

Université de Montréal

**Towards Meaningful & Data-Efficient Learning:
Exploring GAN Losses, Improving Few-shot
Benchmarks, and Multimodal Video Captioning**

par

Gabriel Huang

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée à la Faculté des arts et des sciences
en vue de l'obtention du grade
de Philosophiæ Doctor (Ph.D.) en informatique

Septembre, 2022

© Gabriel Huang, 2022.

Université de Montréal
Faculté des arts et des sciences

Cette thèse intitulée:

**Towards Meaningful & Data-Efficient Learning:
Exploring GAN Losses, Improving Few-shot
Benchmarks, and Multimodal Video Captioning**

présentée par:

Gabriel Huang

黃渤軒

a été évaluée par un jury composé des personnes suivantes:

Pascal Vincent,	président-rapporteur
Simon Lacoste-Julien,	directeur de recherche
Hugo Larochelle,	membre du jury
Lucas Theis,	examineur externe

Thèse acceptée le:

À mon père, l'inventeur dans l'âme, qui m'a partagé son amour des sciences.

To my dad, the inventor at heart, who sparked my love for science.

致我的父親，一位打從心底的發明家，與我分享了他對科學的熱忱



Résumé

Ces dernières années, le domaine de l'apprentissage profond a connu des progrès énormes dans des applications allant de la génération d'images, détection d'objets, modélisation du langage à la réponse aux questions visuelles. Les approches classiques telles que l'apprentissage supervisé nécessitent de grandes quantités de données étiquetées et spécifiques à la tâche. Cependant, celles-ci sont parfois coûteuses, peu pratiques, ou trop longues à collecter. La modélisation efficace en données, qui comprend des techniques comme l'apprentissage *few-shot* (à partir de peu d'exemples) et l'apprentissage *self-supervised* (auto-supervisé), tentent de remédier au manque de données spécifiques à la tâche en exploitant de grandes quantités de données plus "générales". Les progrès de l'apprentissage profond, et en particulier de l'apprentissage *few-shot*, s'appuient sur les *benchmarks* (suites d'évaluation), les métriques d'évaluation et les jeux de données, car ceux-ci sont utilisés pour tester et départager différentes méthodes sur des tâches précises, et identifier l'état de l'art. Cependant, du fait qu'il s'agit de versions idéalisées de la tâche à résoudre, les *benchmarks* sont rarement équivalents à la tâche originelle, et peuvent avoir plusieurs limitations qui entravent leur rôle de sélection des directions de recherche les plus prometteuses. De plus, la définition de métriques d'évaluation pertinentes peut être difficile, en particulier dans le cas de sorties structurées et en haute dimension, telles que des images, de l'audio, de la parole ou encore du texte. Cette thèse discute des limites et des perspectives des *benchmarks* existants, des fonctions de coût (*training losses*) et des métriques d'évaluation (*evaluation metrics*), en mettant l'accent sur la modélisation générative - les Réseaux Antagonistes Génératifs (*GANs*) en particulier - et la modélisation efficace des données, qui comprend l'apprentissage *few-shot* et *self-supervised*. La première contribution est une discussion de la tâche de modélisation générative, suivie d'une exploration des propriétés théoriques et empiriques des fonctions de coût des *GANs*. La deuxième contribution est une discussion sur la limitation des *few-shot classification benchmarks*, certains ne nécessitant pas de généralisation à de nouvelles sémantiques de classe pour être résolus, et la proposition d'une méthode de base pour les résoudre sans étiquettes en phase de *testing*. La troisième contribution est une revue sur les méthodes *few-shot* et *self-supervised* de détection d'objets, qui souligne les limites et directions de recherche prometteuses. Enfin, la quatrième contribution est une méthode efficace en données pour la description de vidéo qui exploite des jeux de données texte et vidéo non supervisés.



Abstract

In recent years, the field of deep learning has seen tremendous progress for applications ranging from image generation, object detection, language modeling, to visual question answering. Classic approaches such as supervised learning require large amounts of task-specific and labeled data, which may be too expensive, time-consuming, or impractical to collect. Data-efficient methods, such as few-shot and self-supervised learning, attempt to deal with the limited availability of task-specific data by leveraging large amounts of general data. Progress in deep learning, and in particular, few-shot learning, is largely driven by the relevant benchmarks, evaluation metrics, and datasets. They are used to test and compare different methods on a given task, and determine the state-of-the-art. However, due to being idealized versions of the task to solve, benchmarks are rarely equivalent to the original task, and can have several limitations which hinder their role of identifying the most promising research directions. Moreover, defining meaningful evaluation metrics can be challenging, especially in the case of high-dimensional and structured outputs, such as images, audio, speech, or text. This thesis discusses the limitations and perspectives of existing benchmarks, training losses, and evaluation metrics, with a focus on generative modeling—Generative Adversarial Networks (GANs) in particular—and data-efficient modeling, which includes few-shot and self-supervised learning. The first contribution is a discussion of the generative modeling task, followed by an exploration of theoretical and empirical properties of the GAN loss. The second contribution is a discussion of a limitation of few-shot classification benchmarks, which is that they may not require *class semantic* generalization to be solved, and the proposal of a baseline method for solving them without test-time labels. The third contribution is a survey of few-shot and self-supervised object detection, which points out the limitations and promising future research for the field. Finally, the fourth contribution is a data-efficient method for video captioning, which leverages unsupervised text and video datasets, and explores several multimodal pretraining strategies.



Keywords—Mots-clés

Keywords. self-supervised learning, few-shot classification, few-shot object detection, low-data learning, object detection, instance segmentation, representation learning, residual network (Resnet), visual transformer (ViT), Faster R-CNN, DETR, parametric adversarial divergence, generative adversarial network (GAN), variational auto-encoder (VAE), maximum-likelihood, structured prediction, optimal discriminator, mutual information, implicit generative model, multimodal pretraining, dense video captioning, cross-attention, YouCook2, HowTo-100M, Youtube-8M, Recipe-1M, Pascal VOC, MSCOCO, LVIS, mutual information neural estimation (MINE)

Mots-clés. apprentissage auto-supervisé, classification *few-shot*, détection d'objets *few-shot*, apprentissage efficace en données, segmentation en instances, apprentissage de représentation, réseau résiduel (Resnet), *transformer* visuel (ViT), Faster R-CNN, DETR, divergences antagonistes paramétriques, auto-encodeur variationnel (VAE), maximum de vraisemblance, prédiction structurée, discriminateur optimal, information mutuelle, modèle génératif implicite, pré-apprentissage multi-modal, description dense de vidéo, attention croisée, YouCook2, HowTo-100M, Youtube-8M, Recipe-1M, Pascal VOC, MSCOCO, LVIS, estimation neuronale d'information mutuelle (MINE)



Contents

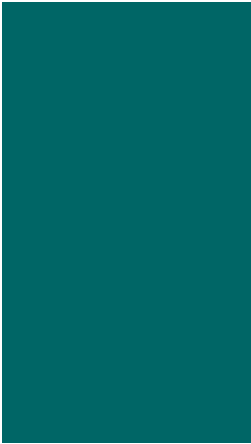
1	Introduction	1
1	Overview of the Thesis Structure	2
2	Research Contributions	3
2.1	Data-efficient Learning	3
2.2	GAN Losses and Generative Modeling	4
2.3	Video Captioning	4
3	Excluded Publications	5
2	Background	6
1	Types of Machine Learning Paradigms	6
1.1	Supervised Learning	7
1.2	Unsupervised Learning	7
1.3	Representation Learning	9
1.4	Transfer Learning	10
1.5	Few-shot and Zero-shot Learning	11
2	Architectures	12
2.1	Neural Network	12
2.2	Convolutional Neural Networks	13
2.3	Transformers	13
3	Training and Evaluation	14
3.1	Evaluation metric	15
3.2	Training loss	15
3.3	Optimization	16
4	Generative Modeling	17
4.1	Maximum Likelihood Estimation	17
4.2	Variational Autoencoders	18
4.3	GANs	18
3	Prologue to First Contribution	21
1	Article Details	21
2	Contributions of the authors	21
3	Limitations and scope of the paper	21

4	Recent Developments	22
4	Exploring Properties of GAN Losses	26
1	Introduction	26
2	Background	28
	2.1 Structured Prediction	28
	2.2 Parametric and Nonparametric Adversarial Divergences	29
3	Divergences as Task Losses	31
	3.1 Formalizing Final Tasks with Statistical Decision Theory	32
	3.2 Parallels between Predictive and Generative Tasks.	33
	3.3 An Analogy with Structured Prediction	35
	3.4 Training and Evaluation in Practice	36
4	Good Divergences Should Scale Well with Data Dimensionality	38
	4.1 Theoretical Sample Complexities in High-Dimensions	39
	4.2 Nonparametric vs. Parametric Wasserstein (Experiment)	41
	4.3 Generating Simple High-dimensional Images (Experiment)	42
5	Good Divergences Should Reflect the Final Task	44
	5.1 Tuning Various Divergences to the Final Task	44
	5.2 Sensitivities of Divergences to Different Aspects of the Sum-25 Distribution	45
	5.3 Generating the Sum-25 Distribution	49
6	Interactions between Generator and Divergence	51
	6.1 Interaction of KL-divergence with Special Generators	51
	6.2 Ability to Train Memorization-based Generators (Experiment)	52
7	Parametric Divergences for Meaningful Mutual Information	53
	7.1 The Corrupted-Label Paradox	54
	7.2 The Hashing Paradox	55
	7.3 Statistical Dependency can be Arbitrarily Complex	57
	7.4 Generalized and Parametric Mutual Information	58
	7.5 MMD-Independence for Integrating Smooth Functions.	61
	7.6 Semi-Discrete Case	64
	7.7 GMI for Corrupted Label Datasets (Experiments)	66
8	Related Work	69
9	Conclusion	71
5	Prologue to Second Contribution	72
	1 Article Details	72
	2 Contributions of the Authors	72
	3 Context and Limitations	72

4	Recent developments	73
6	Solving Few-Shot Learning Benchmarks without Test-Time Labels	75
1	Introduction	75
2	Related Work	77
3	Few-shot Tasks and Evaluation	79
4	Centroid Networks	80
	4.1 Prototypical Networks	80
	4.2 Sinkhorn K-Means	80
	4.3 Combining ProtoNet with Sinkhorn K-Means	81
5	Experiments	82
	5.1 Main Results	82
	5.2 Cross-Domain NLT Few-Shot Classification	82
	5.3 Exploring Other Few-Shot Settings	85
6	Conclusion	87
7	Broader Impact	87
8	Acknowledgements	87
7	Prologue to Third Contribution	88
1	Article Details	88
2	Contributions of the authors	88
3	Recent Developments	88
8	A Survey of Self-Supervised and Few-Shot Object Detection	90
1	Introduction	90
2	Related Surveys	91
3	Background on Object Detection	93
	3.1 Key Concepts	93
	3.2 Datasets and Evaluation Metrics	97
4	Few-Shot Object Detection	97
	4.1 FSOD Framework	98
	4.2 Important Differences with Few-Shot Classification.	100
	4.3 FSOD Datasets	101
	4.4 FSOD Evaluation Metrics	105
	4.5 Few-Shot Object Detection Methods	107
5	Self-Supervised Pretraining	112
	5.1 Image-level Backbone Pretraining	113
	5.2 Issues of Combining Self-Supervised Classification with De- tection	116
	5.3 Object Detection Pretraining	117
	5.4 Comparison of Self-Supervised Object Detection Methods	120

6	Takeaways & Trends	121
6.1	Finetuning is a strong baseline	121
6.2	Impact of self-supervision for object detection	121
6.3	Using heuristics to generate weak labels	122
6.4	Rise of transformers	122
6.5	Problems with current evaluation procedures	123
7	Related Tasks	124
7.1	Weakly-supervised object detection	124
7.2	Self-supervision using other modalities	125
7.3	Low-data and semi-supervised object detection	125
7.4	Few-shot semantic segmentation	126
7.5	Zero-shot object detection	126
8	Conclusion	126
9	Prologue to the Fourth Contribution	128
1	Article Details	128
2	Contributions of the authors	128
3	Context and Limitations	128
4	Recent Developments	129
10	Multimodal Pretraining for Dense Video Captioning	130
1	Introduction	130
2	Related Work	132
3	Data	133
3.1	Dense Video-Captioning Datasets	133
3.2	Pretraining Datasets: ASR+Video	135
3.3	Pretraining Datasets: CAP-style	136
3.4	Differences between Pretraining and Finetuning Datasets	136
4	Method	137
4.1	Separate-Modality Architecture	137
4.2	Pretraining with Text-only MASS	138
4.3	Pretraining with Multimodal MASS	139
4.4	Pretraining with Alignment and Ordering Tasks	139
4.5	Finetuning on Video Captioning	140
5	Experiments	140
5.1	Implementation Details	140
5.2	Main Results	143
6	Conclusions	145
11	Conclusions, Discussions, and Perspectives	146
1	Summary and Conclusions	146

2	Future research directions.	147
A	Appendix for: “Exploring Properties of GAN and VAE Losses”	176
1	Experimental results	176
1.1	Additional Samples for VAE and GAN	176
1.2	Sum-25: Generated samples	180
B	Appendix for “Are Few-Shot Learning Benchmarks too Simple ? Solving them without Test-Time Labels”	181
1	Links to the Code	181
2	Additional Related Work from Clustering Literature	181
3	Additional information on Sinkhorn K-Means.	182
4	Implementation Details	185
5	Additional Few-Shot Clustering Results	187
6	Ablation Study	188
7	Confidence Intervals	189
C	Appendix for “Multimodal Pretraining for Dense Video Captioning”	191
1	The ViTT dataset	191
2	Separated vs. Concatenated-Modality Architecture	195
3	Additional Implementation Details	195
4	Example Predictions	195
5	Full result tables	199



List of Tables

4.1	Sample complexity and other properties of divergences	40
4.2	Log-likelihood of Test -25 w.r.t. Sum-25 constraint	48
6.1	LT and NLT few-shot classification on same-domain benchmarks . .	83
6.2	LT and NLT few-shot classification on cross-domain benchmarks . .	84
6.3	Few-shot clustering on Omniglot with CCN splits	85
6.4	Transductive 5-way 5-shot classification on <i>mini,tiered</i> ImageNet . .	86
8.1	Common few-shot object detection benchmarks	101
8.2	Example computation of precision and recall for object detection . .	106
8.3	FSOD methods with results on PASCAL VOC and MS COCO . . .	108
8.4	Comparison of self-supervised object detection methods pretrained on ImageNet	118
10.1	Datasets used in this work	134
10.2	Pretraining and Fine-tuning objectives	140
10.3	Main results on YouCook2	141
10.4	Main results on ViTT	142
10.5	Ablation study on YouCook2	144
B.1	Clustering accuracies on Omniglot and <i>mini</i> ImageNet	187
B.2	Confidence intervals for same-domain benchmarks	189
B.3	Confidence intervals for cross-domain benchmarks	190
C.1	Standardization of top tags	193
C.2	10 most frequent tags after standardization.	193
C.3	Estimate of human performance on ViTT-All	194
C.4	Estimate of human performance on ViTT-Cooking	194
C.5	Video Captioning Results on YouCook2	200
C.6	Video captioning results on ViTT-All	201
C.7	Video captioning results on ViTT-Cooking	202



List of Figures

3.1	DALL-E 2 failure mode: mixing attributes from distinct characters	23
3.2	DALL-E 2 failure mode: inability to generate text (Gryphon)	24
3.3	DALL-E 2 failure mode: inability to generate text (Wikipedia)	24
3.4	DALL-E 2 failure mode: inability to generate text (Infinite Jest)	25
4.1	“Good” and “Bad” Task Losses	32
4.2	Generated samples using Sinkhorn-Autodiff	42
4.3	Real and generated <i>Thin-8</i> samples	43
4.4	Probing the discriminator	47
4.5	Histogram and recall for Sum-25 task	50
4.6	Generated samples w.r.t. discriminator architecture	53
4.7	Illustrating the hashing paradox	56
4.8	Various GMI w.r.t. number of bins (hashing paradox)	63
4.9	Estimating various image-to-label GMI on MNIST and SVHN	68
8.1	Taxonomy of self-supervised and few-shot object detection methods	92
8.2	Faster R-CNN with Feature Pyramid Network	95
8.3	DETR object detector	96
8.4	Few-shot object detection (FSOD) framework	99
8.5	Importance of evaluating over several episodes	102
8.6	Example precision and recall curves for object detection	106
8.7	Meta-YOLO, a modulation-based FSOD method	110
8.8	DINO’s attention maps	116
10.1	Dense video captioning using ViTT-trained models.	131
10.2	A diagram for the separate-modality architecture	138
A.1	VAE and GAN samples (32×32)	177
A.2	VAE and GAN samples (128×128)	178
A.3	VAE and GAN samples (512×512)	179
A.4	Sum-25 Generated Samples	180
B.1	Omniglot 5-way 5-shot ablation study	188
B.2	<i>mini</i> ImageNet 5-way 5-shot ablation study	188

C.1	Example good and bad predictions on YouCook2	196
C.2	Example good predictions on ViTT-All (Part 1)	197
C.3	Example ok and bad predictions on ViTT (Part 2)	198



List of acronyms and abbreviations

i.e.	<i>id est</i> [that is]
e.g.	<i>exempli gratia</i> [for example]
resp.	respectively
ASR	automated speech recognition
BERT	bidirectional encoder representations from transformers
BLEU $-\{1, 4\}$	language evaluation metric (precision-based)
BYOL	Bootstrap your Own Latent (self-supervised method)
CAP	caption
CCN	constrained clustering network
CIDEr	language evaluation metric
CNN	convolutional neural network
DETR	detection transformer
Faster-RCNN	faster regions with CNN (object detector)
FPN	feature pyramid network
FSC	few-shot classification
FSOD	few-shot object detection
GAN	generative adversarial network
GMI	generalized mutual information
IoU	intersection-over-union
KL	Kullback-Leibler divergence
LT/NLT	labels/no-labels at test time
mAP	mean average precision
MASS	masked sequence-to-sequence pretraining
MI	mutual information
ML	machine learning
MMD	maximum-mean discrepancy
MoCo	momentum-encoder (self-supervised method)
NMS	non-maximum suppression (heuristic)
PixelCNN	image generation architecture
PMI	parametric mutual information
ReLU	rectifier linear unit (nonlinearity)
ResNet	neural architecture with residual connections
RGB	red-green-blue (image color channels)
RKHS	reproducing kernel Hilbert space

RL	reinforcement learning
ROI	region of interest
ROUGE-L	language evaluation metric (recall-based)
RPN	region-proposal network
SDPMI	semi-discrete PMI
SGD	stochastic gradient descent
SimCLR	simple framework for contrastive learning (self-supervised)
SVM	support vector machine
VAE	variational auto-encoder
ViT	visual transformer
WGAN-GP	Wasserstein GAN with gradient penalty



Acknowledgements

I am grateful and forever indebted to the incredible people I encountered during my Ph.D., and the ones who paved the way leading there. It was not just an academic journey, but truly a search for meaning and beauty, a story of human connection, of growth, and cultivating the resilience necessary to undertake any worthwhile challenge.

First and foremost, I would like to thank my adviser Simon Lacoste-Julien, a wonderful mentor and human being, who has always believed in me and has given me hope, guidance, and motivation even during the toughest times. He has genuinely cared for me during this PhD, not just as a student, but also as a young adult who is building his life. He has always taken the hype and trends of deep-learning with a grain of salt, and encouraged me to pursue longer-term, meaningful research, instead of going for the next low-hanging fruit.

Second, I would like to thank Hugo Larochelle, Pascal Vincent, Ioannis Mitliagkas, Bo Pang, Zhenhai Zhu, David Vazquez, Pau Rodriguez, Issam Laradji, Alexandre Lacoste, and Fabrice Michel who have guided and mentored me during the course of my internships and research projects.

Thanks all my collaborators, co-authors and friends, including Ahmed Touati, Hugo Bérard, Gauthier Gidel, Edouard Oyallon, Mohammed Pezeshki, Reyhane Askari Hemmat, Mathew Blaschko, Eugene Belilovsky, Nikos Komodakis, Sergey Zagoruyko, Namyong Kwon, Hwidong Na, Clara Rivera, Radu Soricut.

Thanks to the incredible Mila community, for being so stimulating and positive. I am honored to be part of this grand family, and I am more than grateful to all the people who have contributed to creating such an open, collaborative, and diverse research environment.

Thanks to my friend Rémi Le Priol, who has inspired me in countless ways, taken me out of my comfort zone, and shown me how much influence we have over the course of our own lives. Thanks to my friend and roommate Thierry St-Pierre, who has shared his spiritual discoveries with me, and in some ways knows me better than I know myself. I would like to thank the acroyoga community of Montreal, who have shown me such a playful and healthy way to reconnect with our minds and bodies. For people who spend the majority of their day on a screen, this is a

godsend. And I want to thank my friends who have made my life here so much better: Robin, Nicole, Thiago, Jules, Oscar, Thomas, Henri, Fiona, Elena, Filipe, Alexandre, Elody, Valentin, David, Tess, Will, Pierre, Ornella, Yaël, Stéphanie, Nathan, Tony, Julien, Anita, Daniel, Jacques, Ember, and many more who I haven't forgotten ;)

Thanks to my husband Pang-Ying Peng, who has always supported, understood, and loved me in the deepest ways. You bring me unlimited joy, and you are the pillar of stability that I sometimes so desperately need. We have a bright future together.

And of course, thanks to my little sister Daphné for always being supportive, loving, and understanding, despite our very different personalities! I wish you as much success in your artistic career as you did for me, and I am confident your resilience and work ethic will bring their fruits.

Finally, the biggest thanks go to my mom and dad, Miaowen Wang (王妙文) and Ming Huang (黄明). My dad has nurtured my curiosity and love of science since I was a kid, showing me the magic of technology, the satisfaction of building things with my own hands, and teaching me how to program on my request—that was in *C*, what a rough way to learn for a 10-year-old! My mom has grown my sense of aesthetics and my creativity. She taught me piano when I was just a kid; and when I insisted that I wanted to switch to violin, she accepted, but only on the condition that I would practice a half hour a day, every day, until I turn eighteen. This was tough at times, but she made sure I never went back on my promise—a promise I made when I was six! Together, they carefully planned out a pathway for me in the French schooling system, long before I even knew what options lay ahead, and exposed me early-on to successful stories of higher education. I wouldn't be where I am today without your unconditional love and care. Thank you, truly.

1

Introduction

Machine learning [Mitchell and Mitchell, 1997], a branch of artificial intelligence, can be defined by the approach of training an algorithm to solve problems, make predictions, and take decisions, based on training datasets, human feedback, or interactions with a real or simulated environment. Recent successes in machine learning, such as large language models [Brown et al., 2020], text-to-image generators [Ramesh et al., 2022], visual question answering [Alayrac et al., 2022], and object detectors [Ren et al., 2015, Carion et al., 2020] can be attributed to *deep learning* [Goodfellow et al., 2016], which is the art of using neural networks to learn high-level representations of the data. Besides a substantial amount of elbow grease, deep learning involves designing neural network architectures, training losses, choosing an optimizer to minimize the losses, and tuning the hyperparameters to make everything work, on top of curating learning datasets, collecting labels, and defining appropriate evaluation benchmarks.

Traditional supervised learning methods are hindered by the limited availability of large task-specific datasets. For instance, for fine-grained classification tasks with a long-tailed distribution, such as fungus image classification or human face verification, there may only be a few images available for the rarest varieties (resp. least occurring faces), but we may nevertheless want a system to perform decently on them. Data-efficient methods attempt to circumvent the limited availability of data using various strategies. In this thesis, we focus on few-shot learning [Ravi and Larochelle, 2016, Vinyals et al., 2016b] and self-supervised learning [He et al., 2020, Chen et al., 2020a] which are two complementary approaches which leverage additional data and prior knowledge of the task to learn more efficiently from the task-specific data.

Generative modeling is a different paradigm from supervised learning, where the goal is to generate new data, such as images, from a given distribution, typically with the goal of having *diverse* and *realistic* samples [Arora et al., 2018]. As of this writing, *Generative Adversarial Networks* or GANs, introduced by Goodfellow et al. [2014], are still one of the state-of-the-art models [Karras et al., 2018b]; in terms of image quality, and sampling speed. GANs present a departure from traditional likelihood-maximization approaches, which fit a model density to the training data by maximizing the likelihood. Instead, the GAN generator tries to fool an auxiliary network, the *discriminator*, which is *jointly* trained to recognize

real images from synthetic images. In this thesis, we abstract away the alternating optimization of the generator and the discriminator, instead viewing the GAN generator as minimizing a *parametric divergence*. We study the empirical and theoretical properties of parametric divergences, opposing them to what we call *nonparametric divergences* such as the Kullback-Leibler, Jensen-Shannon [Nowozin et al., 2016], and Wasserstein distance [Arjovsky et al., 2017].

The following sections contain an overview of the thesis structure (Section 1), introduce our research contributions in context (Section 2), and link to publications excluded from this thesis (Section 3).

1 Overview of the Thesis Structure

Beyond this introduction (Chapter 1) and the conclusion (Chapter 11), this thesis contains a background section (Chapter 2) with key notions for understanding the research contributions, followed by the four contributions themselves, each corresponding to a research paper and preceded by a prologue section:

- **First Contribution. “Parametric Adversarial Divergences are Good Losses for Generative Modeling”** by *Gabriel Huang, Hugo Berard, Ahmed Touati, Gauthier Gidel, Pascal Vincent, Simon Lacoste-Julien* [Huang et al., 2017]. This paper was initially accepted as a ICLR 2018 Workshop paper, then presented at the Montreal AI Symposium 2018; a conference version has been submitted to ICML 2018; finally, a journal version was submitted to JMLR. (Chapters 3, 4)
- **Second Contribution. “Are Few-Shot Learning Benchmarks too Simple? Solving them without Test-Time Labels”** by *Gabriel Huang, Hugo Larochelle, Simon Lacoste-Julien* [Huang et al., 2019]. This work was accepted as an ICLR 2019 workshop paper; a full version was submitted to ICLR 2020, ICML 2020 and NeurIPS 2020; this project was also presented at the Montreal AI Symposium 2020. (Chapters 5, 6)
- **Third Contribution. “A Survey of Self-Supervised and Few-Shot Object Detection”** by *Gabriel Huang, Issam Laradji, David Vázquez, Simon Lacoste-Julien, Pau Rodríguez* [Huang et al., 2021]. This paper has been accepted with minor revision at the IEEE Transactions in Pattern Analysis and Machine Intelligence (TPAMI). (Chapters 7, 8)
- **Fourth Contribution. “Multimodal Pretraining for Dense Video Captioning”** by *Gabriel Huang, Bo Pang, Zhenhai Zhu, Clara Rivera, Radu Soricut* [Huang et al., 2020]. This paper has been published at ACL-IJCNLP 2020. (Chapters 9, 10)

2 Research Contributions

2.1 Data-efficient Learning

Traditional methods such as supervised learning require large amounts of *task-specific* and *supervised* datasets, which can be difficult, expensive, and impractical to obtain, hindering the application of deep learning methods to low-data settings [Chen et al., 2019a]. Data-efficient methods attempt to circumvent these shortcomings by exploiting additional datasets or the inherent structure of the dataset. *Few-shot learning* is about training models to learn from a small number of labeled examples, typically by leveraging a larger dataset [Vinyals et al., 2016b, Ravi and Larochelle, 2016]. *Self-supervised learning* is about exploiting prior information (such as data augmentations) or the structure of the data (such as predicting missing words) to learn from unsupervised datasets.

Stronger generalization for few-shot classification. Few-shot classification methods aim to recognize *new* (previously *unseen*) classes or object categories during evaluation, after being exposed to a large number of *known* classes, during training [Ravi and Larochelle, 2016, Finn et al., 2017, Vinyals et al., 2016b]. Omniglot [Lake et al., 2011] and *miniImageNet* [Vinyals et al., 2016b] are among the most popular few-shot classification benchmarks. However, a limitation of these benchmarks is that they do not require learning new class semantics, instead testing methods on *novel classes* only with *known* semantics—*alphabet letters* for Omniglot, *objects* for *miniImageNet*. Such benchmarks may not be representative of few-shot classification benchmarks deployed in the wild, which may be required to recognize new class semantics, such as traffic signs on one occasion, and identify human faces on another.

⇒ Our main contribution to few-shot classification is a baseline method, *centroid networks* [Huang et al., 2019], for solving few-shot classification benchmarks without using any test-time labels, instead relying on clustering of the learned representations to infer the labels. Though our approach has shortcomings of its own, it represents a first step towards quantifying the difficulty of few-shot classification benchmarks in terms of class semantic generalization. As secondary contributions, our method can be used to make inductive few-shot classification methods transductive, solve learning-to-cluster problems; moreover, our clustering algorithm—Sinkhorn K-Means— and its variants, can be used for learning self-supervised representations, as exemplified by SwAV [Caron et al., 2020].

Interplay of self-supervision and few-shot object detection. Object detection consists in localizing, and classifying known objects in an image, typically by predicting bounding box locations and class identities in the image. *Few-shot* object detection consists in learning to detect new objects, based on a small num-

ber of training examples. Previously, few-shot object detection methods such as Meta-RCNN [Yan et al., 2019] or TFA [Wang et al., 2020a] had relied on supervised pretraining, solely of the backbone network. More recent approaches such as DETReg [Bar et al., 2021] leverage self-supervised pretraining, and were able to achieve state-of-the-art performance at the time of publication.

⇒ Our main contribution to the field is a joint survey of self-supervised and few-shot object detection [Huang et al., 2021], which fully explores the interactions between the two paradigms. We create a taxonomy of self-supervised and few-shot object detection techniques, clarify the differences —and there are many!— with few-shot classification, point the the limitations of current evaluation procedures, emphasize the success of simple finetuning-based approaches, and give recommendations for best practices in the future. In particular, we take note of the efficiency of transformers, both at the object detection level, and the backbone level, and advise readers to keep an eye on fully autoregressive approaches which formulate object detection outputs as text.

2.2 GAN Losses and Generative Modeling

⇒ Our main contribution to generative modeling is an investigation of properties of the GAN loss. We start by trying to formalize the problem of image generation by drawing parallels with structured prediction [Taskar et al., 2003], which has long dealt with high-dimensional and structured outputs. Then, we investigate properties such as sample complexity and the ability to enforce certain consistency properties of the distribution (e.g. an arithmetic constraint with the Sum-25 experiment). We point out shortcomings of traditional divergences such as Kullback-Leibler and Jensen-Shannon, explain how integral probability metrics such as MMD Gretton et al. [2007] and the Wasserstein distance Arjovsky et al. [2017], Cuturi [2013] fail to address all of them, and how *parametric divergences* provide a viable alternative. Finally, we explore an extension of parametric divergences to define more intuitive notions of mutual information. In particular, we find out in the finite-data regime that *parametric mutual information* can overcome some apparent paradoxes of traditional mutual information.

2.3 Video Captioning

Video Captioning for Timeline Tagging. Because tutorial videos mainly consist of frames with narrated speech, it can be difficult for a user to navigate through the different steps of a tutorial. Therefore, we propose to automatically caption each step based on two modalities: video frames, and transcripts of the narrated instructions. Previous approaches either only exploit the video information [Zhou et al., 2018c, Sun et al., 2019a, Lei et al., 2020], or exploit both the video and speech information, but are limited by the lack of large annotated datasets [Hessel et al.,

2019]. Existing datasets such as YouCook2 [Zhou et al., 2018a] are limited in size, restricted in scope (recipes only), and demand limited generalization (same recipes are split between training and development sets).

⇒ Our contributions are two-fold: a multimodal pretraining strategy for dense video captioning, and a new dataset for timeline tagging [Huang et al., 2020]. Our method continuously projects video features to embeddings, and features two transformers which cross-attend to each other. We investigate several pretraining strategies, ranging from video-augmented Cloze-style text completion [Song et al., 2019], to text-video alignment and ordering prediction. We also investigate several finetuning strategies, some of which turn out to be beneficial even in the absence of pretraining. Compared to previous methods, our approach is multimodal, leverages nonparallel video and text pretraining sets, and has achieved state-of-the-art results at the time of publication [Huang et al., 2020], with the biggest gain attributable to text-only pretraining. Compared to YouCook2 [Zhou et al., 2018a], our new dataset, *Video Timeline Tagging*, features a wide range of activities, contains shorter labels which are more suitable for timeline tagging, and requires stronger generalization because we do not explicitly split each activity across training and development set. It is publicly available online.¹

3 Excluded Publications

Several publications, which I have contributed to during my PhD, have been excluded from this thesis, in order to keep the manuscript more consistent and succinct:

- **“Scattering Networks for Hybrid Representation Learning”** by *Edouard Oyallon, Sergey Zagoruyko, Gabriel Huang, Nikos Komodakis, Simon Lacoste-Julien, Matthew Blaschko, Eugene Belilovsky*, published in IEEE TPAMI 2018 [Oyallon et al., 2018].
- **“Negative Momentum for Improved Game Dynamics”** by *Gauthier Gidel, Reyhane Askari Hemmat, Mohammad Pezeshki, Rémi Le Priol, Gabriel Huang, Simon Lacoste-Julien, Ioannis Mitliagkas*, published at AISTATS 2019 [Gidel et al., 2019].
- **“Repurposing Pretrained Models for Robust Out-of-domain Few-Shot Learning”** by *Namyeong Kwon, Hwidong Na, Gabriel Huang, Simon Lacoste-Julien*, published at ICLR 2021 [Kwon et al., 2021].

¹<https://github.com/google-research-datasets/Video-Timeline-Tags-ViTT>

2 Background

This chapter briefly introduces the necessary machine learning background to understand the PhD contributions:¹

- Section 1 reviews some common paradigms in machine learning: supervised learning, clustering, generative modeling, representation learning, few-shot and zero-shot learning.
- Section 2 introduces neural networks and some of the architectures we use in our contributions, such as the convolutional neural network and the transformer.
- Section 3 introduces training losses, evaluation metrics, optimizers, data splits and the relationship between them.
- Section 4 introduces the generative modeling task, Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs).

For background information on dense video captioning (Chapter 10) and object detection (Chapter 8) we refer the reader to the corresponding chapters.

1 Types of Machine Learning Paradigms

Machine learning can be characterized by the idea of building models that gradually learn to solve certain tasks, based on training examples, human feedback, or interactions with a real or simulated environment. The field can be subdivided into several paradigms and tasks, which may overlap and are frequently combined together. We introduce the supervised learning (Section 1.1), unsupervised learning (Section 1.2), transfer learning (Section 1.4) and few-shot/zero-shot learning (Section 1.5), with a focus on the challenges of training and evaluation.

¹Please note that “machine learning” and “artificial intelligence” will be used interchangeably through this thesis.

1.1 Supervised Learning

Supervised learning is about learning a **predictor** which takes an input x and outputs a label y . The predictor is a function $h_\theta : \mathcal{X} \rightarrow \mathcal{Y}$, parametrized by its model parameters θ . Typically, the predictor is trained to minimize a **training loss** $\mathcal{L}(\theta)$ on a **training set** consisting of multiple input-output pairs (x_i, y_i) known as *examples*. Then, the predictor is evaluated on a held-out set of examples, known as the **test set**, using an evaluation metric $L(\theta)$. An intermediate **validation set** is often used, to tune the model hyperparameters. The hope is that the model can generalize to new (previously unseen) examples.

Here are some example tasks which can be formulated as supervised learning problems:

- *Image classification*: x is an X-ray of a patient’s wrist, y is whether that there is a broken bone.
- *Speech recognition*: x is an audio signal, y is the transcription of the person’s speech.
- *Machine translation*: x is a sentence in French, and y is a sentence in English.
- *Object detection*: x is an image to analyze, y is the list of detected objects with their categories and location in the image.
- *Video captioning*: x is a video clip, y is a text summary of the video clip.
- *Style transfer*: $x = (c, s)$ is a pair of image and target style, y is the transformed image.
- *Closed-book question answering*: x is a question in natural language, y is the answer.

Challenges. Supervised learning is a general framework which encompasses a wide variety of prediction problems. A challenging aspect remains the *choice of evaluation metric*, especially for evaluating high-dimensional and structured outputs, such as natural language. Another challenge is to acquire sufficient *task-specific labels*, which has motivated the development of transfer learning (Section 1.4) and other low-data approaches (Section 1.5).

1.2 Unsupervised Learning

Unsupervised learning is a different paradigm in that the training set is unlabeled—it has no additional human annotations. Unsupervised learning encompasses a wide range of tasks—generative modeling, representation learning, clustering—with applications ranging from data analysis to prediction and data generation.

Generative Modeling. The goal is to learn a generative model $q_\theta(x)$, which can generate new samples, which we generally require to be *diverse* and *realistic*. We make the distinction between explicit generative models, which can compute the likelihood of a sample numerically, and implicit generative models, which we can only sample from a learned distribution. Examples tasks are:

- *Language modeling.* The goal is to learn a probability distribution—an explicit model—over a sequence of words, corresponding to sentences in a language such as English. Typically, language models are autoregressive and factorize as $p(x_1, x_2, \dots, x_T) = p(x_1) \cdot p(x_2|x_1) \cdots p(x_T|x_{T-1})$. Recurrent neural networks and LSTMs used to be popular for language modeling, but they have been superseded by Transformers in the last few years.
- *Image generation.* The goal is to learn a model that can generate realistic images. Successful implicit models include Generative Adversarial Networks (GANs), while successful explicit models include Diffusion Models [Ho et al., 2020], and Variational Autoencoders [Kingma and Welling, 2014] to some extent [Ramesh et al., 2021].

Evaluating image generation is still an open problem, and has motivated our work on exploring properties of the GAN loss (Chapter 3). The most common metric is the Fréchet Inception Distance [Heusel et al., 2017], which basically ensures that distributions of true and generated images match in the representation space defined by a pretrained feature extractor.

Conditional Generation. The goal is to generate data based on some attribute, text, or other modality. Conditional generation blurs the line between supervised and unsupervised learning, because the models may be trained (or finetuned) on labeled data. One thing that sets them apart from “usual” supervised learning tasks such as classification is that outputs are generally high-dimensional and structured (as opposed to one-hot labels). The high-dimensionality and structured aspect of the outputs is a key part of our discussion on generative losses (see Chapter 4); in particular, losses with sample complexities exponential in the output dimension are generally not applicable in practice, while losses with inductive biases (such as convolutions) have better properties by taking advantage of the structured aspect. Many unsupervised generators can be modified to condition on additional data: for instance, GANs can be conditioned on the object category (e.g., for class-conditional generation), image generators can condition on the visible part of the image (e.g., for image inpainting), and language models can condition on visual data (e.g., for image captioning and visual question answering) or on text (e.g., for text summarization or machine translation).

Clustering. Clustering consists in partitioning a dataset into semantically similar groups (according to some definition). The desired groups can be semantic (e.g. similar object class and attribute), defined mathematically (e.g. small intra-class variance). The groups can also be learned in a supervised way, in the context of learning-to-cluster. Applications of clustering can range from data visualization and analysis, to segmenting images, to quantizing continuous representations into discrete tokens, and can also be used to learn unsupervised representations [Caron et al., 2020]. Common algorithms are K-Means [Lloyd, 1982], which iteratively minimizes the intra-class variance, hierarchical clustering methods [Murtagh and Contreras, 2012], which iteratively merge clusters into bigger ones, and Mean Shift [Comaniciu and Meer, 2002], which iteratively finds the modes of the data distribution. In Chapter 6, we introduce a new clustering algorithm named Sinkhorn K-Means for the purpose of clustering few-shot support sets; our method can also be applied to learning-to-cluster [Hsu et al., 2017] and transductive few-shot classification [Vinyals et al., 2016b].

1.3 Representation Learning

We put representation learning in its own section, as both supervised and unsupervised approaches are possible. The goal of representation learning is to learn a function $g_\theta : \mathcal{X} \rightarrow \mathcal{Z}$, known as **feature extractor**, which maps the raw data x to a more convenient **representation** or **features** z , typically with smaller dimensions, and a more semantic space [Bengio et al., 2013]. The representation can be used explicitly, for instance by performing nearest neighbor or clustering on the representations, or implicitly, for instance when a complete language model is finetuned on the downstream task, instead of extracting intermediate activations.

If the goal is for an user to analyze, visualize, or manipulate the data, then the following properties are often desirable:

- **reconstruction.** Conveniently reconstruct the original data from the representation, such as with a feedforward network $f_\theta : \mathcal{Z} \rightarrow \mathcal{X}$.
- **semantic space.** Things that are close in representation space are semantically close (e.g. face representations for the same person are closer than those of different people). Word embeddings can commonly be used to perform “semantic arithmetic” such as $queen = king - man + woman$. Face embeddings can often be linearly interpolated to reconstruct a meaningful interpolation between two faces.
- **disentanglement.** “Independent” attributes are mapped to separate variables; for a 3D car model representation, this would mean different variables for car size, make, color, form factor, and period. This makes the representation easier to interpret and to manipulate.

If the goal is to solve a downstream task, the representation should either improve the downstream performance or reduce its data requirements. The whole process is then known as **transfer learning**, and the representation learning phase as **pretraining** (Section 1.4).

1.4 Transfer Learning

Transfer learning is a paradigm for solving downstream tasks with limited task-specific labels, which often combines supervised and unsupervised learning techniques. Transfer learning can be decomposed into two stages:

1. **Pretraining stage:** learn a representation which can transfer well to the downstream task; in practice, this often means learning a set of model weights which can be easily finetuned afterwards. The pretraining stage can be supervised or unsupervised. For instance, it is common to perform supervised pretraining on ImageNet for object detection tasks [Ren et al., 2015]. Recent image representations are pretrained on unlabeled Imagenet using self-supervised objectives [Chen et al., 2020a, He et al., 2020]. Language models are also commonly pretrained on unsupervised objectives such as standard language modeling objective [Radford et al., 2019] or masked language modeling [Devlin et al., 2018] prior to solving downstream tasks.
2. **Transfer stage:** use the pretrained representation to solve the final task—downstream task—typically by *finetuning* the pretrained representation on the task, though some approaches rely on *frozen* representations.

Finetuning. The most flexible way to use the representation is to *finetune* it on the downstream task, which means using an optimizer to adjust the model weights on the downstream task. For instance, one can add classification heads or object detection heads on top of the representation, and finetune the resulting model to solve these respective tasks. For language tasks, it is common to finetune the pretrained model with little to no changes to the architecture. For multimodal approaches, one can combine pretrained language and vision models, and finetune them jointly. This is the approach we take in our multimodal pretraining for dense captioning project (Chapter 10).

Frozen representations. Simple strategies such as training a linear classifier on top of fixed (frozen) representations—known as **linear probing**—or using a **nearest neighbors** classifier, have the advantage of simplicity and reliability. They require little to no finetuning, and bear little computational resource requirements once the features are precomputed. However, they cannot adapt the representation to the downstream task, and may have limited performance if it differs substantially from the pretraining task. Linear probing and nearest neighbors are also commonly used

to evaluate self-supervised representations (such as those presented in Section 5.1). Some approaches [Tsimpoukelli et al., 2021] combine a frozen language model with a visual backbone to solve multimodal tasks.

1.5 Few-shot and Zero-shot Learning

Although transfer learning is often a viable approach in low-data scenarios, there exist more specialized paradigms such as few-shot and zero-shot learning, semi-supervised learning, and weakly supervised learning, which we describe here. Such paradigms are usually compatible with transfer learning—they often leverage similar pretraining procedure, but differ in the finetuning step. For instance, recent few-shot classification methods usually leverage pretrained representations [Liu et al., 2021a, Triantafillou et al., 2020].

Few-shot learning. There are different variants with distinct goals and evaluation procedures: few-shot classification (FSC), few-shot object detection (FSOD), and few-shot natural language understanding and generation.

In few-shot classification (FSC), the goal is to learn to classify new classes, based on a small training set, the *support set*. Most benchmarks adopt the K -way N -shot paradigm, in which N annotated examples for each of the K classes are provided [Lake et al., 2011, Vinyals et al., 2016b, Ren et al., 2018]. During (meta-)training, meta-learning-based methods [Finn et al., 2017, Snell et al., 2017b, Vinyals et al., 2016a, Ravi and Larochelle, 2016] commonly sample *episodes* which consist of a support set (for training) and a query set (for validation); the model is adapted on the support set and validated on the query set, and the validation loss is minimized end-to-end. Then, during (meta-)evaluation, episodes with new classes are formed, the model is adapted on the support set and makes predictions on the query set.

A driving goal of FSC is to have strong generalization to new classes, but we argue that popular few-shot benchmarks like Omniglot [Lake et al., 2011] do not require learning new class semantics (e.g. traffic signs or fungi species vs. letters), only new classes with the same semantics (e.g. new letters). This is one of the reasons that motivated our work on quantifying the semantic diversity of existing FSC benchmarks (Chapter 5).

Few-shot object detection (FSOD) is the task of learning to detect, localize and recognize new object categories (*novel* classes) based on a few training examples. In practice, FSOD is formulated very differently from FSC (Section 4.2), and there are several subtleties in the evaluation procedure (Section 4.3). We wrote the survey on FSOD to clarify these subtleties and explore the interactions of FSOD with self-supervised learning (Chapter 8).

Zero-shot learning. In zero-shot learning, the model must learn to recognize new classes or solve new tasks without any prior example, relying instead on features, attributes, metadata or natural language, describing the task to solve or the class to recognize. For instance, in zero-shot image classification, one approach is to describe new classes by their attributes, such as *eats-fish: no, stripes: yes* or *color-brown:no* for animal recognition [Xian et al., 2018]. Other approaches describe new classes by specifying their word embeddings; they may conveniently map image features to the semantic space, which can then be searched using nearest neighbors [Xian et al., 2018, Wang et al., 2018b]. For zero-shot natural language understanding, prompt-based methods conveniently formulate new tasks in natural language; when properly pretrained on language modeling, models may even emerge the capability of solving these tasks [Brown et al., 2020]. Performance may be further improved by training on a wide array of language tasks [Sanh et al., 2021]. Recent multimodal approaches leverage the flexibility of full natural language sentences to describe new classes: for instance CLIP [Nichol and Dhariwal, 2021] and Flamingo [Alayrac et al., 2022] describe new classes (resp. tasks) using natural language.

2 Architectures

Deep learning is a subset of machine learning which attempts to solve tasks by means of representation learning with artificial neural networks, typically trained on large datasets using parallel computation.

2.1 Neural Network

Artificial neural networks—or simply neural networks—are very loosely inspired from biological neural network. From an input data such as an image or English sentence—generally represented as a (sequence of) N -dimensional tensor—they compute activations (non-linear transformations of the data) through repeated application of simple functions, and output a representation or prediction. The functions are parametrized by parameters, which can be learned by means of optimizing the training loss, a cost function defined on a set of examples, known as the training set.

The most basic neural network is the Multi-Layer Perceptron (MLP) which is a composition of **dense**—a.k.a. **fully-connected**—layers $f_L \circ \dots \circ f_2 \circ f_1$. Starting from the input data $x \in \mathbb{R}^{N_0}$, each layer f_i transforms the current activations $x^{(l)} \in \mathbb{R}^{N_i}$ and passes them on to the next layer, until the last layer outputs the prediction or final representation. A basic dense layer is the composition of a linear transformation parametrized by weights $W_i \in \mathbb{R}^{N_{i+1} \times N_i}$ and biases $b_i \in \mathbb{R}^{N_{i+1}}$, and

a fixed nonlinearity a known as the activation function.

$$\begin{aligned} f_i : \mathbb{R}^{N_i} &\longrightarrow \mathbb{R}^{N_{i+1}} \\ x &\mapsto a(W_i x + b_i) \end{aligned}$$

A popular nonlinearity is the rectified linear unit or ReLU [He et al., 2016], which clips activations to be positive $a : u \mapsto \max(0, u)$. The weight and bias parameters are learned using an optimizer (see Sec 3.3).

Neural networks defines a very flexible family of function; it has been shown [Hornik, 1991] that even arbitrarily wide two-layer MLPs are universal function approximators. Additionally, neural networks have the advantage of being relatively smooth (ReLU-based neural networks are continuous and differentiable almost everywhere), and can incorporate inductive biases through the choice of architecture, which can reduce data requirements and improve generalization on unseen data.

In the next sections, we briefly review convolutional neural networks and transformers, which are the building blocks of most modern deep-learning architectures.

2.2 Convolutional Neural Networks

Convolutional neural networks (CNN) are an architecture for processing visual and audio signals with inductive biases inspired from human biology [Li et al., 2021c]. CNNs are a composition of convolutional layers which successively apply nonlinear transformations to feature maps, starting from the input data. For images, feature maps are typically a 3D tensor of size $width \times height \times channels$, while for audio they are 2D tensors of size $timesteps \times channels$. Each feature map is computed by convolving feature maps of the previous layer with a learned kernel. The convolution enforces a strong locality prior—each feature is computed from a local patch—and efficiently reuses parameters (filters) across spatial locations. Recent iterations of CNNs are residual networks or ResNets [He et al., 2016]. Residual networks mix convolutional layers with normalization layers, augmented with residual *skip-connections*, which can bypass convolutional layers and enable deep networks with more than 100 layers. To this day, they are still among state-of-the-art architectures, despite the rising popularity of visual transformers [Dosovitskiy et al., 2021].

2.3 Transformers

Processing and generating sequences is necessary for dealing with modalities such as text, video, or audio, which are most naturally formulated as sequences of tokens, frames, or audio samples. The first models for processing sequences of symbols—such as tokenized sentences—were recurrent neural networks (RNN) which are basically MLPs with additional recurrent connections, which feed a layer’s output to itself at the next time step [Sutskever et al., 2014]. However, RNNs are known to suffer from

short-term memory and vanishing gradient issues, which prompted the resurgence of long short-term memory networks (LSTMs) [Schmidhuber et al., 1997], which side-step some of these issues using gates to retain memory.

Transformers were subsequently introduced by Vaswani et al. [2017] and replaced recurrent connections with attention layers, which enable long-term dependencies and are easily parallelizable. As of today, they have become the standard architecture for natural language understanding (NLU)—with encoder-only architectures like BERT [Devlin et al., 2018] or RoBERTa [Liu et al., 2019b]—for natural language generation (NLG)—with decoder-only architectures like GPT-3 [Brown et al., 2020], Megatron Turing NLG [Smith et al., 2022]—and sequence to sequence problems—with encoder-decoder architectures like T5 [Raffel et al., 2019]. While encoder-only architectures are not suitable for generation, encoder-decoder and decoder-only architectures may handle both NLU and NLG tasks.

Recently, it was shown that transformers can be used as feature extractors for images; the images are cut into 8×8 or 16×16 patches, then the patches are projected and flattened to obtain a sequence of tokens, leading to the Visual Transformer or ViT [Dosovitskiy et al., 2021, Touvron et al., 2020]. Both text and vision transformers have been shown to have favorable scaling properties with respect to model size and training data [Kaplan et al., 2020, Zhai et al., 2021].

3 Training and Evaluation

While some tasks are well-defined and have an obvious evaluation metric, such as classification error for image classification, other tasks, typically those with a structured or generative aspect, may be ill-defined. For instance, it is not obvious how to assess the performance of a machine translation, text-to-speech, or image generation system. Statistical decision theory provides a partial answer, by formalizing these problems as the minimization of a certain evaluation metric, though the definition of such metric can be a challenging problem of its own [Bickel and Doksum, 2015].

To learn a neural network’s weights or parameters, also known as *training* the neural network, we need a **training loss** to minimize, and an **optimizer** to compute the parameter updates. The training loss is sometimes (but not always) a proxy or surrogate of the evaluation metric, but with favorable properties such as differentiability, ease of computation, or good sample complexity.

3.1 Evaluation metric

The evaluation metric $L(\theta)$ measures the quality of the output of a model with parameters θ . Depending on the task considered, choosing the evaluation metric can be trivial or may require engineering various heuristics. For image classification, a trivial evaluation metric is the (expected) classification error, which equals 0 if the predicted class is correct, and 1 otherwise:

$$L(\theta) = \mathbf{E}_{x,y \sim p}[\mathbb{1}_{y \neq h_\theta(x)}]$$

For language modeling, the evaluation metric is typically the log-likelihood or perplexity, which quantify how good the model is at completing the next word in a sentence.

$$L(\theta) = \mathbf{E}_{w_{t+1} \sim p(w_{t+1}|w_{1:t})}[-\log q_\theta(w_{t+1}|w_{1:t})]$$

For object detection, evaluation is less trivial because there is not obvious way to compare two sets of bounding boxes; a common metric is the mean average precision:

$$L(\theta) = \mathbf{E}_{x,y \sim p}[\mathbf{mAP}(y, h_\theta(x))]$$

but such metric has issues such as entangling localization and classification error (see Section 4.4 for a tutorial on mean average precision).

For tasks such as speech recognition, machine translation, or video captioning, evaluation is even more challenging, and may be as hard as defining a semantic distance between two sentences; people typically resort to heuristics such as BLEU [Papineni et al., 2002], an n-gram precision-based metric, ROUGE [Lin, 2004], a recall-based metric, METEOR [Denkowski and Lavie, 2014] or CIDEr [Vedantam et al., 2015], but each of these evaluation metrics has shortcomings and may not always agree with a human annotator [Kilickaya et al., 2016].

For tasks with image outputs such as style transfer, or video next-frame prediction, it is unclear which evaluation metric is the best, as the L2 distance (pixel-wise) is widely known to poorly correlate with human perception. The Fréchet Inception Distance (FID) is often used to evaluate GANs [Heusel et al., 2017] and basically measures the distributional match between generated and real samples

$$\text{FID} = \|\mu - \mu_w\|_2^2 + \text{tr} \left(\Sigma + \Sigma_w - 2 \left(\Sigma^{\frac{1}{2}} \cdot \Sigma_w \cdot \Sigma^{\frac{1}{2}} \right)^{\frac{1}{2}} \right)$$

in the feature space of a pretrained network, such as Inceptionv3 [Szegedy et al., 2015].

3.2 Training loss

The training loss $\mathcal{L}(\theta)$ is a function of the model parameters, which measures the quality of predictions on the training set, and is minimized during training. It is

computed over a training set $\{(x_i, y_i)\}_{1 \leq i \leq N}$, which is generally considered i.i.d. sampled from the true distribution p . In deep learning, we often use stochastic gradient descent for minimization, which requires the training loss to be differentiable almost everywhere. This makes it challenging to use the evaluation metric directly as a training loss, as most of them—such as classification error or mean average precision—are not differentiable. Instead, one may use a differentiable approximation of the evaluation metric, or simply use a different loss.

For instance, a common training loss for classification is the *negative log-likelihood*—a.k.a. *cross-entropy loss*—which can be computed from soft (probabilistic) predictions:

$$\mathcal{L}(\theta) = \mathbf{E}_{x, y \sim \hat{p}}[-\log q_\theta(y|x)] = \frac{1}{N} \sum_{i=1}^N -\log q_\theta(y_i|x_i)$$

the hard predictions are given by the most probable class $h_\theta(x) = \arg \max_y q_\theta(y|x)$. For a single example, a prediction which fully minimizes the cross-entropy loss (by assigning all probabilities to the ground truth label) will also minimize the classification error but the cross-entropy loss has the advantage of being differentiable; it can be thought of as a surrogate loss of the classification error [Osokin et al., 2017].

For natural language outputs, negative log-likelihood is also the training loss of choice due to favorable computational properties with autoregressive models (see Section 6.1), but the connection with evaluation metrics such as BLEU [Papineni et al., 2002] or ROUGE [Lin, 2004] is unclear as soon as predictions don't exactly match the ground-truth.

For image generation, the Fréchet Inception Distance (FID) introduced by Heusel et al. [2017] remains one of the standard evaluation metrics, but is rarely used as a training loss, with most image generation training losses being either adversarial [Goodfellow et al., 2014, Salimans et al., 2016] or likelihood-based [Oord et al., 2016, Ho et al., 2020, Ramesh et al., 2022]. There have been attempts to optimize the FID as an additional loss during training [Mathiasen and Hvilshøj, 2020], which resulted in lower validation FID; but there is no clear evidence whether direct FID optimization can systematically improve image quality.

3.3 Optimization

Training a neural network means finding the parameters θ that minimize the training loss $\mathcal{L}(\theta)$. Except in the case of linear networks, the training loss is usually non-convex, which means there is no guarantee to find the global minimum. This is not necessarily a problem in practice, though this means optimization may be harder to tune and fail more easily [Goodfellow et al., 2016].

Deep learning algorithms are based on stochastic gradient descent. Gradient descent is a first-order iterative optimization algorithm for finding the minimum of a function. The idea behind gradient descent is that if a function $\mathcal{L}(\theta)$ is defined and differentiable in a neighborhood of x then $\mathcal{L}(\theta)$ decreases the fastest when moving θ in the direction of $-\nabla\mathcal{L}(\theta)$, where $\nabla\mathcal{L}(\theta)$ is the gradient of \mathcal{L} at position θ . The update rule for gradient descent is thus defined as follows:

$$\theta_{t+1} = \theta_t - \eta_t \nabla L(\theta_t)$$

where η_t is the **learning rate** (or **step size**).

In practice, the training loss is often defined as an expectation over the training set $\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^N \ell(\theta, x_i)$. Instead of averaging the gradient over the whole training set, it is common to sample only a subset (minibatch) of samples for the update. This procedure is called stochastic gradient descent (SGD) and is proved to converge in the convex case.

Most modern optimizers such as Adam [Kingma and Ba, 2015] are based on SGD, with additional tricks such as *momentum* and *adaptive normalization*.

4 Generative Modeling

The goal of generative modeling is to generate new samples from a given distribution, which are ideally *high-quality* (high precision) and *diverse* (high recall). In this section, we review the VAE and GAN, two popular generative models.

4.1 Maximum Likelihood Estimation

A common divergence used in machine learning is the KL divergence. We can use it to measure the distance between our model p_θ and the empirical distribution of the data $p_{\mathcal{D}}$:

$$\text{KL}(p_{\mathcal{D}}||p_\theta) = \mathbb{E}_{p_{\mathcal{D}}}[\log p_{\mathcal{D}}(x)] - \mathbb{E}_{p_{\mathcal{D}}}[\log p_\theta(x)]$$

If we try to minimize this KL with respect to p_θ , the first term is constant with respect to θ because it is the entropy of the data distribution. We thus get the following objective, called **Maximum Likelihood Estimation** (MLE):

$$\max_{\theta} \mathbb{E}_{p_{\mathcal{D}}}[\log p_\theta(x)]$$

If the density of the model is known this can be approximated with a Monte-Carlo estimate $\mathbb{E}_{p_{\mathcal{D}}}[\log p_\theta(x)] \approx \sum_i \log p_\theta(x_i)$ where the x_i are samples from the dataset.

The KL divergence is minimized only if $p_\theta = p_{\mathcal{D}}$, but $p_{\mathcal{D}}$ can be arbitrarily complicated. We thus need p_θ to be flexible enough to model and capture the complicated dependencies in $p_{\mathcal{D}}$. We thus present two family of models which define flexible family of distributions over high-dimensional space.

4.2 Variational Autoencoders

Variational Autoencoders or VAEs [Kingma and Welling, 2014] consist of an encoder-decoder architecture which encodes data to a latent space, typically with smaller dimensionality, and can reconstruct the input from the latent representation. They are similar in architecture to Denoising Autoencoders [Vincent et al., 2008], but allow sampling from the learned distribution, and can be considered explicit generative models, which means they can numerically evaluate the (approximate) likelihood of a given sample. The VAE can be represented by the joint density $p_\theta(x, z) = p_\theta(x|z)p(z)$, where x is the observed data (e.g. images) and z is the unobserved representation, which is learned. To maximize the log likelihood of the data $p_\theta(x)$, one needs to marginalize with respect to z , $\log p_\theta(x) = \log \int p_\theta(x, z) dz$; unfortunately this integral is usually intractable. However it is possible to derive a lower-bound using Jensen’s inequality and introducing a distribution $q_\phi(z|x)$ called the approximate posterior:

$$\begin{aligned} \log p(x) &\leq \mathbb{E}_{q_\phi(z|x)} \left[\log \frac{p_\theta(x, z)}{q_\phi(z|x)} \right] \\ &= \mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x|z)] - \text{KL}(q_\phi(z|x) || p(z)) \end{aligned}$$

The distributions $p_\theta(x|z)$ and $q_\phi(z|x)$ are typically parametrized by neural networks, respectively the *decoder* and the *encoder*. The above lower-bound is tight when $q_\phi(z|x) = p_\theta(z|x)$, and may require the approximate posterior $q_\phi(z|x)$ to be flexible enough.

The original VAE approximate posterior is a Gaussian distribution with diagonal covariance, and may suffer from a problem known as *posterior collapse*. Modern VAEs, such as the VQ-VAE [Van Den Oord et al., 2017], use discrete latent variables, learnable autoregressive priors, and more powerful approximate posteriors. In our first contribution, we have used original VAEs for simplicity; however, this constitutes a limitation, and a better comparison should have used VAEs with more powerful priors/approximate posteriors.

4.3 GANs

Generative Adversarial Networks or GANs [Goodfellow et al., 2014] have achieved impressive results in image generation tasks, including unconditional image generation [Karras et al., 2018a,b, 2019], conditional image generation [Wang et al., 2018a],

image-to-image translation [Zhu et al., 2017], text-to-image generation [Reed et al., 2016], and human pose manipulation [Ma et al., 2017].

GANs can be conceptualized as an adversarial game between two networks: a *discriminator* network learns to distinguish true samples from fake (generated) samples, while a *generator* tries to fool the discriminator. If learning succeeds, the generator should eventually fool the discriminator enough and generate realistic images. Formally, denote G the generator network and p_G its distribution, from which we can sample but not necessarily compute the density (implicit generator). Denote $D(x)$ the discriminator, a binary image classifier. The GAN objective is the following:

$$\min_{p_G} \max_D \mathbb{E}_{p_D} [\log D(x)] + \mathbb{E}_{p_G} [\log(1 - D(x))]$$

For any given distributions, the optimal discriminator over the class of all functions $\mathcal{X} \rightarrow (0, 1)$ is $D^*(x) = \frac{p_D(x)}{p_D(x) + p_G(x)}$, the objective proposed is then equivalent to minimizing the Jensen-Shannon Divergence between the two distributions:

$$\min_{p_G} \mathbb{E}_{p_D} [\log D^*(x)] + \mathbb{E}_{p_G} [\log(1 - D^*(x))] = \min_G JSD(p_G || p_D)$$

Because this objective does not require having access to the density of the generator but only to sample from it (implicit model), one can use the distribution of $x = g_\theta(z)$, implicitly defined by a latent z sampled from a fixed distribution, and a learnable transformation g_θ .

In practice the second term can lead to vanishing gradient for the generator parameters, an alternative was thus proposed where the generator needs to maximize $\mathbb{E}_{p_G} \log D(x)$ instead. Arjovsky et al. [2017] show that GANs suffer from this vanishing gradient because the Jensen-Shannon Divergence is not well defined for distributions which are not absolutely continuous; this is the case if the data lies on a low dimensional manifold.

Arjovsky et al. [2017] introduced the Wasserstein GAN, which uses a Wasserstein distance instead of the Jensen-Shannon divergence, which is not defined for non-absolutely-continuous distributions:

$$\min_{\theta} \max_{\|f_\phi\|_L \leq 1} \mathbb{E}_{p_D} [f(x)] - \mathbb{E}_{p_Z} [f_\phi(g_\theta(z))]$$

They use a discriminator with clipped weights, to satisfy the Lipschitz-continuity constraint of the variational formulation of the Wasserstein distance. An alternative to weight clipping was proposed in [Gulrajani et al., 2017], where by noticing that the Lipschitz constraint is equivalent to $\|\nabla_x D_\phi(x)\| \leq 1$, they propose to regularize the discriminator with the following gradient penalty:

$$\mathbb{E}_{p(\hat{x})} [(\|\nabla_{\hat{x}} D_\phi(\hat{x})\|_2 - 1)^2]$$

where $p(\hat{x})$ correspond to sampling a point uniformly on the line between a generated sample and a sample from the data distribution. We use this last variant in our experiments, except we directly penalize the norm of the gradient.

3

Prologue to First Contribution

1 Article Details

“**Parametric Adversarial Divergences are Good Losses for Generative Modeling**” by *Gabriel Huang, Hugo Berard, Ahmed Touati, Gauthier Gidel, Pascal Vincent, Simon Lacoste-Julien*. This paper was initially accepted as a ICLR 2018 Workshop paper, then presented at the Montreal AI Symposium 2018; a conference version has been submitted to ICML 2018; finally, a journal version was submitted to JMLR [Huang et al., 2017].

2 Contributions of the authors

Simon Lacoste-Julien has proposed the original connection between generative modeling and structured prediction using statistical decision theory’s task losses. Pascal Vincent and Simon Lacoste-Julien have supervised and provided funding for the project. Hugo Berard, Ahmed Touati and Gauthier Gidel have helped design and run the experiments. Gabriel Huang has led the project, written most of the paper, proposed and performed the majority of experiments, and derived the theoretical analyses related to mutual information.

3 Limitations and scope of the paper

This project was motivated by the intuition that *parametric divergences* (of which GAN losses are a special case) might have interesting advantages for high-dimensional and structured data, and we wanted to characterize their properties better in the specific setting of unconditional data generation. (1) Note that we have adopted a rather restrictive definition of “generative modeling” and limited ourselves to *unconditional* generation of images, excluding other applications such as clustering, topic modeling, compression, or conditional data generation. However, it should be noted that the points discussed about enforcing arbitrary constraints on the generated data are equally valid for conditional generation, which may have more real-world applications. (2) To have a point of comparison, we looked at

maximum-likelihood-based models, specifically Variational Autoencoders [Kingma and Welling, 2014], which can be interpreted as maximizing the likelihood of the data using variational inference. This restricted setting allowed us to factor out architectural differences between GAN and VAE—by using near-identical feedforward encoder-decoders—a parallel that wouldn’t have been possible with PixelCNN [Oord et al., 2016] or diffusion models [Ho et al., 2020]. (3) We argued that parametric mutual information can lead to more intuitive results in the small data regime than regular mutual information; however, we cannot dismiss the fact that mutual information enjoys a solid theoretical foundation—e.g. rate-distortion theory [Blau and Michaeli, 2019]—which parametric mutual information still lacks. Because of those limitations, and as demonstrated by recent developments, we cannot conclude on the superiority of one divergence or style of mutual information over another, but rather hope to gain better theoretical and practical understanding of their properties, under specific settings.

4 Recent Developments

Since we started this project, GAN models have been scaled up by works such as BigGAN [Brock et al., 2019], StyleGAN [Karras et al., 2019], and StyleGANv2 [Karras et al., 2020], reaching resolutions of 1024×1024 and generating crisp and spatially coherent images.

To overcome limitations of the original VAE [Kingma and Welling, 2014], VQ-VAE [Van Den Oord et al., 2017] introduced: (1) a learnable autoregressive prior with more flexibility than the fixed diagonal Gaussian prior of the original VAE [Kingma and Welling, 2014] (2) a vector quantization step to strictly enforce the bottleneck property and prevent the *posterior collapse* issue—when latents are ignored by a powerful decoder. Although the samples are still somewhat blurry, VQ-GAN [Esser et al., 2021] proposed to combine VQ-VAE with two additional losses: a perceptual loss and a parametric divergence loss, resulting in crisp samples.

Diffusion models [Sohl-Dickstein et al., 2015], popularized by Ho et al. [2020] and improved in subsequent works [Nichol and Dhariwal, 2021, Dhariwal and Nichol, 2021] are another type of likelihood-based model. They can be interpreted as training a denoising network to reverse a Gaussian diffusion process. They have achieved state-of-the-art image generation quality, and better diversity than GANs [Nichol and Dhariwal, 2021, Razavi et al., 2019], at the expense of a slower and iterative generating process. In particular, Ramesh et al. [2022] combine diffusion-based prior, decoder, and upsampler networks with a CLIP encoder [Radford et al., 2021] to produce the DALL-E 2 model, resulting in 1024×1024 images of unprecedented quality and diversity.



Figure 3.1: DALL-E 2 fails to generate two distinct characters without blending their attributes together. Prompt: “Captain America and Iron Man standing side by side”. Source: <https://www.lesswrong.com/posts/uKp6tBFStnsvrot5t/what-dall-e-2-can-and-cannot-do>

Interestingly, it has been reported¹ that DALL-E 2 cannot generate coherent English text, despite generating something more English-looking than random letters (Figure 3.2, 3.3, 3.4). Another weakness seems to be mixing various attributes when asked to generate two objects or characters in the same scene (Figure 3.1).

It is possible that scaling up to even more data, or somehow integrating a language model into DALL-E 2 could solve these problems. For instance, the Parti text-to-image model [Yu et al., 2022] improves its text rendering skills as it scales up. However, it is also possible that adding a parametric divergence loss could help overcome the model’s weaknesses more directly and efficiently, an approach already undertaken to enforce temporal consistency [Chu et al., 2020], generate graph-constrained house layouts [Nauata et al., 2020], and enforce semantic segmentation consistency for applications like building facade generation [Arantes et al., 2020].

Overall, these recent developments show that: (1) Maximum-likelihood approaches can generate diverse and photorealistic samples *when combined* with the right generator (such as a diffusion model²), large-scale datasets, and powerful hardware. This further motivates research on data-efficient methods. (2) GANs remain competitive, especially in terms of sampling speed, while combining existing methods with parametric divergences can greatly improve sample quality. (3) Generating data satisfying all the properties we care about (e.g. grammatical text, attribute separation, object placement, temporal coherence) is still an open problem.

¹<https://www.lesswrong.com/posts/uKp6tBFStnsvrot5t/what-dall-e-2-can-and-cannot-do>

²This echoes our discussion on generators with special structure (Section 6.1).

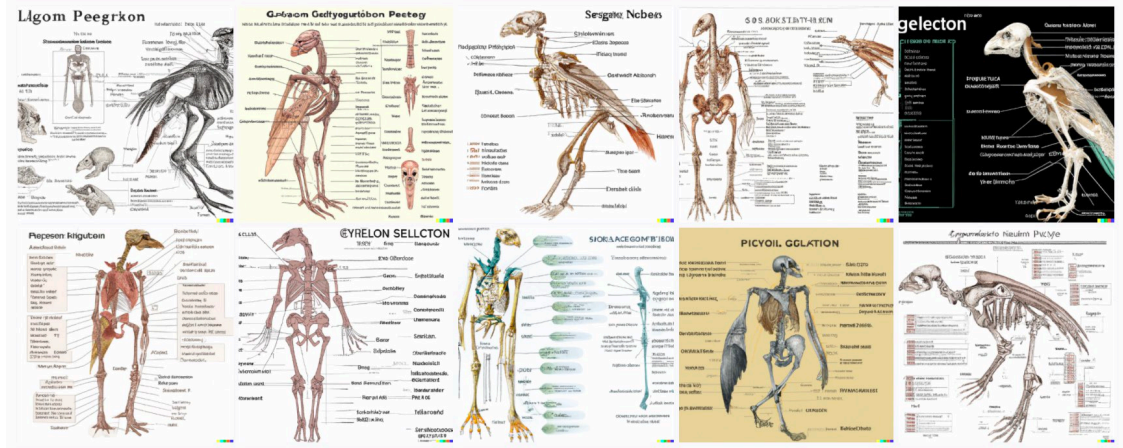


Figure 3.2: DALL-E 2 fails to generate English text. Prompt: “Medical illustration of a gryphon’s skeleton, with labels. High quality, detailed, professional medical illustration.”. Source: https://www.reddit.com/r/dalle2/comments/uh03zs/medical_illustration_of_a_gryphons_skeleton_with/

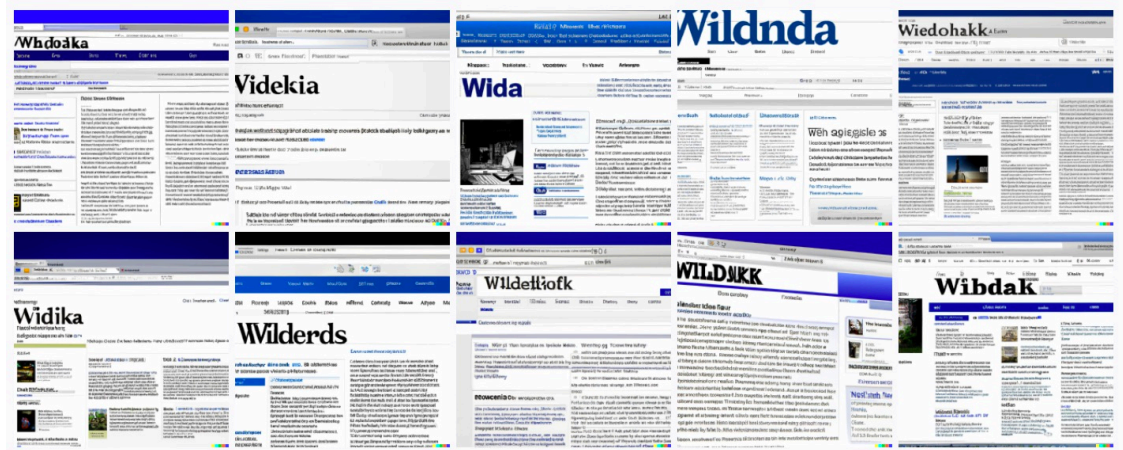


Figure 3.3: DALL-E 2 fails to generate English text. Prompt: “A screenshot of the Wikipedia home page”. Source: <https://www.lesswrong.com/posts/uKp6tBFstnsvrot5t/what-dall-e-2-can-and-cannot-do>



Figure 3.4: DALL-E 2 fails to generate English text. Prompt: “Infinite Jest”.
Source: <https://www.lesswrong.com/posts/uKp6tBFStnsvrot5t/what-dall-e-2-can-and-cannot-do>

4

Exploring Properties of GAN Losses

Abstract

*Parametric adversarial divergences*¹, which are a generalization of the losses used to train generative adversarial networks (GANs), have often been described as being approximations of their nonparametric counterparts, such as the Jensen-Shannon divergence, which can be derived under the so-called *optimal discriminator* assumption. In this position paper, we argue that despite being “non-optimal”, parametric divergences have distinct properties from their nonparametric counterparts which can make them more suitable for learning high-dimensional distributions. A key property is that parametric divergences are only sensitive to certain aspects/moments of the distribution, which depend on the architecture of the discriminator and the loss it was trained with. In contrast, nonparametric divergences such as the Kullback-Leibler divergence are sensitive to moments ignored by the discriminator, but they do not necessarily correlate with sample quality [Theis et al., 2016]. Similarly, we show that mutual information can lead to unintuitive interpretations, and explore more intuitive alternatives based on parametric divergences. We conclude that parametric divergences are a flexible framework for defining statistical quantities relevant to a specific modeling task.

1 Introduction

In traditional statistics, generative modeling is formulated as density estimation. The learning objective and evaluation metric are usually the expected negative-log-likelihood. While maximizing the log-likelihood, or equivalently, minimizing the KL-divergence, works fine for modeling low-dimensional data, there are a number of issues that arise when modeling high-dimensional data, such as images. Maybe the most important issue is the lack of guarantees that log-likelihood is a good proxy for sample quality. This is obviously a problem, because if the goal is to generate realistic data, then the evaluation objective *should* align with sample quality. For instance, Theis et al. [2016] exhibit generative models with high log-likelihood which produce low-quality images, and models with poor log-likelihood which produce

¹We give a formal definition in this paper.

high-quality images. In some cases, they show that the log-likelihood can even be *hacked* to be arbitrarily high, even on test data, without improving the sample quality at all. Another practical issue with f-divergences, a generalization of the KL, is they are either not defined or uninformative whenever the distributions are too far apart, even when tricks such as smoothing are used [Arjovsky et al., 2017].

To address these shortcomings, we look past maximum-likelihood and classic divergences to define better objectives. But first, we need to take a step back and explicit our *final task*, or end goal. Assuming that our end goal is to generate *realistic* and *diverse* samples, how can we formalize such a subjective and ill-defined final task into a *task loss*, a rigorous mathematical objective which can be evaluated and used to derive training (e.g. surrogate) losses? When it comes to defining relevant task losses, it is worthwhile to consider how people choose task losses in structured prediction, for which the label space is often combinatorially large (e.g. sequences of words). For instance, machine translation systems are commonly evaluated using the BLEU-4 metric [Papineni et al., 2002], which essentially counts how many words the predicted and ground truth sentences have in common. Although the BLEU score is only an imperfect approximation of the final task, it is nevertheless more informative than “hard” losses such as the 0 – 1 loss, which give no training signal unless the predicted label matches exactly the ground truth.

Unfortunately, in generative modeling, it is not as obvious how to define a task loss that correlates well with diversity and sample quality. Nevertheless, we argue that the *adversarial framework*, introduced in the context of generative adversarial networks or GANs [Goodfellow et al., 2014], provides an interesting way to define meaningful and practical task losses for generative modeling. For that purpose, we adopt the view² that training a GAN can be seen as training an implicit generator to minimize a special type of task loss, which we call **parametric (adversarial) divergence**:

$$\text{Div}(p||q_\theta) \hat{=} \sup_{\phi \in \Phi} \mathbf{E}_{(\mathbf{x}, \mathbf{x}') \sim p \otimes q_\theta} [\Delta(f_\phi(\mathbf{x}), f_\phi(\mathbf{x}'))] \quad (1.1)$$

where p is the distribution to learn and q_θ is the distribution defined by the implicit generator. The expectation is maximized over a parametrized class of functions $\{f_\phi : \mathcal{X} \rightarrow \mathbb{R}^{d'} ; \phi \in \Phi\}$ which are usually neural networks with a fixed architecture. Those functions are called **discriminators** in the GAN framework [Goodfellow et al., 2014]. The constraints Φ and the formulation $\Delta : \mathbb{R}^{d'} \times \mathbb{R}^{d'} \rightarrow \mathbb{R}$ determine properties of the resulting divergence (see Section 2.2 for concrete examples).

The main contribution of this paper is to show that parametric adversarial diver-

²We focus in this paper on the divergence minimization perspective of GANs. There are other views, such as those based on game theory [Arora et al., 2017, Fedus* et al., 2018], ratio matching and moment matching [Mohamed and Lakshminarayanan, 2016], but these are outside the scope of this paper.

gences can have very different properties from their *nonparametric* counterparts (where the function class is infinite dimensional). Instead of viewing them as imperfect estimators, we argue that parametric divergences are actually better approximators of “generating realistic samples” than likelihood-based objectives. To this end, we start by expliciting the difference between final task and task loss (Section 3). Then, we show that unlike many nonparametric divergences, parametric divergences offer favorable sample complexity while retaining the flexibility to adapt to the final task (Section 4.1). In particular, we show on a toy problem how to tune a parametric divergence in order to enforce properties of interest (Section 5). In practice, combining divergences with specific generators can lead to side-effects, which we discuss in Section 6. Finally, we investigate how to use parametric divergences to define more intuitive notions of mutual information (Section 7).

2 Background

We briefly introduce the structured prediction framework because we will make links between the losses in structured prediction and generative modeling in Section 3.3. We introduce the variational formulation which allows us to consider and compare parametric adversarial divergences and traditional divergences as special cases of adversarial divergences.

2.1 Structured Prediction

The goal of structured prediction is to learn a function $h_\theta : \mathcal{X} \rightarrow \mathcal{Y}$ which predicts a structured output \mathbf{y} from an input \mathbf{x} . Examples of structured outputs include parse-trees, sequences of symbols, alignments between sequences, 3D shapes, and segmentation maps. The key difficulty is that \mathcal{Y} usually has size exponential in the dimension of the input (e.g. for sequence-to-sequence prediction, \mathcal{Y} could be all the sequences of symbols with a given length). Being able to handle this exponentially large set of possible outputs is one of the key challenges in structured prediction. Traditional multi-class classification methods are unsuitable for these problems in general. Standard practice in structured prediction [Taskar et al., 2003, Collins, 2002, Pires et al., 2013] is to consider predictors based on score functions $h_\theta(\mathbf{x}) \triangleq \operatorname{argmax}_{\mathbf{y}' \in \mathcal{Y}} s_\theta(\mathbf{x}, \mathbf{y}')$, where $s_\theta : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$, called the **score/energy function** [LeCun et al., 2006], assigns a score to each possible label \mathbf{y} for an input \mathbf{x} . Typically, as in structured SVMs [Taskar et al., 2003], the score function is linear: $s_\theta(\mathbf{x}, \mathbf{y}) = \langle \theta, g(\mathbf{x}, \mathbf{y}) \rangle$, where $g(\cdot)$ is a predefined feature map. Alternatively, the score function could also be a learned neural network [Belanger and McCallum, 2016].

In order to evaluate the predictions objectively, we need to define a **task-specific** structured loss $\ell(\mathbf{y}', \mathbf{y}; \mathbf{x})$ which expresses the cost of predicting \mathbf{y}' for \mathbf{x} when the ground truth is \mathbf{y} . We discuss the relation between the loss function and the actual final task when we review statistical decision theory in Section 3.1 and 3.2. The goal is then to find a parameter θ which minimizes the generalization error

$$\min_{\theta \in \Theta} \mathbf{E}_{(\mathbf{x}, \mathbf{y}) \sim p} [\ell(h_{\theta}(\mathbf{x}), \mathbf{y}, \mathbf{x})] \quad (2.1)$$

or, in practice, an empirical estimation of it based on an average over a finite sample from p . Directly minimizing this is often intractable, even in simple cases, e.g. when the structured loss ℓ is the 0-1 loss [Arora et al., 1993]. Instead, the usual practice is to minimize a surrogate loss $\mathbf{E}_{(\mathbf{x}, \mathbf{y}) \sim p} [\mathcal{L}(s_{\theta}(\mathbf{x}, \mathbf{y}), \mathbf{y}, \mathbf{x})]$ [Bartlett et al., 2006] which has nicer properties, such as sub-differentiability or convexity, to get a tractable optimization problem. The surrogate loss is said to be consistent [Osokin et al., 2017] when its minimizer is also a minimizer of the task loss.

2.2 Parametric and Nonparametric Adversarial Divergences

The focus of this paper is to analyze whether parametric adversarial divergences are good candidates for generative modeling. In particular, we analyze them relatively to nonparametric divergences. Therefore, we first unify them with a formalism similar to Sriperumbudur et al. [2012], Liu et al. [2017]. We define **adversarial divergences** using the variational formulation:

Definition 2.1 (Adversarial Divergence). We denote adversarial divergences functions which can be written with the following form:

$$\text{Div}(p||q_{\theta}) \triangleq \sup_{f \in \mathcal{F}} \mathbf{E}_{(\mathbf{x}, \mathbf{x}') \sim p \otimes q_{\theta}} [\Delta(f(\mathbf{x}), f(\mathbf{x}'))] \quad (2.2)$$

where we refer to $f : \mathcal{X} \rightarrow \mathbb{R}^{d'} \in \mathcal{F}$ as the discriminator, and $\Delta : \mathbb{R}^{d'} \times \mathbb{R}^{d'} \rightarrow \mathbb{R}$ determines properties of the resulting divergence.

Definition 2.2 (Parametric Divergence). In particular, when the discriminator space \mathcal{F} is *parametric*, such as the set of neural networks with a given architecture, the adversarial divergence is called a parametric (adversarial) divergence.³

For appropriate choices of discriminator class \mathcal{F} and function Δ , we can recover many usual divergences, including f-divergences (such as Kullback-Leibler) and integral probability metrics (such as Wasserstein distances and Maximum Mean discrepancy). For instance,

³Usually, \mathcal{F} is a class of neural networks with fixed architecture. In that case, $\text{Div}(p||q_{\theta})$ has been called a **neural divergence** in Arora et al. [2017]. We will use the slightly more generic **parametric divergence** in our work.

- ψ -divergences with generator function ψ (which we call f-divergences) can be written in dual form [Nowozin et al., 2016]⁴

$$\text{Div}_\psi(p||q_\theta) \hat{=} \sup_{f:\mathcal{X}\rightarrow\mathbb{R}} \mathbf{E}_{\mathbf{x}\sim p}[f(\mathbf{x})] - \mathbf{E}_{\mathbf{x}'\sim q_\theta}[\psi^*(f(\mathbf{x}'))] \quad (2.3)$$

where ψ^* is the convex conjugate. Depending on ψ , one can obtain any ψ -divergence such as the (reverse) Kullback-Leibler, the Jensen-Shannon, the Total Variation, the Chi-Squared.⁵

- Wasserstein-1 distance induced by an arbitrary norm $\|\cdot\|$ and its corresponding dual norm $\|\cdot\|^*$ [Sriperumbudur et al., 2012]:

$$W(p||q_\theta) \hat{=} \sup_{\substack{f:\mathcal{X}\rightarrow\mathbb{R} \\ \forall \mathbf{x}\in\mathcal{X}, \\ \|f'(\mathbf{x})\|^*\leq 1}} \mathbf{E}_{\mathbf{x}\sim p}[f(\mathbf{x})] - \mathbf{E}_{\mathbf{x}'\sim q_\theta}[f(\mathbf{x}')] \quad (2.4)$$

which can be interpreted as the cost to transport all probability mass of p into q , where $\|\mathbf{x} - \mathbf{x}'\|$ is the unit cost of transporting \mathbf{x} to \mathbf{x}' .

- Maximum Mean Discrepancy [Gretton et al., 2012]:

$$\text{MMD}(p||q_\theta) \hat{=} \sup_{\substack{f\in\mathcal{H} \\ \|f\|_{\mathcal{H}}\leq 1}} \mathbf{E}_{\mathbf{x}\sim p}[f(\mathbf{x})] - \mathbf{E}_{\mathbf{x}'\sim q_\theta}[f(\mathbf{x}')] \quad (2.5)$$

where (\mathcal{H}, K) is a Reproducing Kernel Hilbert Space induced by a Kernel $K(\mathbf{x}, \mathbf{x}')$ on \mathcal{X} with the associated norm $\|\cdot\|_{\mathcal{H}}$. The MMD has many interpretations in terms of moment-matching [Li et al., 2017].

Most nonparametric divergences can be made parametric by replacing \mathcal{F} with neural networks: examples are the **parametric Jensen-Shannon**, which is the standard mini-max GAN objective [Goodfellow et al., 2014] and the **parametric Wasserstein** which is the WGAN objective Arjovsky et al. [2017] in essence, modulo some technical tricks.⁶ See Liu et al. [2017] for interpretations and a review and interpretation of other divergences like the Wasserstein with entropic smoothing [Aude et al., 2016], energy-based distances [Li et al., 2017] which can be seen as adversarial MMD, and the WGAN-GP [Gulrajani et al., 2017] objective.

We deliberately chose a somewhat ambiguous terminology – nonparametric v.s. parametric – not to imply a clear-cut distinction between the two (as e.g. neural

⁴The standard form is $\mathbf{E}_{\mathbf{x}\sim q_\theta}[\psi(\frac{p(\mathbf{x})}{q_\theta(\mathbf{x})})]$.

⁵For instance the Kullback-Leibler $\mathbf{E}_{\mathbf{x}\sim p}[\log \frac{p(\mathbf{x})}{q_\theta(\mathbf{x})}]$ has the dual form $\sup_{f:\mathcal{X}\rightarrow\mathbb{R}} \mathbf{E}_{\mathbf{x}\sim p}[f(\mathbf{x})] - \mathbf{E}_{\mathbf{x}'\sim q_\theta}[\exp(f(\mathbf{x}')) - 1]$. Some ψ require additional constraints, such as $\|f\|_\infty \leq 1$ for the Total Variation.

⁶There are subtleties in the way the Lipschitz constraint is enforced. More details in Petzka et al. [2018].

networks can be made to become universal function approximators as we increase their size), but to imply a continuum from least restricted to more restricted function families where the latter are typically expressed through an explicit parametrization.

Under the formalism (2.2), one could argue that parametric divergences are simply estimators –in fact lower-bounds– of their nonparametric counterparts. Our opinion is that parametric divergences are not merely convenient estimators, but can actually be *much better* objectives for generative modeling than nonparametric divergences. We will give practical arguments and experiments to support this statement in the rest of this paper.

3 Divergences as Task Losses

While tasks such as binary and multiclass classification are straightforward to evaluate using classification accuracy (also known as the 0-1 loss), tasks with more structured outputs often require more complex evaluation. For instance, machine translation is often evaluated using the BLEU metric [Papineni et al., 2002], text summarization using the ROUGE-L metric [Lin, 2004], object detection using mean average precision [Ren et al., 2015], semantic segmentation the mean intersection-over-union [Chen et al., 2017], while several metrics such as Multiple Object Tracking Accuracy (MOTA) and Multiple Object Tracking Precision (MOTP) are commonly used for evaluating video tracking [Ciaparrone et al., 2020]. However, optimizing these metrics is not an end in itself, rather the hope is that these metrics are good proxies for the final task (producing an accurate translation, a concise summary, relevant bounding boxes, precise video tracking).

Statistical decision theory is a general framework for modeling the task of acting (in our case, learning a model) under uncertainty (which can come from sampling noise). In particular, ill-defined tasks can be formalized as the minimization of an evaluation metric, called the task loss.

In this work, we propose to consider *parametric divergences* as the task losses for approximating the ill-defined task of *generating realistic samples*. We start by introducing statistical decision theory and discuss desirable properties for task losses (Section 3.1). Then, we unify structured prediction and generative modeling under the statistical decision theory framework (Section 3.2). We point to results that hard task losses such as the 0-1 loss can make learning exponentially slower than using softer losses such as the Hamming loss (Section 3.3). Those results suggest that similarly for generative modeling, it might be beneficial to use “softer” parametric divergences instead of “harder” nonparametric divergences such as the Kullback-Leibler divergence.

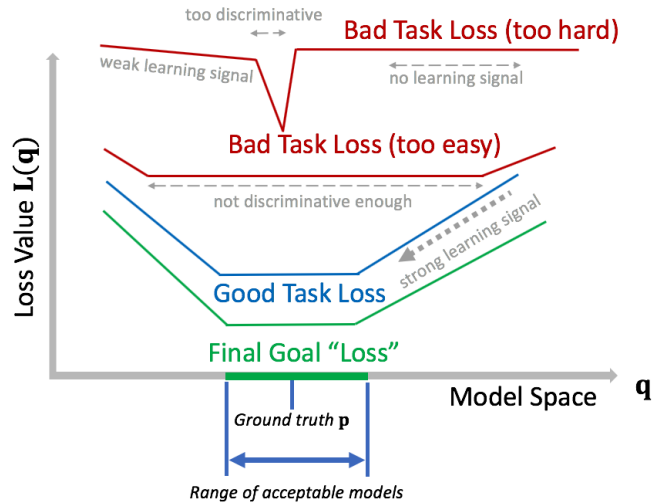


Figure 4.1: Our goal is to solve the final task, represented by the green “loss”. The first task loss (upper red) is not a good loss because it is **too discriminative** and rejects many acceptable models. The second task loss (lower red) is not good either because it is **not discriminative** enough and accepts incorrect models. The third task loss (blue) is a good task loss because it “tracks” the final goal “loss” well.

3.1 Formalizing Final Tasks with Statistical Decision Theory

Statistical decision theory is the standard framework presented in statistic textbooks for modeling and evaluating the task of acting under uncertainty [Bickel and Doksum, 2015]. In our case, *acting* means learning a model such as a classifier or a generative model from data, and the *uncertainty* comes from the fact that we can only access finite samples from the true distribution. Statistical decision theory allows us to formalize ill-defined tasks as the minimization of a clearly-defined evaluation metric, which is called the (statistical) *task loss*. The task loss needs to be mathematically well-defined, and cannot, for instance, require human evaluation in the loop, which would make the metric subjective and ill-defined.

Notation. We denote $p \in \mathcal{P}$ the unknown state of the process (distribution) we want to model, $a \in \mathcal{A}$ the action, and $L_p(a)$ the task loss. In machine learning, the unknown state is typically a probability distribution $p(\mathbf{x})$ or $p(\mathbf{x}, \mathbf{y})$ which we can only access indirectly through a finite training set D_{train} sampled i.i.d. from p , and the learner “plays an action” such as choosing the classifier or generative model which minimizes the negative log-likelihood of the training set.

Designing task losses. What does the ideal task loss look like? Consider the example in Figure 4.1. Our goal is to solve the final task, which is illustrated by the

green final goal “loss”. Such a “loss” might not be expressible as a mathematical loss, for instance it could be a score based on human evaluation or perception. Therefore, we need to approximate the final goal with a task loss $L_p(q_\theta)$, a proper mathematical objective from which we can derive an optimization problem. The task loss is formed with respect to a ground truth p which typically corresponds to the target distribution for generative modeling, and to the ground-truth labels for structured prediction. The first task loss (topmost in red) is not a good loss because it is too “hard” (discriminative). It only accepts the ground-truth p as a good model even though there exists a range of models with equivalent final goal loss. As a consequence, these losses tend to provide little to no learning signal since most models are ruled out as equally bad and the task loss “saturates” without providing adequate search direction. The second task loss (second in red) is not good either because it is not discriminative enough. It assigns low values to models even if they have bad final “loss”. As a result, minimizing that loss is not sufficient to guarantee a good solution. The third task loss (blue) is a good task loss because it “tracks” the final goal “loss” well. In particular, it provides a *strong learning signal* towards the range of acceptable models and is just *discriminative* enough with respect to the final goal.

3.2 Parallels between Predictive and Generative Tasks.

Below, we contrast approaches in predictive (classification and structured prediction) and generative tasks (sampling and likelihood evaluation) with respect to their use of *harder* and *softer* task losses.

Predictive Tasks. We contrast multi-class classification with structured prediction. Assume an unknown distribution $p(\mathbf{x}, \mathbf{y})$. The goal is to learn a function $h_\theta : \mathcal{X} \rightarrow \mathcal{Y}$ which outputs a “correct” prediction \mathbf{y} given an input \mathbf{x} . In multi-class classification, \mathcal{Y} is a finite label space, and all mistakes are penalized equally according to the classification error, also known as the 0-1 loss:

$$L_p(\theta) = \mathbf{E}_{\mathbf{x}, \mathbf{y} \sim p(\mathbf{x}, \mathbf{y})} \left[1_{\{h_\theta(x_i) \neq y_i\}} \right] \quad (3.1)$$

The 0-1 loss is a *hard* loss because the only correct prediction is the ground truth. In contrast, in structured prediction, \mathcal{Y} is a structured space such as sentences, segmentation maps or bounding boxes, and can be exponentially large or even infinite. Mistakes are penalized according to a *structured loss* $\ell(y', y; x)$.⁷

$$L_p(\theta) = \mathbf{E}_{(\mathbf{x}, \mathbf{y}) \sim p} [\ell(h_\theta(\mathbf{x}), \mathbf{y}, \mathbf{x})] \quad (3.2)$$

Although the 0-1 loss $\ell(y', y; x) = 1_{\{h_\theta(x_i) \neq y_i\}}$ could also be used, it is not informative of the type of mistake made, and can be detrimental to learning (Section 3.3).

⁷Depending on the context, both $L_p(a)$ and ℓ are called task losses, as they implicitly define the task.

Instead, people often resort to *softer* losses, such as the Hamming loss, which penalize mistakes more gradually, and result in better learning guarantees (Section 3.3).

Generative Tasks. We contrast traditional likelihood-based generative modeling with GAN-style generative modeling based on parametric divergences. Assume an unknown target distribution $p(\mathbf{x})$. We focus on the (ill-defined) final task of learning a model $q_\theta(\mathbf{x})$ which can generate “realistic” samples from “the same” distribution as p . Traditional generative modeling formalize this task by penalizing models according to the negative log-likelihood:

$$L_p(\theta) = \mathbf{E}_{\mathbf{x} \sim p} [-\log(q_\theta(\mathbf{x}))] \quad (3.3)$$

Although log-likelihood has been the *de facto* learning objective in the past, there is no guarantee that log-likelihood is a good proxy for sample quality. In fact, Theis et al. [2016] have exhibited image models which have high likelihood but produce low quality samples, as well as models which have low likelihood but produce high quality samples. Maximizing the log-likelihood is equivalent to minimizing the Kullback-Leibler divergence

$$\mathbf{KL}(p||q_\theta) = \mathbf{E}_{\mathbf{x} \sim p} \left[\log\left(\frac{p(\mathbf{x})}{q_\theta(\mathbf{x})}\right) \right] \quad (3.4)$$

which can be considered a *hard* loss, in the sense that mistakes are penalized regardless of the metric structure of \mathcal{X} . This is particularly obvious in the special case where $p(x) = \delta(x - x_p)$ and $q(x) = \delta(x - x_q)$ are two Dirac distributions. Then, $KL(p||q_\theta)$ equals 0 if $x_p = x_q$ and infinity otherwise, which means all mistakes are penalized equally regardless of the distance between x_p and x_q . There are ways to make the maximum likelihood “softer”, but they come with their own caveats (Section 6.1).

GAN-style models adopt a very different approach and penalize models according to a parametric divergence :

$$L_p(\theta) = \text{Div}(p||q_\theta) = \sup_{f \in \mathcal{F}} \mathbf{E}_{(\mathbf{x}, \mathbf{x}') \sim p \otimes q_\theta} [\Delta(f(\mathbf{x}), f(\mathbf{x}'))] \quad (3.5)$$

For instance, the saturating GAN can be formulated as the minimization of a parametric Jensen-Shannon divergence:

$$L_p(\theta) = \text{Div}_{\text{ParamJS}}(p||q_\theta) = \sup_{f \in \text{NeuralNet}} \mathbf{E}_{\mathbf{x} \sim p} [\log f(\mathbf{x})] + \mathbf{E}_{\mathbf{x} \sim q} [\log(1 - f(\mathbf{x}))]. \quad (3.6)$$

In general, the parametric divergence only penalizes moments which are captured by the discriminator class. This means that we can arbitrarily tune the “hardness”

of the parametric divergence by tuning \mathcal{F} and Δ . As an extreme example, if \mathcal{F} is the set of linear 1-Lipschitz functions and we consider the Wasserstein-1 formulation $\Delta(f(x), f(x')) = f(x) - f(x')$, then the resulting parametric divergence reduces to the distance between the distribution means $\|\mathbf{E}_{x \sim p}[x] - \mathbf{E}_{x \sim q}[x]\|$. This is substantially *softer* than the corresponding (nonparametric) Wasserstein obtained by removing the linear constraint. In general, parametric divergences are promising task losses for generative modeling as long as we can find ways to tailor the discriminator to approximate the notion of realistic samples well.

3.3 An Analogy with Structured Prediction

We derive an intuitive analogy between “hard” losses such as the 0-1 loss and the KL-divergence and “softer” losses like the Hamming loss and the Wasserstein distance. Then, we draw insights from the convergence results of Osokin et al. [2017] in structured prediction, which parallel the intuition in generative modeling that learning with weaker⁸ divergences is easier [Arjovsky et al., 2017] and more intuitive [Liu et al., 2017] than with stronger divergences.

Analogy between Structured Prediction Losses and Divergences. A loose analogy can be made between “hard” losses like the 0-1 loss and the KL-divergence. A similar analogy can be made between “softer” losses like the Hamming loss and the Wasserstein distance. Consider two Dirac distributions $p(y) = \delta(y - y_p)$ and $q(y) = \delta(y - y_q)$ defined over \mathbb{R}^D . We compute the Jensen-Shannon divergence, which can be thought of as a symmetrized KL:

$$\mathbf{JS}(p||q) = \frac{1}{2}\mathbf{KL}(p||\frac{p+q}{2}) + \frac{1}{2}\mathbf{KL}(q||\frac{p+q}{2}) = 1_{\{y_p \neq y_q\}} \cdot \log 2.$$

In this case, the Jensen-Shannon divergence reduces to a scaled 0-1 loss between the atoms. We now consider the Wasserstein distance $\mathbf{W}(p, q)$ with base distance L_2 , which yields here:

$$\mathbf{W}(p, q) = \|y_p - y_q\|_2.$$

The Wasserstein distance reduces to the L_2 distance between the atoms, or equivalently the Hamming distance, if we consider only binary vectors. In that sense, the KL-divergence could be considered a “hard” divergence, while the Wasserstein distance could be considered a “softer” divergence.

“Softer” losses are better in structured prediction. Consider a “hard” structured loss, the 0-1 loss, defined as $\ell_{0-1}(\mathbf{y}, \mathbf{y}') \triangleq \mathbf{1}\{\mathbf{y} \neq \mathbf{y}'\}$, and a “softer” loss, the Hamming loss, defined as $\ell_{\text{Ham}}(\mathbf{y}, \mathbf{y}') \triangleq \frac{1}{T} \sum_{t=1}^T \mathbf{1}\{\mathbf{y}_t \neq \mathbf{y}'_t\}$, when \mathbf{y} decomposes as $T = \log_2 |\mathcal{Y}|$ binary variables $(y_t)_{1 \leq t \leq T}$. Because “softer” losses like the Hamming

⁸In the topological sense.

loss are more informative about the mistakes, we can expect that fewer examples are needed to learn the model. This is, with caveats, what was shown by [Osokin et al. \[2017\]](#) in a nonparametric setting.⁹ [Osokin et al. \[2017\]](#) derive a worst case sample complexity needed for a simple learner based on surrogate loss minimization via stochastic gradient descent to achieve a fixed regret $\epsilon > 0$ with respect to the best generalization error possible. The sample complexity quantifies how many samples are needed for the simple learner to guarantee a regret of ϵ . When the task is the 0-1 loss, they get a sample complexity of $O(|\mathcal{Y}|/\epsilon^2)$, which is exponential in the dimension of y . However, when the task loss is the Hamming loss, they get a much better sample complexity of $O(\log_2 |\mathcal{Y}|/\epsilon^2)$ which is linear in the number of dimensions, whenever certain constraints are imposed on the score function [see [Osokin et al., 2017](#), section on exact calibration functions]. In contrast, if the surrogate loss of the learner is based on the 0-1 loss (the analog of the top red curve in [Figure 4.1](#)), but the task loss is the Hamming loss (giving rise to the analog of the bottom green curve in [Figure 4.1](#)), then the sample complexity of the learner is still exponential.

Thus their results suggest that choosing the *right* structured loss, like the “softer” Hamming loss, might make training *exponentially* faster. According to the previous analogy, these results could mean that it might also be more efficient to use softer losses than the KL-divergence for training generative models. These observations echo results in generative modeling [[Arjovsky et al., 2017](#), [Liu et al., 2017](#)] showing that it can be easier to learn with weaker divergences than with stronger ones (in the topological sense). In particular, one of their arguments is that distributions with disjoint support can be compared in weaker topologies like the one induced by the Wasserstein but not in stronger ones like the one induced by the Jensen-Shannon.

However, we will show in [Section 4.1](#) that the Wasserstein distance has other issues, such as poor sample complexity. On the other hand, parametric adversarial divergences, which are also a softer alternative to the KL-divergence, have reasonable sample complexity and other good properties which are discussed in [Sections 4, 5 and 7](#).

3.4 Training and Evaluation in Practice

Training vs. Evaluation Loss. Strictly speaking, task losses should be regarded as evaluation metrics, from which we can then derive training losses which are easier to optimize (e.g. 0-1 task loss is approximated with cross-entropy training

⁹The analysis of [Osokin et al. \[2017\]](#) is nonparametric in the sense that it ignores the dependence on x (it allows an arbitrary dependence on x for the score functions by using an infinite dimensional RKHS function space). Additionally, they only consider convex consistent surrogate losses in their analysis, and they give upper bounds but not lower bounds on the sample complexity. It is possible that optimizing approximately-consistent surrogate losses instead of consistent ones, or making additional assumptions on the distribution of the data could yield better sample complexities.

loss). In practice, parametric divergences are only used as training losses but not as evaluation metrics, because optimization instability makes it difficult to compute them reliably. However, future work might make it possible to use parametric divergences as evaluation metrics as well.

Estimation Biases. Consider two fixed distributions p and q . We are usually interested in the *population* divergence

$$L_p(\theta) \triangleq \text{Div}(p||q_\theta) = \sup_{f \in \mathcal{F}} \mathbf{E}_{(\mathbf{x}, \mathbf{x}') \sim p \otimes q_\theta} [\Delta(f(\mathbf{x}), f(\mathbf{x}'))] \quad (3.7)$$

In practice, p is unknown and we only have access to samples. We might also only have access to samples from the model when it is an implicit one. Denoting $D_{train} = \{\mathbf{x}_{train}^{(i)} \stackrel{\text{iid}}{\sim} p, \mathbf{y}_{train}^{(i)} \stackrel{\text{iid}}{\sim} q\}$, we compute the *training* divergence as follows :

$$\text{Div}(p||q_\theta)_{train} = \sup_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^N \Delta(f(\mathbf{x}_{train}^{(i)}), f(\mathbf{y}_{train}^{(i)})) \quad (3.8)$$

Denote \hat{f}_{ERM} the previous minimizer. We define the *validation* divergence as the evaluation of \hat{f}_{ERM} over a validation set $D_{val} = \{\mathbf{x}_{val}^{(i)} \stackrel{\text{iid}}{\sim} p, \mathbf{y}_{val}^{(i)} \stackrel{\text{iid}}{\sim} q\}$:

$$\text{Div}(p||q_\theta)_{val} = \frac{1}{N'} \sum_{i=1}^{N'} \Delta(\hat{f}_{ERM}(\mathbf{x}_{val}^{(i)}), \hat{f}_{ERM}(\mathbf{y}_{val}^{(i)})) \quad (3.9)$$

Analogously to usual classification bounds (the discriminator can be seen as a classifier), the expected training divergence is *higher* than the population divergence, while the expected validation divergence is *lower* than the population divergence, where the expectations are taken over the sampling of D_{train}, D_{val} .

Theorem 1 (Parametric Divergence Biases). The following inequalities hold :

$$\underbrace{\mathbf{E}_{D_{train}, D_{val}}[\text{Div}(p||q)_{val}]}_{\text{validation divergence}} \leq \underbrace{\text{Div}(p||q)}_{\text{population divergence}} \leq \underbrace{\mathbf{E}_{D_{train}}[\text{Div}(p||q)_{train}]}_{\text{training divergence}} \quad (3.10)$$

Proof. The second inequality results from the fact that taking the supremum inside the expectation $\mathbf{E}_{D_{train}}[\cdot]$ is always higher than outside it.

$$\sup_{f \in \mathcal{F}} \underbrace{\mathbf{E}_{D_{train}} \left[\frac{1}{N} \sum_{i=1}^N \Delta(f(\mathbf{x}_{train}^{(i)}), f(\mathbf{y}_{train}^{(i)})) \right]}_{\mathbf{E}_{(\mathbf{x}, \mathbf{x}') \sim p \otimes q_\theta} [\Delta(f(\mathbf{x}), f(\mathbf{x}'))]} \leq \underbrace{\mathbf{E}_{D_{train}} \left[\sup_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^N \Delta(f(\mathbf{x}_{train}^{(i)}), f(\mathbf{y}_{train}^{(i)})) \right]}_{\text{Div}(p||q)_{train}} \quad (3.11)$$

For the first inequality, taking the expectation of the validation divergence with respect to the sampling of the validation set gives

$$\mathbf{E}_{D_{val}} \left[\frac{1}{N} \sum_{i=1}^N \Delta(\hat{f}_{ERM}(\mathbf{x}_{val}^{(i)}), \hat{f}_{ERM}(\mathbf{y}_{val}^{(i)})) \right] = \mathbf{E}_{(\mathbf{x}, \mathbf{x}') \sim p \otimes q_\theta} [\Delta(\hat{f}_{ERM}(\mathbf{x}), \hat{f}_{ERM}(\mathbf{x}'))] \quad (3.12)$$

$$\leq \sup_{f \in \mathcal{F}} \mathbf{E}_{(\mathbf{x}, \mathbf{x}') \sim p \otimes q_\theta} [\Delta(f(\mathbf{x}), f(\mathbf{x}'))] \quad (3.13)$$

where the inequality comes from the definition of the supremum. Now, we take expectations with respect to the sampling of the training set, which \hat{f}_{ERM} depends on:

$$\mathbf{E}_{D_{train}, D_{val}} \left[\underbrace{\frac{1}{N} \sum_{i=1}^N \Delta(\hat{f}_{ERM}(\mathbf{x}_{val}^{(i)}), \hat{f}_{ERM}(\mathbf{y}_{val}^{(i)}))}_{\text{Div}(p||q)_{val}} \right] \leq \underbrace{\sup_{f \in \mathcal{F}} \mathbf{E}_{(\mathbf{x}, \mathbf{x}') \sim p \otimes q_\theta} [\Delta(f(\mathbf{x}), f(\mathbf{x}'))]}_{\text{Div}(p||q)} \quad (3.14)$$

■

Notice that the validation divergence is not an unbiased estimator of the population divergence, because the discriminator was only optimized over the training set. Therefore, it is always worthwhile to look both at training and validation divergence to bound the population divergence. If the discriminator is overfitting to the training set, then there would be a large *generalization* gap between the two, while if the values are close, then we have a good estimate of the population divergence. To the best of our knowledge, there is no unbiased estimators of the population divergence in the general case. An exception is the family of f-divergences which can be estimated with Monte-Carlo in the special case where p, q have a density which can be evaluated. Another exception is MMD [Bellemare et al., 2017], which admits a closed-form unbiased estimator. However, such estimator is only unbiased for a fixed kernel. If the kernel were to be optimized to maximize some discrepancy between empirical distributions, in the same way a discriminator is optimized, then the MMD estimator would become biased as well [Bińkowski et al., 2018].

4 Good Divergences Should Scale Well with Data Dimensionality

One of the main difficulties in generative modeling is to deal with high-dimensional data because of the curse of dimensionality, which states that in order to fill a

given volume in data space, it takes an amount of data which is exponential in its dimension. The sample complexity, which characterizes how much data is needed to approximate a given divergence, is discussed for nonparametric and parametric divergences in Section 4.1. We also compare some nonparametric and parametric divergences experimentally in Sections 4.2 and 4.3.

4.1 Theoretical Sample Complexities in High-Dimensions

Consider two distributions p, q and their associated empirical distributions \hat{p}_n, \hat{q}_n . Typically, p is the unknown distribution to learn, and q is the model (or generator) distribution. How much data is required to approximate the true (population) divergence $\text{Div}(p||q)$ with the empirical divergence $\text{Div}(\hat{p}_n||\hat{q}_n)$? Formally, we can define the sample complexity as the minimal number of samples n such that $|\text{Div}(p||q) - \text{Div}(\hat{p}_n||\hat{q}_n)| \leq \epsilon$ with high probability for $\epsilon > 0$.

Following the terminology of Mohamed and Lakshminarayanan [2016], we distinguish the case of *explicit* model, where the density $q(x)$ can be numerically evaluated, and the case of *implicit* model, where it is only possible to sample from q . For instance, GAN generators are implicit models, and have a distribution which is typically supported in a low-dimensional manifold and does not admit a density with respect to the usual measure. Explicit models are more restrictive and need to have a full-dimensional support. For instance, VAEs and PixelCNNs are explicit models, but they each come with their own problems (see Section 6.1). We summarize the sample complexities for some parametric and nonparametric divergences in Table 4.1.

Parametric Divergences. Parametric adversarial divergences can be formulated as a classification problem between p and q , with a loss depending on the specific adversarial divergence. They can be estimated for implicit models and have a reasonable sample complexity of $O(p/\epsilon^2)$, where p is the VC-dimension/number of parameters of the discriminator [Arora et al., 2017]. They are usually computed using stochastic gradient descent (SGD), which provides no guarantees of finding a global minimum. This is not necessarily a bad thing, as it has been theoretically shown in the supervised case that SGD induces some form of bias or regularization, leading for instance to maximum-margin solutions in the separable-data case [Soudry et al., 2018]. We might expect the same type of implicit regularization to be beneficial for parametric divergences too.

(Nonparametric) Wasserstein. A straightforward estimator of the (nonparametric) Wasserstein is simply the Wasserstein distance between the empirical distributions \hat{p}_n and \hat{q}_n , for which smoothed versions can be computed in $O(n^2)$ using specialized algorithms such as Sinkhorn’s algorithm [Cuturi, 2013] –it is then

Divergence	Sample Complexity	Computation	Tunable to Final Task
Explicit model: Can evaluate $p(x)$			
f-Divergences	$O(1/\epsilon^2)$	Monte-Carlo, $O(n)$	No
Implicit model: Can only sample $x \sim p$			
f-Divergences		undefined in general	
Nonparametric Wasserstein	$O(1/\epsilon^{d/2})$	Sinkhorn, $O(n^2)$	in base distance
MMD	$O(1/\epsilon^2)$	analytic, $O(n^2)$	in kernel
Parametric Divergence	$O(p/\epsilon^2)$	SGD	in discriminator
Parametric Wasserstein Div.	$O(p/\epsilon^2)$	SGD	in discriminator & base distance

Table 4.1: Properties of Divergences. Note that although f-divergences can be estimated efficiently for explicit models, they are usually not defined for implicit models (see text). MMD can be estimated efficiently in closed form and can be tuned through the choice of kernel, but is known to lack discriminative power for generic kernels. The nonparametric Wasserstein can be computed iteratively with the Sinkhorn algorithm, and can integrate the final loss in its base distance, but requires exponentially many samples to estimate which makes it impractical in high dimensions. Parametric divergences have reasonable sample complexities, can be computed iteratively with SGD, and can integrate the final loss through the choice of class of discriminators and the choice of side-tasks. In particular, the parametric Wasserstein has the additional possibility of integrating the final loss into the base distance.

known as the plug-in estimator– or iterative Bregman projections [Benamou et al., 2015]. However, the empirical Wasserstein is a biased estimator and has sample complexity $n = O(1/\epsilon^d)$ which is exponential in the number of dimensions [see Sriperumbudur et al., 2012, Corollary 3.5]. This bound was recently improved to $O(1/\epsilon^{d/2})$ by Chizat et al. [2020]. Thus, the theory suggests that empirical Wasserstein is not a viable estimator in high-dimensions. In practice, we compare nonparametric and parametric Wasserstein on two generation tasks in Section 4.2.

f-divergences. For explicit models which allow evaluating the density $q_\theta(x)$, one could use Monte-Carlo to evaluate the f-divergence with sample complexity $n = O(1/\epsilon^2)$, according to the Central-Limit theorem. However, for implicit models, there is no one good way of estimating f-divergences from samples. And in fact, most f-divergences are generally not defined for empirical distributions because they might not be absolutely continuous with another. There are some techniques for estimation [Nguyen et al., 2010, Moon and Hero, 2014, Ruderman et al., 2012], but they all make additional assumptions about the underlying densities (such as smoothness), or they solve the dual in a restricted family, such as a RKHS, which makes the divergences no longer f-divergences.

MMD. Maximum Mean Discrepancy admits an estimator with sample complexity $n = O(1/\epsilon^2)$, which can be computed analytically using U-statistics $O(n^2)$, or even

in $O(n)$ using the linear estimator [Gretton et al., 2007]. One should note that MMD depends fundamentally on the choice of kernel. As the sample complexity is independent of the dimension of the data, one might believe that the MMD estimator behaves well in high dimensions. However, it was experimentally illustrated in Dziugaite et al. [2015] that with generic kernels like RBF, MMD performs poorly for MNIST and Toronto face datasets, as the generated images have many artifacts and are clearly distinguishable from the training dataset.

It was shown theoretically in [Reddi et al., 2015] that the power of the MMD statistical test can drop polynomially with increasing dimension, which means that with generic kernels, MMD might be unable to discriminate well between high-dimensional generated and training distributions. Specifically, consider a Gaussian kernel with bandwidth γ and compute the MMD^2 between two isotropic Gaussians with different means. Then, for $0 < \epsilon \leq 1/2$, and $d \rightarrow \infty$, MMD^2 goes to zero:

- polynomially as $1/d$ if $\gamma = \sqrt{d}$
- polynomially as $1/d^{1+2\epsilon}$ if $\gamma = d^{1/2-\epsilon}$
- exponentially as $\exp(d^{2\epsilon}/2)$ if $\gamma = d^{1/2+\epsilon}$, all that while the KL divergence between the two Gaussians stays constant.

which suggest MMD with fixed kernel cannot separate high-dimensional distributions well.

Limitations of Sample Complexity Analysis. Note that comparing divergences in terms of sample complexity can give good insights on what is a good divergence, but should be taken with a grain of salt as well. On the one hand, the sample complexities we give are upper-bounds, which means the estimators could potentially converge faster. On the other hand, one might not need a very good estimator of the divergence in order to learn in some cases. This is illustrated in our experiments with the nonparametric Wasserstein which has bad sample complexity but yields reasonable results in some cases (Section 4.2).

4.2 Nonparametric vs. Parametric Wasserstein (Experiment)

Since the sample complexity of the nonparametric Wasserstein is exponential in the dimension (Section 4.1), we verify experimentally whether training a generator to minimize the *nonparametric Wasserstein* distance works in high dimensions. We implement the Sinkhorn-AutoDiff algorithm [Genevay et al., 2018] to compute the entropy-regularized L_2 -Wasserstein distance between minibatches of training images and generated images, and minimize the divergence using stochastic gradient

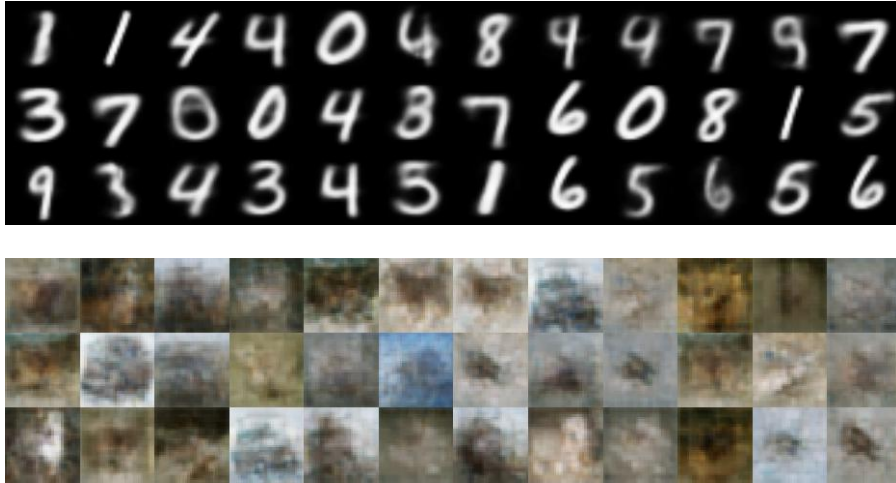


Figure 4.2: Images generated by the network after training with the Sinkhorn-Autodiff algorithm on MNIST dataset (top) and CIFAR-10 dataset (bottom). One can observe that although the network succeeds in learning MNIST, it is unable to produce convincing and diverse samples on the more complex CIFAR-10.

descent.¹⁰ **Figure 4.2** shows generated samples after training with the Sinkhorn-Autodiff algorithm on both MNIST and CIFAR-10 dataset. We then minimize the estimated divergence using stochastic gradient descent. On MNIST, the network manages to produce decent but blurry images. However on CIFAR-10, which has higher intrinsic dimensionality, the generator fails to produce meaningful samples. This is in stark contrast with the high quality generators displayed in the literature with a *parametric Wasserstein* (Wasserstein-GAN). This result would suggest that indeed the nonparametric Wasserstein should not be used for generative modeling when the (effective) dimensionality is high.

4.3 Generating Simple High-dimensional Images (Experiment)

There has recently been a number of successes in modelling high-dimensional images with GANs, such as 1024×1024 faces [Karras et al., 2018a] and 512×512 photos [Brock et al., 2019], which does suggest that parametric divergences are very successful in modelling high-dimensional data. We collect *Thin-8*, a dataset of 1585 grayscale handwritten images of the digit 8, with a very high resolution of 512×512 .¹¹ The *Thin-8* task differs from the aforementioned tasks because the images to model have **very low intrinsic dimensionality** (each digit is one curve which can be parametrized using only a handful of points), which allows us

¹⁰<https://github.com/gpeyre/SinkhornAutoDiff>

¹¹Thin-8 dataset can be download from <https://gabrielhuang.github.io/#thin>

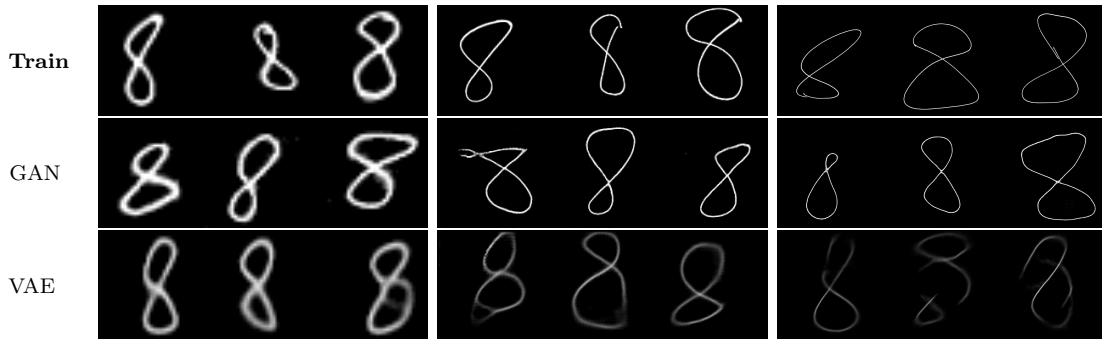


Figure 4.3: Samples from *Thin-8* training set (**top row**), WGAN-GP (**middle row**) and Convolutional VAE (**bottom row**) with 16 latent variables. Resolutions are 32×32 (**left column**), 128×128 (**middle column**), and 512×512 (**right column**). Note how the GAN samples are always crisp and realistic across all resolutions, while the VAE samples tend to be blurry with gray pixel values in high-resolution. We can also observe some averaging artifacts in the top-right 512×512 VAE sample, which looks like the average of two “8”. More samples can be found in Section 1.1 of the Appendix.

to factor out the usual complexity of high-dimensional photos. Therefore, even a simple convolutional network with few filters and few latent-dimensions (16 in our experiments) should be able to generate the images.

We train¹² a convolutional VAE and a WGAN-GP [Gulrajani et al., 2017], henceforth simply denoted GAN, using nearly the same architectures (VAE decoder similar to GAN generator, VAE encoder similar to GAN discriminator), with 16 latent variables, on the following resolutions: 32×32 , 128×128 and 512×512 . We optimize the losses using Adam, and augment the samples with random elastic deformation during training.

Generated samples are shown in **Figure 4.3**. We observe that the VAE, trained to minimize the evidence lower bound on maximum-likelihood, fails to generate convincing samples in high-dimensions: they are blurry, pixel values are gray instead of being white, and some samples look like the average of many digits. This can be explained by the fact that the smoothing used in the VAE reduces the maximum likelihood to a pixel-wise reconstruction loss, which is problematic here. Indeed, because the images are dominated by background pixels and the strokes are very thin, with high probability, any two “8” will intersect on no more than a little area, so pixel-wise distances are not meaningful.

On the contrary, the GAN which is trained with a parametric Wasserstein distance can generate sharp and realistic samples even in 512×512 . Our hypothesis is that

¹²Code available at <https://github.com/gabrielhuang/adversarial-divergence-code>

the discriminator learns only the moments that matter for visual quality, and that matching only those moments is easier than matching all of the moments, which is required for really maximizing the likelihood.

In conclusion, the high-dimensional aspect was clearly an obstacle for successfully training a fairly simple generator using maximum-likelihood, while using a parametric divergence to train the *exact* same generator allowed to overcome the high-dimensional aspect and resulted in high-quality samples.

5 Good Divergences Should Reflect the Final Task

In Section 3, we discussed the necessity and importance of designing task losses which reflect the final task. We showed that in structured prediction, optimizing more informative task losses can make learning considerably easier under some conditions. Similarly, in generative modeling, we would like divergences to be as informative and close to the final task as possible. Although not all divergences can easily integrate final task-related criteria, parametric divergences can be tuned through side-tasks and indirectly through their architecture (Section 5.1). We study the synthetic task of generating images of digits that sum up to 25, and compare KL-based and parametric divergences in their ability to enforce that constraint (Section 5.2 and 5.3).

5.1 Tuning Various Divergences to the Final Task

Pure f-divergences cannot directly integrate *any* notion of final task. By default, f-divergences might not have the expected properties, as we show experimentally in Section 5.2. To some extent, there is a possibility of tweaking f-divergences by combining them with generators that have a *special structure*; this is discussed in Section 6.1. One could also attempt to induce properties of interest by adding a *regularization* term to the f-divergence. However, if we assume that maximum likelihood is itself often not a meaningful task loss, then there is no guarantee that minimizing a tradeoff between maximum likelihood and a regularization term is more meaningful or easier.

The Wasserstein distance and MMD are respectively induced by a base metric $d(\mathbf{x}, \mathbf{x}')$ and a kernel $K(\mathbf{x}, \mathbf{x}')$. The metric and kernel give us the opportunity to specify a task by letting us express a (subjective) notion of *similarity*. However, the metric and kernel traditionally had to be defined by hand. For instance, [Genevay et al. \[2018\]](#) learn to generate MNIST by minimizing a smooth Wasserstein based on the L_2 -distance, while [Dziugaite et al. \[2015\]](#), [Li et al. \[2015\]](#) also learn to generate

MNIST by minimizing the MMD induced by kernels obtained externally: either generic kernels based on the L_2 -distance or on autoencoder features. However, the results seem to be limited to simple datasets. There is no *obvious* or generally accepted way to learn the metric or kernel in an end-to-end fashion; this is an active research direction. In particular, MMD has recently been combined with adversarial kernel learning, with convincing results on LSUN, CelebA and ImageNet images: Mroueh et al. [2017] learn a feature map and try to match its mean and covariance, Li et al. [2017] learn kernels end-to-end, while Bellemare et al. [2017] do end-to-end learning of energy distances, which are closely related to MMD. See Bińkowski et al. [2018] for a recent review of MMD-based GANs.

Parametric adversarial divergences can be tweaked to fit the final task in several ways. The first knob is the choice of discriminator *architecture*, which implicitly determines what aspects of the data the divergence is more sensitive or blind to. A typical choice of discriminator architecture for image generation are convolutional neural networks [Radford et al., 2016], since CNNs have several good properties for assessing whether an image is realistic: ability to detect blurriness, edges and textures, while being robust to translations and small deformations. The second knob is the use of a *side-task*. Instead of solely training the discriminator to distinguish true and generated data, one can also train the discriminator to solve a relevant side-task at the same-time, with the hope that it induces properties of interest on the resulting divergence. We will show in Sections 5.2 and 5.3 how a discriminator can be made much more sensitive to certain aspects of the data using a side-task.

5.2 Sensitivities of Divergences to Different Aspects of the Sum-25 Distribution

In this section, we introduce the Sum-25 task as a benchmark for comparing the sensitivities of divergences to different aspects of the data. We then add a side-task to a parametric divergence and show that it can improve sensitivity to important aspects of the data.

Sum-25 Task. The Sum-25 task consists in generating combinations of 5 digits that sum up to 25. We devise the following, *on-the-fly* dataset. First, we enumerate all 5631 combinations of 5 digits (out of 100,000) such that these digits sum up to 25. Then, we split them into disjoint train (50%) and test (50%) sets. The sampling process consists in uniformly sampling a random combination from the train/test set, then sampling corresponding digit images from MNIST, and finally concatenating them to yield the final image containing the 5 digits in a row summing up to 25.

Visual and Symbolic Constraints. The Sum-25 task can be thought of as a toy version of more complex tasks which have their own constraints. An example real task could be to generate pictures which satisfy perspective (e.g. objects must be skewed appropriately) and physical constraints (e.g., no floating objects). In the case of the Sum-25 task, the generator needs to model *two* aspects of the data accurately:

- *Visual constraint* : Digits should be recognizable and have good visual quality.
- *Symbolic constraint* : Digits must sum up to 25.

Therefore, can we define divergences which enforce both visual and symbolic constraints ?

Factorized Distributions. To factor out the effects of the learning process, we compute divergences between fixed reference p and candidate q distributions. Additionally, we restrict p and q distributions to be factorizable into a symbolic model and a conditional visual model as follows

$$q(x) = \sum_z q(x, z) = \sum_z \underbrace{q(z)}_{\text{symbolic model}} \prod_{i=1}^5 \underbrace{q(x_i|z_i)}_{\text{conditional visual model}} \quad (5.1)$$

where z_i are symbolic digits and x_i images of digits. We denote *Test*-25 the reference distribution, which is uniform over combinations of 5 digits from the MNIST test-set which sum up to 25 (symbolic constraint satisfied).

Sensitivity of KL-divergence to Constraints. In our first experiment, we consider 6 candidate distributions based on combining 3 conditional single-digit visual models with 2 symbolic models. The visual models are VAEs which are trained with Adam to generate single MNIST digits for 10, 70, and 80 epochs (prefixes Vae_{10-} , Vae_{70-} , Vae_{80-}). The symbolic models **-25** (resp. **-Non25**) are the uniform distributions over combinations of 5 digits that sum up (resp. do not sum up) to 25. We compute the $\mathbf{E}_p[-\log q]$ using (5.1), but replacing $q(x_i|z_i)$ with the variational lower-bound of the VAE, since these can be naturally computed. The numbers in **Table 4.2** show that the negative log-likelihood is *overly sensitive* to the visual appearance, but barely cares about the higher-level Sum-25 constraint. Therefore, by training with such an objective we might end up with digits that don't sum up to 25. More generally, this suggests that models trained with nonparametric divergences such as KL might not be able to enforce certain constraints which might be important to the final task.

Sensitivity of Parametric Divergence (a,b) For our second experiment, we would like to compute the parametric Jensen-Shannon between the reference

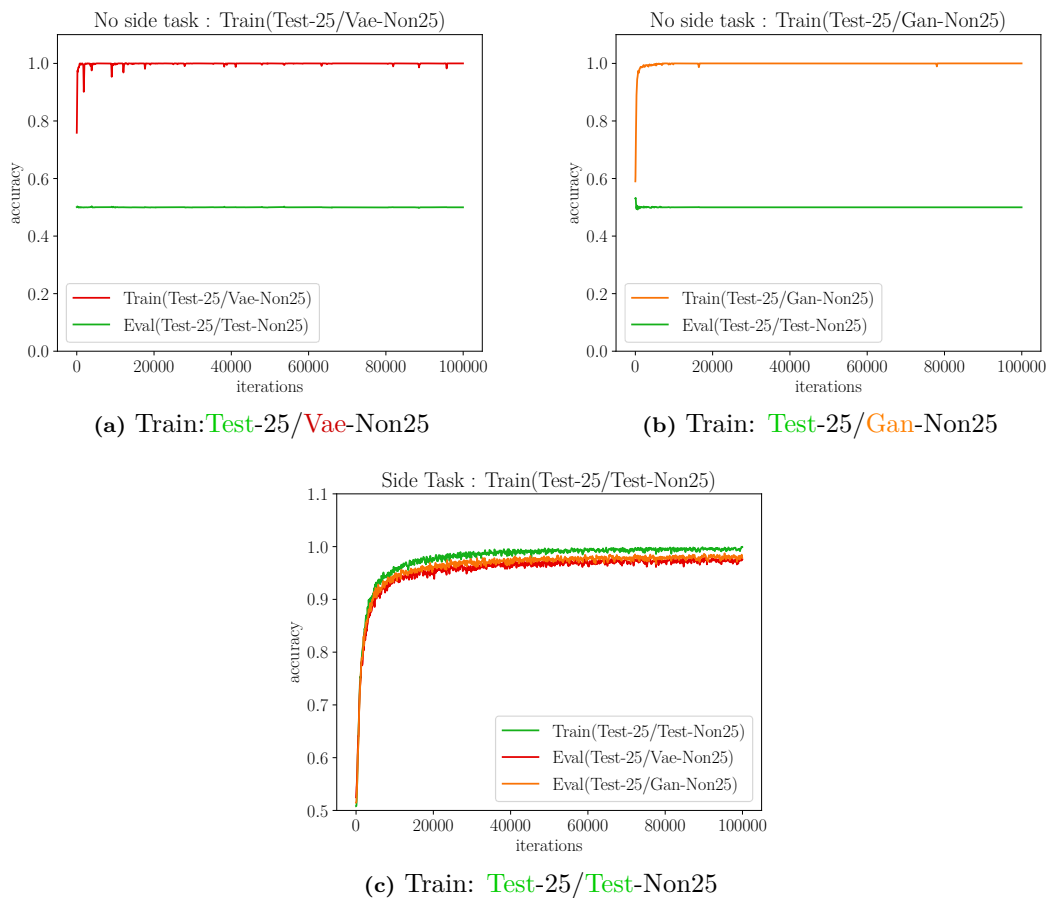


Figure 4.4: Probing the discriminator. Experiment **a** (resp **b**) : we train the discriminator to separate **Test-25/Vae-Non25** (resp. **Test-25/Gan-Non25**) training accuracy (red) goes to 1 very quickly. We probe the discriminator to see if it can separate **Test-25/Test-Non25**, and observe bad accuracy (green), which means the discriminator was unable to learn the sum-25 constraint, and discriminated solely on visual properties. In Experiment **(c)**, we train the discriminator to separate **Test-25/Test-Non25** as a side task. Good training accuracy (green) is evidence that the discriminator learned the sum-25 constraint, since it is the only way to separate the distributions. We then evaluate if the discriminator can separate **Test-25/Vae-Non25** and get evaluation accuracy (red) almost as good training accuracy. Slightly higher evaluation accuracy is observed for **Test-25/Gan-Non25**, which makes sense since **Gan** digits are visually more similar to **Test**. Thus the discriminator can detect the sum-25 constraint despite the **Test** \rightarrow **Vae** and **Test** \rightarrow **Gan** domain shifts, which suggests such the SideTask could help enforce Sum-25 during end-to-end training (i.e. training a GAN).

Visual Model $q(x_i z_i)$	Sum-25 satisfied	Sum-25 NOT satisfied
Vae_{10}	572.3 ± 1.4	575.4 ± 1.5
Vae_{70}	488.9 ± 1.2	487.3 ± 1.2
Vae_{80}	484.5 ± 1.2	483.7 ± 1.2

Table 4.2: Estimated Negative Log-Likelihoods of **Test-25** when Sum-25 constraint is and is NOT satisfied by $q(z)$, for VAE conditional visual models trained for 10, 70 and 80 epochs. Notice how there is barely any improvement from having digits sum up to 25 or not (compare “Sum-25 satisfied” to “NOT satisfied”), while even tiny improvements in visual quality yield substantial gains in NLL (compare Vae_{70}, Vae_{80} which have no perceptible difference).

distribution **Test-25** and candidate distributions based on various conditional and symbolic models. We consider **Vae-25**, **Vae-Non25**, **Gan-25** and **Gan-Non25**, where **Vae** is a VAE trained for 80 epochs, and **Gan** is a WGAN-GP trained for 50 epochs. However, we cannot present a similar table as previously because the numerical values of parametric divergences are unstable/random and dependent both on discriminator initialization and random sampling of data (even with gradient-penalty/other formulations). Instead, we propose an alternative approach which we call *probing* the discriminator. Specifically, we first train the discriminator to classify **Test-25/Vae-Non25** (resp. **Test-25/Gan-Non25**), but we evaluate its accuracy on the different problem of separating **Test-25/Test-Non25** to see whether the discriminator has learned to detect the sum-25 constraint. As shown in plot (a) of Figure 4.4, the discriminator fails to discriminate **Test-25/Test-Non25**, which suggests that it has only focused on the visual constraint during training. This is actually not that surprising as the discriminator can actually get excellent accuracy just by examining the visual quality of the first digit.

Sensitivity of Parametric Divergence w/ Side-Task (c) In our third experiment, we explore whether we can better enforce the Sum25 constraint through the discriminator. For this purpose, we train the discriminator on the side-task of separating **Test-25/Test-Non25**. Since the individual digits are now indistinguishable, the only way for the discriminator to get maximum accuracy on the side-task is to consider digits jointly and detect if they sum to 25. After training the discriminator on the side task, we probe the discriminator to see whether it can separate **Test-25/Vae-Non25** (resp. **Test-25/Gan-Non25**) which simulates the target and (imperfect) model distribution one could encounter while training a generative model. As shown in plot (c) of Figure 4.4, it turns out the discriminator can separate **Test-25/Vae-Non25** (resp. **Test-25/Gan-Non25**) fairly well, and moreover, it can also generalize to new combinations it has not seen before. This suggests that the side-task approach can help enforce sum-25 when learning q end-to-end, in a GAN setting.

5.3 Generating the Sum-25 Distribution

The previous section suggested that KL-divergence does not care about the symbolic constraint, that the vanilla parametric divergence does not either, but the parametric divergence + side-task cares about the constraint. We investigate if our intuitions hold when actually training a generator using these divergences. We train three models to generate the training set: a **VAE** (objective is upper bound on nonparametric Kullback-Leibler), a WGAN-GP (objective is parametric Wasserstein, simply denoted **GAN**), and another WGAN-GP with the additional side-task of discriminating **Test-25/Test-Non25** (objective is parametric Wasserstein with side task, denoted **GAN-SideTask**). All models share the same architecture for their generator network and use 200 latent variables. After training, with the help of a MNIST classifier, we automatically recognize and sum up the digits in each generated sample. As usual, the VAE samples are a bit blurry while the GAN samples are more realistic and crisp. Generated samples can be found in Section 1.2 of the appendix. We then compare how well VAE, GAN and GAN-SideTask *enforce* and *generalize* the constraint that the digits sum to 25.

Enforcing the Symbolic Constraint [Figure 4.5 left]. We plot the distributions of the sums of the combinations generated by the three models. The GAN-SideTask samples are the most concentrated around the target 25, followed by the GAN, and then VAE which is barely better than the independent baseline (digits follow marginal distribution). In that respect, the GAN-SideTask (though still far from nailing the problem) was much better than the VAE at capturing and enforcing the particular aspects and constraints of the data distribution (summing up to 25). This corroborates our prior hypothesis that side-task can help better align parametric divergences with the final task. Surprisingly, the GAN without side-task also outperforms the VAE. One hypothesis is that when GAN samples become visually too hard to discriminate, the discriminator will eventually start to detect the sum-25 constraint, even if such effect was not observed during fixed distribution experiments. Another possibility is that since the generator is always adapting to the discriminator (joint training), the discriminator does not have time to focus too much on visual details.

Generalizing the Symbolic Constraint [Figure 4.5 right]. We plot the train and test recall scores, which are defined as the proportions of the train/test combinations covered by a generative model after generating a fixed number of samples. High *train* recall means the generators cover the training combinations well (no symbolic **mode dropping**), while high *test* recall means they are able to generate new combinations which were not in the training set (symbolic **generalization**). Again, GAN-SideTask has best train/test recalls, followed by GAN, and VAE ranks last with same recall as the Independent Baseline. There is a very slight

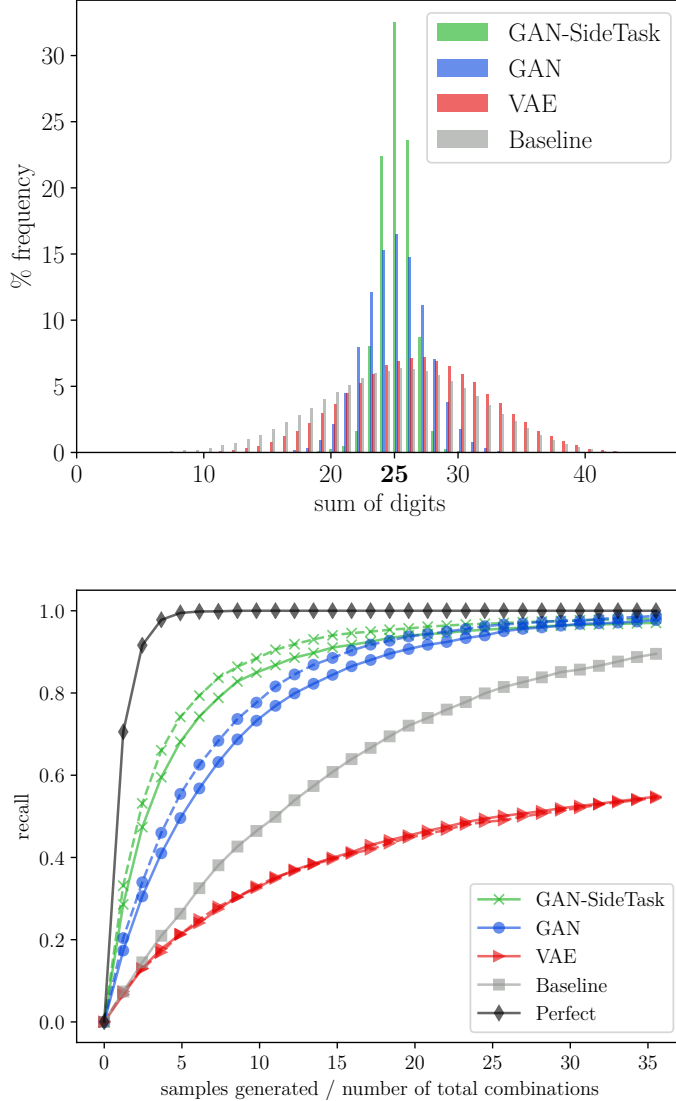


Figure 4.5: Top: Histograms of the sums of digits generated by VAE (red), WGAN-GP (green) and Independent Baseline (gray). The baseline draws digits independently according to their empirical marginal probabilities, which corresponds to fitting independent multinomial distributions over digits using maximum likelihood. WGAN-GP beats largely both VAE and Independent Baseline as it gives a sharper distribution centered on the target sum 25. **Bottom:** Train (dashed) and test (solid) which tell us how well the models cover the train and test set. The best theoretical recall is given by the Perfect generator (black) which samples uniformly among the 5631 combinations. Higher train recall means *less mode dropping*, while higher test recall means *better ability to generalize constraints*. WGAN-GP has the best recall (green), followed by the independent baseline (gray) and the VAE (red). Plots averaged over 5 runs. Train (dashed) and test (solid) recalls are identical for independent baseline and perfect generator.

overfitting to the training set for the GANs (solid line lower than dashed line), but GAN-SideTask seems to generalize very well. For models with very low recalls such as VAE, train and test recalls are equal, which is not surprising (it is the case for the uniform distribution). We leave for future work to investigate whether the generalization of the constraint is due more to the discriminator or the generator.

Conclusion of Sum-25. To conclude the Sum-25 experiments, parametric divergences have allowed us, through the addition of a side-task, to better enforce the symbolic constraint, and thus yield generative models which better solve the Sum-25 task than ones trained with nonparametric divergences or vanilla parametric divergences. We can imagine that for more complex tasks, practitioners can create side-tasks to make the discriminators more sensitive to certain aspects of the distribution.

6 Interactions between Generator and Divergence

Previously in Sections 4 and 5, we gave arguments against using f-divergences for general generators because f-divergences cannot directly handle implicit generators or be tuned to reflect the final task. Here, we discuss the fact that certain generators with a special structure can compensate for the shortcomings of the KL-divergence, but this special structure can also bring other problems (Section 6.1). We also consider the *converse* question of whether we can train a memorization-based generators with no generalization ability, but using a parametric divergence (Section 6.2).

6.1 Interaction of KL-divergence with Special Generators

Certain generators with a special structure can compensate for the shortcomings of the KL-divergence. Here, the two special structures we discuss are : smoothing the generator distribution with a (Gaussian) observation model and autogressive models.

Smoothing observation model. By adding an observation model such as a Gaussian model, on top of any generator, one can artificially extend its support to the whole input space. In particular, this makes the KL-divergence well-defined, and makes it possible to train models such as variational autoencoders [Kingma and Welling, 2014] (VAEs). The observation model makes the log-likelihood involve a “reconstruction loss”, a pixel-wise L_2 distance between images analogous to the Hamming loss, which makes the training relatively easy and very stable. However, the Gaussian is partly responsible for the VAE’s inability to learn sharp distributions.

Indeed it is a known problem that VAEs produce blurry samples [Arjovsky et al., 2017], and in fact even if the approximate posterior matches exactly the true posterior, which would correspond to the evidence lower-bound being tight, the output of the VAE would still be blurry [Bousquet et al., 2017].

Autoregressive models. Another example of special structure is autoregressive models, such as recurrent neural networks [Mikolov et al., 2010], which factorize naturally as $q_\theta(x) = \prod_i q_\theta(x_i|x_1, \dots, x_{i-1})$, and PixelCNNs [Oord et al., 2016]. Training autoregressive models using maximum likelihood results in teacher-forcing [Lamb et al., 2016]: each ground-truth symbol is fed to the RNN, which then has to maximize the likelihood of the next symbol. Since teacher-forcing induces a lot of supervision, it is possible to learn using maximum-likelihood. Once again, there are similarities with the Hamming loss because each predicted symbol is compared with its associated ground truth symbol. However, among other problems, there is a discrepancy between training and generation. Sampling from q_θ would require iteratively sampling each symbol and feeding it back to the RNN, giving the potential to accumulate errors, which is not something that is accounted for during training. See Leblond et al. [2018] and references therein for more principled approaches to sequence prediction with autoregressive models.

6.2 Ability to Train Memorization-based Generators (Experiment)

Previously, we discussed using generators with special structure to compensate shortcomings of the KL-divergence. Here we explore the converse case. Can we train a memorization-based generator, which has no generalization abilities, using a parametric divergence? Obviously, we cannot expect the generator to do any generalization, but this experiment is a good sanity check to see whether a given divergence will enforce realistic samples. Additionally, the memorized data can be plotted as a summary of the target distribution.

We compare the parametric Wasserstein divergences induced by three different discriminators (linear, dense, and CNN) under the WGAN-GP [Gulrajani et al., 2017] formulation. The memorization-based generator is a mixture of 100 prototypes, which can be also thought of as a mixture of 100 Gaussians with zero-variance. To sample a new image, the generator randomly returns one of 100 learned images. The model “density” is $q_\theta(\mathbf{x}) = \frac{1}{K} \sum_z \delta(\mathbf{x} - \mathbf{x}_z)$, where x_z are the prototypes (images) and δ is the Dirac distribution.

Prototypes learned from MNIST are shown in **Figure 4.6**. The first observation is that the linear discriminator is too weak of a divergence: all prototypes only learn the mean of the training set. Now, the dense discriminator learns prototypes which sometimes look like digits, but are blurry or unrecognizable most the time.



Figure 4.6: All 100 Prototypes learned using linear (**left**), dense (**middle**), and CNN discriminator (**right**). We observe that with the linear discriminator, only the mean of the training set is learned, while using the dense discriminator yields blurry prototypes. Only the CNN discriminator yields clear prototypes.

The samples from the CNN discriminator are never blurry and recognizable in the majority of cases. Our results confirms that indeed, even for simplistic models like a mixture of Diracs, using a CNN discriminator provides a better task loss for generative modeling of images.

Note that it would have been impossible to directly fit the same generator using a KL-divergence, because such a model does not admit a continuous density. A workaround approach is to relax the generator into a model that admits a density with respect to the usual measure, by replacing the Diracs with Gaussians. Maximizing the likelihood could then be done using an algorithm similar to K-means, and would likely result in blurry prototypes, since each prototype would be expressed as the average of several images.

7 Parametric Divergences for Meaningful Mutual Information

Recall that the mutual information between two variables X and Y can be defined as the Kullback-Leibler divergence between the joint distribution and the product of marginals¹³

$$\mathbf{MI}(X, Y) = \int_{x,y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} dx dy = \mathbf{KL}(p(x, y) || p(x)p(y)) \quad (7.1)$$

Mutual information quantifies the amount of information, in *nats* or *bits*, which is shared between X and Y , independently of the way these variables are represented.

¹³We use densities for simplicity, but more general definitions based on measure theory exist.

We argue that the problems of the KL-divergence presented previously can lead to problematic properties of mutual information. We illustrate that mutual information is not always an intuitive concept and can have a paradoxical behavior (Sections 7.1 and 7.2). We give an explanation of these paradoxes in Section 7.3, and propose more intuitive and meaningful notions of generalized mutual information based on parametric divergences in Section 7.4. Finally, we apply the proposed generalized mutual information to the paradoxes and discuss the results.

7.1 The Corrupted-Label Paradox

We showcase a simple example of joint distribution $p(x, y)$ where the mutual information is high between x and y even if there is no meaningful dependency between them. This can be thought of as *overfitting* to the empirical distributions, which is unavoidable because mutual information makes no assumption whatsoever.

Consider $\{(x_i, y_i)\}_{1 \leq i \leq N}$ the labeled training examples of MNIST (although the paradox can be derived for any classification problem). There are $N = 60,000$ examples and $K = 10$ balanced classes, each class containing exactly $N/K = 6,000$ examples. Consider two joint distributions for which we compute the mutual information using the difference between marginal and conditional entropy $\mathbf{MI}(X, Y) = H(Y) - H(Y|X)$:

- **Empirical:** Define $p_1(x, y)$ as the empirical distribution of the training set $p_1(x, y) = \frac{1}{N} \sum_{i=1}^N \delta(x - x_i) \mathbf{1}(y = y_i)$, where $\delta(\cdot)$ is the Dirac distribution on \mathcal{X} . The marginal distribution of Y is uniform over 10 classes, so the marginal entropy of Y is $H(Y) = \log 10 = 2.30$ nats. For the conditional entropy, when x is known and $p_1(x) > 0$, the label is fully determined by $y = \sum_{i=1}^N \mathbf{1}(x = x_i) y_i$, so $H(Y|X) = 0$ nats, and the mutual information is $\mathbf{MI}(X, Y) = 2.3$ nats.
- **Pseudo-Random labels:** Sample and fix random permutation $\sigma(i)$ over 60,000 elements, and permute all labels by considering the distribution $p_2(x, y) = \sum_{i=1}^N \delta(x = x_i) \delta(y = y_{\sigma(i)})$. The marginal distribution of Y does not change, so $H(Y) = \log 10 = 2.30$ nats again. For the conditional entropy, the label is now fully determined¹⁴ by $y = \sum_{i=1}^N \mathbf{1}(x = x_i) y_{\sigma(i)}$, so $H(Y|X) = 0$ nats, and the mutual information is $\mathbf{MI}(X, Y) = 2.3$ nats again (assuming fixed σ).

It can be rather surprising that the mutual information in both cases is always approximately 2.3 nats, which corresponds the amount of information for disambiguating between 10 balanced classes. While the labels for the empirical distribution are very natural and correspond to the identity of the digit represented, the labels for the permuted-label distribution have no meaning at all and would appear random

¹⁴Assuming no duplicate images.

to any human. Clearly, the mutual information does not care whether X and Y having a *meaningful* dependency. Instead, it will always equal 2.3 nats as long as the same label is consistently assigned to each image.

7.2 The Hashing Paradox

We showcase a family of distributions $p_K(X, Y)$ – this time over continuous variables $X, Y \in \mathbb{R}$ – for which the mutual information $\mathbf{MI}(X, Y)$ goes to infinity, while X and Y appear to be increasingly independent. Recall that continuous mutual information is always non-negative and independent of the base-measure, and preserves the meaning that $\mathbf{MI}(X, Y) = 0$ if and only if X and Y are independent. If $Y = X$, then the mutual information equals infinity.

For any positive $K \in \mathbb{N}$, define the distribution $p_K(X, Y)$ as follows. Consider two independent uniform random variables $X, W \sim_{iid} U([0, 1])$, and a (deterministic) permutation σ_K of the range $\{0, \dots, K - 1\}$, which we call the *hash function* and will be defined later. We define the random variable $Y = \frac{\sigma_K(\text{floor}(K * X))}{K} + \frac{W}{K}$. It is very easy to verify that the marginal distribution of Y is also uniform on $[0, 1]$, so its marginal differential entropy is $H(Y) = 0$. For the conditional entropy, when X is known, we know that Y is uniform in an interval of measure $1/K$, so $H(Y|X) = \log \frac{1}{K} = -\log K$, regardless of the actual permutation σ_K . Therefore, the mutual information is always $\log K$ nats, and grows to infinity as the number of bins goes to infinity.

We consider two families of hash functions σ_K . For increasing K , we represent samples $(X, Y) \sim p_K$ along with the mutual information $\mathbf{MI}(X, Y)$ [Figure 4.7]:

- **Identity function** $\sigma_K(i) = i$. As K grows, Y has to be closer and closer to X , while their mutual information goes to infinity. This is intuitive because Y is essentially converging towards X and giving more and more information over X .
- **Pseudo-random permutation.** Consider any given implementation of Python. Seed the pseudo-random number generator (PRNG) to 0 (or any fixed number) and define a permutation σ by shuffling an array containing the range $\{1, \dots, K\}$. As shown in Figure 4.7, when K grows to infinity, Y looks visually more and more independent from X , thus we would intuitively expect the mutual information to vanish to zero. However, as proved previously, the mutual information does not depend on σ , and actually grows in $\log K$, which *contradicts* (in appearance) the fact that X and Y are *visually* more and more independent. Beyond visual independence, it would be easy to show that for large enough K , the random variables X and Y can be considered numerically independent for practical purposes, such as numerical integration $\int_{x=0}^1 \int_{y=0}^1 f(x, y) dp_K(x, y)$, as long as the function f is sufficiently smooth.

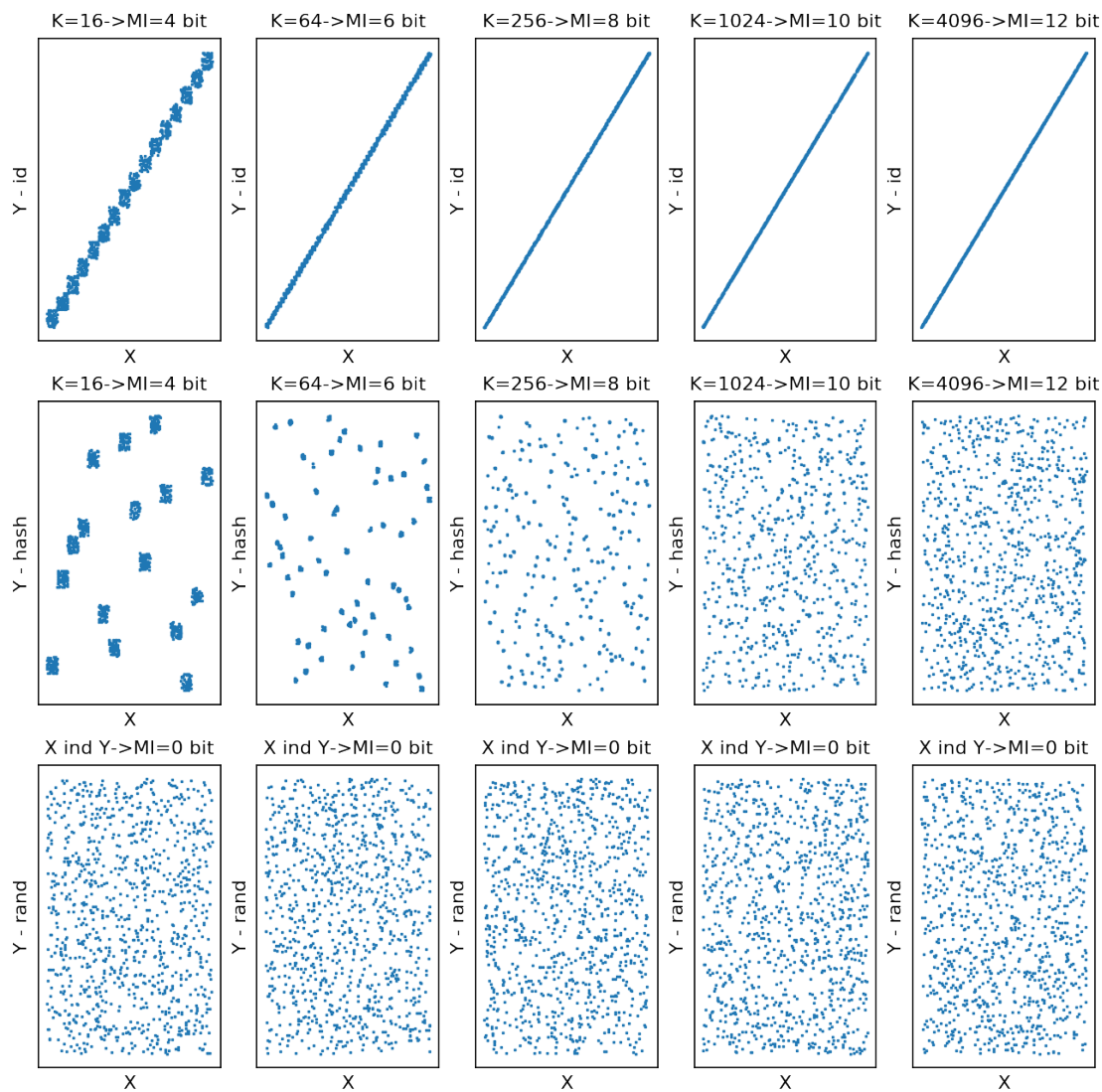


Figure 4.7: The Hashing Paradox. Samples of the distribution $p_K(x, y)$ for increasing numbers of bins $K \in \{16, 64, 256, 1024, 4096\}$, with corresponding mutual information. **Row 1:** the permutation σ_K is the identity. **Row 2:** the permutation is a pseudo-random permutation. **Row 3:** X and Y are sampled independently. Observe how the samples from Row 2 appear to be more and more independent (and distributed similarly to Row 3), despite having increasingly higher mutual information. In fact, for high enough K , it is impossible to tell Row 2 from Row 3.

7.3 Statistical Dependency can be Arbitrarily Complex

The previous paradoxes comes from the fact that mutual information does not care about any underlying *metric* or *similarity* on the spaces \mathcal{X} and \mathcal{Y} . Two points are either considered equal or not equal, which is a property that stems from the KL divergence. Moreover, the mutual information also does not care about the nature of the relationship between X and Y . Indeed, no matter how “complex” the relationship is, as long as there exists some function such that $Y = h(X)$, then all of Y is determined by X , and the mutual information between Y and X is maximal, i.e., equal to the entropy of Y .

If the relationship between X and Y is too *complex*, i.e., it is too *hard* to predict Y from X , then can we really consider that X and Y are dependent for any practical purposes? Consider the hashing example, where Y appears to be visually random from X . For applications like numerical integration of smooth functions, it is likely that we can expect the same type of guarantees as if X and Y were truly independent. However, for cryptographic applications where randomness requirements are much more stringent, we can probably not consider that X and Y are independent. In fact, we could even push the reasoning one step further and argue that pseudo-random number generators are not truly random. However, PRNGs such as linear congruential generators (LCG) are omnipresent in machine learning (e.g. SGD sampling, network initialization, sampling latent variables) and generally accepted to be truly random for all practical purposes, although successive samples are known to be highly dependent in a strict sense.

These apparent paradoxes motivate the need for defining more *meaningful* notions of mutual information, which are tailored to the specificities of the final application (e.g. numerical integration, sampling, assessing independence, learning disentangled latent variables). One way to do this is to control the complexity of the relationship $Y = h(X, \epsilon)$ between X and Y , where h is deterministic and ϵ is some noise independent from X . For instance, we can restrict the class of possible relationships $h \in \mathcal{H}$, or define a Bayesian prior $p(h)$ over them:

- **Linear Dependence.** If only linear dependence is considered simple, then the resulting notion of independence corresponds to the usual notions of linear correlation.
- **Smoothness.** Define some metric on \mathcal{X} and some metric \mathcal{Y} . We could restrict \mathcal{H} to the class of L -Lipschitz functions for some L , which means only smooth relationships are deemed interpretable.
- **Neural Networks.** Consider the set of neural networks with a fixed neural network architecture. We could restrict h to be representable with one of these neural networks. Alternatively, we could also restrict h to be a neural network which can be fitted within a limited number of SGD steps, in order

to take advantage of its implicit regularization properties.

- **Kolmogorov complexity.** Given a programming language, we could theoretically define the complexity of a function as the length of the shortest program for implementing that function. For instance, this language could be Python, restricted to its standard library.
- **Arbitrary Complexity / Memorization-based** If we consider all the functions $h \in \mathcal{H} = \mathcal{Y}^{\mathcal{X}}$, then any arbitrary pair (X, Y) can just be *memorized* inside h . By definition, any h is considered simple in the memorization-sense. In this case, we recover the classic definition of mutual information, which allows h to be arbitrarily complex.

We will be introducing complexity-aware variants of mutual information in Section 7.4, and experimenting with some of their properties in Section 7.6 and 7.7.

Explaining the corrupted label paradox. In the corrupted-label paradox for the random label case, each label Y is assigned according to an arbitrary rule σ , and cannot be deduced from the image X other than through memorization. Here the relationship $Y = h(X)$ is complex in the smoothness, neural network, and Kolmogorov sense, and only simple for the memorization-based sense. Therefore, if complexity in the sense of smoothness, neural network, and Kolmogorov complexity are meaningful to the final task (e.g. identifying semantic correlation), then an intuitive notion of mutual information should predict X and Y to be *independent*.

Explaining the Hashing Paradox In the hashing paradox, when $\sigma_K = \text{hash}$ is a pseudo-random permutation, the relationship h_K between X and Y is *complex* in the smoothness and neural network sense, but *simple* in the Kolmogorov sense because h_K can be implemented in a few lines of python. Therefore, if complexity in the sense of smoothness and neural networks are meaningful to the application (e.g. numerical integration of smooth functions), we should consider the variables to be *independent*. However, if complexity in the sense of Kolmogorov are meaningful to the application (e.g. cryptography), then the variables *cannot* be considered independent.

7.4 Generalized and Parametric Mutual Information

We explain how to generalize mutual information to account for arbitrary properties of the distribution, such as the complexity of the relationship $Y = h(X)$ and underlying metrics of the spaces X, Y . Recall that the mutual information $\mathbf{MI}(X, Y)$ between two random variables X and Y can be written as the KL-divergence between the joint $p_{X,Y}$ and the product of marginals $p_X \otimes p_Y$.

$$\mathbf{MI}(X, Y) = \mathbf{KL}(p_{X,Y} || p_X \otimes p_Y) \tag{7.2}$$

We define the *generalized mutual information* by replacing the KL-divergence with another nonparametric and parametric divergences.

Definition 7.1 (Generalized Mutual Information). Given a divergence $\mathbf{Div}(\cdot||\cdot)$, we define the \mathbf{Div} -generalized mutual information (GMI) as:

$$\mathbf{GMI}_{\mathbf{Div}}(X, Y) = \mathbf{Div}(p_{X,Y}||p_X \otimes p_Y) \quad (7.3)$$

Additionally, if $\mathbf{Div}(\cdot||\cdot)$ is a parametric divergence, then we say that $\mathbf{GMI}_{\mathbf{Div}}$ is a parametric mutual information (PMI).

Definition 7.2 (Generalized Independence). Given a divergence $\mathbf{Div}(\cdot||\cdot)$, we say that X and Y are \mathbf{Div} -independent if and only if $\mathbf{GMI}_{\mathbf{Div}}(X, Y) = 0$. We abuse the terminology and say that X and Y are \mathbf{Div} -independent even when \mathbf{Div} is not a proper divergence.

For instance, we could consider the class of f-divergences (KL, Jensen-Shannon), integral probability metrics (MMD, Wasserstein) or parametric adversarial divergences. We argue that the properties of parametric divergences, discussed in Sections 4 and 5 transfer over to the induced parametric mutual information.

Using MMD for independence-testing has been proposed in Gretton et al. [2012]. The kernel defines a similarity metric over $\mathcal{X} \times \mathcal{Y}$. It should be noted that MMD-generalized-mutual-information can only be as meaningful as the kernel considered. In particular, for generic kernels MMD-GMI might not be powerful enough to find some dependencies, for the same reasons MMD can fail to discriminate distributions in high dimensions (Section 4.1). Similarly, Wasserstein distance and variants have been proposed for independence-testing [Ramdas et al., 2017]. Just like for MMD, the choice of base-metric determines the properties of the generalize mutual information. However, similarly to the Wasserstein distance, Wasserstein-GMI also suffers from poor sample complexity in high dimensions. KL-parametric mutual information (KL-PMI) has been proposed in [Belghazi et al., 2018] under the name of Mutual Information Neural Estimator (MINE), as an approximator of the true mutual information. We argue that KL-parametric mutual information should not be seen as a mere approximator of mutual information, and that it can sometimes be a more natural and meaningful concept than traditional mutual information. We provide evidence of this in the experimental section 7.7.

Training and Validation GMI. In practice, generalized mutual information is estimated from a finite dataset, and it is handy to lower and upper bound the population generalized mutual information. We independently sample a training set $D_{train} = \{(\mathbf{x}_{train}^{(i)}, \mathbf{y}_{train}^{(i)})\}$ and a validation set $D_{val} = \{(\mathbf{x}_{val}^{(i)}, \mathbf{y}_{val}^{(i)})\}$ from the joint distribution $p_{X,Y}$.

Definition 7.3 (Training GMI). Denote N the size of the training set. We define the *training* GMI as:

$$\mathbf{GMI}(X, Y)_{train} = \sup_{f \in \mathcal{F}} \frac{1}{N(N-1)(N-2)} \sum_{i=1}^N \sum_{j \neq i}^N \sum_{k \neq i, j}^N \Delta(f(\mathbf{x}_{train}^{(i)}, \mathbf{y}_{train}^{(i)}), f(\mathbf{x}_{train}^{(j)}, \mathbf{y}_{train}^{(k)})) \quad (7.4)$$

In a similar spirit as the U-statistics-based MMD estimator proposed in Lemma 6 of [Gretton et al. \[2012\]](#), we remove some indices $j \neq i$ and $k \neq i, j$ so that we can reuse samples from the joint $p_{X,Y}$ as if $(\mathbf{x}_{train}^{(i)}, \mathbf{y}_{train}^{(i)})$, $(\mathbf{x}_{train}^{(j)}, \mathbf{y}_{train}^{(k)})$ were sampled from $p_{X,Y} \otimes (p_X \otimes p_Y)$. The resulting term inside the $\sup_{f \in \mathcal{F}}$ is an unbiased estimator of the term inside the $\sup_{f \in \mathcal{F}}$ of the corresponding adversarial divergence:

$$\mathbf{E}_{D_{train}} \left[\frac{1}{N(N-1)(N-2)} \sum_{i=1}^N \sum_{j \neq i}^N \sum_{k \neq i, j}^N \Delta(f(\mathbf{x}_{train}^{(i)}, \mathbf{y}_{train}^{(i)}), f(\mathbf{x}_{train}^{(j)}, \mathbf{y}_{train}^{(k)})) \right] \quad (7.5)$$

$$= \underbrace{\mathbf{E}_{(x,y),(x',y') \sim p_{X,Y} \otimes (p_X \otimes p_Y)} [\Delta(f(x, y), f(x', y'))]}_{\text{take } \sup_{f \in \mathcal{F}} \text{ to get } \mathbf{GMI}(X, Y)} \quad (7.6)$$

However, it is important to note that (7.4) is not an unbiased estimator of the GMI because of the supremum. We will only derive a bound in expectation in [Theorem 2](#).

Definition 7.4 (Validation GMI). Denote \hat{f} the previous maximizer, and N' the size of the validation set. We define the *validation* GMI as:

$$\mathbf{GMI}(X, Y)_{val} = \frac{1}{N'(N'-1)(N'-2)} \sum_{i=1}^{N'} \sum_{j \neq i}^{N'} \sum_{k \neq i, j}^{N'} \Delta(\hat{f}(\mathbf{x}_{val}^{(i)}, \mathbf{y}_{val}^{(i)}), \hat{f}(\mathbf{x}_{val}^{(j)}, \mathbf{y}_{val}^{(k)})). \quad (7.7)$$

Using the same principles as for supervised classification ([Section 3.4](#)) the true (population) GMI can be lower (resp. upper) bounded by the training (resp. validation) GMI ([7.8](#)).

Theorem 2 (Bounds for Generalized Mutual Information). For any generalized mutual information, the following bounds hold:

$$\mathbf{E}_{D_{train}, D_{val}} [\mathbf{GMI}_{val}(X, Y)] \leq \mathbf{GMI}(X, Y) \leq \mathbf{E}_{D_{train}} [\mathbf{GMI}_{train}(X, Y)]. \quad (7.8)$$

Proof. For conciseness, denote:

$$\hat{\Delta}_{train} = \frac{1}{N(N-1)(N-2)} \sum_{i=1}^N \sum_{j \neq i}^N \sum_{k \neq i, j}^N \Delta(f(\mathbf{x}_{train}^{(i)}, \mathbf{y}_{train}^{(i)}), f(\mathbf{x}_{train}^{(j)}, \mathbf{y}_{train}^{(k)})). \quad (7.9)$$

The right inequality results from the fact that taking the supremum inside the expectation $\mathbf{E}_{D_{train}}[\cdot]$ is always higher than outside it, and the fact that $\hat{\Delta}_{train}$ is an unbiased estimator of $\mathbf{GMI}(X, Y)$ without the supremum:

$$\mathbf{GMI}(X, Y) = \sup_{f \in \mathcal{F}} \underbrace{\mathbf{E}_{D_{train}}[\hat{\Delta}_{train}]}_{\mathbf{E}_{(x,y),(x',y') \sim p_{X,Y} \otimes (p_X \otimes p_Y)}[\Delta(f(x,y), f(x',y'))]} \leq \mathbf{E}_{D_{train}} \left[\underbrace{\sup_{f \in \mathcal{F}} \hat{\Delta}_{train}}_{\mathbf{GMI}_{train}(X,Y)} \right]. \quad (7.10)$$

For the left inequality, taking the expectation of $\mathbf{GMI}(X, Y)_{val}$ with respect to the sampling of the validation set gives:

$$\mathbf{E}_{D_{val}}[\mathbf{GMI}(X, Y)_{val}] = \mathbf{E}_{(x,y),(x',y') \sim p_{X,Y} \otimes (p_X \otimes p_Y)} [\Delta(\hat{f}(x, y), \hat{f}(x', y'))] \quad (7.11)$$

$$\leq \sup_{f \in \mathcal{F}} \mathbf{E}_{(x,x') \sim p \otimes q_\theta} [\Delta(f(x, y), f(x', y'))] \quad (7.12)$$

where the inequality comes from the definition of the supremum. Now, we take expectations with respect to the sampling of the training set, which \hat{f} depends on:¹⁵

$$\mathbf{E}_{D_{train}, D_{val}}[\mathbf{GMI}(X, Y)_{val}] \leq \mathbf{GMI}(X, Y) \quad (7.13)$$

■

We will be using these bounds to estimate the GMI in the experiments of Section 7.7.

7.5 MMD-Independence for Integrating Smooth Functions.

We take the example of numerical integration of smooth functions to illustrate that weaker notions of independence can be sufficient for practical purposes. Specifically, consider a smooth function $f(X, Y)$, in the sense that $f \in RKHS$ and $\|f\|_{RKHS} < \infty$. We want to compute its integral $I(f)$ over $[0, 1]^2$, which can also be written as an expectation by defining two *independent* random variables X, Y which are uniform over $[0, 1]$:

$$I(f) = \int_{x=0}^1 \int_{y=0}^1 f(x, y) dy dx = \int_{x \in X} \int_{y \in Y} f(x, y) p_Y(y) p_X(x) dy dx = \langle f, p_X \otimes p_Y \rangle_{L_2} \quad (7.14)$$

However, we can actually relax the independence assumption and only assume that X and Y are approximately MMD-independent, in the sense that their weak MMD-mutual information $\mathbf{GMI}_{\text{MMD}}(X, Y)$ is small or equal to zero. Under that assumption, the new integral is

$$J(f) = \int_{x \in X} \int_{y \in Y} f(x, y) p_{X,Y}(x, y) dy dx = \langle f, p_{X,Y} \rangle_{L_2} \quad (7.15)$$

¹⁵That last step is optional: the expectation with respect to D_{train} could be removed.

To show that $J(f)$ approximates $I(f)$ better as X and Y become more and more weakly MMD-independent, we write

$$|J(f) - I(f)| = | \langle f, p_{X,Y} - p_X \otimes p_Y \rangle_{L_2} | = | \langle f, \mu_{p_{X,Y}} - \mu_{p_X \otimes p_Y} \rangle_{RKHS} | \quad (7.16)$$

where the second equality comes from the definition of MMD embeddings $\mu_{p_{X,Y}}, \mu_{p_X \otimes p_Y}$ and the fact that f is in $RKHS$. Applying the Cauchy-Schwartz inequality and the definition of MMD-based mutual information yields

$$|J(f) - I(f)| \leq \|f\|_{RKHS} * \|\mu_{p_{X,Y}} - \mu_{p_X \otimes p_Y}\|_{RKHS} \leq \|f\|_{RKHS} * \underbrace{\mathbf{GMI}_{\text{MMD}}(X, Y)}_{\text{small or zero}} \quad (7.17)$$

Therefore, whether X and Y are strictly independent or only MMD-independent does not make any difference for integrating smooth functions in RKHS, because in both cases we will have $J(f) = I(f)$.

GMI for the Hashing Paradox. Recall the distributions $p_{X,Y}$ introduced for the hashing paradox (Section 7.2):

- Identity case: $X \sim U([0, 1])$ and $Y = \frac{\text{floor}(K*X)}{K} + \frac{W}{K}$. Variables X and Y appear more and more correlated, which is consistent with $\mathbf{MI}(X, Y) = \log K \rightarrow_{K \rightarrow \infty} \infty$.
- Hashing case: $X \sim U([0, 1])$ and $Y = \frac{\text{hash}_K(\text{floor}(K*X))}{K} + \frac{W}{K}$. For large K , variables X and Y appear visually more and more independent, which seems incompatible with $\mathbf{MI}(X, Y) = \log K \rightarrow_{K \rightarrow \infty} \infty$.
- Independent case: $X \sim U([0, 1])$ and $Y \sim U([0, 1])$ are independent and $\mathbf{MI}(X, Y) = 0$, which is consistent with the visual appearance.

[Figure 4.8] For these 3 distributions, we compute the usual KL-based mutual information analytically. We also compute their MMD-GMI with Gaussian kernels of bandwidths 0.01, 0.1, 10 using the unbiased U-estimator of MMD^2 proposed in Lemma 6 of [Gretton et al. \[2007\]](#). It turns out that MMD-GMI with Gaussian kernel and appropriate bandwidth *matches the visual intuition* well. For the hashing distribution, when the number of bins K is small, MMD-GMI increases with K as the bins get smaller and Y looks more determined by X , closely tracking the identity distribution curve, until the relationship $Y = h(X)$ becomes too irregular. However, for large enough K , the size of each bin becomes smaller than the bandwidth, at which point MMD-GMI goes to zero, and tracks the independent distribution curve.

From the previous result (7.17), it turns out that when $\sigma_K = \text{hash}$, the distribution $p_{X,Y}$ is fit for numerical integration of functions in the Gaussian RKHS (i.e., functions with only low-frequencies). However, it is not the case when σ_K is the identity

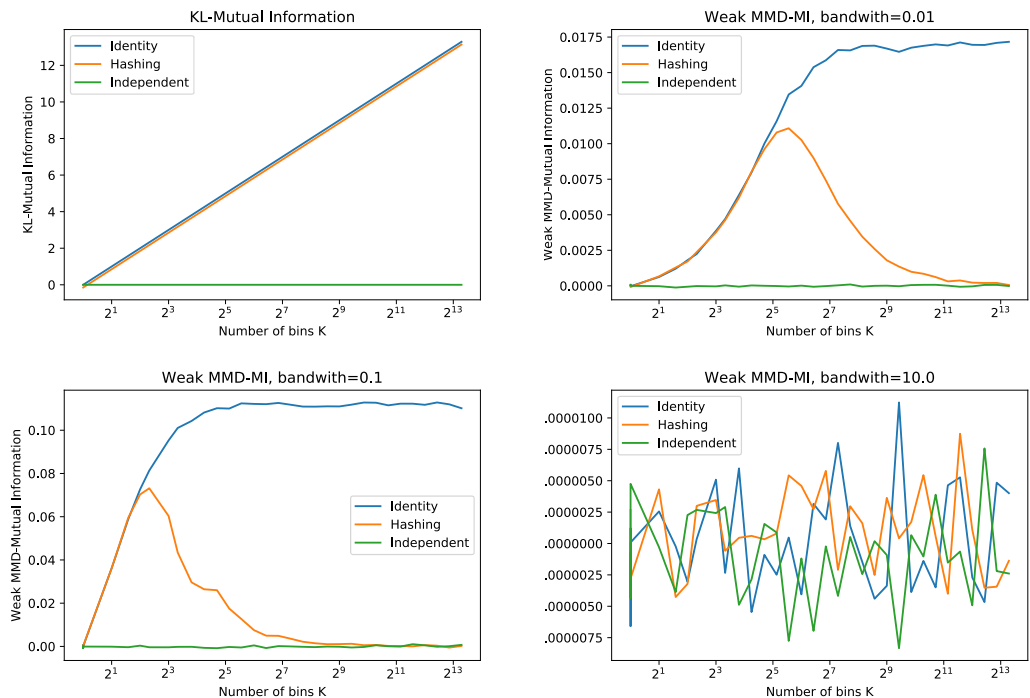


Figure 4.8: Various generalized mutual information (GMI) as a function of the number of bins K for the **Identity** (blue), **Hashing** (orange), and **Independent** (green) distributions. An intuitive notion of mutual information should go to zero for the hashing distribution (orange) as K increases. For smaller values of K , MMD-GMI with bandwidths 0.01 and 0.1 increases originally for the hashing distribution, closely following the identity distribution curve (blue), peaks for K roughly equal to half the bandwidth, and then goes to zero to closely follow the independent distribution curve (green). However, when the bandwidth is very large (e.g. 10), all distributions look the same, and MMD-GMI has no discriminative power.

function, and this can be simply shown by considering a function with support far from the diagonal $X = Y$.

7.6 Semi-Discrete Case

When \mathcal{Y} is discrete with few values (e.g. labels), it makes sense to try to explicitly predict $Y = h(X)$, instead of considering discriminators which take both variables as input. We show that KL-based parametric mutual information (KL-PMI) can be upper (lower) bounded simply by training a classifier to predict Y from X by minimizing the cross-entropy loss, and subsequently subtracting its training (validation) loss from the entropy of Y , which is straightforward to estimate for discrete variables with finite values.

Theorem 3 (Semi-Discrete Mutual Information). When Y is finite, the mutual information between X and Y can be written as the solution to the following optimization problem :

$$\mathbf{MI}(X, Y)(X, Y) = H(Y) - \inf_{T: \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}} \underbrace{\mathbf{E}_{x, y \sim p(x, y)} [-\log q(y|x)]}_{\mathcal{L}(T)} \quad (7.18)$$

where $H(Y)$ is the discrete entropy of the marginal Y and $\mathcal{L}(T)$ can be reinterpreted as the log-loss or cross-entropy loss between a candidate distribution $q(y|x)$ and $p(y|x)$. Specifically, we define a ‘‘predictor’’ $q(y|x) = \frac{p(y)e^{T(x, y)}}{\sum_{y'=1}^{|\mathcal{Y}|} p(y')e^{T(x, y')}}$, where $T(x, y)$ can be interpreted as scores for predicting y given x , and the inf can be interpreted as finding the score T which maximizes the conditional likelihood for the data from p .

Proof. We start from the KL form of mutual information and take conditional expectations using the factorization $p(x, y) = p(x)p(y|x)$

$$\mathbf{MI}(X, Y) = \mathbf{KL}(p(x, y) || p(x)p(y)) = \mathbf{E}_{x, y \sim p(x, y)} \left[\log \left(\frac{p(x, y)}{p(x)p(y)} \right) \right] \quad (7.19)$$

$$= \mathbf{E}_{x \sim p(x)} \mathbf{E}_{y \sim p(y|x)} \left[\log \frac{p(y|x)}{p(y)} \right] = \mathbf{E}_{x \sim p(x)} [\mathbf{KL}(p(y|x) || p(y))] \quad (7.20)$$

We take the Donsker-Varadhan [Belghazi et al., 2018] representation of $\mathbf{KL}(p(y|x) || p(y))$:

$$\mathbf{KL}(p(y|x) || p(y)) = \sup_{M: \mathcal{Y} \rightarrow \mathbb{R}} \mathbf{E}_{y \sim p(y|x)} [M(y)] - \log \mathbf{E}_{y \sim p(y)} [\exp(M(y))] \quad (7.21)$$

Injecting this expression in Equation (7.20) yields

$$\mathbf{MI}(X, Y) = \mathbf{E}_{x \sim p(x)} \mathbf{E}_{y \sim p(y|x)} \left[\log \frac{p(y|x)}{p(y)} \right] \quad (7.22)$$

$$= \mathbf{E}_{x \sim p(x)} \left[\sup_{M: \mathcal{Y} \rightarrow \mathbb{R}} \left\{ \mathbf{E}_{y \sim p(y|x)} [M(y)] - \log \mathbf{E}_{y \sim p(y)} [\exp(M(y))] \right\} \right] \quad (7.23)$$

The sup can be taken out of the expectation by considering all functions $T : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$,

$$\mathbf{MI}(X, Y) = \sup_{T: \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}} \mathbf{E}_{x \sim p(x)} \left[\mathbf{E}_{y \sim p(y|x)} [T(x, y)] - \log \mathbf{E}_{y \sim p(y)} [\exp(T(x, y))] \right] \quad (7.24)$$

Using the fact that \mathcal{Y} is discrete and finite, we rewrite the expectation in $p(y)$ as a sum :

$$\mathbf{MI}(X, Y) = \sup_{T: \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}} \left\{ \mathbf{E}_{x \sim p(x)} \mathbf{E}_{y \sim p(y|x)} \log \frac{e^{T(x, y)}}{\sum_{y'=1}^{|\mathcal{Y}|} p(y') e^{T(x, y')}} \right\} \quad (7.25)$$

We subtract the marginal entropy $H(Y) = \sum_{y'=1}^{|\mathcal{Y}|} p(y') \log p(y')$ which does not depend on T ,

$$\mathbf{MI}(X, Y) - H(Y) = \sup_{T: \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}} \mathbf{E}_{x \sim p(x)} \mathbf{E}_{y \sim p(y|x)} \log \frac{p(y) e^{T(x, y)}}{\underbrace{\sum_{y'=1}^{|\mathcal{Y}|} p(y') e^{T(x, y')}}_{q(y|x)}} \quad (7.26)$$

We recognize a weighted softmax over the $T(x, y)$, which we denote $q(y|x)$. After re-arranging, we get :

$$\mathbf{MI}(X, Y) = H(Y) - \inf_{T: \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}} \mathbf{E}_{x, y \sim p(x, y)} [-\log q(y|x)] \quad (7.27)$$

■

We can modify Equation (7.18) to define semi-discrete parametric mutual information by restricting the function T to be in a parametric family \mathcal{T} .

Definition 7.5 (Semi-Discrete Parametric-KL Mutual Information). We define the semi-discrete parametric-KL mutual information (KL-SDPMI) as follows:

$$\mathbf{SDPMI}(X, Y) = H(Y) - \inf_{T \in \mathcal{T}} \underbrace{\mathbf{E}_{x, y \sim p(x, y)} [-\log q(y|x)]}_{\mathcal{L}(T)} \quad (7.28)$$

where $q(y|x) = \frac{p(y)e^{T(x,y)}}{\sum_{y'=1}^{|Y|} p(y')e^{T(x,y'')}}$ and \mathcal{T} is a parametric family of functions, such as the neural networks with a given architecture. Although it might be possible to extend the semi-discrete formulation to other divergences, we focus on KL due to its interpretability.

Since we recognize a negative log-likelihood loss, we propose the following approach for estimating lower and upper bounds for SDPMI. We independently sample a training set $D_{train} = \{(x_{train}^{(i)}, y_{train}^{(i)}) \sim p_{x,y}\}$ and a validation set $D_{val} = \{(x_{val}^{(i)}, y_{val}^{(i)}) \sim p_{x,y}\}$ from the joint distribution $p_{x,y}$. Then, we train a neural network $q(y|x)$ to predict y from x on the D_{train} , and compute the training and validation losses $\mathcal{L}_{train}, L_{val}$.

Theorem 4 (KL-SDPMI Bounds). We can bound the SDPMI using the training and validation SDPMI :

$$\underbrace{H(Y) - \mathbf{E}_{D_{train}, D_{val}}[\mathcal{L}_{val}(\hat{T})]}_{\text{Validation SDPMI}} \leq \text{SDPMI}(X, Y) \leq \underbrace{H(Y) - \mathbf{E}_{D_{train}}[\mathcal{L}_{train}(\hat{T})]}_{\text{Training SDPMI}} \quad (7.29)$$

where $\hat{T} \in \mathcal{T}$ is the minimizer of the training loss $\mathcal{L}_{train}(T) = -\sum_{i=1}^N \log \frac{p(y_{train}^{(i)})e^{T(x_{train}^{(i)}, y_{train}^{(i)})}}{\sum_{y'=1}^{|Y|} p(y')e^{T(x_{train}^{(i)}, y')}}$, $\mathcal{L}_{val}(\hat{T}) = -\sum_{i=1}^{N'} \log \frac{p(y_{val}^{(i)})e^{T(x_{val}^{(i)}, y_{val}^{(i)})}}{\sum_{y'=1}^{|Y|} p(y')e^{T(x_{val}^{(i)}, y')}}$ is the validation loss, and the expectations are over the sampling of the training and validation sets. Moreover, the (nonparametric) mutual information can be lower bounded using the validation loss:

$$H(Y) - \mathbf{E}_{D_{train}, D_{val}}[\mathcal{L}_{val}(\hat{T})] \leq \text{SDPMI}(X, Y) \leq \text{MI}(X, Y) \quad (7.30)$$

Proof. The first inequality comes from the fact that for supervised learning, the training loss (validation loss) is a lower bound (upper bound) of the population loss in expectation. We make use of this inequality to bound the SDPMI in the experiments (Section 7.7). The second inequality comes from the fact that \mathcal{T} is a subset of the functions $\mathcal{X} \rightarrow \mathcal{Y}$. ■

7.7 GMI for Corrupted Label Datasets (Experiments)

We define a range of distributions q_α from less to more corrupted by corrupting the labels of MNIST and SVHN. Specifically, we replace the ground-truth label distribution by taking a mixture between the ground truth (assumed to be deterministic) and uniform distribution over all classes

$$q_\alpha(y|x) = (1 - \alpha)p(y|x) + \frac{\alpha}{K}$$

where $K = 10$ is the number of classes and we take $\alpha \in \{0, 0.25, 0.5, 0.75, 1\}$. For $\alpha = 0$ we recover the original distribution with deterministic labels, while for $\alpha = 1$ we recover the random-label distribution presented in Section 7.1, with labels independent of the image. In practice, we simulate the sampling of the training and test set by corrupting a random fraction α of the ground-truth labels. Because each image x occurs only once in the dataset, it is not obvious that some labels are non-deterministic (corrupted).

Can we define generalized mutual information which capture the fact that the labels are increasingly corrupted? An ideal notion of mutual information should vary *smoothly* and *perceptibly* with respect to α . In **Figure 4.9**, we compute and plot various GMI as a function of the degree of corruption:

- **Mutual Information (Analytical)**. For the specific q_α considered, the true (population) mutual information can be computed analytically and equals $\mathbf{MI}_\alpha(X, Y) = \log K + \log(1 - (1 - \frac{1}{K}) * \alpha)$, which is a smooth function that gradually transitions from $\log K$ to 0, and captures the fact that labels are more and more random.
- **Mutual Information (Empirical)**. On the “reference easy” problem (see caption of Figure 4.9), the empirical mutual information approximates the population mutual information closely because each “image” appears many times, since there is only one possible image per class. However, for corrupted MNIST and SVHN, each image appears only once (the odds that the same image appears twice is zero for continuous distributions) and each label appears to be fully determined by the associated image. As a consequence, the empirical mutual information is constant and equal to $\log K$, and fails to capture the fact that labels are increasingly corrupted.
- **MMD-GMI (Empirical)**. We compute $\mathbf{MMD}_K(p_{X,Y}, p_X \otimes p_Y)$ with kernel $K(x, x', y, y') = \exp(-\|x - x'\|^2/d) \exp(-1_{y \neq y'})$ over the training set, where d is the number of dimensions. On the reference tasks, MMD-GMI (purple) varies smoothly with respect to the amount of corruption. However, on corrupted MNIST, MMD-GMI is always close to zero and only slightly sensitive to α , which means it considers the label to be mostly independent from the data. This effect is even more obvious for corrupted SVHN. This is consistent with the observation that for generic kernels MMD fails to separate distributions in high-dimensions (Section 4.1).
- **Wasserstein-GMI (Empirical)**. Using the Sinkhorn algorithm [Cuturi, 2013], we estimate $\mathbf{W}_d(p_{X,Y}, p_X \otimes p_Y)$ with base-distance $d((x, y), (x', y')) = \|x - x'\|/\sqrt{d} + 1_{y \neq y'}$ over the training set, where d is the number of dimensions. On the reference tasks, Wasserstein-GMI (pink) varies smoothly with respect to the amount of corruption. However, for corrupted MNIST, Wasserstein-GMI is only slightly sensitive to α , and seems to be heavily biased, as even

Name	Underlying divergence	Function class	Upper Bound	Lower Bound
MI	KL	Nonparametric	Empirical KL	-
MMD-GMI	MMD	RKHS (Gaussian Kernel)	Empirical MMD	-
Wasserstein-GMI	Wasserstein-1	L -Lipschitz functions	Empirical Wasserstein	-
SDPMI-Linear	KL	Linear	Training SDPMI	Validation SDPMI
SDPMI-RF	KL	Random Forest	Training SDPMI	Validation SDPMI
SDPMI-CNN	KL	CNN	Training SDPMI	Validation SDPMI

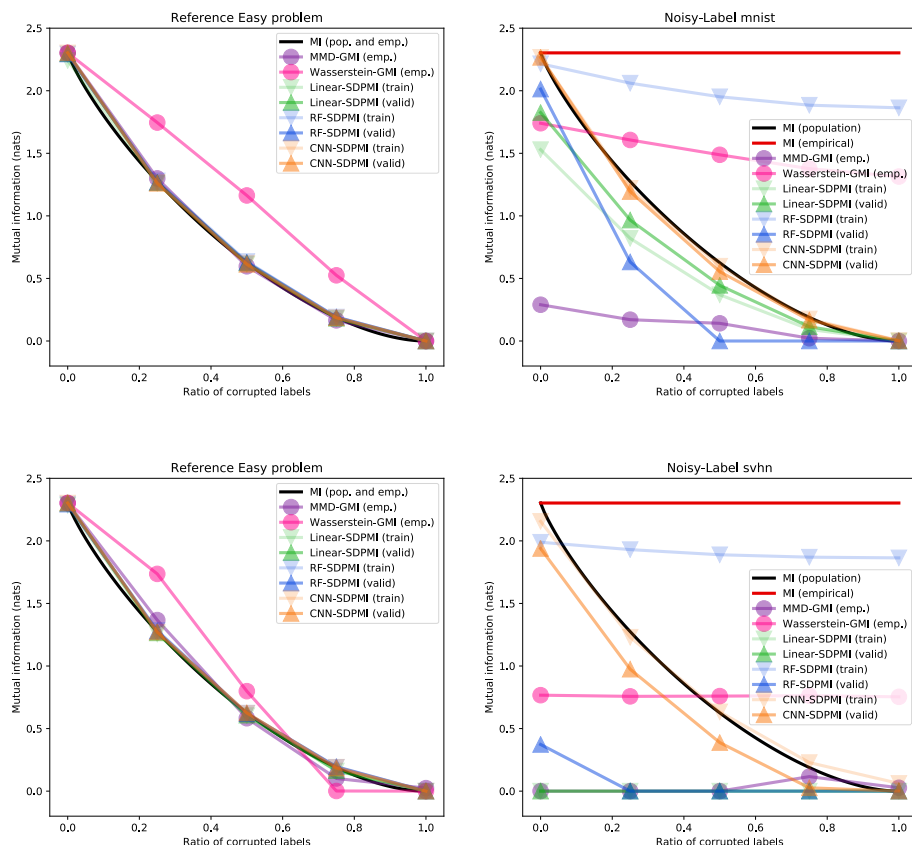


Figure 4.9: Estimating various generalized mutual information between image and label on MNIST (**middle row right column**) and SVHN (**bottom row right column**) as a function of α . For sanity check and scaling purposes, we also estimate generalized mutual information on reference easy tasks of the same dimensionality (left column plots). The mutual information and the semi-discrete parametric mutual information (with linear, random forest and CNN predictors) are expressed in nats. MMD-GMI and Wasserstein-GMI cannot be directly compared to KL-MI because they have different scales. In order to calibrate the various GMI to each other, we define a *reference* easy task from corrupted MNIST (resp. SVHN), except that the pixels of each flattened image are partitioned into K equal contiguous groups and set to 0 except for the group which index corresponds to the ground-truth label. MMD-GMI and Wasserstein-GMI curves are rescaled such that the maximum value reaches $\log(10) \approx 2.3$ on the reference task.

the independent case $\alpha = 1$ gets assigned a high Wasserstein-GMI. Again, the effect is more obvious for corrupted SVHN, where Wasserstein-GMI does not vary with respect to α . This is likely due to the poor sample complexity of the empirical Wasserstein distance, as discussed in Section 4.1.

- **KL-SDPMI (Training and Validation)**. We compute the semi-discrete parametric mutual information based on linear (logistic), random-forest, and CNN classifiers, and use Equation (7.30) (without the expectations) for estimating upper (training) and lower (validation) bounds for the Semi-Discrete Parametric KL-MI. For the reference tasks, all three Parametric-MI closely approximate the True MI with no apparent gap between training and validation MI. The random-forest Parametric-MI has a large training-validation gap on corrupted MNIST and even more on corrupted SVHN, which indicates severe overfitting. The logistic Parametric-MI varies smoothly with respect to α on corrupted MNIST, although it slightly underestimates the mutual information (compared to the reference problem). However, on corrupted SVHN, the logistic parametric-MI is not flexible enough to capture the dependence and equals zero for all values of α . The CNN Parametric-MI varies gradually with respect to α both for corrupted MNIST and SVHN, with a noticeable but limited gap between train and test-MI. In this case, the CNN Parametric-MI is the most intuitive notion of mutual information.

All divergences are equally informative in the reference easy cases, which act both as a sanity check and a way to calibrate them. For the harder corrupted MNIST, Logistic and CNN-based SDPMI are the most informative, most likely because they make decent implicit assumptions on the distribution (linear separability, convolutional prior). MMD-GMI and Wasserstein-GMI are not as informative but still somewhat sensitive to α , which is intuitive given that they are based pixel-wise metric on the images, which are still a little relevant in the MNIST case. For the hardest noisy-label SVHN case, only the CNN-SDPMI remains informative. The Linear-SPBMI has no sensitivity to α because SVHN is not as linearly separable (as MNIST). MMD-GMI and Wasserstein-GMI fail completely as pixel-wise distances are no longer meaningful for SVHN. Random-forest is a terrible estimator for MNIST and SVHN and overfits severely (large training-validation gap) due to its inefficient priors (rule-based classification where each rule is a threshold on a single pixel value).

8 Related Work

Closest to our work are the following two papers. Arora et al. [2017] argue that analyzing GANs with a nonparametric (optimal discriminator) view does not really make sense, because the usual nonparametric divergences considered have bad

sample complexity. They also prove sample complexities for parametric divergences. [Liu et al. \[2017\]](#) prove under some conditions that globally minimizing a neural divergence is equivalent to matching all moments that can be represented within the discriminator family. They unify parametric divergences with nonparametric divergences and introduce the notions of strong and weak divergence. However neither of these works focuses on the meaning and practical properties of parametric divergences, as we do here, regarding their suitability for a final task, and paralleling similar questions studied in structured prediction.

Throughout this paper, we have also used several results from the literature to discuss the properties of parametric divergences. Before the first GAN paper, [Sriperumbudur et al. \[2012\]](#) unified traditional Integral Probability Metrics (IPM), analyzed their statistical properties, and proposed to view them as classification problems. Similarly, [Reid and Williamson \[2011\]](#) show that computing a divergence can be formulated as a classification problem. Later, [Nowozin et al. \[2016\]](#) generalize the GAN objective to any adversarial f-divergence. However, the first papers to actually study the effect of restricting the discriminator to be a neural network instead of any function are the MMD-GAN papers: [Li et al. \[2015\]](#), [Dziugaite et al. \[2015\]](#), [Li et al. \[2017\]](#), [Mroueh et al. \[2017\]](#) and [Bellemare et al. \[2017\]](#) who give an interpretation of their energy distance framework in terms of moment matching. [Mohamed and Lakshminarayanan \[2016\]](#) give many interpretations of generative modeling, including moment-matching, divergence minimization, and density ratio matching. On the other hand, work has been done to better understand the GAN objective in order to improve its stability [[Salimans et al., 2016](#)]. Subsequently, [Arjovsky et al. \[2017\]](#) introduce the adversarial Wasserstein distance which makes training much more stable, and [Gulrajani et al. \[2017\]](#) improve the objective to make it more practical. Regarding model evaluation, [Theis et al. \[2016\]](#) contains an excellent discussion on the evaluation of generative models, they show in particular that log-likelihood is not a good proxy for the visual quality of samples. [Danihelka et al. \[2017\]](#) compare parametric adversarial divergence and likelihood objectives in the special case of RealNVP, a generator with explicit density, and obtain better visual results with the adversarial divergence. [Belghazi et al. \[2018\]](#) propose to use neural networks to estimate mutual information; we propose an alternative formulation in the semi-discrete case and highlight its distinct properties on some toy distributions. Concerning theoretical understanding of learning in structured prediction, several recent papers are devoted to theoretical understanding of structured prediction such as [Cortes et al. \[2016\]](#) and [London et al. \[2016\]](#) which propose generalization error bounds in the same vein as [Osokin et al. \[2017\]](#) but with data dependencies.

Our perspective on generative modeling is novel because we ground it on the notion of final task – what we ultimately care about – and highlight the multiple reasons why parametric divergences offer a superior framework to define good task losses

with respect to a final task; in essence, they provide a more effective and meaningful training signal. We also perform experiments to determine properties of some parametric divergences, such as invariance/robustness, ability to enforce constraints and properties of interest, as well as the difference with their nonparametric counterparts. To the best of our knowledge, this is the first work that links the task loss generalization error of structured prediction and the adversarial divergences used in generative modeling.

9 Conclusion

We have shown that parametric adversarial divergences, a generalization of the GAN loss, are not merely lower bounds of nonparametric divergences, but instead have distinct properties which makes them favorable for high dimensional generative modeling. Among these properties, parametric divergences can scale up to high-dimensional data, and can be tuned to be sensitive to specific aspects of the target distribution. We have also explored using parametric divergences to define more meaningful notions of mutual information. An important area of improvement that remains is to compute parametric divergences reliably enough for using them as practical evaluation metrics for generative models.

Acknowledgements

This research was partially supported by the Canada Excellence Research Chair in “Data Science for Real-time Decision-making”, by the NSERC Discovery Grant RGPIN2017-06936 and by a Google Research Award. These funds were all received and administered by University of Montreal.

5

Prologue to Second Contribution

1 Article Details

Are Few-Shot Learning Benchmarks too Simple? Solving them without Test-Time Labels by *Gabriel Huang, Hugo Larochelle, Simon Lacoste-Julien*. This work was accepted as an ICLR 2019 workshop paper; a full version was submitted to ICLR 2020, ICML 2020 and NeurIPS 2020; this project was also presented at the Montreal AI Symposium 2020.

2 Contributions of the Authors

Gabriel Huang proposed the original idea, wrote the paper, and ran the experiments. Hugo Larochelle provided supervision and compute credits, helped survey the few-shot literature, and helped apply the method to the Meta-Dataset. Simon Lacoste-Julien provided supervision, funding, and helped formalize the experimental setting.

3 Context and Limitations

This project stemmed from the observation that popular few-shot classification benchmarks such as Omniglot and *miniImageNet* do not evaluate generalizing to new class semantics—e.g. *colors* vs *shapes*—but only to new classes with same semantics—e.g. *characters* for Omniglot, *object categories* for *miniImageNet*. We proposed a baseline, *Centroid Networks*, which ignores labels from the support set at test-time, and instead clusters the learned representations to recover the target classes (up to permutation). (1) Centroid Networks is useful for gauging the difficulty of few-shot classification benchmarks, but we stopped one step short of numerically quantifying the diversity of class semantics. Under our framework, doing so would require taking the difference between two lower bounds (say, the CentroidNet vs. ProtoNet performance) which is unreliable. Instead the provided numbers can be interpreted as the minimum performance to expect below which test-time labels are not even required. (2) The proposed Sinkhorn K-Means method

is not applicable to general few-shot clustering because we assume that the classes are uniformly distributed. However, it can have applications in quantization and self-supervised learning [Caron et al., 2020], where the uniform constraint can be desirable.

4 Recent developments

Since we started working on this project, harder few-shot image classification benchmarks have been proposed, featuring diverse class semantics and requiring skills such as concept learning, compositionality, and abstract reasoning. *Bongard-LOGO* [Nie et al., 2020], a procedurally-generated image classification benchmark, features diverse class semantics such as stroke type, convexity, shape category, generative process, and symmetry. Another challenging benchmark is the *Procedurally Generated Matrices* (PGM) dataset [Barrett et al., 2018] which is inspired by Raven’s Progressive Matrices [Raven, 1941], and consists in picking the most “logical” answer to complete a matrix of images, in a similar fashion to human IQ tests. Kandinsky Patterns [Müller and Holzinger, 2021] are another benchmark consisting of images with one to many colored shapes. Class semantics are defined in natural language, and can involve intrinsic concepts (shape, color), existence, counting, comparison, spatial arrangement, or even emerging patterns. More recently, the *Compositional Reasoning Under Uncertainty* (CURI) benchmark [Vedantam et al., 2021] was proposed, featuring compositionality, relational concepts and concept uncertainty. Class semantics involve intrinsic properties (color, shape, material), extrinsic properties (object location), boolean operators, counting, and more.

Beyond image few-shot classification, researchers have also explored other modalities and the zero-shot setting. Recent advances in language modeling, such as GPT-3 [Brown et al., 2020], FLAN [Wei et al., 2021b], T0 Sanh et al. [2021] and PaLM [Chowdhery et al., 2022] have shown that large language models (i.e. ≥ 10 B parameters) are zero-shot and few-shot learners which can solve new and challenging tasks such as *Big Bench*¹, described in text form. Multimodal approaches have also achieved impressive results. The CLIP model [Radford et al., 2021], which is trained on 400M image-text pairs, can leverage the expressivity of natural language to do zero-shot classification of new categories. More recently, Alayrac et al. [2022] introduced Flamingo, a vision-language model from pretrained frozen unimodal vision and language models, capable of solving object and action classification, scene/event understanding, visual question answering, on image and video, for the zero and few-shot setting.

¹<https://github.com/google/BIG-bench>

The meaning of zero-shot and few-shot learning is constantly evolving, far beyond image classification. Apart from scaling models up, a constant trend seems to be in multimodal approaches which leverage the expressivity and implicit knowledge contained in natural language.

6

Solving Few-Shot Learning Benchmarks without Test-Time Labels

Abstract

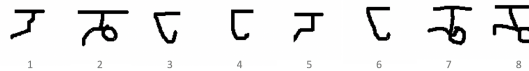
We show that several popular few-shot learning benchmarks can be solved with varying degrees of success without using support set Labels at Test-time (LT). To this end, we introduce a new baseline called *Centroid Networks*, a modification of Prototypical Networks in which the support set labels are hidden from the method at test-time and have to be recovered through clustering. A benchmark that can be solved perfectly without LT does not require proper task adaptation and is therefore inadequate for evaluating few-shot methods. In practice, most benchmarks cannot be solved perfectly without LT, but running our baseline on any new combinations of architectures and datasets gives insights on the baseline performance to be expected from leveraging a good representation, before any adaptation to the test-time labels.

1 Introduction

Supervised few-shot classification, sometimes simply called few-shot learning, consists in learning a method that can adapt to new classification tasks from a small number of examples. Being able to learn new classes from a small number of labeled examples is desirable from a practical perspective because it removes the need for the end-user to label large datasets. Instead, a central organization with access to large generic datasets could “pre-train” the method (training phase) so that when it is shipped to end-users, each user only needs small amounts of labeled data to adapt the method on its own classification task (testing phase). Supervised few-shot classification is typically formulated as a distribution $P(T)$ of classification tasks, also called episodes, which are split into training, validation, and testing sets. Each episode comes with two small sets of labeled examples called the *support* and *query* sets. The goal is to learn a classifier that can learn (task adaptation) from the task-specific support set (X_S, Y_S) and classify the query set (X_Q, Y_Q) with maximum accuracy, despite limited training data. Typically, this is achieved by training the model on a large number of training episodes beforehand. The Omniglot [Lake et al., 2011] and *mini*ImageNet [Vinyals et al., 2016a] benchmarks have been heavily used to evaluate and compare supervised few-shot classification methods in the last

few years [Vinyals et al., 2016a, Ravi and Larochelle, 2016, Snell et al., 2017a, Finn et al., 2017, Sung et al., 2018b].

An important question is whether those benchmarks are actually evaluating the task adaptation capabilities of few-shot methods, or rather something else. Consider the following (transductive) classification task, extracted from the “Mongolian” alphabet of Omniglot:



It is relatively easy to solve the task even without being familiar with “Mongolian” characters because we are already familiar with the “group by character” task.¹ We can solve the task without receiving a single labeled example of the new classes, that is, without using a support set. Conversely, there are task distributions that do require learning from a support set. For instance, consider this other transductive classification task :



The task is now ambiguous because there are many possible semantics (by shape, color, or border style). In this case, we would need a support set to learn (task adaptation) which criterion should be used. Proper task adaptation requires using the support set labels at test-time. Therefore, one way to investigate which few-shot benchmarks require proper task adaptation is to ask *which benchmarks can be solved without using support set labels at test-time ?*

Definition (LT vs. NLT methods). We say that a few-shot classification method is **LT** if it uses the support set **L**abels at **T**est-time. Conversely, we say that a few-shot classification method is **NLT** if it uses **No L**abels at **T**est-time. Both **LT** and **NLT** methods are allowed to use labels during training. Note that all usual few-shot methods [Vinyals et al., 2016a, Finn et al., 2017, Snell et al., 2017a] are **LT** methods

Given this definition, we want to know if we can reach high performance on a benchmark with NLT methods. In particular, benchmarks that can be solved by NLT methods are definitely too “simple” and do not require proper task adaptation, in the sense of using test-time labels. We have reasons to suspect that NLT methods can reach high performance on Omniglot and miniImageNet because their class semantics are invariant across different episodes (Omniglot classes are always alphabet characters, *miniImageNet* classes are always object categories as defined by the WordNet taxonomy [Miller, 1995, Russakovsky et al., 2015]), and because the classes are randomly split between training and testing (there is no domain

¹Solution: there are 3 classes {1, 5}, {2, 7, 8}, {3, 4, 6}.

shift).

We introduce a new baseline called *Centroid Networks* (or CentroidNet), which is an NLT modification of Prototypical Networks in which the support set labels are hidden from the method and recovered through clustering. We attempt to solve several popular few-shot benchmarks using our NLT baseline and report the resulting accuracies. Those numbers can be interpreted as *the baseline performance to be expected from leveraging a good representation, before any LT-based task adaptation takes place*. Therefore, we recommend running our NLT baseline on any published combinations of architectures and datasets in order to disentangle the fraction of the performance can be attributed to the architecture/representation versus its ability to learn from new labels.

Our contributions are the following :

- We propose a simple NLT baseline for solving few-shot tasks without support set labels at test-time. This baseline helps determine how much performance is to be expected from leveraging a good representation without learning from task-specific labels at test-time.
- We show that Centroid Networks achieve 99.1% and 98.1% NLT accuracies for Omniglot 5-way/5-shot and 20-way/5-shot, effectively showing that Omniglot can be solved without LT.
- We report NLT accuracies of our baseline on several other popular benchmarks, giving an idea of the performance to be expected on any (dataset, architecture) combination from leveraging a good representation before any LT-based task adaptation takes place. We observe that support set labels are much more critical for cross-domain benchmarks.

Finally, we also explore applying our method to few-shot clustering and (LT) transductive few-shot learning.

2 Related Work

Simple Baselines. [Chen et al. \[2019a\]](#) propose a simple baseline for LT few-shot classification, and show that even simple baselines can solve few-shot benchmarks with good performance when combined with the right architecture. Our NLT baseline is even simpler in the sense that we neither use LT or change the representation at test-time.

Task Overfitting vs. Task Generalization. [Yin et al. \[2019\]](#) introduce the notion of non-mutually exclusive tasks, which is a family of tasks that can be solved by a single function. They argue that non-mutually exclusive tasks are not diverse

enough and that few-shot classification methods are at risk of “task overfitting”. They propose a regularization technique to prevent task overfitting. In our case, we argue that even the original (mutually-exclusive) Omniglot and *miniImageNet* do not have sufficient task diversity in terms of class semantics. Our motivation is different and is to show that one can memorize the class semantics of the training set and still achieve high accuracy on the test set, thereby making those benchmarks too “simple”.

Task Adaptation vs. Feature Reuse. [Raghu et al. \[2019\]](#) investigate whether the performance of MAML [[Finn et al., 2017](#)] comes from rapid adaptation to new tasks or from reusing good features. They propose a version of MAML in which the features are kept constant in the inner loop, without any loss in performance. Our work shares the same motivation in showing that the performance of meta-learning algorithms might come more from using good universal features rather than doing task adaptation, where we mean “without using LT” and they mean “without adapting the features”. We take it one step further than fixing the features by proposing a NLT baseline.

Meta-Learning and Semi/Un-supervised Learning. Some recent work has explored combinations of unsupervised learning and meta-learning, to address various other tasks. [Metz et al. \[2018\]](#) meta-learn an unsupervised representation update rule that produces useful features for LT few-shot learning. Similarly, [Hsu et al. \[2018\]](#), [Khodadadeh et al. \[2018\]](#) learn a LT method using no labels a training time. Thus, all these works consider the opposite setting from us : they use no labels at training-time to learn a LT method, while we use labels at training time to learn a NLT method. Our work is also related to Semi-Supervised Prototypical Networks [[Ren et al., 2018](#)], in which the support set contains both labeled and unlabeled examples. In a sense, we go beyond their work by requiring no labels at all to infer centroids at test-time.

Zero-Shot Learning. Strictly speaking, *true* zero-shot learning would mean solving the few-shot classification problem without using the support set at all. Centroid Networks are related to true zero-shot learning because they do not use the labels of the support set. However, our method uses the images and some partial information (Section 3) and therefore cannot be considered pure zero-shot learning. In practice, zero-shot methods also have access to partial information in the form of text descriptions of the new classes [[Socher et al., 2013](#), [Romera-Paredes and Torr, 2015](#)].

Our work is also related to the supervised clustering, learning to cluster, and other clustering-related literature. We have moved them to Appendix 2 due to a lack of space and because learning-to-cluster is not the central contribution of this paper.

3 Few-shot Tasks and Evaluation

We give some background on meta-learning and standard (LT) few-shot classification. Then, we will introduce NLT few-shot classification, which formalizes what it means to *solve few-shot classification without test-time labels*.

Meta-learning consists in learning a method that can solve a distribution of tasks or episodes $P(T)$ well, with respect to some external evaluation metric (e.g. accuracy). Typically, independently sampled tasks from $P(T)$ are split into training, validation, and testing sets, although the meta-learning tasks generally contain additional supervision as we discuss below. In this framework, an algorithm is first trained to solve the training tasks (train-time), then it is evaluated on the validation/testing tasks (test-time).

LT few-shot classification. Each episode comes with a small *support* set $S = (X_S, Y_S)$ and a small *query* set $Q = (X_Q, Y_Q)$, where X_S, X_Q denote images or data, and Y_S, Y_Q their labels. The task is to predict labels \hat{Y}_Q for the query images X_Q and the learner has access to the task-specific images X_S and labels Y_S for task adaptation. Finally, the classification accuracy is computed between \hat{Y}_Q and Y_Q . We denote it **LT-accuracy** to differentiate it from the metrics of other tasks.

NLT few-shot classification. The NLT setting is the same as its LT counterpart except that we hide the support set labels at test-time. Before making predictions, we need to introduce an additional step in which the learner attempts to recover Y_S by clustering X_S into K clusters. Denote $\hat{C}_S^{(i)} \in [1, K]$ the cluster index for the i -th example $X_S^{(i)}$. Because the cluster indices are only defined up to permutation of the values, we use the Hungarian algorithm² to find the permutation $\sigma : [1, K] \rightarrow [1, K]$ that maximizes the accuracy $\sum_i \mathbf{1}_{\{\sigma(\hat{C}_S^{(i)}) \neq Y_S^{(i)}\}}$ on the support set, and denote $\hat{Y}_S^{(i)} = \sigma(Y_S^{(i)})$ the optimal indices. The final task is to predict labels \hat{Y}_Q for X_Q as usual, after learning from the reconstructed support set (X_S, \hat{Y}_S) . The predictions \hat{Y}_Q are compared with the ground-truth Y_Q . We denote the resulting accuracy **NLT-accuracy**. Note that any LT-method can be made into a NLT-method by combining it with a clustering algorithm. This is exactly the approach of our NLT baseline, in which ProtoNet is combined with Sinkhorn K-means.

Note that this section is about the evaluation tasks (the testing tasks) and does not put restrictions on the usable labels during training. Typically, standard LT few-shot classification methods have access to Y_Q during training but Y_Q is reserved for external evaluation during testing. In the same spirit, NLT few-shot classification methods are allowed to use Y_S and Y_Q during training, but both Y_S and Y_Q are

²https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linear_sum_assignment.html

reserved for external evaluation at testing time.

4 Centroid Networks

We mentioned previously that any LT method can be made into a NLT method by combining it with a clustering step at test-time (Section 3). Our NLT baseline, *Centroid Networks* or *CentroidNet*, is a combination of ProtoNet and Sinkhorn K-Means, which is used to recover the support set labels at test-time.

4.1 Prototypical Networks

Prototypical Networks or ProtoNets [Snell et al., 2017a] is one of the simplest supervised few-shot classification methods, and yet it has been shown to be competitive with more complex methods when combined with ResNet backbones [Ye et al., 2020]. The only learnable component of ProtoNets is the (backbone) embedding function $h_\theta : \mathcal{X} \rightarrow \mathcal{Z}$ which maps images to an embedding (feature) space. Given an episode, ProtoNet takes the support set images, maps them to the feature space, and computes the average (the prototype) of each class $\mu_j = \frac{1}{M} \sum_i h_\theta(x_i^s) * \mathbb{1}\{y_i^s = j\}$. Each point from the query set is then classified according to a soft nearest-neighbor scheme $p_\theta(y = j|x) = \text{softmax}(\|h_\theta(x) - \mu_j\|^2)$. ProtoNets are trained end-to-end by minimizing the classification cross-entropy on the query set.

4.2 Sinkhorn K-Means

We propose *Sinkhorn K-Means* as the clustering module of Centroid Networks. The idea of Sinkhorn K-Means has been mentioned sporadically in the literature (see Appendix 2) but to the best of our knowledge the algorithm has never been explicitly described or applied to few-shot learning before.

Sinkhorn K-Means takes as input a set of N points $x \in \mathbb{R}^{n \times d}$ (in our case, ProtoNet embeddings) and outputs a set of K centroids $c_j \in \mathbb{R}^{k \times d}$ with data-centroid soft assignments $p \in \mathbb{R}^{n \times k}$. We start by initializing the centroids around zero with a small amount of Gaussian noise to break symmetries. Then, we attempt to find the centroids that minimize the Sinkhorn distance [Cuturi and Doucet, 2014] between the empirical distributions defined by the data $p(x) = \frac{1}{N} \sum_{i=1}^N \delta(x - x_i)$ and the centroids $q(x) = \frac{1}{K} \sum_{j=1}^k \delta(x - c_j)$. To do so, we alternatively compute the optimal transport plan between p and q using Sinkhorn distances (Algorithm 1) and update each centroid to the weighted average of its assigned data points. For simplicity, Algorithm 2 describes the procedure in the case where clusters are balanced. When the clusters are not balanced but the cluster weights are known (e.g. Meta-Dataset),

Algorithm 1 Sinkhorn(x, c, γ) for empirical distributions.

Input: data $(x_i)_{1 \leq i \leq n} \in \mathbb{R}^{n \times d}$, centroids $(c_j)_{1 \leq j \leq k} \in \mathbb{R}^{k \times d}$, regularization constant $\gamma > 0$.

Output: optimal transport plan $(p_{i,j}) \in \mathbb{R}^{n \times k}$.

$K_{i,j} \leftarrow \exp(-\|x_i - c_j\|_2^2 / \gamma) \in \mathbb{R}^{n \times d}$

$R_i \leftarrow 1/n \quad 1 \leq i \leq n$

$v_j \leftarrow 1, C_j \leftarrow 1/k \quad 1 \leq j \leq d$

while not converged **do**

$u_i \leftarrow R_i / (\sum_{j=1}^k K_{i,j} v_j), \quad 1 \leq i \leq n$

$v_j \leftarrow C_j / (\sum_{i=1}^n K_{i,j} u_i), \quad 1 \leq j \leq k$

end while

$p_{i,j} \leftarrow u_i K_{i,j} v_j, \quad 1 \leq i \leq n, 1 \leq j \leq k$

return assignments p

Algorithm 2 Sinkhorn K-Means(x, c, γ)

Input: data $(x_i)_{1 \leq i \leq N}$, initial centroids $(c_j)_{1 \leq j \leq K}$, regularization constant $\gamma > 0$.

Output: final centroids $(c_j)_{1 \leq j \leq K}$, optimal assignment $(p_{i,j}) \in \mathbb{R}^{N \times K}$.

while not converged **do**

$(p_{i,j}) \leftarrow \text{Sinkhorn}(x, c, \gamma)$

$c_j \leftarrow k \sum_{i=1}^n p_{i,j} x_i, \quad 1 \leq j \leq k$

end while

return centroids c , assignments

p .

the weights can be taken into account by the Sinkhorn distance. All details can be found in our code.

Sinkhorn K-Means can be seen as a version of K-Means where the greedy nearest-centroid hard assignment (expectation step) is replaced with a global regularized optimal transport soft-assignment. More discussions about the Sinkhorn distance, the differences between Sinkhorn/Regular K-Means, and an empirical comparison are given in Sections 3 and 6 of the Appendix.

4.3 Combining ProtoNet with Sinkhorn K-Means

For training, we teach ProtoNet to solve LT few-shot classification tasks (regular ProtoNet training). For testing, we combined Sinkhorn K-Means as described below.

Training (all tasks). Training is the same regardless of the evaluation task. Using standard training [Snell et al., 2017a], we fit the ProtoNet backbone to solve LT few-shot classification. Given a training episode, we embed the support set, compute the prototypes, make predictions on the query set, compute the cross-entropy loss after revealing the query labels, and minimize it with gradient descent with respect to the parameters of the backbone. In some cases, we have found helpful to use an additional center loss [Wen et al., 2016] in order to pull the embeddings of the same class together. This is discussed in the ablation study in

Appendix 6.

Testing on NLT few-shot classification. Given the support and query images X_S, X_Q , the number of classes K and number of shots, we embed the support images and compute centroids (c_j) and the optimal transport plan ($p_{i,j}$) using Sinkhorn K-Means. To get hard assignments, we can either classify query points according to their nearest centroid (*softmax assignments*), or return their majority assigned centroid $\operatorname{argmax}_j p_{i,j}$ (*sinkhorn assignments*). According to the ablation study, the choice of assignment strategy has little effect on the performance (Appendix 6).

5 Experiments

We start with our main results for NLT few-shot classification (Section 5.1). We then apply our method to cross-domain benchmarks, which we expect to be harder without test-time labels (Section 5.2). We also explore applying our method in other few-shot settings such as few-shot clustering with partial information and transductive few-shot classification (Section 5.3). Details about the implementation and datasets can be found in Appendix 4.

5.1 Main Results

[Table 6.1] We run our NLT baseline on four popular few-shot classification benchmarks: Omniglot [Lake et al., 2011], *miniImageNet* [Vinyals et al., 2016a], *tieredImageNet* [Ren et al., 2018], and CUB [Wah et al., 2011]. We consider the classic four layer convolutional architecture [Snell et al., 2017a] (which we denote Conv), and the ResNet-12 [Ye et al., 2020]. We combine Sinkhorn K-Means with the ProtoNet implementation of [Ye et al., 2020], except for Omniglot where we use the original implementation [Snell et al., 2017a]. We report NLT accuracies alongside several state-of-the art LT methods for comparison. Given the very high numbers for Omniglot, we can conclude that both the 5-way 5-shot and 20 way 5-shot settings can be solved without LT. The same type of definitive conclusions cannot be made about the other benchmarks, but we do observe that for *tieredImageNet* our NLT baseline surpasses a number of LT methods implemented in a recent survey [Chen et al., 2019a].

5.2 Cross-Domain NLT Few-Shot Classification

[Table 6.2] We run our NLT baseline on two cross-domain benchmarks, *miniImageNet*→CUB [Chen et al., 2019a] and Meta-Dataset [Triantafillou et al., 2020] which was recently proposed as a harder benchmark. We report NLT accuracies alongside several LT methods, including the ProtoNet implementations

Table 6.1: LT and NLT few-shot classification on same-domain benchmarks. We consider 5-way 5-shot episodes, plus an additional 20-way 5-shot for Omniglot. Test accuracies are computed over 600 test episodes. The ResNet architectures are ResNet-12 except for SimpleShot and CTM which use ResNet-18. We denote ProtoNet* the implementation from which we derive CentroidNet. For the CUB dataset, some results [Ye et al., 2020] that we could not reproduce are in gray (they might have accidentally reported validation accuracies).

<i>miniImageNet</i>		
LT Methods	LT Accuracy	
backbone \rightarrow	Conv	ResNet
MatchNet [Vinyals et al., 2016a]	51.09	-
MAML [Finn et al., 2017]	63.11	-
RelationNet [Sung et al., 2018b]	67.07	-
ProtoNet [Snell et al., 2017a]	68.20	-
FEAT [Ye et al., 2020]	71.61	-
TADAM [Oreshkin et al., 2018]	-	76.70
MetaOptNet [Lee et al., 2019]	-	78.63
SimpleShot [Wang et al., 2019b]	-	80.02
CTM [Li et al., 2019b]	-	80.51
ProtoNet* [Ye et al., 2020]	71.33	80.53
FEAT [Ye et al., 2020]	-	82.05
NLT Methods	NLT Accuracy	
backbone \rightarrow	Conv	ResNet
CentroidNet (ours)	57.57	69.86

<i>tieredImageNet</i>		
LT Methods	LT Accuracy	
backbone \rightarrow	ResNet	
ProtoNet [Snell et al., 2017a]	72.69	
RelationNet [Sung et al., 2018b]	71.32	
MetaOptNet [Lee et al., 2019]	81.56	
CTM [Li et al., 2019b]	84.28	
SimpleShot [Wang et al., 2019b]	84.58	
ProtoNet* [Ye et al., 2020]	84.03	
FEAT [Ye et al., 2020]	84.79	
NLT Methods	NLT Accuracy	
backbone \rightarrow	ResNet	
CentroidNet (ours)	75.36	

Omniglot		
LT Methods	5-way	20-way
backbone \rightarrow	ConvNet	
SiameseNet [Koch et al., 2015]	98.4	97.0
MatchNet [Vinyals et al., 2016a]	98.9	98.5
NeuralStat [Edwards and Storkey, 2016]	99.5	98.1
MemoryMod [Kaiser et al., 2017]	99.6	98.6
ProtoNet* [Snell et al., 2017a]	99.7	98.9
MAML [Finn et al., 2017]	99.9	98.9
NLT Methods	5-way	20-way
backbone \rightarrow	ConvNet	
CentroidNet (ours)	99.1	98.1

CUB		
LT Methods	LT Accuracy	
backbone \rightarrow	ConvNet	
MatchNet [Vinyals et al., 2016a]	72.86	
MAML [Finn et al., 2017]	72.09	
ProtoNet [Snell et al., 2017a]	70.77	
ProtoNet* (repro) [Ye et al., 2020]	75.33	
RelationNet [Sung et al., 2018b]	76.11	
MatchNet [Ye et al., 2020]	79.00	
ProtoNet [Ye et al., 2020]	81.50	
FEAT [Ye et al., 2020]	82.90	
NLT Methods	NLT Accuracy	
backbone \rightarrow	ConvNet	
CentroidNet (ours)	66.13	

Table 6.2: LT and NLT few-shot classification on cross-domain benchmarks. *miniImageNet*→CUB is 5-way 5-shot, and Meta-Dataset involves variable numbers of ways and shots [Triantafillou et al., 2020]. Test accuracies are averaged over 600 episodes. See Appendix 7 for confidence intervals.

Meta-Dataset								
Test Dataset method →	<i>Train on ILSVRC</i>				<i>Train on all datasets</i>			
	LT			NLT	LT			NLT
	Proto	CNAPs	SUR	Centro	Proto	CNAPs	SUR	Centro
ILSVRC	44.12	50.6	56.3	26.40	41.79	52.3	56.3	23.84
Omniglot	53.40	45.2	67.5	36.83	81.93	88.4	93.1	66.25
Aircraft	45.29	36.0	50.4	24.15	69.43	80.5	85.4	57.50
Birds	63.59	60.7	71.7	41.08	64.73	72.2	71.4	43.56
Textures	61.78	67.5	70.2	39.63	66.35	58.3	71.5	43.50
QuickDraw	49.58	42.3	52.4	31.04	67.74	72.5	81.3	46.96
Fungi	35.27	30.1	39.1	18.11	38.94	47.4	63.1	21.76
VGG Flower	78.09	70.7	84.3	47.98	84.45	86.0	82.8	55.11
Traffic Sign	46.08	53.3	63.1	22.03	49.91	60.2	70.4	22.71
MSCOCO	35.63	45.2	52.8	18.19	36.64	42.6	52.4	17.60

MiniImageNet → CUB		
LT Methods backbone →	LT Accuracy	
	ConvNet	ResNet
MAML [Chen et al., 2019a]	-	51.34
MatchNet [Chen et al., 2019a]	-	53.07
RelationNet [Chen et al., 2019a]	-	57.71
ProtoNet* (repro)	62.52	61.38
ProtoNet [Chen et al., 2019a]	-	62.02
Baseline [Chen et al., 2019a]	-	65.57
GNN-FT [Tseng et al., 2020]	-	66.32
Neg-Softmax [Liu et al., 2020]	-	69.30
NLT Methods backbone →	NLT Accuracy	
	ConvNet	ResNet
CentroidNet (ours)	47.01	44.62

Table 6.3: Few-shot clustering on Omniglot with CCN splits. We consider the usual 4-layer “Conv-4” architecture [Snell et al., 2017a], the “CCN” architecture used by Hsu et al. [2017] which has more filters, and using raw images (32×32). The differences between the two architectures are not significant. Our results are reported with 95% confidence intervals and averaged over 1000 test episodes with a fixed model. Numbers with a star* are from Hsu et al. [2019].

Omniglot-CCN			
Clustering Methods backbone \rightarrow	Clustering Accuracy		
	32×32	Conv-4	CCN
K-Means	21.7*	69.4 \pm 0.5	-
CCN (KCL) [Hsu et al., 2017]	-	-	82.4*
CCN (MCL) [Hsu et al., 2019]	-	-	83.3*
CentroidNet-FSC (ours)	-	86.8 \pm 0.6	86.6 \pm 0.6

that we are based on [Triantafillou et al., 2020, Ye et al., 2020] and recent state-of-the-art LT-methods on Meta-Dataset : CNAPs [Requeima et al., 2019] and SUR [Dvornik et al., 2020]. For both cross-domain benchmarks, we observe that the gap between our NLT baseline and the LT state-of-the-art is significantly larger than in the same-domain case, thereby confirming that cross-domain benchmarks are much more dependent on using test-time labels, which would make them more appropriate benchmarks for validating the task-adaptation capabilities of LT methods.

5.3 Exploring Other Few-Shot Settings

Few-Shot Clustering with Partial Information [Table 6.3]. In few-shot clustering, the goal is to cluster new sets of data according to semantics learned during training. We investigate using CentroidNet for few-shot clustering. Please note that our method is less flexible than other learning to cluster approaches [Hsu et al., 2018] in that it requires knowing the number of examples per cluster (partial information). We will be comparing with Constrained Clustering Networks [Hsu et al., 2017, 2019], a recent state-of-the-art learning to cluster method, on their split of Omniglot which we denote Omniglot-CCN.³ Omniglot-CCN is harder than the usual split, because the training and testing splits contain different alphabets. Furthermore, each episode consists of characters of the same alphabet (more fine-grained) and the number of ways varies (20 to 47 characters per set). We start by training ProtoNet on classic LT few-shot classification with a center loss of 0.1. At

³We make the choice to apply our method on their task rather than the opposite because their method is much slower and more complicated to run. By solving the same task as them using the same architectures, we can compare directly with the results from their paper.

Table 6.4: Transductive 5-way 5-shot classification on *miniImageNet* and *tieredImageNet* with the ResNet-12 backbone. ProtoNet numbers are reproduced from the implementation of Ye et al. [2020].

Transductive Methods dataset \longrightarrow	Transductive Accuracy	
	<i>miniImageNet</i>	<i>tieredImageNet</i>
TPN [Liu et al., 2018]	75.65	-
TEAM [Qiao et al., 2019]	75.90	-
ProtoNet [Ye et al., 2020] (non-transductive)	80.40 \pm 0.57	84.24 \pm 0.65
CAN+ [Hou et al., 2019]	80.64	84.93
FEAT [Ye et al., 2020] (non-transductive)	80.99 \pm 0.61	84.58 \pm 0.63
ProtoNet+Sinkhorn (ours)	82.79\pm0.57	86.35\pm0.63
FEAT+Sinkhorn (ours)	83.54\pm0.58	87.47\pm0.60

test-time, we embed the set to cluster and cluster the embeddings using Sinkhorn K-Means. Because the predicted cluster indices are permutation invariant, we run the Hungarian algorithm to find the permutation that maximizes the accuracy. The resulting accuracy is called the **clustering accuracy** [Hsu et al., 2017]. We find that CentroidNet outperform all “flavors” of CCN by a margin (86.8% vs. 83.3% highest), while being simpler and running about 100 times faster (the data is only embedded once). We provide additional few-shot clustering results in Section 5 of the appendix.

Transductive Few-Shot Classification [Table 6.4]. In transductive few-shot classification, the learner is allowed to make predictions jointly on the query set. We explore using Sinkhorn K-Means to post-process non-transductive predictions in order to solve transductive few-shot classification. We start from the non-transductive predictions given by the ProtoNet and FEAT implementations of Ye et al. [2020]. Using Sinkhorn K-Means we search for the optimal transport plan between query points and classes. The cost of assigning a query point to a class is taken equal to its predicted negative log-likelihood for that class. We solve transductive few-shot classification on *miniImageNet* and *tieredImageNet* and compare the results with TPN [Liu et al., 2018], TEAM [Liu et al., 2018], and CAN+[Hou et al., 2019]. We achieve the best scores after starting from a strong non-transductive baseline. Some people might argue that our comparison is unfair because we explicitly use the uniform distribution of query labels unlike other methods [Vinyals et al., 2016a, Liu et al., 2018, Qiao et al., 2019]. However, we wish to point out that there is no reason to believe that other methods aren’t implicitly leveraging this assumption (even in our case, we could infer the label distribution by treating it as a hyperparameter).

6 Conclusion

Motivated by the apparent lack of diversity of some popular few-shot classification benchmarks, we have proposed a new baseline that attempts to solve them without using support set labels at test-time (LT). We find that Omniglot can be solved without LT, and report accuracies on several popular same-domain and cross-domain benchmarks. By comparing our NLT baseline to state-of-the-art LT methods, we confirm that cross-domain few-shot classification is significantly harder and dependent on using LT. In general, our NLT accuracies on any combination of dataset and architecture can be taken as a future reference of the baseline performance to be expected from only using a good representation, without any adaptation to test-time labels. We hope that our work has raised awareness about some limitations of current few-shot learning benchmarks. Our results support recent developments of harder multi-domain benchmarks.

7 Broader Impact

Our work discusses the way researchers evaluate their methods, in a specific topic of machine learning. We are motivated by the hypothesis that benchmarks which are not representative of real world situations might lead to overconfidence and could result in unpredictable behavior on deployment.

8 Acknowledgements

We thank Min Lin and Eugene Belilovsky for insightful discussions on few-shot classification. We thank Jose Gallego for insightful discussions on Sinkhorn K-Means. We thank Pascal Lamblin and Eleni Triantafillou for helping with the Meta-Dataset benchmark. This research was partially supported by the NSERC Discovery Grant RGPIN2017-06936, a Google Focused Research Award and the Canada CIFAR AI Chair Program. We also thank Google for providing Google Cloud credits.

7

Prologue to Third Contribution

1 Article Details

A Survey of Self-Supervised and Few-Shot Object Detection by *Gabriel Huang, Issam Laradji, David Vázquez, Simon Lacoste-Julien, Pau Rodríguez*. This paper has been accepted with minor revision at the IEEE Transactions in Pattern Analysis and Machine Intelligence (TPAMI).

2 Contributions of the authors

Originally, we were trying to beat the state-of-the-art in few-shot object detection by using self-supervised pretraining. Upon reviewing the literature, we realized that the field was somewhat disorganized, with inconsistent evaluation protocols that were substantially different from few-shot classification; there were even different methods sharing the same name (e.g., *Meta-RCNN*). David Vázquez suggested that writing a survey would be a valuable contribution to the community. All co-authors participated in the writing of the survey, with contributions proportional to the ordering, except for Pau Rodríguez, who supervised me directly and also contributed significantly to the writing. David Vázquez and Simon Lacoste-Julien helped proofread the survey and provided overall mentoring.

3 Recent Developments

Pix2Seq [Chen et al., 2021b] proposed to formulate object detection as a language modeling task. Given an image to process, a transformer is conditioned on the image to produce a sequence of bounding box coordinates and labels (e.g. [123, 450, 170, 480], "pedestrian"). Embeddings are learned for each object category and box coordinates—normalized and discretized into bins. The flexibility of training autoregressive models with maximum likelihood completely sidesteps the traditional difficulty of matching predicted boxes with ground-truth, removing the need for ad-hoc heuristics like Non-Maximum Suppression or Hungarian Matching, which are not fully differentiable (the matching part of the Hungarian loss is not differentiable).

One limitation of Pix2Seq is that embeddings are learned for each category instead of taking advantage of the full potential of language models, which is an important feature in the context of zero-shot and few-shot object detection. Another issue is the quantization of bounding box coordinates, which may potentially be an issue for high-precision applications.

The last months have seen a rise of general-purpose vision-language models. For instance, Unicorn [Yang et al., 2021b] unifies visual grounding, grounded captioning, image captioning, question answering, and object localization into the same model. More recently, Flamingo [Alayrac et al., 2022] was proposed as a general-purpose vision-language model. We expect the field of general-purpose vision-language models, featuring multi-task finetuning [Sanh et al., 2021] and emergent zero-shot capabilities [Brown et al., 2020], to continue growing, with object detection as one of the many tasks they can solve.

8

A Survey of Self-Supervised and Few-Shot Object Detection

Abstract

Labeling data is often expensive and time-consuming, especially for tasks such as object detection and instance segmentation, which require dense labeling of the image. While few-shot object detection is about training a model on *novel* (unseen) object classes with little data, it still requires prior training on many labeled examples of *base* (seen) classes. On the other hand, self-supervised methods aim at learning representations from unlabeled data which transfer well to downstream tasks such as object detection. Combining few-shot and self-supervised object detection is a promising research direction. In this survey, we review and characterize the most recent approaches on few-shot and self-supervised object detection. Then, we give our main takeaways and discuss future research directions.

Project page: <https://gabrielhuang.github.io/fsod-survey/>

1 Introduction

Traditional object detectors rely on large supervised object detection datasets such as PASCAL VOC [Everingham et al. \[2010\]](#) and MS COCO [Lin et al. \[2014\]](#), which have over hundreds and thousands of annotated examples per object category. However, labeling data is often expensive and time-consuming. This is especially true in the case of object detection and instance segmentation, which require dense labeling of bounding boxes/masks for each object, a process that is slower and requires more annotator training than for object classification. Moreover, for fine-grained object detection applications such as plant or animal species recognition, pre-labeled datasets may not exist, and labels may have to be collected on the spot by expert annotators.

To try to solve these problems, few-shot object detection (FSOD) methods attempt to recognize *novel* (unseen) object classes based only on a few examples, after training on many labeled examples of *base* (seen) classes. Until recently, the standard approach in few-shot object detection was to pretrain a backbone for ImageNet classification, then train an object detector on top of this backbone on the base classes, and finally finetune on the novel classes [[Kang et al., 2019](#), [Yan et al., 2019](#),

Wang et al., 2020a, Wu et al., 2020a, Zhang et al., 2021b]. However, because of the tremendous progress in learning self-supervised representations, several (few-shot) detection methods now initialize their backbone from representations pretrained with unsupervised pretext tasks on ImageNet and MS COCO [Bar et al., 2021, Wei et al., 2021a, Yang et al., 2021a, Xiao et al., 2021, Li et al., 2021b, Pinheiro et al., 2020].

The problem with typical self-supervised pretrained methods such as SimCLR [Chen et al., 2020a] or MoCo [He et al., 2020] is that they are geared towards classification, and often engineered to maximize *Top-1* performance on ImageNet Wang et al. [2021]. However, some of the learned invariances in classification (e.g. to translation) might not be desirable in localization tasks, and thus the representation might discard critical information for object detection. Moreover, it has been shown that higher ImageNet Top-1 accuracy does not necessarily guarantee higher object detection performance Wang et al. [2021].

In response to such shortcomings, there has been a growing number of methods for self-supervised object detection. These methods [Dai et al., 2021, Wang et al., 2021, Yang et al., 2021a, Xiao et al., 2021, Wei et al., 2021a] not only attempt to remedy the shortcomings of classification-geared representations, but also pretrain more components in addition to the feature extractor, such as the region proposal network (RPN) and detection head, in the case of Faster R-CNN based methods. In particular, the current state of the art for FSOD on MS COCO is a method which does self-supervised pretraining of both the backbone and the object detector [Bar et al., 2021].

Thus, this motivates a survey combining the most recent approaches on few-shot and self-supervised object detection, both of which having not been surveyed before (see Section 2). In the following sections, we briefly summarize key object detection concepts (Section 3). Then we review the few-shot object detection task and benchmarks (Section 4) and we discuss the most recent developments in few-shot object detection (Section 4) and self-supervised object detection pretraining (Section 5). We conclude this survey by summing up the main takeaways, future trends, and related tasks (Sections 6 and 7). We provide a taxonomy of popular few-shot and self-supervised object detection methods in Figure 8.1, on the base of which this survey is structured.

2 Related Surveys

Jiao et al. [2019] and Zaidi et al. [2021] survey modern deep-learning based object detection methods. They review several state-of-the-art backbones and compare their parameter counts. They present common datasets benchmarks and evaluation

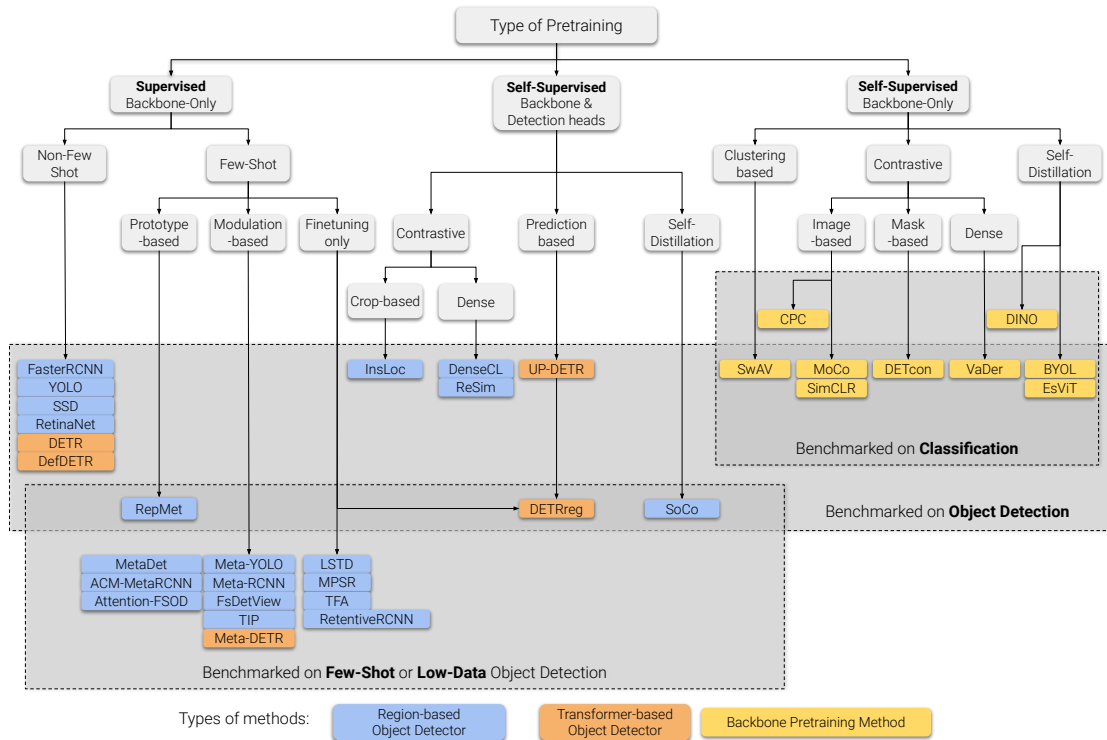


Figure 8.1: A taxonomy of object detection methods reviewed in this survey. We categorize them based on the following hierarchy: methods using supervised backbone pretraining, methods using self-supervised pretraining of backbone and detection heads, and self-supervised backbone pretraining methods. In parallel, we also tag (shaded rectangles) those methods depending on whether they have been benchmarked on regular object detection, few-shot/low-shot object detection, and ImageNet classification. As discussed in Section 5, many self-supervised classification methods have also been used to initialize object detection backbones and evaluated on object detection benchmarks. DETReg [Bar et al., 2021], which is a self-supervised object detection method, obtained state-of-the-art FSOD results on MS COCO and uses self-supervised pretraining of the entire architecture.

metrics. Jiao et al. [2019] categorize object detectors into single-stage and two-stage detectors, and report their respective performances in a large table. Zaidi et al. [2021] is more recent and discusses backbone choices extensively, including transformer-based backbones such as Swin transformers [Liu et al., 2021b], and lightweight architectures such as MobileNet [Howard et al., 2017] and SqueezeNet [Iandola et al., 2016]. Although both works focus on traditional object detection, they briefly mention weakly-supervised, few-shot and unsupervised object detection as future trends. Note that these surveys do not cover the newer transformer-based object detectors such as DETR [Carion et al., 2020] or Deformable DETR [Zhu et al., 2021], which we will briefly introduce in this survey.

Jing and Tian [2020] present a survey on self-supervised visual feature learning. They perform an extensive review of self-supervised pretext tasks, backbones, and downstream tasks for image and video recognition. In this work, we also introduce a number of generic self-supervised pretraining techniques but we focus on methods particularly designed for object detection.

Regarding few-shot classification, a simpler task than few-shot object detection, Chen et al. [2019a] introduce a comparative analysis of several representative few-shot classification algorithms. Wang et al. [2020b] propose a more extensive survey on methods and datasets. However, they do not explore few-shot object detection methods.

Khan et al. [2021] show that transformers have achieved impressive results in image classification, object detection, action recognition and segmentation. In this survey, we review object detection methods using transformers as backbones and as detection heads with DETR and variants [Carion et al., 2020, Zhu et al., 2021]. We also discuss the emergent properties of visual transformers as showcased by Caron et al. [2021].

3 Background on Object Detection

3.1 Key Concepts

For clarity, we start by reviewing key concepts in object detection, and introduce relevant vocabulary. Readers already familiar with object detection can skip directly to Sections 4 and 5 for few-shot and self-supervised object detection. We illustrate those concepts in the context of Faster R-CNN [Ren et al., 2015] with Feature Pyramid Network [Lin et al., 2017a], a multi-scale two-stage object detector represented in Figure 8.2, and DETR [Carion et al., 2020] represented in Figure 8.3. A more in-depth analysis of object detection concepts can be found in the object detection surveys by Jiao et al. [2019], Zaidi et al. [2021].

Object detection is the task of jointly localizing and recognizing objects of interest in an image. Specifically, the object detector has to predict a bounding box around each object, and predict the correct object category. Object detectors are traditionally trained on labeled object detection datasets such as PASCAL VOC [Everingham et al., 2010] or MS COCO [Lin et al., 2014]; the objects of interest are simply the categories that the model is trained to recognize.

The **backbone** network is a feature extractor which takes as input an RGB image and outputs one or several feature maps [Lin et al., 2017a]. Typically, the backbone is a residual network such as the ResNet-50 [He et al., 2016], and is pretrained on ImageNet classification before finetuning it to downstream tasks [Wang et al., 2020a, Yan et al., 2019, Bar et al., 2021]. Alternatively, an increasing number of works have considered using visual transformers instead [Redmon and Farhadi, 2018, Li et al., 2021b, Caron et al., 2021]. The RGB image is a 3D tensor in $\mathbb{R}^{W \times H \times 3}$, where typically $W = H = 224$ for classification, and $W, H \approx 1300, 800$ for object detection (as per Detectron2’s¹ default parameters). For few-shot object detection, a ground-truth mask delimiting the support object is sometimes appended to a fourth channel of the image, and the backbone is modified to take as input tensors in $\mathbb{R}^{W \times H \times 4}$.

Single-scale features consist of a single 3D tensor obtained by taking the outputs of a certain backbone layer. Typically, the C4 layer (corresponding to the output of “res5” the 4th residual block) of the ResNet-50 is used for object detection. The feature map is of size $Z \in \mathbb{R}^{w \times h \times c}$ where c is the number of channels and w, h are the spatial dimensions of the feature map, which are much smaller than the image due to strided convolutions.

Multi-scale features consist of several 3D tensors at different scales. Merely combining several layer outputs from the backbone would result in the high-resolution lower layers having limited semantic information. A common solution is to implement top-down and lateral pathways using a Feature Pyramid Network (**FPN**) [Lin et al., 2017a] to propagate information from the higher-levels of the backbone back to the lower levels (illustrated in Figure 8.2).

Faster R-CNN [Ren et al., 2015], represented in Figure 8.2, is a popular two-stage object detector. To detect objects, they start by feeding an image to the backbone to get single or multi-scale features. Then, they apply the following two stages:

- *Stage 1*: They feed the features to the **Region Proposal Network** (RPN) to extract **object proposals**, which are bounding boxes susceptible to contain objects. The object proposals are predicted at predefined locations, scales and aspect ratios (known as **anchors**), refined using a regression head (anchor deltas), and scored for “**objectness**”. They use Non-Maximum Suppression

¹<https://github.com/facebookresearch/detectron2>

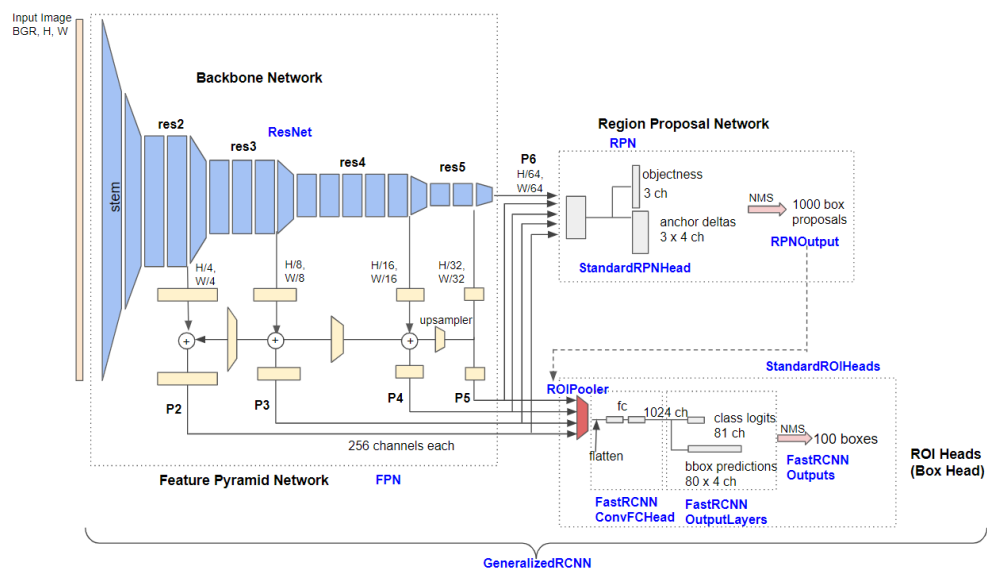


Figure 8.2: A Faster R-CNN with Feature Pyramid Network. The input image is fed to the backbone network, then the feature pyramid network (light yellow) computes multi-scale features. The region proposal network proposes candidate boxes, which are filtered with non-maximum suppression (NMS). Features for the remaining boxes are pooled with RoIAlign and fed to the box head, which predicts object category and refined box coordinates. Finally, redundant and low-quality predictions are removed with NMS. Blue labels are class names in the detectron2 implementation. Figure courtesy of Hiroto Honda. <https://medium.com/@hirotoschwert/digging-into-detectron-2-47b2e794fabd>

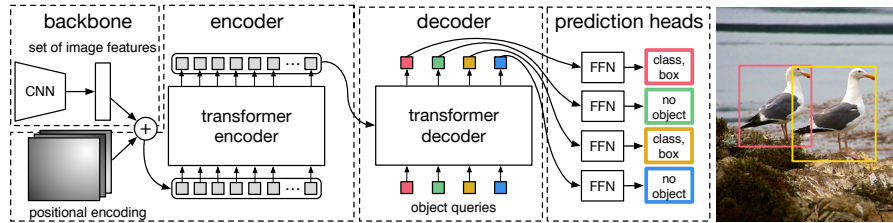


Figure 8.3: The DETR object detector. The image is fed to the backbone, then positional encodings are added to the features and fed to the transformer encoder. The decoder takes as input object query embeddings, cross-attends to the encoded representation while performing self-attention on the transformed query embeddings, and outputs a fixed number of object detections, which are finally thresholded, without need for NMS [Carion et al., 2020]. Image courtesy of Carion et al. [2020].

(NMS) to remove redundant and low-quality object proposals.

- *Stage 2:* For each object proposal they extract a pooled feature map by resampling the features inside its bounding box to a fixed size, using RoIAlign or ROI Pool (pooling strategies). For multiscale-features, the appropriate level is picked using a heuristic. Then, they feed the pooled features into the **Box Head** or **Region-of-Interest** (ROI) head, which predicts the object category and refines the bounding box with another regression head. Finally, they run NMS again to remove redundant and low-confidence predictions.

In this survey, we will refer to the union of the RPN and box head as the **detection heads**.

Mask R-CNN [He et al., 2017] is an improvement on top of Fast R-CNN to solve instance segmentation. At the simplest level, Mask R-CNN predicts segmentation masks for each detected instance, additionally to the bounding box and class predictions.

Single-stage object detectors, such as You Only Look Once (**YOLO**) [Redmon et al., 2016, Redmon and Farhadi, 2018], Single-Shot Detector (**SSD**) [Liu et al., 2016], and newer methods such as RetinaNet [Lin et al., 2017b] and CenterNet [Zhou et al., 2019], are generally simpler and faster than two-stage detectors, at the cost of lower prediction quality. Single-stage detectors directly predict objects at predefined locations from the feature map, with the possibility of subsequently refining the box locations and aspect ratios. Please refer to object detection surveys for an in-depth review [Jiao et al., 2019, Zaidi et al., 2021].

DETR [Carion et al., 2020] represented in Figure 8.3, which stands for DEtection TRansformer, is a recent transformer-based architecture for end-to-end object detection. Notably, DETR has a much simpler overall architecture than Faster R-CNN and removes the need for the NMS heuristic –which is non-differentiable–

by learning to remove redundant detections. However, being set-based, DETR relies on the Hungarian algorithm [Munkres, 1957] for computing the prediction loss, which has been shown to be difficult to optimize [Sun et al., 2020].

To detect objects, they start by feeding an image to the backbone to get features. Then, they feed the features to the transformer **encoder** to obtain encoded features. Finally, they feed 100 (or any number of) learned “query embeddings” to the transformer **decoder**. The transformer decoder attends to the encoded features and outputs up to 100 predictions, which consist of bounding box locations, object categories and confidence scores. The highest-confidence predictions are returned. There is no need for removing redundant predictions using NMS, as the model learns to jointly make non-redundant predictions thanks to the Hungarian loss. During training, the *Hungarian loss* is computed by finding the optimal matching between detected and ground-truth boxes in terms of box location and predicted class. The loss is minimized using stochastic gradient descent (SGD). We will also refer to the union of transformer encoder and decoder as the **detection heads**.

Deformable DETR [Zhu et al., 2021] is a commonly used improvement over DETR. Deformable DETR uses multi-scale deformable attention modules, which can attend to a small set of learned locations over multiple feature scales, instead of attending uniformly over a whole single-scale feature map. The authors manage to train their model using 10 times fewer epochs than DETR [Zhu et al., 2021].

3.2 Datasets and Evaluation Metrics

The most popular datasets for traditional object detection are PASCAL VOC [Everingham et al., 2010] and MS COCO [Lin et al., 2014]. Since they have already been widely discussed in the literature, we refer the reader to previous object detection surveys [Jiao et al., 2019, Zaidi et al., 2021]. PASCAL VOC and MS COCO have also been adopted by the few-shot object detection (FSOD) and self-supervised object detection (SSOD) communities. We provide an extensive discussion on their use as few-shot benchmarks in Section 4.3. Please also refer to Section 4.3 for a detailed explanation of the mean average precision (mAP) evaluation metric and the differences between PASCAL VOC and MS COCO implementations.

4 Few-Shot Object Detection

Informally, few-shot object detection (FSOD) is the task of learning to detect new categories of objects using only *one* or a *few* training examples per class. In this section, we describe the FSOD framework, its differences with few-shot classification, common datasets, evaluation metrics, and FSOD methods. We provide a taxonomy of popular few-shot and self-supervised object detection methods in Figure 8.1.

4.1 FSOD Framework

We formally introduce the dominant FSOD framework, as formalized by Kang et al. [2019] (**Figure 8.4**). FSOD partitions objects into two disjoint sets of categories: **base** or known/source classes, which are object categories for which we have access to a large number of training examples; and **novel** or unseen/target classes, for which we have only a few training examples (shots) per class. In the vast majority of the FSOD literature, we assume that the object detector’s backbone has already been pretrained on an image classification dataset such as ImageNet (usually a ResNet-50 or 101). Then, the FSOD task is formalized as follows:

- (1) **Base training.**² Annotations are given only for the base classes, with a large number of training examples per class (*bikes* in the example). We train the FSOD method on the base classes.
- (2) **Few-shot finetuning.** Annotations are given for the *support set*, a very small number of training examples from *both* the base and novel classes (one *bike* and one *human* in the example). Most methods finetune the FSOD model on the support set, but some methods might only use the support set for conditioning during evaluation (finetuning-free methods).
- (3) **Few-shot evaluation.** We evaluate the FSOD to jointly detect base and novel classes from the test set (few-shot refers to the size of the support set). The performance metrics are reported separately for base and novel classes. Common evaluation metrics are variants of the mean average precision: mAP50 for Pascal and COCO-style mAP for COCO. They are often denoted bAP50, bAP75, bAP (resp. nAP50, nAP75, nAP) for the base and novel classes respectively, where the number is the IoU-threshold in percentage (see Section 4.4 for full explanation).

We encourage researchers to report both base *and* novel class performance, a setting sometimes called Generalized FSOD [Fan et al., 2021]. Note that “training” and “test” set refer to the splits used in traditional object detection. Base and novel classes are typically present in both the training and testing sets; however, the novel class annotations are filtered out from the training set during base training; during few-shot finetuning, the support set is typically taken to be a (fixed) subset of the training set; during few-shot evaluation, all of the test set is used to reduce uncertainty [Kang et al., 2019].

Special case—no fine-tuning Conditioning-based methods skip the fine-tuning step; instead, novel examples are used as support examples to condition the model,

²In the context of self-supervised learning, base-training may also be referred to as *finetuning* or *training*. This should not be confused with *base training* in the meta-learning framework; rather this is similar to the meta-training phase [Finn et al., 2017].

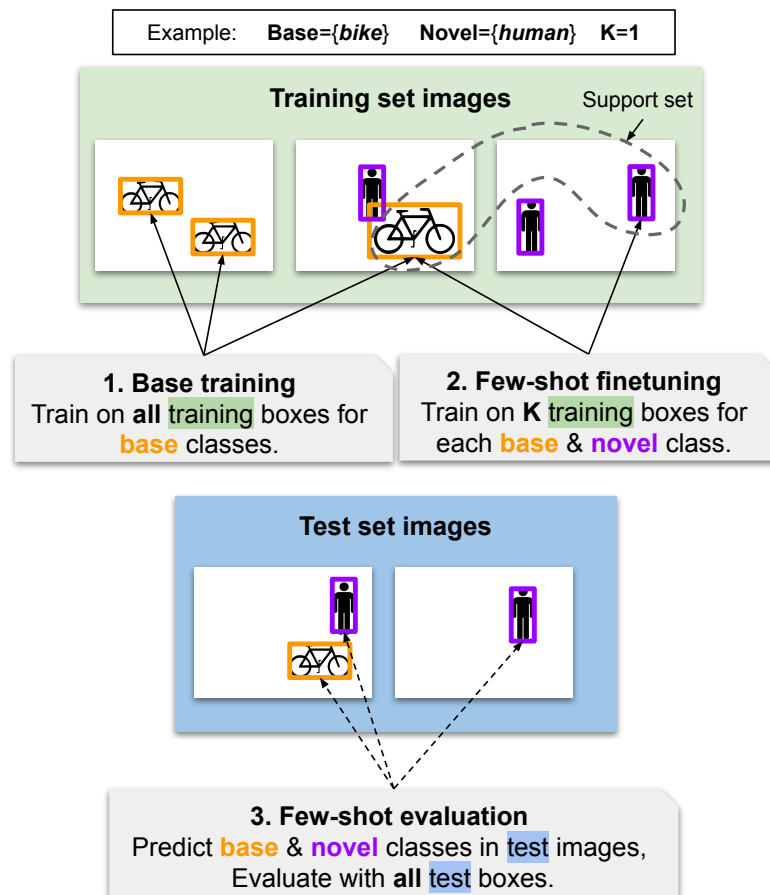


Figure 8.4: Few-shot object detection protocol, as proposed by Kang et al. [2019]. During base-training, the method is trained on base classes. Then during few-shot finetuning, the model is finetuned or conditioned on the support set. Finally, during few-shot evaluation, the method is evaluated on base and novel class detection.

and predictions are made directly on the test set. In practice, the majority of conditioning-based methods reviewed in this survey do benefit from some form of finetuning.

Special case—no base classes The standard FSOD framework might seem counter-intuitive as it assumes a large set of annotated base class images. DETReg [Bar et al., 2021] have investigated removing the base-training phase and replacing it with self-supervised pretraining, relying solely on a small number of novel labels, which came at the cost of lower nAP.

4.2 Important Differences with Few-Shot Classification.

As this may not be obvious to readers unfamiliar with both fields, we explicit several practical differences between the FSOD finetuning-based paradigm and the learning-to-learn paradigm [Ravi and Larochelle, 2016, Vinyals et al., 2016a, Chen et al., 2019a, Huang et al., 2019] commonly used in few-shot classification (FSC):

- **Several objects per image.** In FSOD there can be several instances from base and novel classes in the same image, whereas FSC assumes only one dominant object per image. While FSC filters out all novel *images* during base-training, FSOD removes only the novel object annotations but keeps the images that also contain base objects.
- **Joint prediction on base and novel.** During few-shot finetuning and evaluation, both base and novel classes are present and have to be jointly detected. On the contrary, few-shot classifiers are typically only finetuned and evaluated on novel classes. Note however that many papers only report average precision for novel classes under metric names such as *nAP* or *nAP50*.
- **Learning-to-learn vs. finetuning.** Gradient-based few-shot classification methods such as MAML [Finn et al., 2017] or Learning-to-Optimize [Ravi and Larochelle, 2016] rely heavily on the *learning-to-learn* paradigm; during (meta)training, N-way K-shot episodes are generated by sampling a small training set (support set) and a small validation set (query set) from the base classes. The classifier is finetuned with gradient descent on the support set, makes predictions on the query set, and the query-set loss is minimized using gradient descent, which propagates the gradient through support set tuning. On the contrary, a majority of FSOD methods do not generally consider episodes or backpropagate through gradient descent. Pure finetuning FSOD methods Wang et al. [2020a], Chen et al. [2018], Wu et al. [2020a], Bar et al. [2021] are first trained on all base classes during base training, and finetuned only once on a fixed support set before few-shot evaluation. Moreover, because the few-shot finetuning step (e.g., optimizer learning rates) and the pre-finetuning weights are not calibrated over several episodes using

Table 8.1: Common FSOD benchmarks. Image counts are after filtering out the ones containing no relevant bounding boxes.

Benchmark	Classes		1. Base Training		2. Few-shot Finetuning			3. Few-shot Evaluation		
	Base	Novel	#images	#base-bb	#shots	#base-bb	#novel-bb	#images	#base-bb	#novel-bb
PASCAL VOC/split 1	15	5	14,631	41,084	1:2:3:5:10	15–150	5–50	4,952	13,052	1,924
PASCAL VOC/split 2	15	5	14,779	40,397	1:2:3:5:10	15–150	5–50	4,952	12,888	2,088
PASCAL VOC/split 3	15	5	14,318	40,511	1:2:3:5:10	15–150	5–50	4,952	13,137	1,839
MS COCO 2014	60	20	98,459	367,702	10:30	600–1,800	200–600	5,000	15,318	20,193
LVIS v0.5	776	454	68,568	688,029	8.57 (variable)	7,760	2,786	5,000	50,334	429

learning-to-learn on a separate query set, they might not be optimal. This is partially mitigated by hyperparameter tuning, which can help find optimal learning rates, but not find the optimal pre-finetuning weights.

- **Episodic Evaluation.** For FSC evaluation, several episodes are sampled from the novel classes [Vinyals et al., 2016a, Oreshkin et al., 2018, Rodriguez et al., 2020]; the classifier is finetuned on the support set and classifies the query set, and the results are averaged over hundreds of runs, which have the advantage of reducing variance and estimating confidence intervals [Chen et al., 2019a, Huang et al., 2019]. On the contrary, each of Kang’s splits [Yan et al., 2019] feature only *one* fixed support set (the exact instances are prespecified), which is known to cause overfitting and overestimating performance especially in the case of 1-shot object detection, when the support set is tiny [Wang et al., 2020a]. See Figure 8.5 for the impact of using several episodes on PASCAL VOC. In response to those issues, Wang et al. [2020a] sample multiple support sets (30 seeds) to lower the variance of the benchmark.
- **Separate validation and test sets.** Whereas FSC methods generally validate and test on separate splits for common benchmarks such as Omniglot [Lake et al., 2015], *mini*ImageNet [Vinyals et al., 2016a], or Synbols Lacoste et al. [2020], FSOD detection methods follow the standard practice in object detection of training on the union of the training and validation sets, and using the test set for both hyperparameter tuning and evaluation, which inevitably leads to overestimating the generalization ability of the methods.

4.3 FSOD Datasets

We describe the dominant FSOD benchmarks, as introduced by Meta-YOLO [Kang et al., 2019] and improved by TFA (Two-stage Fine-tuning Approach) [Wang et al., 2020a] to mitigate variance issues. These are also the benchmarks that we will use to compare FSOD methods in Table 8.3. We compute data statistics in **Table 8.1**.³ Caveats and future best practices are discussed in Section 4.3 and Section 6.5.

³Note that the reported number of images is after removing those containing no relevant annotations (e.g., for base training, the images which contained only novel objects are removed).

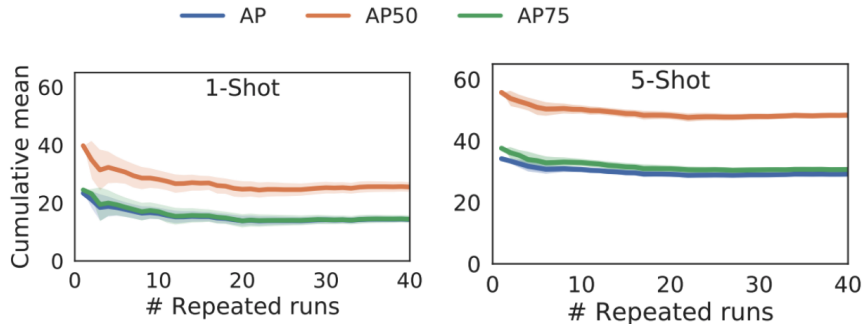


Figure 8.5: Importance of evaluating over several episodes. The nAP, nAP50 and nAP75 of PASCAL VOC Split-1 are averaged using a variable number of episodes. Note how the means and variances only become stable after around 20 episodes. Figure courtesy of [Wang et al., 2020a].

PASCAL VOC [Everingham et al., 2010]

PASCAL VOC is arguably one of the most popular smaller benchmarks for traditional object detection. For few-shot object detection, the object categories are split between **15 base classes** and **5 novel classes**. Three different base/novel splits are usually considered, denoted splits 1, 2 and 3. For base training, the training and validation (*trainval*) images from VOC2007 and VOC2012 are used, which might lead to overfitting (Section 6.5). This amounts to **40k base boxes** spread over 15k images for an average of **2700 boxes / base class**, with 2.8 boxes / image. For few-shot finetuning, a fixed subset of the VOC2007 and VOC2012 *trainval* sets is taken as the support set. Kang et al. [2019] consider the **1,2,3,5,10-shot** settings, which correspond to 15-150 base bounding boxes and 5-50 novel bounding boxes. The fact that the instances are fixed may lead to overestimating performance (Section 4.3). For few-shot evaluation, roughly 5k images from the VOC2007 test set are considered, with 13k base and 2k novel boxes, which adds up to an average of **870 boxes/base** and **400 boxes/novel** class. The method has to detect both base and novel classes, and several evaluation metrics are reported separately for base and novel classes: mAP50, mAP75, and COCO-style mAP. The main comparison metric is the novel mAP50.

MS COCO [Lin et al., 2014]

MS COCO or Microsoft Common Objects in COntext [Lin et al., 2014], is another popular benchmark for object detection. For FSOD, the object categories are split between **20 novel classes** which are shared with PASCAL VOC, and the remaining **60 base classes**. Following Kang et al. [2019] and Wang et al. [2020a], a 5k subset of the COCO2014 validation images are used for few-shot evaluation (denoted *val5k*), while the remaining COCO2014 training and validation images are

used for base-training and few-shot finetuning (denoted *trainvalno5k*). Specifically for base training, roughly **367k base boxes** are considered (10 times more than PASCAL VOC), spread over 98k images from *trainvalno5k*. This is an average of **6k boxes / base class** and 3.7 boxes/image. For few-shot finetuning, a fixed subset of *trainvalno5k* is taken as the support set. Kang et al. [2019] consider the **10,30-shot** settings, which correspond to 600-1800 base boxes and 200-600 novel boxes, and suffer less from overfitting than PASCAL VOC [Wang et al., 2020a] despite also using fixed instances. For few-shot evaluation, the *val5k* are used, consisting in 15k base boxes and 20k novel boxes, which amounts to **250 boxes / base class** and **1k boxes / novel class**. Several evaluation metrics are reported separately for base and novel categories: COCO-style mAP, mAP50, mAP75, and mAP for small, medium and large objects. The main comparison metric is novel mAP.

LVIS [Gupta et al., 2019]

LVIS or Large Vocabulary Instance Segmentation [Gupta et al., 2019] is a newer object detection dataset featuring 1230 classes, categorized as *frequent*, *common* (10 or more annotations) and *rare* (fewer than 10). TFA [Wang et al., 2020a] have proposed using the *v0.5* version of this dataset for FSOD, dividing it into **776 base classes** (frequent and common) and **454 novel classes**, making it by far the FSOD benchmark with the most number of categories (10 times more categories than COCO, 50 times more than Pascal). For this description, we follow the reference implementation from TFA⁴ as we could not find all the details in the paper. For base training, **688k base boxes** are considered, spread over 69k images from the training set, which amounts to **887 boxes / base class** and 10 boxes / image. For few-shot finetuning, up to 10 shots from the training set are considered, depending on the number of available examples. This corresponds to an average of 8.57 boxes/class, spread over 7.7k base boxes and 2.8k novel boxes. For few-shot evaluation, the validation set of LVIS is used, consisting of 50,334 base and 429 novel boxes, spread over 5,000 evaluation images. Evaluation metrics are COCO-style mAP, mAP50 and mAP75, reported separately for frequent, common, rare objects, and also aggregated over all three categories.

Discussion

We discuss some of the advantages and issues associated with the aforementioned benchmarks (see also Section 6.5).

Overfitting issues For PASCAL VOC, the support set is very small (20-200 examples) and the specific instances are predefined by Kang’s splits. This can cause overfitting and result in overestimating the novel average precision, especially in

⁴<https://github.com/ucbdrive/few-shot-object-detection>

the 1-shot case, an issue illustrated by TFA [Wang et al., 2020a] in Figure 8.5. To mitigate this issue, TFA [Wang et al., 2020a] propose to randomly sample multiple support sets (30 seeds) and to average the results in a benchmark which we will denote the **TFA-splits** as opposed to the benchmark using a single fixed support set, which we will denote the **Kang-splits**. This is a good first step, but not as reliable as the common practice in few-shot classification of averaging metrics over 600 episodes [Chen et al., 2019a].

Reliability With a substantial support set of 800-2400 bounding boxes for few-shot finetuning, and plenty of few-shot evaluation boxes for base and novel categories, MS COCO is arguably the most reliable benchmark for comparing different methods. In fact, we sort methods according to 30-shot MS COCO nAP in Table 8.3. Because the 20 novel classes were chosen to be in common with Pascal, a very natural benchmark to consider is the MS COCO→Pascal cross-domain scenario, which has been considered by some of the earlier and subsequent works [Chen et al., 2018, Wu et al., 2020a, Fan et al., 2020b].

Limitations of LVIS v0.5 With 1230 classes, LVIS has more than an order of magnitude more object categories than MS COCO, and a lot of potential for few-shot object detection. However there are some shortcomings with directly using TFA’s splits. One problem is that only 705 out of 776 base classes and 125 out of 454 novel classes appear in the validation set, which means that the majority of novel classes will never be evaluated on.⁵ Moreover, because there are almost 100 more times base than novel boxes in the validation set, performance is completely dominated by base objects for metrics aggregated on base and novel classes. Finally, even the 125 novel classes that are evaluated on only have an average of 3.4 boxes / class, which means evaluation is potentially very noisy. Due to those issues, we do not recommend the current TFA splits for evaluating few-shot object detection methods. However, LVIS—especially LVIS v1.0 which has more data and stricter quality control—has a lot of potential in FSOD given the large diversity of objects. Therefore, proposing more balanced splits and evaluation sets would definitely be beneficial to the FSOD community.

Class overlap “Novel” FSOD classes may overlap with ImageNet classes used for backbone pretraining, potentially leading to performance overestimation. For instance, all of the 20 PASCAL VOC categories—used as MS COCO novel classes—except “person” appear in some form inside ImageNet. Related categories may be parts of one another (car mirror, aircraft carrier), subcategories (sport car, rocking chair, flowerpot vs. potted plant), or functionally similar objects

⁵As found by running TFA’s data loader: <https://github.com/ucbdrive/few-shot-object-detection>

(sofa vs. studio couch, motorbike vs. scooter). Because of the large number of ImageNet labels, we acknowledge the challenge of designing novel classes with no overlap with ImageNet for general-purpose object detectors. However, for niche FSOD applications, we do encourage practitioners to sanitize backbone and FSOD datasets for cross-contamination.

4.4 FSOD Evaluation Metrics

By design, object detectors such as Faster R-CNN and DETR output a *fixed* number of predictions paired with a confidence score. This means they use a threshold to cut off low-confidence predictions. Therefore, they have to trade off between using a higher threshold, which will lead to higher precision (most predicted objects are actually real objects) but low recall (miss out many of the real objects); and using a lower threshold, which could increase recall at the expense of precision.

The **mean average precision (mAP)** is a standard metric for evaluating object detection and FSOD methods, and is defined as the mean of the individual average precisions (AP) for each object class. The individual APs are defined as the area under the precision-recall curve – discussed below – which can be plotted by varying the confidence threshold of the object detector.

To compute the AP for a class, we first rank the detections for this class by decreasing confidence. Starting from the top-ranked detections ($k = 1$), we consider them as *True Positives* if their intersection over union (IoU) with any ground-truth true object is above a given IoU-threshold (typically 50% or 75%). If the IoU is below the threshold or the ground-truth has already been detected, then we consider them to be *False Positives*. For each rank k , corresponding to a different choice of confidence-threshold, we can compute the $\text{precision@}k$, a measure of relevance defined as the number of true positives among the top- k boxes divided by k , and the $\text{recall@}k$, a measure of sensitivity defined as the number of true positives among the top- k boxes divided by the total number of ground truth boxes.

We give an example of AP computation in Table 8.2. Notice how recall is non-decreasing as a function of k , while precision can fluctuate up and down. By varying k between 1 and the total number of detections, we can plot a precision vs. recall curve (see Figure 8.6). The precision-recall curve (orange) is made non-increasing by taking the *interpolated* precision (green), defined as $p_{\text{interp}}(r) = \max_{r' \geq r} p(r')$.⁶ The average precision is defined as the area under that curve. The exact way the area is computed depends on the specific benchmark.

For PASCAL VOC’s 2007 test set [Everingham et al., 2010], which is used for

⁶Interpolation is a natural thing to do because it means that for a minimum recall requirement, there exists a confidence-threshold which results in a detector with better precision if we allow the recall to be higher than that threshold, which is never a detrimental.

Table 8.2: Example computation of precision@k and recall@k. There are 10 detections ranked by decreasing confidence and 5 ground-truth boxes. Example from <https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173>

Rank k	True Positive?	Precision@ k	Recall@ k
1	True	1.0	0.2
2	True	1.0	0.4
3	False	0.67	0.4
4	False	0.5	0.4
5	False	0.4	0.4
6	True	0.5	0.6
7	True	0.57	0.8
8	False	0.5	0.8
9	False	0.44	0.8
10	True	0.5	1.0

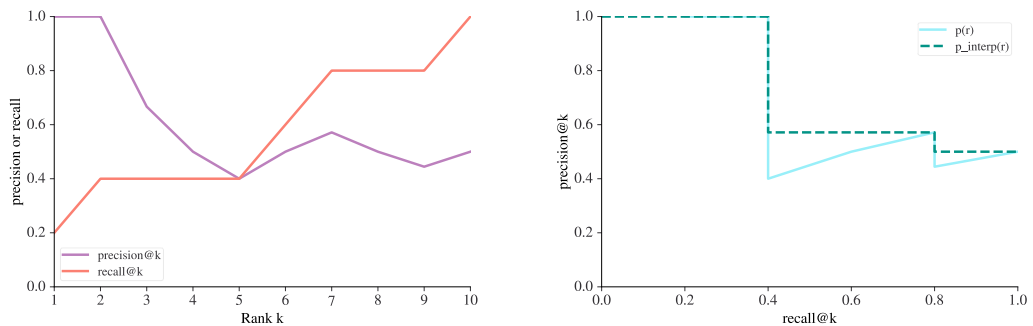


Figure 8.6: Top: precision@k and recall@k as a function of k the number of boxes considered. **Bottom:** precision@k and interpolated-precision@k as functions of the recall@k (precision-recall curve)

FSOD evaluation, the main IoU-threshold used is 0.5 and the area under the curve is approximated by sampling and averaging the interpolated precision at 11 points $\{0, 0.1, 0.2, \dots, 1\}$. IoU-thresholds of 0.75 and COCO-style mAP are also sometimes reported [Wang et al., 2020a]. For MS COCO, the area under the interpolated precision curve is computed exactly by adding up the area of each rectangle (see Figure). COCO-style mAP is defined by averaging the mAP at different thresholds from 0.5 to 0.95 in increments of 0.05. For COCO, it is common to report mAP scores for objects of different sizes: small, medium and large [Wang et al., 2020a, Xiao and Marlet, 2020]. Additionally, mAP50 and mAP75 are often provided [Wang et al., 2020a, Fan et al., 2020a, Zhang et al., 2021b], which are computed by computing the exact area under the curve for IoU-thresholds of 0.5 and 0.75. Generally in FSOD, it is also common to report mAP separately for base and novel classes, denoted as bAP and nAP [Wang et al., 2020a]. Most works put greater emphasis on the nAP [Kang et al., 2019, Yan et al., 2019], though some claim that maximizing bAP is also important to avoid catastrophic forgetting [Fan et al., 2021, Wang et al., 2020a]. These metrics and datasets give us a comprehensive overview of how different models perform for few-shot object detection. Note that for LVIS v0.5, the metrics are the same as for MS COCO, and reported separately for frequent, common and rare objects. However, we do not recommend following the TFA splits for LVIS due to the issues outlined in Section 4.3.

4.5 Few-Shot Object Detection Methods

We review several FSOD methods from the literature. In our description, we assume that backbones have already been pretrained on ImageNet. We summarize most of these methods in Table 8.3.⁷

Finetuning-only methods ⁸

Finetuning-only methods generally start from a traditional object detector such as Faster-RCNN [Ren et al., 2015] with only minor architecture modifications. They do base training on many base class examples, then do few-shot finetuning on a support set containing both base and novel classes. The rationale behind this two-step process is to deal with the extreme imbalance between the base and novel classes, and to avoid overfitting on the novel classes.

LSTD [Chen et al., 2018] proposed the first finetuning strategy for FSOD. A hybrid SSD/Faster-RCNN “source” network is trained on the source classes, then at finetuning time, its weights are copied into a “target” network, except for the

⁷We only included methods evaluated on at least one of the dominant FSOD benchmarks/splits.

⁸Use colors for quick reference to Table 8.3.

Table 8.3: Few-Shot Object Detection methods with results on PASCAL VOC and MS COCO. Methods are categorized as **finetuning**-only, **prototype**-based, and **modulation**-based. TIP is a general add-on strategy for two-stage detectors. Faster RCNN+FT numbers are from TFA [Wang et al., 2020a]. RepMet and Attention-FSOD numbers are from Meta-DETR [Zhang et al., 2021b]. Methods are sorted by MS COCO 30-shot nAP.

Find the most up-to-date table at <https://github.com/gabrielhuang/awesome-few-shot-object-detection>

Name	Type	VOC TFA-split (nAP50)			VOC Kang-split (nAP50)			MS COCO (nAP)	
		1-shot	3-shot	10-shot	1-shot	3-shot	10-shot	10-shot	30-shot
LSTD [Chen et al., 2018]	finetuning	-	-	-	8.2	12.4	38.5	-	-
RepMet [Karlinsky et al., 2019]	prototype	-	-	-	26.1	34.4	41.3	-	-
Meta-YOLO [Kang et al., 2019]	modulation	14.2	29.8	-	14.8	26.7	47.2	5.6	9.1
MetaDet [Wang et al., 2019c]	modulation	-	-	-	18.9	30.2	49.6	7.1	11.3
Meta-RCNN [Yan et al., 2019]	modulation	-	-	-	19.9	35.0	51.5	8.7	12.4
Faster RCNN+FT [Wang et al., 2020a]	finetuning	9.9	21.6	35.6	15.2	29.0	45.5	9.2	12.5
ACM-MetaRCNN [Wu et al., 2020b]	modulation	-	-	-	31.9	35.9	53.1	9.4	12.8
TFA w/fc [Wang et al., 2020a]	finetuning	22.9	40.4	52.0	36.8	43.6	57.0	10.0	13.4
TFA w/cos [Wang et al., 2020a]	finetuning	25.3	42.1	52.8	39.8	44.7	56.0	10.0	13.7
Retentive RCNN [Fan et al., 2021]	finetuning	-	-	-	42.0	46.0	56.0	10.5	13.8
MPSR [Wu et al., 2020a]	finetuning	-	-	-	41.7	51.4	61.8	9.8	14.1
Attention-FSOD [Fan et al., 2020a]	modulation	-	-	-	-	-	-	12.0	-
FsDetView [Xiao and Marlet, 2020]	modulation	24.2	42.2	57.4	-	-	-	12.5	14.7
CME [Li et al., 2021a]	finetuning	-	-	-	41.5	50.4	60.9	15.1	16.9
TIP [Li and Li, 2021]	add-on	27.7	43.3	59.6	-	-	-	16.3	18.3
DAnA [Chen et al., 2021a]	modulation	-	-	-	-	-	-	18.6	21.6
DeFRCN [Qiao et al., 2021]	prototype	-	-	-	53.6	61.5	60.8	18.5	22.6
Meta-DETR [Zhang et al., 2021b]	modulation	35.1	53.2	62.0	-	-	-	19.0	22.2
DETRReg [Bar et al., 2021]	finetuning	-	-	-	-	-	-	25.0	30.0

classification layer, which is randomly initialized, and finetuned on the target classes.⁹ Additionally, the authors propose to regularize the finetuning stage by penalizing the activation of background features with L_2 loss (Background Depression), and another “Transfer-Knowledge” loss which pulls target network predictions closer to the source network predictions. Subsequently, TFA or Frustratingly Simple Few-Shot Object Detection Wang et al. [2020a] showed that even a simple finetuning approach with minimal modifications to Faster R-CNN can actually yield competitive performance for FSOD. TFA replaces the fully-connected classification heads of Faster R-CNN with *cosine* similarities; the authors argue that such feature normalization leads to reduced intra-class variance, and less accuracy decrease on the base classes. First, a Faster R-CNN with cosine classification heads is trained on the base classes using the usual loss. During few-shot finetuning, new classification weights are randomly initialized for novel classes, appended to the base weights, and the last layers of the model are finetuned on the base+novel classes, while keeping the backbone and RPN frozen. MPSR [Wu et al., 2020a] is also a finetuning approach. They propose

⁹Slightly different from the FSOD framework presented in Section 4.1, the authors exclusively consider a cross-dataset scenario; therefore, target classes may or may not include sources classes.

an even more scale-aware Faster RCNN by combining the Feature Pyramid Network with traditional object pyramids [Adelson et al., 1984]. After training the model on the base classes, the classification layer is simply discarded, a new classification layer is initialized, and the model does few-shot finetuning on the base+novel classes without freezing any layers. RetentiveRCNN [Fan et al., 2021] extend TFA to generalized FSOD, where the goal is to perform well on the novel classes without losing performance on the base classes. They observe a “massive variation of norms between base classes and unseen novel classes”, which could explain why using cosine classification layers is better than fully connected ones. After training a Faster RCNN on the base classes, they freeze the base RPN and detection branches, and in parallel they introduce new finetuned RPN and detection branches to detect both base and novel classes. They also use a consistency loss, similar in spirit to LSTD’s transfer-knowledge loss, to make predictions on base objects more similar in the base/base-and-novel branches. DETReg [Bar et al., 2021] use a finetuning approach on the Deformable DETR [Zhu et al., 2021] architecture, and achieve state-of-the-art results on few-shot COCO object detection after proposing a self-supervised strategy for pretraining the detection heads, which is discussed in more depth in Section 5.3.

Conditioning-based methods

For clarity, we will refer to the image to process (to detect objects from) as the **query image**. In addition to the query image, conditioning-based methods are also fed with annotated **support images**, which are reference examples of each class to detect. Each support image generally has a single bounding-box around the object to detect. In the context of the FSOD framework presented in Section 4.1, support images are randomly sampled from all base classes during training (step 3), while a predefined (few-shot) set of base and novel images are used during finetuning and evaluation (steps 4 and 5). In this section, we review two types of conditioning-based methods: prototype-based, and modulation-based.

Prototype-based methods RepMet [Karlinsky et al., 2019], which stands for representative-based metric learning, is based on Faster-RCNN with Deformable Feature Pyramid Network (FPN), and learns representatives/prototypes for each object category. At base-training time, RepMet samples several supporting examples for each class, computes their ROI-pooled representations (representatives), and classifies object proposals according to their distance to the representatives. The gradients are propagated through both the proposals and the prototypes. At few-shot finetuning and evaluation time, representatives for the novel classes are computed, and objects proposals are classified using those new representatives. Optionally, the authors propose to finetune the novel prototypes by maximizing the detection loss on novel objects, which they find beneficial (denoted as “episode fine-

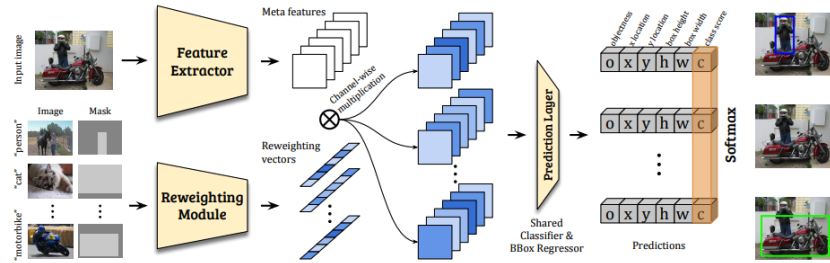


Figure 8.7: Meta-YOLO, a modulation-based FSOD method.

tuning” in Table 3 of [Karlinsky et al. \[2019\]](#)). ACM-MetaRCNN [[Wu et al., 2020b](#)] have also proposed a baseline which combines Faster R-CNN with prototypical networks by replacing the classification layer with the non-parametric prototypical network equivalent. They investigate this baseline with and without finetuning, and find that it is always beneficial to finetune.

Modulation-based methods Modulation-based methods generally compute **support weights** (also known as *class embeddings*, *weights*, *prototypes*, or *attentive-vectors*) from the support features using a separate **conditioning branch** (sometimes called *reweighting module* [[Kang et al., 2019](#)], *guidance module* [[Li and Li, 2021](#)], or *remodeling network* [[Yan et al., 2019](#)]). Each class has its own support weights, which are usually computed by applying global average-pooling to the support features and have a shape $1 \times 1 \times C$ shape, where C is the number of channels. The support weights then are multiplied channel-wise with the query features to obtain class-specific query features, in a process known as **modulation** or **aggregation**. Finally, binary detections are predicted (*object* vs. *background*) on each set of class-specific features, and the results are merged to get multiclass detections. For faster inference, support weights can be precomputed and stored during the finetuning/conditioning step.

To the best of our knowledge, one of the first modulation-based method was Meta-YOLO [Kang et al. \[2019\]](#), which is also the work that introduces the standardized FSOD splits (see [Figure 8.7](#)). Meta-YOLO uses both conditioning and finetuning. The model uses a feature extractor module to obtain the image features of the query image, and a reweighting module that extracts features from the bounding boxes of the support images to obtain reweighting vectors. The object mask binary matrices are added as an extra layer to the RGB images to form 4-channel images which are fed to the reweighting module. These vectors are used to condition the image features which allows the prediction layer to output the bounding boxes corresponding to the classes represented by those vectors. During base training, the two modules and the prediction layer are trained on the base classes. During few-shot finetuning, the model is finetuned using the K support examples per class,

from the base and novel classes.¹⁰ At few-shot evaluation time, reweighting vectors for each class are precomputed by averaging the corresponding vectors, and used to modulate the main features.

Since then, several improved conditioning-based methods have been introduced. Due to using different architectures and design choices, these works employ a diverse range of modulation strategies, which we discuss below. In **Meta-YOLO** [Kang et al., 2019], which is based on the single-stage detector YOLO, the features f_{qry} output by the backbone are directly multiplied channel-wise by the class embeddings f_{cls} , resulting in the modulated features $[f_{qry} \otimes f_{cls}]$. In **Meta-RCNN** [Yan et al., 2019] which is based on the two-stage detector Faster RCNN, the features are only multiplied after they have been pooled with RoIAlign; the consequence is that the Region Proposal Network (RPN) is agnostic to the object category. In **ACM-MetaRCNN** [Wu et al., 2020b], also based on Faster RCNN, the feature maps are multiplied twice by the class prototypes: before running the RPN, and after pooling the features of each box, which means the RPN may produce different region proposals for each class. In **Attention-FSOD** [Fan et al., 2020a], based on Faster RCNN, the query features are convolved channel-wise with support features (used as kernels) before feeding them to the RPN. In practice the authors find 1×1 support features to be optimal, and the convolution reduces to a channel-wise multiplication. The proposals are fed to a relational detection head, which classifies objects using matching scores by comparing query features with support features. Additionally, the authors explore three ways to model support-to-query relations in the detection head: globally, locally, and patch-wise. They find it beneficial to use all three types of relation heads. In **Fully Guided Network (FGN)** [Fan et al., 2020b], based on Faster RCNN, the support features are global-average-pooled and averaged for each class to obtain the class-attentive vectors. Then, the query features are multiplied channel-wise with the class-attentive vectors and fed to the RPN to obtain class-specific proposals, which are aggregated between the different classes. Finally, in the relational detection head, the aligned query features and N support class averages are concatenated altogether and fed into a MLP to obtain box and multiclass predictions. Relational detection heads [Sung et al., 2018a] have the ability to jointly predict boxes for all classes, which differs from other conditioning-based approaches which predict boxes independently for each class by relying on class-specific modulated features. In **FsDetView** [Xiao and Marlet, 2020], the query features are modulated by the support weights after ROI pooling. Instead of simply multiplying the features together, the authors propose to also concatenate and subtract them, resulting in the modulated features $[f_{qry} \otimes f_{cls}, f_{qry} - f_{cls}, f_{qry}]$. **Meta-DETR** [Zhang et al., 2021b] adapts FsDetView’s modulation strategy to the DETR architecture [Carion et al., 2020]. At the output of the backbone, both

¹⁰Since only K labeled bounding boxes are available for the novel classes, to balance between samples from the base and novel classes, only K boxes are included for each base class

support and query features are fed to the transformer encoder, then the query features are multiplied with global-average-pooled support vectors, and fed to the transformer decoder to make binary predictions.

Necessity of finetuning Only a few FSOD methods such as Li et al. [2020] or Attention-FSOD [Fan et al., 2020a] present themselves as methods that do not require finetuning. However, we should note that most if not all of the conditioning-based methods presented in the previous section could technically be used without finetuning on base+novel classes, by directly conditioning on the support examples at few-shot evaluation time. In practice, most works find it beneficial to finetune, and in fact many of the conditioning-based methods reviewed above do not even report numbers without finetuning. For instance, ACM-MetaRCNN, a conditioning-based model, finetune their model, except for 1-shot and 2-shot on PASCAL VOC, where they do not finetune “to avoid overfitting”. Even Attention-FSOD [Fan et al., 2020a], which claims to be usable without finetuning, achieves its best performance after finetuning (see for instance Table 8.3).

Add-on methods

Some FSOD methods do not propose a specific architecture, but instead propose add-on tricks that can be combined with many of the existing FSOD methods to boost performance. For instance, Transformation Invariant Principle (TIP) [Li and Li, 2021] is a regularization strategy based on data augmentation transformations, which can be applied to any two-stage FSOD method. Specifically, TIP proposes to minimize several consistency losses: the guidance vectors (class weights) for a support object (first view) and its transformed version (second view) should be close in feature space (the authors find the L2 distance to give best results). Additionally, TIP pools features from one view using proposals generated from another view; the resulting detections are used to compute another detection loss (regression and classification).

5 Self-Supervised Pretraining

Until recently, the standard approach in deep object detection was to pretrain the backbone on supervised ImageNet Deng et al. [2009] classification. This still holds for modern iterations of two-stage detectors such as Faster R-CNN Ren et al. [2015] – as per its `detectron2` implementation – as well as one-stage detectors such as YOLOv3 Redmon and Farhadi [2018], SSD Liu et al. [2016], RetinaNet Lin et al. [2017b] and recent transformer-based detectors like DETR Carion et al. [2020] and Deformable-DETR Zhu et al. [2021].

Self-supervised pretraining has emerged as an effective alternative to supervised pretraining where the supervision comes from the data itself. The key idea is to automatically generate labels from unlabeled data, and learning to predict those labels back. This process is known as solving a pretext task. For instance, a common pretext task is to predict the relative position of two random crops from the same image [Pathak et al., 2016]. This broad definition could potentially include many unsupervised methods such as VAEs [Kingma and Welling, 2014] and GANs [Goodfellow et al., 2014, Huang et al., 2017] but in practice the self-supervised term is used for methods for which the pretext task differs from the downstream task [He et al., 2020, Chen et al., 2020a, Grill et al., 2020]. Some language models such as word2vec [Mikolov et al., 2013] are also considered to be self-supervised.

Starting with SimCLR Chen et al. [2020a] and MoCo He et al. [2020], people have experimented initializing object detection backbones with unsupervised representations learned on ImageNet (or COCO) instead of supervised ones. Since the pretext tasks are fairly general, there is the hope that unsupervised representations might generalize better to downstream tasks than classification-based ones. Recent works [Wei et al., 2021a, Bar et al., 2021, Yang et al., 2021a, Wang et al., 2021] which we will refer to as *self-supervised object detection* methods go beyond backbone-pretraining by also pretraining the detection heads specifically for object detection.

In Section 5.1 we review self-supervised *classification* methods; then in Section 5.2 we discuss their limitations for initializing object detection backbones; finally in Section 5.3 we review self-supervised *object detection* approaches, which unlike the previous methods, are specifically tailored to object detection.

5.1 Image-level Backbone Pretraining

In the image classification domain, self-supervised learning has emerged as a strong alternative to supervised pretraining, especially in domains where images are abundant but annotations are scarce Mañas et al. [2021]. Self-supervised classification methods are not limited to classification, as the learned feature extractor can be used to initialize the backbone of common object detection architectures. We categorize backbone pretraining strategies into contrastive, clustering-based and self-distillative. We will omit reconstruction [Vincent et al., 2008, Pathak et al., 2016, Zhang et al., 2016, 2017] methods and visual common sense [Doersch et al., 2015, Noroozi and Favaro, 2016, Gidaris et al., 2018] based tasks, as to our knowledge, they have not been used for object detection.

Global Contrastive Learning \mathbf{X} ¹¹

These approaches leverage the InfoNCE [Oord et al., 2018] loss to compare pairs of samples which can be positive pairs or negative pairs. Usually, positive pairs are generated from different *views* (data augmentations) of the same image, while negative pairs are generated from different images. Contrastive Predictive Coding (CPC) is one of the first approaches to be competitive with respect to supervised classification [Oord et al., 2018, Henaff, 2020]. In CPC, the goal is to predict a future part of a sequential signal $p(x|c)$ given some previous context of that signal (c). Since reconstructing x from c is difficult in high-dimensional spaces, they propose a contrastive objective instead. Given a set of random samples, containing one *positive* sample $x^+ \sim p(x|c)$, and $N - 1$ *negative* samples $x_1^-, \dots, x_N^- \sim p(x)$ from the “proposal” distribution, they propose to learn a function $f_\theta(x, c)$ which minimizes the InfoNCE loss:

$$\mathcal{L}_{\text{InfoNCE}}(\theta) = -\mathbb{E}_X \left[\log \frac{f_\theta(x^+, c)}{\sum_i f_\theta(x_i^-, c)} \right]. \quad (5.1)$$

The density ratio $f_\theta(x, c)$ can be interpreted as an affinity score, which should be high for the real sample and low for negative samples.

More recent approaches such as momentum contrast (MoCo) [He et al., 2020, Chen et al., 2020c, 2021c], or SimCLR Chen et al. [2020a,b] reinterpret the InfoNCE loss in a different context. Given a reference image x_0 , positive samples $x^+ \sim p(x|x_0)$ are generated using data augmentation on x_0 , and negative samples $x_1^-, \dots, x_N^- \sim p(x)$ are other images sampled from the dataset. The InfoNCE loss becomes:

$$\mathcal{L}_{\text{InfoNCE}}(\theta) = -\mathbb{E}_X \left[\log \frac{f_\theta(x^+, x_0)}{\sum_i f_\theta(x_i^-, x_0)} \right]. \quad (5.2)$$

The goal is to learn representations which maximize the affinity $f_\theta(x, x_0)$ between different data-augmentations (**views**) of the same image, and minimize the affinity between different images.

While both MoCo and SimCLR find crucial to use a large set of negative examples, they differ in their approach to obtain them. SimCLR uses a large batch size, while MoCo uses smaller batches but stores the embeddings in a queue, which is used to retrieve negative examples. To prevent drift between queued embeddings and positive examples, MoCo encodes negative examples with an exponential moving average of the weights.

Most self-supervised pre-training methods in the literature aim to learn a global image representation to transfer to a given downstream task [He et al., 2020, Chen et al., 2020c, 2021c, 2020a, Caron et al., 2020, 2021]. However, global image

¹¹Use colors for quick reference to Table 8.4. The \mathbf{X} means backbone-only pretraining.

representations might not be optimal for dense prediction tasks such as detection and segmentation. In order to bridge the gap between self-supervised pre-training and dense prediction, [Pinheiro et al. \[2020\]](#) proposed VADeR, a pixel-level contrastive learning task for dense visual representation. Different from global contrastive learning approaches such as SimCLR, VADeR uses an encoder-decoder architecture. Then, given the output feature maps for a positive sample, an augmented sample, and a negative sample, the InfoNCE loss is applied between the decoder’s pixel features rather than being applied to the average of the decoder’s output. As a result, VADeR achieves encouraging results when compared to strong baselines in many structured prediction tasks, ranging from recognition to geometry.

Clustering-Based ✗ ¹²

Clustering-based methods rely on unsupervised clustering algorithms to generate pseudo-labels for training deep learning models [[Xie et al., 2016](#), [Yang et al., 2016](#)]. A key idea is to alternate between clustering learned representations, and using the predicted cluster assignments to improve representations in return. [Caron et al. \[2018, 2019\]](#) show that k-means cluster assignments are an effective supervisory signal for learning visual representations. [Asano et al. \[2019\]](#) show that cluster assignments can be solved as an optimal transport problem. Based on previous approaches, Swapping Assignments between multiple Views of the same image (SwAV) [[Caron et al., 2020](#)] was proposed. SwAV attempts to predict the cluster assignments for one view from another view (data augmentation) of the same image. It uses the Sinkhorn-Knopp algorithm for clustering [[Cuturi, 2013](#)], which has previously been explored for recovering labels in few-shot classification [[Huang et al., 2019](#)], and has good properties such as quick convergence and differentiability [[Cuturi, 2013](#)]. SwAV avoids trivial solutions where all features collapse to the same representation by using the appropriate amount of entropy regularization [[Caron et al., 2020](#)].

Knowledge Self-distillation (BYOL) ✗

Self-distillative approaches such as Bring Your Own Latent (BYOL) [[Grill et al., 2020](#)] and mean teacher [[Tarvainen and Valpola, 2017](#)] move away from contrastive learning by maximizing the similarity between the predictions of a teacher and a student model. The student model is optimized using SGD, while the teacher model is instantiated as an exponential moving average of the student weights [[Grill et al., 2020](#)]. In order to prevent them from collapsing to the same representation, [Grill et al. \[2020\]](#) found it helpful to use other tricks such as softmax sharpening and recentering. Subsequent approaches such as DINO [[Caron et al., 2021](#)] and EsViT [[Li et al., 2021b](#)] leverage vision transformers (ViT) [[Dosovitskiy et al.,](#)

¹²Use colors for quick reference to Table 8.4. Checkmarks ✓ and ✗ indicate whether the detection heads are pretrained.

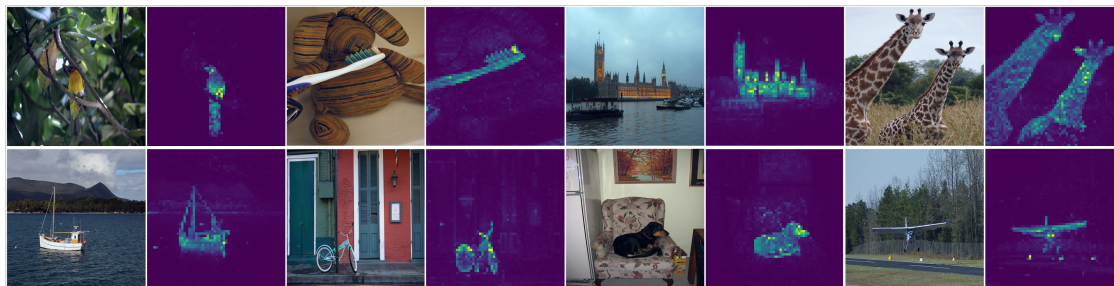


Figure 8.8: DINO’s attention maps. Since DINO is based on a visual transformer, the attention maps corresponding to the [CLS] token can be plotted. Despite being trained with no supervision, different attention heads are found to segment different objects. Source: Caron et al. [2019].

2021] instead of residual networks, following the trend of using self-supervision in natural language processing [Vaswani et al., 2017, Radford et al., 2018, Brown et al., 2020]. They divide the input image into a grid of small patches (8×8 pixels for DINO) and feed them to a ViT. The last feature map, which could be used as a dense representation, is average-pooled into a single vector and compared between teacher and student models using cross-entropy. The main difference between the two is that EsViT uses a two-stage architecture and a part-based loss. The first improvement reduces the amount of image patches in the second stage, which makes the model more efficient. The second improvement introduces an additional loss besides DINO’s teacher-student loss that encourages matching regions across multiple views to match their respective student and teacher representations.

Interestingly, the authors of DINO show that semantic segmentation masks naturally emerge from the attention masks of the visual transformer (see Figure 8.8). This is very surprising given that the model was trained with no supervision whatsoever. This suggests that combining dense pretext tasks such as VADeR [Pinheiro et al., 2020] and DINO [Caron et al., 2021] with attention-based models such as transformers could improve the transferability of self-supervised learning methods to dense downstream tasks.

5.2 Issues of Combining Self-Supervised Classification with Detection

Conceptually, there are some issues with transferring classification-based representations to object detectors, whether the representations were supervised or self-supervised.

Untrained Detection Heads

The first issue is with untrained detection heads, due to the architecture mismatch. For instance, it is a common practice to combine Resnet-50 and Resnet-101 backbones with a Feature Pyramid Network (FPN). However, the FPN is initialized from scratch and does not generally benefit from pretraining (though [Pinheiro et al. \[2020\]](#) propose to pretrain the FPN). The same goes for the Region Proposal Network (RPN) and the detection regression and classification heads, in the case of Faster RCNN; and for the encoder and decoder, in the case of DETR-style architectures.

Task Mismatch

Secondly, ImageNet top-1 classification accuracy does not necessarily correlate with object detection performance. While certain properties such as translation and scale invariance are desirable for classification, they might actually hinder object localization [[Newell and Deng, 2020](#), [Xiao et al., 2020](#)]. Classification-based representations such as MoCo [He et al. \[2020\]](#) or SimCLR [Chen et al. \[2020a\]](#) might discard spatial information that is useful for localization, because they are irrelevant for solving the pretraining tasks. Moreover, data augmentation strategies such as random cropping and jittering may introduce undesirable invariances into the network. In fact, [Yang et al. \[2021a\]](#) show that MoCo can perform better on object detection than BYOL and SwAV, despite having worse classification performance. Identifying whether the performance bottleneck comes from localization or classification errors is as hard of a problem as evaluating object detection itself. Metrics like mAP cannot disentangle localization and classification errors—AP is computed per-category, so if class predictions are wrong they will impact the localization error. mAP with higher IoU threshold put more emphasis to precise localization, but they are still contingent on having correct class assignments.

5.3 Object Detection Pretraining

Self-supervised object detection approaches attempt to remedy those issues by pretraining the object detection pipeline on a variety of unsupervised pretext tasks.

Predictive approaches ✓

Predictive approaches such as UP-DETR [Dai et al. \[2021\]](#) and DETReg [Bar et al. \[2021\]](#) pretrain the detection heads of DETR by making them re-predict the position of automatically generated “ground-truth” crops. These crops are generated either randomly for UP-DETR or using Selective Search [[Uijlings et al., 2013](#)] for DETReg, a training-free heuristic based on iteratively merging regions with similar colors,

Table 8.4: Comparison of Self-Supervised Object Detection Methods pretrained on unlabeled ImageNet. The check marks ✓ and ✗ refer to whether the methods pretrain both the backbone and detection heads vs. only the backbone.

Name	Pretrains Detector	Loss	View Matching	Region Proposal Mechanism	Pascal AP50	COCO AP	Best Backbone	Object Detector
DETReg [Bar et al., 2021]	✓	predictive ✓	-	selective search	83.3	45.5	R50	Def.DETR
SoCo [Wei et al., 2021a]	✓	BYOL ✓	crop	selective search	83.8	44.3	R50-FPN	F-RCNN
InsLoc [Yang et al., 2021a]	✓	contrastive ✓	crop	random crop	83.0	43.3	R50-C4	F-RCNN
UP-DETR [Dai et al., 2021]	✓	predictive ✓	-	random crop	80.1	42.8	R50-C4	DETR
ReSim [Xiao et al., 2021]	✓	contrastive ✓	sliding window	random crop	83.1	41.4	R50-FPN/C4	F-RCNN
DenseCL [Wang et al., 2021]	✓	contrastive ✓	feature	random crop	82.8	41.2	R50-FPN/C4	F-RCNN
VaDer [Pinheiro et al., 2020]	FPN-only	contrastive ✗	feature	-	-	39.2	R50-FPN	F-RCNN
EsViT [Li et al., 2021b]	✗	BYOL ✗	image	-	-	46.2	Swin-ViT	F-RCNN
DETRcon [Hénaff et al., 2021]	✗	contrastive ✗	mask	grid/FH/MCG	82.8	43.4	R50-FPN	F-RCNN
BYOL [Grill et al., 2020]	✗	BYOL ✗	image	-	81.0	42.3	R50-FPN/C4	F-RCNN
DI✗ [Caron et al., 2021]	✗	BYOL ✗	image	-	-	-	ViT/R-50	-
SwAV [Caron et al., 2020]	✗	clustering ✗	image	-	77.4	42.3	ResNet	F-RCNN
MoCo [Chen et al., 2020c]	✗	contrastive ✗	image	-	82.5	41.7	R50-FPN/C4	F-RCNN
SimCLR [Chen et al., 2020a]	✗	contrastive ✗	image	-	81.9	39.6	ResNet	F-RCNN

textures, and other local characteristics. ¹³

To pretrain DETR on unsupervised images, the multi-class classification heads (see Figure 8.3) which would normally predict the object category or *background* are replaced with a *binary* classification head. In DETReg, the goal is to detect the top proposals generated by Selective Search, as if they were ground-truth foreground objects. The usual DETR loss is used, except that the matching cost used to compute correspondences between ground-truth boxes and detections is a function of the predicted binary labels and locations – instead of ground truth and predicted multiclass labels. In UP-DETR, the goal is to predict back the positions of the random crops. Specifically, the transformer decoder is conditioned on the random crops by adding their corresponding features to the decoder input (see *object queries* in Figure 8.3). This is done by partitioning the object queries into K groups,¹⁴ and adding a different random crop to each group. The loss is computed by finding the optimal matching between the predicted boxes and the “ground-truth” random crops using the Hungarian algorithm [Munkres, 1957], where the cost of matching two boxes is a function of their location and predicted binary label.

On top of the DETR loss, an additional reconstruction loss is used to force the decoder transformer to reconstruct its input. DETReg uses a simple L1 loss

¹³Note that Selective Search used to be a popular training-free heuristic for generating high-recall low-precision region proposals, and was used in RCNN Girshick et al. [2014] and Fast RCNN Girshick [2015] before it was replaced by a trained Region Proposal Network (RPN).

¹⁴The way UP-DETR matches predictions and ground-truth from different groups instead of matching only within the same groups might not necessarily be an intended feature, but rather a consequence of building on top of existing DETR code. In practice, this does not make much difference as the groups are matched correctly (personal communication with the authors).

$\mathcal{L}_{rec}(z_i, z_j) = \|z_i - z_j\|_1$, while UP-DETR uses cosine similarity $\mathcal{L}_{rec}(z_i, z_j) = \left\| \frac{z_i}{\|z_i\|} - \frac{z_j}{\|z_j\|} \right\|_1$. Also, an interesting by-product of UP-DETR is that it learns a conditioning branch, which can be reused directly for one-shot object detection by replacing the random crops with support images. The paper provides some results on PASCAL VOC Dai et al. [2021].¹⁵

Local Contrastive Learning ✓

Unlike global contrastive methods (Section 5.1) which contrast global representations at the image level, local contrastive methods contrast backbone representations locally, either at the *feature* or *crop* level, with the hope of learning location-aware representations. Some of them, such as InsLoc [Yang et al., 2021a], also pretrain detection heads.

In InsLoc [Yang et al., 2021a], features for each crop are computed using RoIAlign [Ren et al., 2015], then transformed by the RoI heads to a single $1 \times 1 \times d$ vector. Positive pairs are generated by randomly cropping two views of the same image, while negative pairs are generated by using different images. Specifically in InsLoc, positive pairs are generated by pasting two views of the same object (the foreground) at random locations and scales onto other images of the dataset (the background). The authors also introduce a cut-and-paste scheme in order to force the receptive field of RoIAlign to ignore distractor features outside the bounding box.

In ReSim [Xiao et al., 2021], two overlapping crops are generated from two different views of the same image. Then, a sliding window is moved across the overlapping area, and the pooled representations are compared at each of the final convolutional layers. Positive pairs consist of aligned sliding windows across two views of the same image, while negative pairs either consist of unaligned sliding windows, or sliding windows from two different images.

In DenseCL [Wang et al., 2021], instead of using spatial correspondence, positive pairs are generated by matching each feature from one view to the feature with highest cosine similarity in another view of the same image. Negative examples are simply features from different images. Additionally, the authors combine this dense loss with a global MoCo-style loss, which they claim is necessary to bootstrap correct correspondences.

Self-distillative approaches (BYOL) ✓

Self-distillative (BYOL-based) approaches such as SoCo Wei et al. [2021a] depart significantly from contrastive approaches as there is no need for negative examples.

¹⁵Not reported in Table 8.4 due to using different splits.

Selective object COnstrastive learning (SoCo) builds on top of BYOL Grill et al. [2020] and pretrains both the backbone, feature pyramid, and RoI heads of a FPN-based Faster RCNN by training two networks simultaneously. The “student” network f_θ uses SGD to copy the feature maps of a “teacher” network f_ξ , which is an exponential moving average of the student network, and the student network is optimized using SGD. For a given image, object proposals (denote the bounding boxes b) are generated unsupervisedly using Selective Search [Uijlings et al., 2013]. Then, two views V_1, V_2 are generated and respectively fed into student and teacher FPNs to get feature maps v_1, v_2 . Box features are computed for each bounding box b by pooling v_1, v_2 with RoIAlign and passing them to the RoI heads:

$$h_1 = f_\theta^H(\text{RoIAlign}(v_1, b)), \quad h_2 = f_\xi^H(\text{RoIAlign}(v_2, b)).$$

The box features h_1, h_2 are then projected to obtain latent embeddings e_1, e_2 , and their cosine similarity is minimized

$$\mathcal{L}(\theta) = -\frac{\langle e_1, e_2 \rangle}{\|e_1\|_2 \cdot \|e_2\|_2}.$$

In practice, SoCo introduces several other tricks, such as using more than two views and resizing them at multiple scales, jittering the proposed box coordinates, and filtering proposals by aspect ratio Wei et al. [2021a].

5.4 Comparison of Self-Supervised Object Detection Methods

In Table 8.4, we review the self-supervised object detection methods discussed previously, and report their performance on PASCAL VOC and MS COCO object detection (non few-shot). Note that the numbers are not directly comparable in absolute value, due to variations in model architectures, hyperparameters, learning rate schedules, data augmentation schemes, and other implementation details. Instead, we encourage the reader to dig into the corresponding ablation studies of those works.

This table contains methods specifically geared towards object detection as presented in Section 5.3, which train the backbone (“Pretrains Detector: Yes”), and general purpose representations as presented in Section 5.1 which only pretrain the backbone on top of which an object detector was fitted a posteriori, often by a subsequent work (“Pretrains Detector: No”). For instance, most of the numbers for DenseCL, BYOL, DETcon, MoCo, SimCLR and SwAV are taken from the SoCo paper [Wei et al., 2021a]. VaDer is in between, as it does not pretrain detection heads but does pretrain a feature pyramid network (FPN) alongside the backbone.

The methods can be categorized into four types of losses: self-distillative (BYOL), predictive (predictive), contrastive (contrastive) and clustering-based

(clustering). The check marks ✓ and ✗ refer to whether these methods also pretrain the detection heads. **View Matching** refers to the way different views of the same image are matched. From most global to most local: image > crop, mask > sliding window > feature. *Region proposal mechanism* describes how unsupervised regions are generated for the purpose of pretraining (not to be confused with the candidate proposals in two-stage detectors). DETcon generates masks from: “grid” a fixed-size grid, “FH” the Felzenszwalb-Huttenlocher algorithm [Felzenszwalb and Huttenlocher, 2004], or “MCG” Multiscale Combinatorial Grouping [Arbeláez et al., 2014]. “R50-FPN” means ResNet-50 with Feature Pyramid Network. “R50-C4” means using ResNet-50’s C4 layer. “ViT” is the visual transformer [Dosovitskiy et al., 2021]. “Swin” is a type of hierarchical visual transformer [Liu et al., 2021b]. “F-RCNN” stands for Faster R-CNN, “Def. DETR” for deformable DETR.

Many FSOD methods [Wang et al., 2021, Xiao et al., 2021, Grill et al., 2020, He et al., 2020] are found to perform better using multi-scale features with a FPN for MS COCO, but with single-scale C4 features for PASCAL VOC, which may be a consequence of its limited size.

6 Takeaways & Trends

We discuss our main takeaways and forecasted trends from this survey.

6.1 Finetuning is a strong baseline

Almost every few-shot object detection method we have reviewed finetunes on the novel classes. This is the case even for conditioning-based methods, which could technically be used without finetuning by conditioning on the support examples, but have been found to benefit from finetuning anyways. The problem is that finetuning approaches are slower and may require more hyperparameter tuning. This could be a serious obstacle to deploying such methods in the real world. In general, we hope to more see competitive finetuning-free methods in the future.

6.2 Impact of self-supervision for object detection

It is somewhat surprising that self-supervised object detection pretraining only brings limited improvements for traditional object detection. This could be explained by the fact that post-pretraining, the object detector is finetuned on the labeled dataset, which could render self-supervision redundant. It could also be that current experiments are mainly limited to ImageNet and MS COCO pretraining, whilst self-supervision could potentially benefit from larger unlabeled datasets. However, the impact of self-supervised pretraining seems to be more significant for few-shot

and low-data object detection. In fact, the state-of-the-art FSOD results on MS COCO are from DETReg [Bar et al., 2021], a self-supervised object detection method.

6.3 Using heuristics to generate weak labels

A general trend in self-supervised learning is to use heuristics to generate weak or noisy labels. Data augmentations are now widely used for generating positive pairs in the context of self-supervised classification [He et al., 2020, Chen et al., 2020a, Grill et al., 2020]. Specifically to object detection, DETReg [Bar et al., 2021] and SoCo [Wei et al., 2021a] have adopted Selective Search [Uijlings et al., 2013], for generating crops which are more likely to contain objects. On the other hand, DetCon [Hénaff et al., 2021] have explored using the Felzenszwalb-Huttenlocher algorithm and Multiscale Combinatorial Grouping to generate better segmentation masks for feature pooling. Since these heuristics come from traditional computer vision, we expect practitioners to continue adapting more of them to improve self-supervised training in the future. An important question is how such heuristics can be integrated in an iterative bootstrapping procedure: as better representations are learned, it might be worthwhile to gradually replace the initial heuristics with learned and improved ones (e.g., replacing selective search with a learned RPN). One possible direction of research could be to develop differentiable/learnable versions of these traditional heuristics.

6.4 Rise of transformers

Visual transformers have gained increasing traction in object detection, both as backbones and as end-to-end detection heads. For using them as backbones, works such as DINO [Caron et al., 2021] have shown that fully unsupervised pretraining of visual transformers can lead to the emergence of object segmentation capabilities. Specifically in their case, the multi-head attention modules learn to segment foreground objects as a byproduct of solving the pretext task, as shown in Figure 8.8. More generally, there is growing belief from the study of scaling laws for foundational models that visual transformers can generalize better than ResNets to large scale training [Zhai et al., 2021]. Some self-supervised methods, such as EsViT [Li et al., 2021b], also rely on recent iterations of visual transformers such as Swin [Liu et al., 2021b] to obtain state-of-the-art results. When using them as detection heads, DETR [Carion et al., 2020] has shown that transformer-based detection heads can be trained end-to-end. In particular, they are capable of making joint predictions and dealing with redundant detections, thus removing the need to rely on heuristics such as non-maximum suppression (NMS). More recent work such as Pix2Seq [Chen et al., 2021b] has shown that object detection can be formulated as a language modeling task. The flexibility that comes with

language modeling could blur the line between pure vision tasks (such as object detection) and vision-to-language tasks (such as image captioning or visual question answering), lead to simpler architectures, more end-to-end approaches with less heuristics (such as NMS), and pave the way to foundational models for vision and language tasks.

6.5 Problems with current evaluation procedures

Comparisons such as Table 8.3 and Table 8.4 should only be used to get a general idea of the performance of those systems. The numbers themselves often not directly comparable, due to variations in backbone architecture, use of multi-scale features (FPN), varying detection architectures, types of data augmentations used, learning rate scheduling, or even things as trivial as input image size resizing.

Differences in implementation details

In fact, many of the modulation-based FSOD methods we have reviewed in Section 4.5 have quite a similar structure, differing only in the modulation strategy, backbone architecture and object detector used. This raises the question of how much of the performance of state-of-the-art methods is owed to new ideas rather than better hyperparameter tuning or using better architectures. One way would be to see how much performance we can get with running older methods in newer frameworks, or building an unifying benchmark.

Issues with data splits

Specifically to the FSOD use of PASCAL VOC, there has been a shift from using Kang’s splits to TFA’s splits which were introduced later to alleviate the variance problems with Kang’s splits. Despite the fact that the two splits can yield wildly different numbers (see for instance the line on TFA w/cos), several works mistakenly mix them up in the same tables [Xiao and Marlet, 2020, Li and Li, 2021, Zhang et al., 2021b]. More generally, the fact that virtually every FSOD paper – regular object detection papers too – trains on the union of training and validation sets (trainval) and uses the test set for hyperparameter tuning can lead to overfitting and overestimating the actual generalization performance.

Proposed guidelines

Therefore, we propose the following guidelines for having more comparable results:

1. Define and use proper train/val/test splits. Researchers should agree on newer splits or benchmarks, as no single researcher has any incentive to stop overfitting on the test set.

-
2. Do not propose benchmarks which are prone to high variance, such as Kang’s splits for Pascal or TFA splits for LVIS. Prior work on few-shot classification has consistently provided confidence intervals by averaging results over multiple episodes, and sampling the few-shot training set instead of fixing the instances [Snell et al., 2017a, Chen et al., 2019a, Vinyals et al., 2016a].
 3. Standardize implementation details such as image resizing, whitening, and data augmentations. Define standard backbone and detector architectures to be explored for each benchmark. Results could be presented in two categories: fixed architecture and best architecture. The introduction of Detectron2 has already led to more standardization and code sharing, providing among other things a standard implementation of Faster R-CNN with FPN.¹⁶
 4. Relate new tasks to existing tasks. For instance, the dominant FSOD and few-shot classification frameworks use different terminologies, training and evaluation procedures, and FSOD could have benefited from FSC best practices. We found it necessary to clarify the differences and subtleties in Section 4.2.

7 Related Tasks

We briefly discuss other related tasks, which are out of the scope of this survey.

7.1 Weakly-supervised object detection

Image-level and point-level annotations are cheaper and faster to obtain, and noisy image-level labels could even be generated automatically using image search engines. Weakly supervised object detectors are trained using only image-level annotations without requiring bounding boxes [Bilen and Vedaldi, 2016, Jie et al., 2017, Tang et al., 2018] and could therefore benefit from a larger pool of labels. Many weakly supervised detection methods fall under multiple-instance learning (MIL) Dietterich et al. [1997] where each image corresponds to a bag of object proposals. A bag is given the class label based on whether the label exists in the image. Li et al. [2016] present a two-step approach that includes selecting good object proposals; then training a Faster RCNN Ren et al. [2015]. Tang et al. [2017] use a refinement learning strategy to select good quality proposals. C-MIL Wan et al. [2019] introduces an optimization method to avoid selecting poor proposals, C-WSL Gao et al. [2018] uses object count information to obtain the highest scoring proposals, WISE Laradji et al.

¹⁶Despite the valuable efforts of Detectron2 towards standardization and open-sourcing, we did find the framework overwhelming for some use-cases. This resulted in additional difficulties when working with Detectron2-based projects due to the highly abstract nature of the framework. We hope that future frameworks will be more user-centric. For instance, a micro-framework with independently-usable modules might lead to more readable user code.

[2019b] that uses class activation maps to score proposals and LOOC Laradji et al. [2020a] can be used to detect objects in crowded scenes. Point-level annotations can also be used for object detection like in Laradji et al. [2019a, 2021b, 2020b], but they require slightly more human effort.

7.2 Self-supervision using other modalities

Instructional videos are a natural source of self-supervision, as they contain both speech and visual information. For instance, Amrani et al. [2020] recently proposed using unlabeled narrated instructional videos to learn an object detector by exploiting the correlations between narration and video. They start by extracting video transcripts with an external method. For a given object, they generate positive frames from the temporal period where the object is mentioned, and negative frames from videos that do not mention the object. They extract bounding boxes using Selective Search [Uijlings et al., 2013], compute their features using a pretrained backbone, cluster them in feature space, assign a score to each cluster, and filter out the noisiest examples. Finally, they train a detector on the remaining bounding boxes. Laradji et al. [2021a] deal with 3D ct scans using self-supervision and weak supervision to train a model to predict regions in the lungs that are infected with COVID. Liu et al. [2019a] used self-supervision by making sure outputs are consistent between different viewpoints and augmentations which in turn helped improve 2D to 3D reconstruction.

7.3 Low-data and semi-supervised object detection

Low-data object detection (LSOD) and semi-supervised object detection (SemiOD) are closely related to few-shot object detection (FSOD). Instead of having a distinction between base classes (many examples) and novel classes (few-shot), LSOD and SemiOD both assume that the number of examples for all classes is limited—which is generally simulated by considering a fraction of the labels of traditional object detection datasets (e.g. *miniCOCO* is a 1%, 5% or 10% subset of MS COCO). Several of the self-supervised object detection methods reviewed in this survey report numbers in the low-data regime; see for instance the results on *miniCOCO* of DETReg [Bar et al., 2021], SoCo [Wei et al., 2021a], InsLoc [Yang et al., 2021a], DenseCL [Wang et al., 2021]. Compared to LSOD, SemiOD methods generally leverage additional unlabeled datasets. For instance, Adaptive Class-Rebalancing [Zhang et al., 2021a] and SoftTeacher [Xu et al., 2021b] report results on *miniCOCO* but also leverage additional unlabeled MS COCO images to improve performance on the regular MS COCO benchmark.

7.4 Few-shot semantic segmentation

Few-shot object detection has many similarities to few-shot semantic segmentation as they both require us to identify the objects of interest in the images. However, semantic segmentation only considers the class of individual pixels and does not require to identify the individual objects in the image. Semantic segmentation models tend to be simpler as they are usually based on architectures that only have a downsampling path and an upsampling path Long et al. [2015], Ronneberger et al. [2015], as opposed to a proposal generator and additional networks for classification and regression Girshick [2015]. Further, people have explored few-shot semantic segmentation using weaker labels than the full segmentation masks. For instance, Rakelly et al. [2018], Siam et al. [2020] allow annotators to label only few pixels per object in the support or image-level labels for meta-testing. Using weakly supervised methods for few-shot object detection is an interesting direction that is fairly unexplored.

7.5 Zero-shot object detection

Zero-shot object detection and instance segmentation are about learning to detect (resp. segment) novel objects based on a non-visual descriptions of them. These descriptions could be in the form of semantic attributes, such as “long tail” and “orange beak” in the case of bird classification. In practice, the attribute vector often consists of pretrained word embeddings, since those are readily available and contain implicit world knowledge from large unlabeled datasets. When using word embeddings, a common strategy is to modify existing object detection/instance segmentation heads by projecting object feature maps to have same dimensionality as word embeddings. This is the case of ViLD [Gu et al., 2021], ZSI [Zheng et al., 2021], BLC [Zheng et al., 2020b], PL [Rahman et al., 2020], DSES [Bansal et al., 2018], who propose many tricks to improve performance, such as distilling pretrained vision-language models like CLIP [Radford et al., 2021], learning improved word embeddings for “foreground” and “background” for the RPN, using separate pathways for seen and unseen classes to avoid catastrophic forgetting, and explore different ways to mingle semantic and visual information. Overall, several innovations from zero-shot classification and object detection, few-shot object detection, and instance segmentation could be shared in the future.

8 Conclusion

We have formalized the few-shot object detection framework and reviewed the main benchmarks and evaluation metrics. We have categorized, reviewed, and compared several few-shot and self-supervised object detection methods. Finally, we

have summarized our main takeaways, made future best practice recommendations, highlighted trends to follow, and given pointers to related tasks.

Acknowledgments

This research was partially supported by the Canada CIFAR AI Chair Program, the NSERC Discovery Grant RGPIN-2017-06936, by project PID2020-120611RB-I00/AEI/10.13039/501100011033 and by Mitacs through the Mitacs Accelerate program. Simon Lacoste-Julien is a CIFAR Associate Fellow in the Learning in Machines & Brains program.

9

Prologue to the Fourth Contribution

1 Article Details

Multimodal Pretraining for Dense Video Captioning by *Gabriel Huang, Bo Pang, Zhenhai Zhu, Clara Rivera, Radu Soricut*. This paper has been accepted at ACL-IJCNLP 2020.

2 Contributions of the authors

This project was done while Gabriel Huang was an intern at Google Research in the GARCON team. Bo Pang led the creation of the ViTT dataset, defined the project and mentored Gabriel Huang; Zhenhai Zhu helped with the code, data processing, and launching experiments; Clara Rivera contributed to creating the ViTT dataset; Radu Soricut supervised the team and edited the paper. All authors helped write the paper. Gabriel Huang led the project, wrote a significant part of the paper, and conducted most experiments.

3 Context and Limitations

This project was motivated by the goal of helping users navigate tutorial videos more easily, by automatically generating captions for each instructional step, such as “beat eggs” or “put cake into oven”. Compared to previous work, our approach has certain advantages, such as taking multimodal inputs (augmenting video with speech-to-text) and relying on self-supervised pretraining, thus reducing requirements on labeled data. However, a limitation of our method is to assume that the video is already time-segmented into instructional steps, a challenging task by itself, which should ideally be performed jointly with video captioning. Another limitation, due to practical constraints at the time of the project, lies in the use of precomputed image and video features, which should ideally be learned end-to-end on the downstream and pretraining tasks. Finally, another limitation is the use of an external speech-to-text engine, which discards non-verbal cues and sounds that may contain important clues.

4 Recent Developments

There has been subsequent progress in vision-language architectures and pretraining/masking schemes [Shin et al., 2022] leading to improved results on video captioning. Some of these progresses have come from separate improvements in vision or language. For instance, the success of Vision Transformers (ViT) [Dosovitskiy et al., 2021] and the DETR [Carion et al., 2020] object detector have shown the potential of transformers in the vision domain. Similarly, there has been a lot of progress on language models from scaling them up to unprecedented model and dataset sizes [Brown et al., 2020], [Sanh et al., 2021], [Chowdhery et al., 2022]. Because of the prohibitive pretraining costs associated with learning state-of-the-art language models, cross-modal methods, such as *Frozen* [Tsimpoukelli et al., 2021] that can reuse frozen language models are becoming increasingly relevant.

Recently, *Flamingo* [Alayrac et al., 2022] was proposed, which combines a pretrained 70B parameter language model known as *Chinchilla* [Hoffmann et al., 2022] with a variant of ResNet known as NFNet [Brock et al., 2021], separately pretrained on a CLIP-style contrastive objective [Radford et al., 2021]. During the main training phase both models are frozen; the language model blocks are interleaved with learnable cross-attention blocks which attend to the visual features. The model can handle interleaved text-image-video data, and can solve tasks such as video captioning and visual question answering. Although their results are still behind ours and the state-of-the-art on YouCook2 [Xu et al., 2021a], the flexibility of their approach represents a promising direction towards foundational vision-language models and solving generic zero and few-shot vision-language tasks.

10

Multimodal Pretraining for Dense Video Captioning

Abstract

Learning specific hands-on skills such as cooking, car maintenance, and home repairs increasingly happens via instructional videos. The user experience with such videos is known to be improved by meta-information such as time-stamped annotations for the main steps involved. Generating such annotations automatically is challenging, and we describe here two relevant contributions. First, we construct and release a new dense video captioning dataset, **Video Timeline Tags (ViTT)**, featuring a variety of instructional videos together with time-stamped annotations. Second, we explore several multimodal sequence-to-sequence pretraining strategies that leverage large unsupervised datasets of videos and caption-like texts. We pretrain and subsequently finetune dense video captioning models using both YouCook2 and ViTT. We show that such models generalize well and are robust over a wide variety of instructional videos.

1 Introduction

YouTube recently reported that a billion hours of videos were being watched on the platform every day [YouTubeBlog, 2017]. In addition, the amount of time people spent watching online videos was estimated to grow at an average rate of 32% a year between 2013 and 2018, with an average person forecasted to watch 100 minutes of online videos per day in 2021 [ZenithMedia, 2019].

An important reason for this fast-growing video consumption is information-seeking. For instance, people turn to YouTube “hungry for how-to and learning content” O’Neil-Hart [2018]. Indeed, compared to traditional content format such as text, video carries richer information to satisfy such needs. But as a content media, videos are also inherently more difficult to skim through, making it harder to quickly target the relevant part(s) of a video. Recognizing this difficulty, search engines started showing links to “key moments” within videos in search results, based on timestamps and short descriptions provided by the content creators themselves.¹

¹<https://www.blog.google/products/\search/key-moments-video-search/>



Groundtruth	<i>Varying stitching speeds</i>
Ø-Pretraining	<i>Showing other parts</i>
MASS-Pretraining	<i>Explaining how to do a stitch</i>

Figure 10.1: Dense video captioning using ViTT-trained models. For the given video scene, we show the ViTT annotation (Groundtruth) and model outputs (no pretraining and MASS-based pretraining).

This enables users to get a quick sense of what the video covers, and also to jump to a particular time in the video if so desired. This effort echoes prior work in the literature showing how users of instructional videos can benefit from human-curated meta-data, such as a timeline pointing to the successive steps of a tutorial Kim et al. [2014], Margulieux et al. [2012], Weir et al. [2015]. Producing such meta-data in an automatic way would greatly scale up the efforts of providing easier information access to videos. This task is closely related to the dense video captioning task considered in prior work Zhou et al. [2018b,c], Krishna et al. [2017], where an instructional video is first segmented into its main steps, followed by segment-level caption generation.

To date, the YouCook2 data set Zhou et al. [2018b] is the largest annotated data set for dense video captioning. It contains annotations for 2,000 cooking videos covering 89 recipes, with per-recipe training / validation split. Restricting to a small number of recipes is helpful for early exploratory work, but such restrictions impose barriers to model generalization and adoption that are hard to overcome. We directly address this problem by constructing a larger and broader-coverage annotated dataset that covers a wide range of instructional topics (cooking, repairs, maintenance, etc.) We make the results of our annotation efforts publicly available as **Video Timeline Tags (ViTT)**², consisting of around 8,000 videos annotated with timelines (on average 7.1 segments per video, each segment with a short free-text description).

Using YouCook2 and the new ViTT dataset as benchmarks for testing model performance and generalization, we further focus on the sub-problem of video-

²Available at <https://github.com/google-research-datasets/Video-Timeline-Tags-ViTT>

segment-level caption generation, assuming segment boundaries are given Hessel et al. [2019], Sun et al. [2019b], Luo et al. [2020]. Motivated by the high cost of collecting human annotations, we investigate pretraining a video segment captioning model using unsupervised signals – ASR (Automatic Speech Recognition) tokens and visual features from instructional videos, and *unpaired* instruction steps extracted from independent sources: Recipe1M Marin et al. [2019] and WikiHow Koupae and Wang [2018]. In contrast to prior work that focused on BERT-style pretraining of encoder networks [Sun et al., 2019b,a], our approach entails jointly pretraining both multimodal encoder and text-based decoder models via MASS-style pretraining Song et al. [2019]. Our experiments show that pretraining with either text-only or multimodal data provides significant gains over no pretraining, on both the established YouCook2 benchmark and the new ViTT benchmark. The results we obtain establish state-of-the-art performance on YouCook2, and present strong performance numbers on the ViTT benchmark. These findings help us conclude that the resulting models generalize well and are quite robust over a wide variety of instructional videos.

2 Related Work

Text-only Pretraining. Language pretraining models based on the Transformer neural network architecture Vaswani et al. [2017] such as BERT [Devlin et al., 2018], GPT Radford et al. [2018], RoBERTa Liu et al. [2019b], MASS Song et al. [2019] and ALBERT Lan et al. [2020] have achieved state-of-the-art results on many NLP tasks. MASS [Song et al., 2019] has been recently proposed as a joint encoder-decoder pretraining strategy. For sequence-to-sequence tasks, this strategy is shown to outperform approaches that separately pretrain the encoder (using a BERT-style objective) and the decoder (using a language modeling objective). UniLM Dong et al. [2019], BART Lewis et al. [2019], and T5 Raffel et al. [2019] propose unified pretraining approaches for both understanding and generation tasks.

Multimodal Pretraining. VideoBERT Sun et al. [2019b], CBT Sun et al. [2019a] and ActBERT Zhu and Yang [2020] use a BERT-style objective to train both video and ASR text encoders. Alayrac et al. [2016] and Miech et al. [2020] use margin-based loss functions to learn joint representations for video and ASR, and evaluate them on downstream tasks such as video captioning, action segmentation and anticipation, and action localization. An independent and concurrent work (UniViLM) by Luo et al. [2020] is closely related to ours in that we share some similar pretraining objectives, some of the pretraining setup – HowTo100M Alayrac et al. [2016], and the down-stream video captioning benchmark using YouCook2 Zhou et al. [2018b]. The main difference is that they use BERT-style pretraining for encoder and language-modeling style pretraining for decoder, whereas we use MASS-style pre-training to pretrain encoder and decoder jointly.

Other approaches such as ViLBERT [Lu et al., 2019], LXMERT [Tan and Bansal, 2019], Unicoder-VL [Li et al., 2019a], VL-BERT [Su et al., 2019], and UNITER [Chen et al., 2019b] focus on pretraining joint representations for text and image, evaluating them on downstream tasks such as visual question answering, image-text retrieval and referring expressions.

Dense Video Captioning. In this paper, we focus on generating captions at the segment-level, which is a sub-task of the so-called dense video captioning task Krishna et al. [2017], where fine-grained captions are generated for video segments, conditioned on an input video with pre-defined event segments. This is different from the video captioning models that generate a single summary for the entire video Wang et al. [2019a].

Hessel et al. [2019] make use of ASR and video for segment-level captioning on YouCook2 and show that most of the performance comes from ASR. Shi et al. [2019], Luo et al. [2020] train their dense video captioning models on both video frames and ASR text and demonstrate the benefits of adding ASR as an input to the model. There are also a number of video captioning approaches that do not use ASR directly Zhou et al. [2018c], Pan et al. [2020], Zheng et al. [2020a], Zhang et al. [2020], Lei et al. [2020].

Instructional video captioning data sets. In addition to YouCook2 Zhou et al. [2018b], there are two other smaller data sets in the instructional video captioning category. Epic Kitchen Damen et al. [2018] features 55 hours of video consisting of 11.5M frames, which were densely labeled for a total of 39.6K action segments and 454.3K object bounding boxes. How2 Sanabria et al. [2018] consists of instructional videos with video-level (as opposed to segment-level) descriptions, authored by the video creators themselves.

3 Data

We present the datasets used for pretraining, finetuning, and evaluation in Table 10.1. We also describe in detail the newly introduced dense video captioning dataset, **V**ideo **T**imeline **T**ags (ViTT).

3.1 Dense Video-Captioning Datasets

Our goal is to generate captions (CAP) for video segments. We consider two datasets with segment-level captions for fine-tuning and evaluating ASR+Video→CAP models.

Name	Type	# segments
<i>Pretraining datasets</i>		
YT8M-cook	ASR+video	186 K
HowTo100M	ASR+video	8.0 M
Recipe1M	CAP-style	10.8 M
WikiHow	CAP-style	1.3 M
<i>Finetuning datasets</i>		
YouCook2	ASR+video+CAP	11.5 K
ViTT-All	ASR+video+CAP	88.5 K

Table 10.1: Datasets used in this work, along with size of the data measured by the total number of segments.

YouCook2. Up to this point, YouCook2 [Zhou et al., 2018b] has been the largest human-annotated dense-captioning dataset of instructional videos publicly available. It originally contained 2,000 cooking videos from YouTube. Starting from 110 recipe types (e.g., “shrimp tempura”), 25 unique videos per recipe type were collected; the recipe types that did not gather enough videos were dropped, resulting in a total of 89 recipe types in the final dataset. In addition, Zhou et al. [2018a] “randomly split the videos belonging to each recipe into 67%:23%:10% as training, validation and test sets³,” which effectively guarantees that videos in the validation and test sets are never about unseen recipes. Annotators were then asked to construct recipe steps for each video — that is, identify the start and end times for each step, and provide a recipe-like description of each step. Overall, they reported an average of 7.7 segments per video, and 8.8 words per description. After removing videos that had been deleted by users, we obtained a total of 11,549 segments.

ViTT. One limitation of the YouCook2 dataset is the artificially imposed (almost) uniform distribution of videos over 89 recipes. While this may help making the task more tractable, it is difficult to judge whether performance on its validation / test sets can be generalized to unseen topics.

The design of our ViTT dataset annotation process is aimed at fixing some of these drawbacks. We started by collecting a large dataset of videos containing a broader variety of topics to better reflect topic distribution in the wild. Specifically, we randomly sampled instructional videos from the YouTube-8M dataset [Abu-El-Haija et al., 2016], a large-scale collection of YouTube videos that also contain topical labels. Since much of prior work in this area revolved around cooking videos, we aimed at sampling a significant proportion of our data from videos with cooking

³Note that no annotations are provided for the test split; we conducted our own training/dev/test split over available videos.

labels (specifically, “Cooking” and “Recipe”). Aside from the intentional bias regarding cooking videos, the rest of the videos were selected by randomly sampling non-cooking videos, including only those that were considered to be instructional videos by our human annotators.

Once candidate videos were identified, timeline annotations and descriptive tags were collected. Our motivation was to enable downstream applications to allow navigating to specific content sections. Therefore, annotators were asked to identify the main steps in a video and mark their start time. They were also asked to produce a descriptive-yet-concise, free-text tag for each step (e.g., “shaping the cookies”, “removing any leftover glass”). A subset of the videos has received more than one complete annotation (main steps plus tags).

The resulting ViTT dataset consists of a total of 8,169 videos, of which 3,381 are cooking-related. A total of 5,840 videos have received only one annotation, and have been designated as the training split. Videos with more than one annotation have been designated as validation / test data. Overall, there are 7.1 segments per video on average (max: 19). Given the dataset design, descriptions are much shorter in length compared to YouCook2: on average there are 2.97 words per tag (max: 16) — 20% of the captions are single-word, 22% are two-words, and 25% are three words. Note that the average caption length is significantly shorter than for YouCook2, which is not surprising given our motivation of providing short and concise timeline tags for video navigation. We standardized the paraphrases among the top-20 most frequent captions. For instance, {“intro”, “introduction”} → “intro”. Otherwise, we have preserved the original tags as-is, even though additional paraphrasing most definitely exists. Annotators were instructed to start and end the video with an opening and closing segment as possible. As a result, the most frequent tag (post-standardization) in the dataset is “intro”, which accounts for roughly 11% of the 88,455 segments. More details on the data collection process and additional analysis can be found in the Supplementary Material (Section 1).

Overall, this results in 56,027 unique tags, with a vocabulary size of 12,509 token types over 88,455 segments. In this paper, we consider two variants: the full dataset (ViTT-All), and the cooking subset (ViTT-Cooking).

3.2 Pretraining Datasets: ASR+Video

We consider two large-scale unannotated video datasets for pretraining, as described below. Time-stamped ASR tokens were obtained via YouTube Data API,⁴ and split into ASR segments if the timestamps of two consecutive words are more than 2 seconds apart, or if a segment is longer than a pre-specified max length (in our case, 320 words). They were paired with concurrent video frames in the same segment.

⁴<https://developers.google.com/youtube/v3/docs/captions>

YT8M-cook We extract the cooking subset of YouTube-8M [Abu-El-Haija et al., 2016] by taking, from its training split, videos with “Cooking” or “Recipe” labels, and retain those with English ASR, subject to YouTube policies. After preprocessing, we obtain 186K ASR+video segments with an average length of 64 words (24 seconds) per segment.

HowTo100M. This is based on the 1.2M YouTube instructional videos released by Miech et al. [2019], covering a broad range of topics. After preprocessing, we obtain 7.99M ASR+video segments with an average of 78 words (28.7 seconds) per segment.

3.3 Pretraining Datasets: CAP-style

We also consider two text-only datasets for *pretraining*, containing *unpaired* instruction steps similar in style to the target captions.

Recipe1M is a collection of 1M recipes scraped from a number of popular cooking websites [Marin et al., 2019]. We use the sequence of instructions extracted for each recipe in this dataset, and treat each recipe step as a separate example during pretraining. This results in 10,767,594 CAP-style segments, with 12.8 words per segment.

WikiHow is a collection of 230,843 articles extracted from the WikiHow knowledge base [Koupae and Wang, 2018]. Each article comes with a title starting with “How to”. Each associated step starts with a step summary (in bold) followed by a detailed explanation. We extract all the step summaries, resulting in 1,360,145 CAP-style segments, with 8.2 words per segment. Again, each instruction step is considered as a separate example during pretraining.

3.4 Differences between Pretraining and Finetuning Datasets

First, note that *video segments* are defined differently for pretraining and finetuning datasets, and may not match exactly. For ASR+Video pretraining datasets, which are unsupervised, the segments are divided following a simple heuristic (e.g., two consecutive words more than 2 seconds apart), whereas for finetuning ASR+Video→CAP datasets, which are supervised, the segments are defined by human annotators to correspond to instruction steps. Otherwise, the ASR data are relatively similar between pretraining and finetuning datasets, as both come from instructional videos and are in the style of spoken language.

Second, compared to the target captions in finetuning datasets, the CAP-like pretraining datasets are similar in spirit — they all represent summaries of *steps*, but they may differ in length, style and granularity. In particular, the CAP-like pretraining datasets are closer in style to captions in YouCook2, where annotators were instructed to produce a recipe-like description for each step. This is reflected in their similar average length (YouCook2: 8.8 words, Recipe1M: 12.8 words, WikiHow: 8.2 words); whereas captions in ViTT are significantly shorter (2.97 words on average).

Despite these differences — some are inevitable due to the unsupervised nature of pretraining datasets — the pretraining data is very helpful for our task as shown in the experimental results.

4 Method

To model segment-level caption generation, we adopt MASS-style pretraining Song et al. [2019] with Transformer Vaswani et al. [2017] as the backbone architecture. For both pre-training and fine-tuning objectives, we have considered two variants: text-only and multi-modal. They are summarized in Table 10.2 and more details are given below.

4.1 Separate-Modality Architecture

Both ASR tokens and video segment features are given as input in the multimodal variants. We consider an architecture with a separate transformer for each modality (text or video), see Figure 10.2 for details. When available, the text and video encoders attend to each other at every layer using cross-modal attention, as in ViLBERT Lu et al. [2019]. The text decoder attends over the final-layer output of both encoders. We discuss in more detail the differences between using a separate-modality architecture vs. a vanilla-Transformer approach for all modalities in Appendix 2.

The inputs to the text encoder is the sum of three components: text token embeddings, positional embeddings and the corresponding style embeddings,⁵ depending on the style of the text (ASR or Caption-like). The inputs to the video encoder could be either precomputed frame-level 2D CNN features or 3D CNN features, pretrained on the Kinetics Carreira and Zisserman [2017], Kay et al. [2017] data set. The visual features are projected with fully-connected layers to the same dimension as the text embeddings.

⁵This is similar to the way language-ID embeddings are used in machine translation.

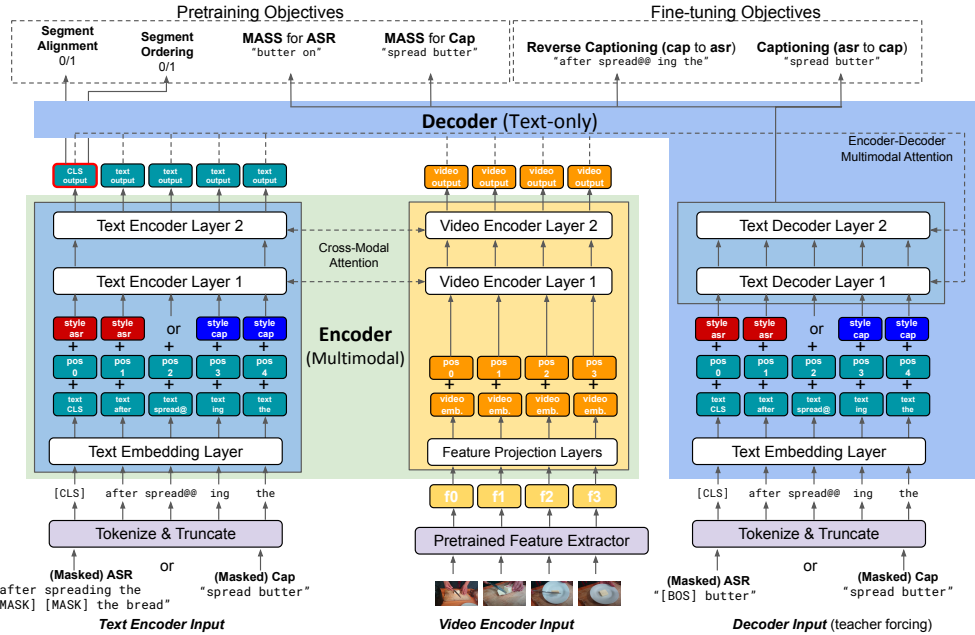


Figure 10.2: A diagram for the separate-modality architecture. It consists of a two-stream (text and video inputs) encoder with cross-modal attention and a text-only decoder, jointly trained using the MASS objective.

The main architecture we consider is a 2-layer encoder (E2), 6-layer decoder (D6) Transformer. We use **E2D6** to refer to the text-only version, and **E2vidD6** to refer to the multimodal version with an active video encoder. We also experiment with E2D2 and E2vidD2 (2-layer decoder).⁶

4.2 Pretraining with Text-only MASS

Text-only pretraining is essentially the unsupervised learning of the style transfer between ASR-style and caption-style texts using *unpaired* data sources: ASR strings from video segments in YT8M-cook or HowTo100M; and CAP-style instruction steps found in Recipe1M or HowTo100M. Just like using MASS for unsupervised machine translation involves pretraining the model on unpaired monolingual datasets, we alternate between ASR→ASR and CAP→CAP MASS steps during our pretraining stage, which does not require the “source” (ASR) and “target” (CAP-style) data to be aligned.

In an ASR→ASR step, we mask a random subsequence of the ASR and feed the masked ASR to the text encoder. The text decoder must reconstruct the hidden subsequence while attending to the encoder output. A CAP→CAP step works

⁶We found in a preliminary study that using 6-layer encoders did not improve performance for our application.

similarly by trying to reconstruct a masked sequence of a CAP-style text. The encoder and decoder are trained jointly using teacher-forcing on the decoder. We denote this text-only strategy as **MASS** in the experiments.

4.3 Pretraining with Multimodal MASS

During multimodal pretraining, we alternate between text-only CAP→CAP MASS steps and multimodal MASS steps. During each multimodal MASS step ASR+video→ASR, we feed a masked ASR to the text-encoder and the co-occurring video features to the video-encoder. The text decoder must reconstruct the masked ASR subsequence. We denote this pretraining strategy as **MASSvid** in the experiments. This trains cross-modal attention between the text-encoder and video-encoder at every layer, jointly with the text decoder that attends to the output layer of both the text and video encoders.⁷

To force more cross-modal attention between encoder and decoder, we also investigate a strategy of hiding the text-encoder output from the decoder for some fraction of training examples. We refer to this strategy as **MASSdrop** in the experiments.

4.4 Pretraining with Alignment and Ordering Tasks

We also explore encoder-only multimodal pretraining strategies. We take the last-layer representation for the CLS (beginning of sentence) token from the encoder, and add a multi-layer perceptron on top of it for binary predictions (Figure 10.2). Given a pair of ASR and video segment, we train the encoder to predict the following objectives:

- Segment-Level **Alignment**. An (ASR, video) pair is *aligned* if they occur in the same pretraining segment; negative examples are constructed by sampling pairs from the same video but at least 2 segments away.
- Segment-Level **Ordering**. We sample (ASR, video) pairs that are at least 2 segments away, and train the model to predict whether the ASR occurs before or after the video clip.

During this **MASSalign** pretraining stage, we alternate between two text-only MASS steps (CAP→CAP, ASR→ASR) and the two binary predictions (**Alignment** and **Ordering**) described above.

⁷In preliminary experiments, we had attempted to directly adapt the MASS objective [Song et al., 2019] to video reconstruction — by masking a subsequence of the input video and making the video decoder reconstruct the input using the Noise Contrastive Estimator Loss [Sun et al., 2019a]. Due to limited success, we did not further pursue this approach.

<i>Pretraining Objectives</i>			
Name	T	V	Input→Output
MASS	✓	✗	CAP→CAP, ASR→ASR
MASSvid	✓	✓	CAP→CAP, ASR+video→ASR
MASSdrop	✓	✓	CAP→CAP, ASR+video→ASR
MASSalign	✓	✓	CAP→CAP, ASR→ASR, ASR+video→{0, 1}
<i>Finetuning Objectives</i>			
Name	T	V	Input→Output
UniD	✓	✗	ASR→CAP
BiD	✓	✗	ASR→CAP, CAP→ASR
UniD	✓	✓	ASR+video→CAP
BiD	✓	✓	ASR+video→CAP, CAP→ASR
BiDalt	✓	✓	ASR+video→CAP, CAP+video→ASR

Table 10.2: Pretraining and Fine-tuning objectives. For each strategy, ✓ indicates whether the text (**T**) and video (**V**) encoders are active, followed by a summary of training objectives involved in one training step.

4.5 Finetuning on Video Captioning

For text-only finetuning, we feed ASR to the text encoder and the decoder has to predict the corresponding CAP (ASR→CAP). For multimodal finetuning, we also feed additional video representations to the video encoder (ASR+video→CAP). When finetuning a multimodal model from text-only pretraining, everything related to video (weights in the video encoder and any cross-modal attention modules) will be initialized randomly. In addition to these *uni-directional* (**UniD**) finetuning, we also experiment with several variants of *bidirectional* (**BiD**) finetuning (Table 10.2). For instance, adding CAP→ASR (predicting ASR from CAP) to text-only finetuning. In the experiments, we find some variants of bidirectional finetuning beneficial whether training from scratch or finetuning from a pretrained model.

5 Experiments

5.1 Implementation Details

We tokenize ASR and CAP inputs using byte-pair-encoding subwords [Sennrich et al., 2015], and truncate them to 240 subwords. We truncate video sequences to 40 frames (40 seconds of video), compute the 128-dim features proposed by Wang et al. [2014] (which we will refer to as Compact 2D features), and project them to the embedding space using a two-layer perceptron with layer normalization and GeLU activations.

Method	Input	Pretraining	BLEU-4	METEOR	ROUGE-L	CIDER
Constant Pred [Hessel et al., 2019]	-	-	2.70	10.30	21.70	0.15
MART Lei et al. [2020]	Video	-	8.00	15.90	-	0.36
EMT Zhou et al. [2018c]	Video	-	4.38	11.55	27.44	0.38
CBT Sun et al. [2019a]	Video	Kinetics + HowTo100M	5.12	12.97	30.44	0.64
AT [Hessel et al., 2019]	ASR	-	8.55	16.93	35.54	1.06
AT+Video [Hessel et al., 2019]	Video + ASR	-	9.01	17.77	36.65	1.12
UniViLM #1 [Luo et al., 2020]	Video	-	6.06	12.47	31.48	0.64
UniViLM #2 [Luo et al., 2020]	Video + ASR	-	8.67	15.38	35.02	1.00
UniViLM #5 [Luo et al., 2020]	Video + ASR	HowTo100M	10.42	16.93	38.02	1.20
<i>∅ Pretraining</i>						
E2D6-BiD	ASR	-	7.90	15.70	34.86	0.93
E2vidD6-BiD	Video + ASR	-	8.01	16.19	34.66	0.91
<i>Text Pretraining</i>						
E2D6-MASS-BiD	ASR	YT8M-cook + Recipe1M	10.60	17.42	38.08	1.20
E2vidD6-MASS-BiD	Video + ASR	YT8M-cook + Recipe1M	11.47	17.70	38.80	1.25
<i>Multimodal Pretraining</i>						
E2vidD6-MASSalign-BiD	Video + ASR	YT8M-cook + Recipe1M	11.53	17.62	39.03	1.22
E2vidD6-MASSvid-BiD	Video + ASR	YT8M-cook + Recipe1M	12.04	18.32	39.03	1.23
E2vidD6-MASSdrop-BiD	Video + ASR	YT8M-cook + Recipe1M	10.45	17.74	38.82	1.22
Human [Hessel et al., 2019]	Video + ASR	-	15.20	25.90	45.10	3.80

Table 10.3: Segment-level captioning results on YouCook2. We use YT8M-cook and Recipe1M for pretraining. The numbers for the related work (first group) are directly reported from the corresponding papers. The last line is an estimate of human performance as reported by Hessel et al. [2019], and can be taken as a rough upper bound of the best performance achievable.

We instantiate the E2xDx models defined in Section 4.1 with 128-dimensional embeddings and 8 heads respectively for self-attention, encoder-decoder, and cross-modal attention modules. We define each epoch to be 3,125 iterations, where each iteration contains one repetition of each training step as represented in Table 10.2. We pretrain for 200 epochs and finetune for 30 epochs.

For evaluation, we consider BLEU-4 [Papineni et al., 2002], METEOR [Denkowski and Lavie, 2014], ROUGE-L [Lin and Och, 2004] and CIDEr [Vedantam et al., 2015] metrics.

Please refer to Appendix 3 for full implementation details, hyperparameters and computation cost.

Notes on ViTT evaluation: With the exception of ROUGE-L, all other metrics are sensitive to short groundtruth. 67% of the groundtruth tags in ViTT have less than 4 words, where a perfect prediction will not yield a full score in, say, BLEU-4. Thus, we focus mainly on ROUGE-L, report BLEU-1 instead of BLEU-4 for ViTT, and provide the other two metrics only as reference points.

We had originally decided to use videos with multiple annotations as validation and test data, so that we could explore evaluation with multiple reference groundtruth

Method	Input	ViTT-All				ViTT-Cooking				
		BLEU-1	METEOR	ROUGE-L	CIDEr	BLEU-1	METEOR	ROUGE-L	CIDEr	
Constant ("intro")	baseline -	1.42	3.32	11.15	0.28	1.16	2.93	10.21	0.25	
<i>∅ Pretraining</i>										
E2D6-BiD	ASR	19.60	9.12	27.88	0.68	20.77	10.08	28.63	0.72	
E2vidD6-BiD	Video + ASR	19.49	9.23	28.53	0.69	20.45	9.88	28.88	0.69	
<i>Text Pretraining</i>										
E2D6-MASS-BiD	ASR	21.93	10.60	30.45	0.79	24.79	12.25	32.40	0.88	
E2vidD6-MASS-BiD	Video + ASR	22.44	10.83	31.27	0.81	24.22	12.22	32.60	0.89	
<i>Multimodal Pretraining</i>										
E2vidD6-MASSalign-BiD	Video + ASR	22.31	10.66	31.13	0.79	24.92	12.25	33.09	0.90	
E2vidD6-MASSvid-BiD	Video + ASR	22.45	10.76	31.49	0.80	24.87	12.43	32.97	0.90	
E2vidD6-MASSdrop-BiD	Video + ASR	22.37	11.00	31.40	0.82	24.48	12.22	33.10	0.89	
Human	Video + ASR	43.34	33.56	41.88	1.26	41.61	32.50	41.59	1.21	

Table 10.4: Segment-level captioning results on ViTT. For ViTT-All we pretrain on HowTo100M and WikiHow; for ViTT-Cooking we pretrain on YT8M-cook and Recipe1M. We report baseline scores for predicting the most common caption “intro”. We also estimate the human performance as a rough upper bound (details in Supplementary Material 1; Tables C.3,C.4).

captions. But as annotators do not always yield the same set of segment boundaries, this became tricky. Instead, we simply treat each segment as a separate instance with one single reference caption. Note that all segments annotated for the same video will be in either validation or test to ensure no content overlap.

5.2 Main Results

We run several variants of our method on YouCook2, ViTT-All and ViTT-Cooking, using different architectures, modalities, pretraining datasets, pretraining and finetuning strategies.

Comparing with other methods on YouCook2 For YouCook2, we report our method alongside several methods from the literature [Hessel et al., 2019, Sun et al., 2019b, Zhou et al., 2018c, Lei et al., 2020], as well as state-of-the-art concurrent work [Luo et al., 2020]. The related work is provided for reference and to give a ballpark estimate of the relative performance of each method, but results are not always strictly and directly comparable. Beyond the usual sources of discrepancy in data processing, tokenization, or even different splits, an additional source of complication comes from the fact that videos are regularly deleted by content creators, causing video datasets to shrink over time. Additionally, when comparing to other work incorporating pretraining, we could differ in (videos available in) pretraining datasets, segmentation strategies, etc. To this end, we perform an extensive ablation study, which at least helps us to understand the effectiveness of different components in our approach.

Effect of pretraining The main experimental results for the three datasets we consider are summarized in Table 10.3 (YouCook2) and Table 10.4 (ViTT-All and ViTT-Cooking). Across all three datasets, the best performance is achieved by finetuning a multimodal captioning model under the *Multimodal Pretraining* condition. For instance, on YouCook2, E2vidD6-MASSvid-BiD improves over the no-pretraining model E2vidD6-BiD by 4.37 ROUGE-L, a larger improvement than UniViLM with pretraining (#5) vs without (#2) Luo et al. [2020]. This improvement also holds in ViTT-Cooking (+4.22 in ROUGE-L) and ViTT-All (+2.97 in ROUGE-L). We do not observe consistent and significant trends among the different multimodal pretraining strategies: MASS pretraining with video (**MASSvid**), with video and droptext (**MASSdrop**), or with alignment tasks (**MASSalign**).⁸ Furthermore, we observe that most of the pretraining improvement is achievable via text-only MASS pretraining. Across all three datasets, while *Multimodal Pretraining* (E2vidD6-MASSvid-BiD) is consistently better than *Text Pretraining* (E2vidD6-MASS-BiD), the differences are quite small (under one ROUGE-L point).

⁸Limited improvement with MASSalign suggests that such alignment tasks are better suited for retrieval [Luo et al., 2020].

Method	BLEU-4	METEOR	ROUGE-L	CIDEr
D2-UniD	10.84	17.39	38.24	1.16
D6-UniD	11.39	18.00	38.71	1.22
D2-BiD	11.38	18.04	38.67	1.19
D6-BiD	11.47	17.70	38.80	1.25
D6-BiDalt	11.07	17.68	38.43	1.22
D6-BiD (S3D)	11.64	18.04	38.75	1.24

Table 10.5: Ablation study on YouCook2. We finetune a multimodal captioning model (E2vid) with either 2-layer decoder (D2) or 6-layer decoder (D6) using YT8M-cook /Recipe1M for MASS pretraining, combined with either unidirectional (UniD) or bidirectional (BiD) finetuning. We find no significant difference between using 2D and 3D features (marked as S3D).

It’s worth noting that for MASSalign, the best validation accuracies for the pre-training tasks are reasonably high: for YT8M-cook, we observed 90% accuracy for the alignment task, and 80% for the ordering task (for HowTo100M: 87% and 71.4%, respectively), where random guess would yield 50%. This suggests that our video features are reasonably strong, and the findings above are not due to weak visual representations.

Effect of other modeling choices We experiment with 2-layer decoder (D2) vs 6-layer decoder (D6), combined with either unidirectional fine-tuning (**UniD**) or bidirectional fine-tuning (**BiD**). Table 10.5 shows ablation results of the four possible combinations when finetuning a multimodal model using text-only pretraining on YouCook2 (a more complete list of results can be found in Appendix 5, showing similar trends). The D6xBiD combination tends to yield the best performance, with the differences among the four configurations being relatively small (under one ROUGE-L point). For visual features, we also explored using 3D features [Xie et al., 2018] instead of 2D features during finetuning (with no pretraining or text-only pretraining), and do not find much difference in model performance on YouCook2. As a result, we use the simpler 2D features in our multimodal pretraining. We leave more extensive experiments with visual features as future work.

Generalization implications An important motivation for constructing the ViTT dataset and evaluating our models on it has been related to generalization. Since the YouCook2 benchmark is restricted to a small number of cooking recipes, there is little to be understood about how well models trained and evaluated on it generalize. In contrast, the ViTT benchmark has a much wider coverage (for both cooking-related videos and general instructional videos), and no imposed topic overlap between train/dev/test. As such, there are two findings here that

are relevant with respect to generalization: (a) the absolute performance of the models on the ViTT benchmark is quite high (ROUGE-L scores above 0.30 are usually indicative of decent performance), and (b) the performance on ViTT vs. YouCook2 is clearly lower (31.5 ROUGE-L vs. 39.0 ROUGE-L, reflecting the increased difficulty of the new benchmark), but it is maximized under similar pretraining and finetuning conditions, which allows us to claim that the resulting models generalize well and are quite robust over a wide variety of instructional videos.

6 Conclusions

Motivated to improve information-seeking capabilities for videos, we have collected and annotated a new dense video captioning dataset, ViTT, which is larger with higher diversity compared to YouCook2. We investigated several multimodal pretraining strategies for segment-level video captioning, and conducted extensive ablation studies. We concluded that text-only pretraining is the most decisive factor in improving the performance on all the benchmarks used. Even more to the point, our results indicate that most of the performance can be attributed to leveraging the ASR signal. We achieve new state-of-the-art results on the YouCook2 benchmark, and establish strong performance baselines for the new ViTT benchmark, which can be used as starting points for driving more progress in this direction.

Acknowledgements

We send warm thanks to Ashish Thapliyal for helping the first author debug his code and navigate the computing infrastructure, and to Sebastian Goodman for his technical help (and lightning fast responses!). We also thank the anonymous reviewers for their comments and suggestions.

11

Conclusions, Discussions, and Perspectives

1 Summary and Conclusions

During the course of this thesis, we have attempted to understand and propose ways to make deep learning practices more *meaningful* and *data-efficient*. Our first three contributions were exploratory, and intended to deepen our understanding of existing methods, benchmarks, training losses and evaluation metrics, with a focus on generative modeling and data-efficient methods. The last contribution was a new method for more data-efficient video captioning.

The first contribution was an exploration of the GAN loss, originally motivated by previous work on the relationship between training loss and evaluation metric in the context of structured prediction [Osokin and Kohli, 2014]. Starting from the task at hand, generative modeling, we discussed some desirable properties for a training loss and evaluation metric: namely, to encourage realistic and diverse samples, consistent with human perception, while being easy to optimize. We pointed out several shortcomings of traditional divergences such as the Kullback-Leibler divergence, and how integral probability metrics such as MMD and Wasserstein fail to overcome them all. We then proceeded with a theoretical and empirical analysis of GAN-based divergences, and extended them to define mutual information. We concluded that rather than being crude approximations of traditional divergences, GAN losses have the potential to better match human perception, while retaining attractive computational and statistical properties.

The second contribution started by pointing out a desirable feature of few-shot classification: generalizing to new class semantics. We argued that meaningful few-shot classification benchmarks should test for class semantic generalization, but that does not seem the case for popular benchmarks such as *Omniglot*. We proposed a baseline method for solving few-shot classification benchmarks without test-time labels, effectively concluding that *Omniglot* does not require class semantic generalization to be solved. We then provided baseline numbers for solving other benchmarks without test-time labels, and observed that cross-domain benchmarks were significantly harder for our baseline, consistent with the idea that cross-domain benchmarks require generalizing to new class semantics. In the process, we introduced a new clustering algorithm, Sinkhorn K-Means, which to our knowledge is the first application of optimal transport to few-shot classification.

The third contribution was a survey of few-shot object detection, which we have extended to include self-supervised object detection, a natural combination under data-efficiency constraints. An important goal was to clarify the field of low-data object detection and ease newcomers into the field; we covered the differences with few-shot classification, reorganized the literature, pointed out the inconsistencies in the evaluation strategies—which make some comparisons almost meaningless!—and explained the main few-shot and self-supervised object detection strategies. Another goal was to extract the most meaningful trends and future research directions: we observed that self-supervision can be beneficial for few-shot object detection, that finetuning remains a strong baseline, that traditional computer vision methods can generate weak supervision, and that both visual transformer backbones and transformer-based object detection have promising properties.

The fourth contribution was a data-efficient model for captioning tutorial videos, which achieved state-of-the-art performance on YouCook2 [Zhou et al., 2018a] at the time of publication, and a new dataset, ViTT, that was tailored to the task at hand. The novelty of our work was to combine several *modalities*—video features and speech-to-text transcripts—with several *multimodal pretraining objectives*, allowing us to leverage unpaired unlabeled video and caption-like text datasets. We found, perhaps surprisingly, that the majority of the performance came from text-only pretraining and using the transcribed speech, with only a small performance gain from using video features and multimodal pretraining. After investigating the issue, we concluded that the limited gain may be due to the limited power of the video features used, or due to the fact that learning cross-modal attention from fully unpaired data remains a hard task.

2 Future research directions.

Meaningful training losses & evaluation metrics. Defining evaluation metrics which correlate better with human perception, and more generally with the task at hand, will remain an important challenge of the future. Since its introduction in 2017, the Fréchet Inception Distance has remained the gold standard of image generation [Heusel et al., 2017], while GANs have remained some of the preferred generation methods, though diffusion-based models [Ho et al., 2020] have recently gained much traction, especially in the context of text-to-image systems [Ramesh et al., 2022]. Newer approaches have often involved humans-in-the-loop for evaluating various characteristics of deep learning systems, such as image quality [Ramesh et al., 2022, Yu et al., 2022], sample diversity [Arora et al., 2018], helpfulness and harmlessness [Bai et al., 2022]; or quality, safety, and groundedness of conversational agents [Thoppilan et al., 2022]. Several works, especially in the language domain, have started evaluating models beyond pure performance, and assess them

for bias/fairness, privacy/data memorization, and political correctness [Thoppilan et al., 2022]. Considering the scale at which models will be deployed, such “ethical ” metrics will play increasingly important roles, to ensure AI has a positive impact where deployed. Defining meaningful metrics to measure these ethical aspects, beyond preventing stereotypes and inappropriate language, is a challenging but essential future research direction.

Language-based zero-shot learning. In this thesis, we have covered how existing few-shot benchmarks and evaluation protocols have certain issues such as requiring limited generalization. In particular, we have discussed how Omniglot has fixed class semantics [Huang et al., 2019], how YouCook2 has the same recipes across training and validation splits [Huang et al., 2020], and the variance and data contamination problems in few-shot object detection [Huang et al., 2021]. On the other hand, recent developments have shown that *prompting*—formulating natural language understanding and generation tasks in natural language—is a viable strategy for eliciting zero or few-shot capabilities in large language models [Brown et al., 2020, Schick and Schütze, 2020, Sanh et al., 2021]. Compared to traditional ways of expressing new tasks such as using support sets (which is limited to categorical predictions), meta-attributes (which are typically fixed in advance), or word embeddings (which have limited expressivity), using natural language may be the most flexible way of formulating truly novel tasks, which require strong generalization to solve. Indeed, multimodal methods trained on large-scale datasets, such as CLIP [Radford et al., 2021], Frozen [Tsimpoukelli et al., 2021] and Flamingo [Alayrac et al., 2022], have shown that the expressivity of natural language can be extended to solve vision and vision-language tasks. Therefore, we believe that benchmarks and methods which formulate zero and few-shot tasks in a mix of natural language and other modalities—such as images, videos, speech—are the most promising approaches towards stronger generalization and general-purpose methods.



Bibliography

- S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016.
- E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden. Pyramid methods in image processing. *RCA engineer*, 29(6):33–41, 1984.
- J.-B. Alayrac, P. Bojanowski, N. Agrawal, I. L. Josef Sivic, and S. Lacoste-Julien. Unsupervised learning from narrated instruction videos. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016.
- J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, R. Ring, E. Rutherford, S. Cabi, T. Han, Z. Gong, S. Samangooei, M. Monteiro, J. Menick, S. Borgeaud, A. Brock, A. Nematzadeh, S. Sharifzadeh, M. Binkowski, R. Barreira, O. Vinyals, A. Zisserman, and K. Simonyan. Flamingo: a visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198*, 2022.
- E. Amrani, R. Ben-Ari, I. Shapira, T. Hakim, and A. Bronstein. Self-supervised object detection and retrieval using unlabeled videos. In *CVPR Workshops*, 2020.
- R. B. Arantes, G. Vogiatzis, and D. R. Faria. Csc-gan: Cycle and semantic consistency for dataset augmentation. In *International Symposium on Visual Computing*, pages 170–181. Springer, 2020.
- P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *CVPR*, 2014.
- M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. In *ICML*, 2017.
- S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. In *FOCS*, 1993.
- S. Arora, R. Ge, Y. Liang, T. Ma, and Y. Zhang. Generalization and equilibrium in generative adversarial nets (GANs). In *ICML*, 2017.
- S. Arora, A. Risteski, and Y. Zhang. Do gans learn the distribution? some theory and empirics. In *International Conference on Learning Representations*, 2018.

-
- Y. M. Asano, C. Rupprecht, and A. Vedaldi. Self-labelling via simultaneous clustering and representation learning. *arXiv:1911.05371*, 2019.
- G. Aude, M. Cuturi, G. Peyré, and F. Bach. Stochastic optimization for large-scale optimal transport. In *NIPS*, 2016.
- P. Awasthi and R. B. Zadeh. Supervised clustering. In *Neural Information Processing Systems*, 2010.
- Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.
- A. Bansal, K. Sikka, G. Sharma, R. Chellappa, and A. Divakaran. Zero-shot object detection. In *ECCV*, 2018.
- A. Bar, X. Wang, V. Kantorov, C. J. Reed, R. Herzig, G. Chechik, A. Rohrbach, T. Darrell, and A. Globerson. Detreg: Unsupervised pretraining with region priors for object detection. *arXiv:2106.04550*, 2021.
- D. Barrett, F. Hill, A. Santoro, A. Morcos, and T. Lillicrap. Measuring abstract reasoning in neural networks. In *International conference on machine learning*, pages 511–520. PMLR, 2018.
- P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.
- D. Belanger and A. McCallum. Structured prediction energy networks. In *ICML*, 2016.
- M. I. Belghazi, A. Baratin, S. Rajeshwar, S. Ozair, Y. Bengio, A. Courville, and D. Hjelm. Mutual information neural estimation. In *International Conference on Machine Learning*, pages 531–540. PMLR, 2018.
- M. G. Bellemare, I. Danihelka, W. Dabney, S. Mohamed, B. Lakshminarayanan, S. Hoyer, and R. Munos. The Cramer distance as a solution to biased Wasserstein gradients. *arXiv:1705.10743*, 2017.
- J.-D. Benamou, G. Carlier, M. Cuturi, L. Nenna, and G. Peyré. Iterative bregman projections for regularized transportation problems. *SIAM Journal on Scientific Computing*, 37(2):A1111–A1138, 2015.
- Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

-
- P. J. Bickel and K. A. Doksum. *Mathematical Statistics: Basic Ideas and Selected Topics, Volume I*. Chapman and Hall/CRC, 2015.
- H. Bilen and A. Vedaldi. Weakly supervised deep detection networks. In *CVPR*, 2016.
- M. Bilenko, S. Basu, and R. J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *International Conference on Machine Learning*, 2004.
- M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton. Demystifying MMD GANs. In *International Conference on Learning Representations*, 2018. URL <https://openreview.net/forum?id=r11UOzWCW>.
- Y. Blau and T. Michaeli. Rethinking lossy compression: The rate-distortion-perception tradeoff. In *International Conference on Machine Learning*, pages 675–685. PMLR, 2019.
- O. Bousquet, S. Gelly, I. Tolstikhin, C.-J. Simon-Gabriel, and B. Schoelkopf. From optimal transport to generative modeling: the VEGAN cookbook. *arXiv:1705.07642*, 2017.
- A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=B1xsqj09Fm>.
- A. Brock, S. De, S. L. Smith, and K. Simonyan. High-performance large-scale image recognition without normalization. In *International Conference on Machine Learning*, pages 1059–1071. PMLR, 2021.
- T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In *NeurIPS*, 2020.
- G. Canas and L. Rosasco. Learning probability measures with respect to optimal transport metrics. In *Neural Information Processing Systems*, 2012.
- N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.
- M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018.

-
- M. Caron, P. Bojanowski, J. Mairal, and A. Joulin. Unsupervised pre-training of image features on non-curated data. In *CVPR*, 2019.
- M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020.
- M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers. *arXiv:2104.14294*, 2021.
- J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733, 2017.
- H. Chen, Y. Wang, G. Wang, and Y. Qiao. Lstd: A low-shot transfer detector for object detection. In *AAAI*, 2018.
- L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.
- T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020a.
- T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton. Big self-supervised models are strong semi-supervised learners. *NeurIPS*, 2020b.
- T. Chen, Y. Liu, H. Su, Y. Chan, Y. Lin, J. Yeh, W. Chen, and W. H. Hsu. Dual-awareness attention for few-shot object detection. *IEEE TM*, 23, 2021a.
- T. Chen, S. Saxena, L. Li, D. J. Fleet, and G. Hinton. Pix2seq: A language modeling framework for object detection. *arXiv:2109.10852*, 2021b.
- W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019a.
- X. Chen, H. Fan, R. Girshick, and K. He. Improved baselines with momentum contrastive learning. *arXiv:2003.04297*, 2020c.
- X. Chen, S. Xie, and K. He. An empirical study of training self-supervised vision transformers. *arXiv:2104.02057*, 2021c.
- Y.-C. Chen, L. Li, L. Yu, A. E. Kholy, F. Ahmed, Z. Gan, Y. Cheng, and J. Liu. Uniter: Learning universal image-text representations. *arXiv preprint arXiv:1909.11740*, 2019b.
- L. Chizat, P. Roussillon, F. Léger, F.-X. Vialard, and G. Peyré. Faster wasserstein distance estimation with the sinkhorn divergence. *Advances in Neural Information Processing Systems*, 33, 2020.

-
- A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.
- M. Chu, Y. Xie, J. Mayer, L. Leal-Taixé, and N. Thuerey. Learning temporal coherence via self-supervision for gan-based video generation. *ACM Transactions on Graphics (TOG)*, 39(4):75–1, 2020.
- G. Ciaparrone, F. L. Sánchez, S. Tabik, L. Troiano, R. Tagliaferri, and F. Herrera. Deep learning in video multi-object tracking: A survey. *Neurocomputing*, 381: 61–88, 2020.
- M. Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *EMNLP*, 2002.
- D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5): 603–619, 2002.
- C. Cortes, V. Kuznetsov, M. Mohri, and S. Yang. Structured prediction theory based on factor graph complexity. In *NIPS*, 2016.
- M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *NIPS*, 2013.
- M. Cuturi and A. Doucet. Fast computation of wasserstein barycenters. In *International Conference on Machine Learning*, 2014.
- Z. Dai, B. Cai, Y. Lin, and J. Chen. Up-detr: Unsupervised pre-training for object detection with transformers. In *CVPR*, 2021.
- D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray. Scaling egocentric vision: The EPIC-KITCHENS dataset. In *European Conference on Computer Vision (ECCV)*, 2018.
- I. Danihelka, B. Lakshminarayanan, B. Uria, D. Wierstra, and P. Dayan. Comparison of maximum likelihood and GAN-based training of real NVPs. *arXiv:1705.05263*, 2017.
- J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.
- M. Denkowski and A. Lavie. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on statistical machine translation*, pages 376–380, 2014.

-
- J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34, 2021.
- T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1-2):31–71, 1997.
- C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015.
- L. Dong, N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou, and H. Hon. Unified language model pre-training for natural language understanding and generation. *arXiv preprint arXiv:1905.03197*, 2019.
- A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.
- N. Dvornik, C. Schmid, and J. Mairal. Selecting relevant features from a universal representation for few-shot classification. *arXiv preprint arXiv:2003.09338*, 2020.
- G. K. Dziugaite, D. M. Roy, and Z. Ghahramani. Training generative neural networks via maximum mean discrepancy optimization. In *UAI*, 2015.
- H. Edwards and A. Storkey. Towards a neural statistician. *arXiv preprint arXiv:1606.02185*, 2016.
- P. Esser, R. Rombach, and B. Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12873–12883, 2021.
- M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010.
- Q. Fan, W. Zhuo, C.-K. Tang, and Y.-W. Tai. Few-shot object detection with attention-rpn and multi-relation detector. In *CVPR*, 2020a.
- Z. Fan, J.-G. Yu, Z. Liang, J. Ou, C. Gao, G.-S. Xia, and Y. Li. Fgn: Fully guided network for few-shot instance segmentation. In *CVPR*, 2020b.
- Z. Fan, Y. Ma, Z. Li, and J. Sun. Generalized few-shot object detection without forgetting. In *CVPR*, 2021.

-
- W. Fedus*, M. Rosca*, B. Lakshminarayanan, A. M. Dai, S. Mohamed, and I. Goodfellow. Many paths to equilibrium: GANs do not need to decrease a divergence at every step. In *International Conference on Learning Representations*, 2018. URL <https://openreview.net/forum?id=ByQpn1ZA->.
- P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 2004.
- T. Finley and T. Joachims. Supervised clustering with support vector machines. In *International Conference on Machine Learning*, 2005.
- T. Finley and T. Joachims. Supervised k-means clustering. In *Cornell Computing and Information Science Technical Report*, 2008.
- C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.
- M. Gao, A. Li, R. Yu, V. I. Morariu, and L. S. Davis. C-wsl: Count-guided weakly supervised localization. In *ECCV*, 2018.
- A. Genevay, G. Peyré, and M. Cuturi. Learning generative models with sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics*, pages 1608–1617. PMLR, 2018.
- S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018.
- G. Gidel, R. A. Hemmat, M. Pezeshki, R. Lepriol, G. Huang, S. Lacoste-Julien, and I. Mitliagkas. Negative momentum for improved game dynamics. In *AISTATS*, 2019.
- R. Girshick. Fast r-cnn. In *ICCV*, 2015.
- R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.
- I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. In *NeurIPS*, 2014.
- I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- A. Gretton, K. M. Borgwardt, M. Rasch, B. Schölkopf, A. J. Smola, et al. A kernel method for the two-sample-problem. In *NIPS*, 2007.
- A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012.

-
- J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, B. Piot, k. kavukcuoglu, R. Munos, and M. Valko. Bootstrap your own latent - a new approach to self-supervised learning. In *NeurIPS*, volume 33, pages 21271–21284, 2020.
- X. Gu, T.-Y. Lin, W. Kuo, and Y. Cui. Zero-shot detection via vision and language knowledge distillation. *arXiv:2104.13921*, 2021.
- I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL <https://proceedings.neurips.cc/paper/2017/file/892c3b1c6dccd52936e27cbd0ff683d6-Paper.pdf>.
- A. Gupta, P. Dollar, and R. Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *CVPR*, 2019.
- K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *ICCV*, 2017.
- K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.
- O. Henaff. Data-efficient image recognition with contrastive predictive coding. In *ICML*, 2020.
- O. J. Hénaff, S. Koppula, J.-B. Alayrac, A. v. d. Oord, O. Vinyals, and J. Carreira. Efficient visual pretraining with contrastive detection. *arXiv:2103.10957*, 2021.
- J. Hessel, B. Pang, Z. Zhu, and R. Soricut. A case study on combining asr and visual features for generating instructional video captions. In *Proceedings of CoNLL*, 2019.
- M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.
- J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.
- J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

-
- K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.
- R. Hou, H. Chang, M. Bingpeng, S. Shan, and X. Chen. Cross attention network for few-shot classification. In *Advances in Neural Information Processing Systems*, pages 4005–4016, 2019.
- A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv:1704.04861*, 2017.
- K. Hsu, S. Levine, and C. Finn. Unsupervised learning via meta-learning. *arXiv preprint 1810.02334*, 2018.
- Y.-C. Hsu, Z. Lv, and Z. Kira. Learning to cluster in order to transfer across domains and tasks. *arXiv preprint 1711.10125*, 2017.
- Y.-C. Hsu, Z. Lv, J. Schlosser, P. Odom, and Z. Kira. Multi-class classification without multi-class labels. In *International Conference on Learning Representations*, 2019.
- G. Huang, H. Berard, A. Touati, G. Gidel, P. Vincent, and S. Lacoste-Julien. Parametric adversarial divergences are good task losses for generative modeling. *arXiv:1708.02511*, 2017.
- G. Huang, H. Larochelle, and S. Lacoste-Julien. Are few-shot learning benchmarks too simple? solving them without task supervision at test-time. *arXiv:1902.08605*, 2019.
- G. Huang, B. Pang, Z. Zhu, C. Rivera, and R. Soricut. Multimodal pretraining for dense video captioning. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, Suzhou, China, Dec. 2020. Association for Computational Linguistics.
- G. Huang, I. Laradji, D. Vazquez, S. Lacoste-Julien, and P. Rodriguez. A survey of self-supervised and few-shot object detection. *To appear in IEEE TPAMI 2022*, 2021.
- F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and < 0.5 mb model size. *arXiv:1602.07360*, 2016.
- L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu. A survey of deep learning-based object detection. *IEEE access*, 7:128837–128868, 2019.

-
- Z. Jie, Y. Wei, X. Jin, J. Feng, and W. Liu. Deep self-taught learning for weakly supervised object localization. In *CVPR*, 2017.
- L. Jing and Y. Tian. Self-supervised visual feature learning with deep neural networks: A survey. *T-PAMI*, 2020.
- L. Kaiser, O. Nachum, A. Roy, and S. Bengio. Learning to remember rare events. *arXiv preprint arXiv:1703.03129*, 2017.
- B. Kang, Z. Liu, X. Wang, F. Yu, J. Feng, and T. Darrell. Few-shot object detection via feature reweighting. In *ICCV*, 2019.
- J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.
- L. Karlinsky, J. Shtok, S. Harary, E. Schwartz, A. Aides, R. Feris, R. Giryes, and A. M. Bronstein. Repmet: Representative-based metric learning for classification and few-shot object detection. In *CVPR*, 2019.
- T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018a. URL <https://openreview.net/forum?id=Hk99zCeAb>.
- T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *ICLR*, 2018b.
- T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.
- T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and improving the image quality of stylegan. In *CVPR*, 2020.
- W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, A. Natsev, M. Suleyman, and A. Zisserman. The kinetics human action video dataset. *ArXiv*, abs/1705.06950, 2017.
- S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah. Transformers in vision: A survey. *arXiv:2101.01169*, 2021.
- S. Khodadadeh, L. Bölöni, and M. Shah. Unsupervised meta-learning for few-shot image and video classification. *arXiv preprint 1811.11819*, 2018.
- M. Kilickaya, A. Erdem, N. Ikizler-Cinbis, and E. Erdem. Re-evaluating automatic metrics for image captioning. *arXiv preprint arXiv:1612.07600*, 2016.

-
- J. Kim, P. T. Nguyen, S. Weir, P. J. Guo, R. C. Miller, and K. Z. Gajos. Crowd-sourcing step-by-step information extraction to enhance existing how-to videos. In *CHI*, 2014.
- D. Kingma and J. Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.
- D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *ICLR*, 2014.
- G. Koch, R. Zemel, R. Salakhutdinov, et al. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, 2015.
- M. Koupaee and W. Y. Wang. Wikihow: A large scale text summarization dataset. *arXiv preprint arXiv:1810.09305*, 2018.
- R. Krishna, K. Hata, F. Ren, L. Fei-Fei, and J. C. Niebles. Dense-captioning events in videos. In *International Conference on Computer Vision (ICCV)*, 2017.
- N. Kwon, H. Na, G. Huang, and S. Lacoste-Julien. Repurposing pretrained models for robust out-of-domain few-shot learning. *ICLR*, 2021.
- A. Lacoste, P. Rodriguez, F. Branchaud-Charron, P. Atighehchian, M. Caccia, I. H. Laradji, A. Drouin, M. Craddock, L. Charlin, and D. Vázquez. Symbols: Probing learning algorithms with synthetic datasets. In *NeurIPS*, 2020.
- B. Lake, R. Salakhutdinov, J. Gross, and J. Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 33, 2011.
- B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.
- A. M. Lamb, A. Goyal, Y. Zhang, S. Zhang, A. C. Courville, and Y. Bengio. Professor forcing: A new algorithm for training recurrent networks. In *NIPS*, 2016.
- Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. Albert: A lite bert for self-supervised learning of language representations. *ArXiv*, abs/1909.11942, 2020.
- I. Laradji, P. Rodriguez, O. Manas, K. Lensink, M. Law, L. Kurzman, W. Parker, D. Vazquez, and D. Nowrouzezahrai. A weakly supervised consistency-based learning method for covid-19 segmentation in ct images. In *WACV*, 2021a.
- I. H. Laradji, N. Rostamzadeh, P. O. Pinheiro, D. Vázquez, and M. Schmidt. Instance segmentation with point supervision. *arXiv:1906.06392*, 2019a.

-
- I. H. Laradji, D. Vazquez, and M. Schmidt. Where are the masks: Instance segmentation with image-level supervision. *arXiv:1907.01430*, 2019b.
- I. H. Laradji, R. Pardinias, P. Rodriguez, and D. Vazquez. Looc: Localize overlapping objects with count supervision. In *ICIP*, 2020a.
- I. H. Laradji, N. Rostamzadeh, P. O. Pinheiro, D. Vazquez, and M. Schmidt. Proposal-based instance segmentation with point supervision. In *ICIP*, 2020b.
- I. H. Laradji, A. Saleh, P. Rodriguez, D. Nowrouzezahrai, M. R. Azghadi, and D. Vazquez. Weakly supervised underwater fish segmentation using affinity lfcn. *Scientific Reports*, 11(1):1–10, 2021b.
- R. Leblond, J.-B. Alayrac, A. Osokin, and S. Lacoste-Julien. SEARNN: Training RNNs with global-local losses. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HkUR_y-RZ.
- Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang. A tutorial on energy-based learning. *Predicting structured data*, 2006.
- K. Lee, S. Maji, A. Ravichandran, and S. Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10657–10665, 2019.
- J. Lei, L. Wang, Y. Shen, D. Yu, T. L. Berg, and M. Bansal. Mart: Memory-augmented recurrent transformer for coherent video paragraph captioning. In *The 58th Annual Meeting of the Association for Computational Linguistics*, 2020.
- M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *ArXiv*, abs/1910.13461, 2019.
- A. Li and Z. Li. Transformation invariant few-shot object detection. In *CVPR*, 2021.
- B. Li, B. Yang, C. Liu, F. Liu, R. Ji, and Q. Ye. Beyond max-margin: Class margin equilibrium for few-shot object detection. In *CVPR*, 2021a.
- C. Li, J. Yang, P. Zhang, M. Gao, B. Xiao, X. Dai, L. Yuan, and J. Gao. Efficient self-supervised vision transformers for representation learning. *arXiv:2106.09785*, 2021b.
- C.-L. Li, W.-C. Chang, Y. Cheng, Y. Yang, and B. Póczos. MMD GAN: Towards deeper understanding of moment matching network. In *NIPS*, 2017. (to appear).

-
- D. Li, J.-B. Huang, Y. Li, S. Wang, and M.-H. Yang. Weakly supervised object localization with progressive domain adaptation. In *CVPR*, 2016.
- G. Li, N. Duan, Y. Fang, D. Jiang, and M. Zhou. Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training. *arXiv preprint arXiv:1908.06066*, 2019a.
- H. Li, D. Eigen, S. Dodge, M. Zeiler, and X. Wang. Finding task-relevant features for few-shot learning by category traversal. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–10, 2019b.
- X. Li, L. Zhang, Y. P. Chen, Y.-W. Tai, and C.-K. Tang. One-shot object detection without fine-tuning. *arXiv:2005.03819*, 2020.
- Y. Li, K. Swersky, and R. Zemel. Generative moment matching networks. In *ICML*, 2015.
- Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 2021c.
- C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.
- C.-Y. Lin and F. J. Och. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 605. Association for Computational Linguistics, 2004.
- T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.
- T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017a.
- T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *ICCV*, 2017b.
- B. Liu, Y. Cao, Y. Lin, Q. Li, Z. Zhang, M. Long, and H. Hu. Negative margin matters: Understanding margin in few-shot classification. *arXiv preprint arXiv:2003.12060*, 2020.
- P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*, 2021a.

-
- S. Liu, O. Bousquet, and K. Chaudhuri. Approximation and convergence properties of generative adversarial learning. In *Advances in Neural Information Processing Systems*, pages 5551–5559, 2017.
- S. Liu, T. Li, W. Chen, and H. Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *ICCV*, 2019a.
- W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016.
- Y. Liu, J. Lee, M. Park, S. Kim, E. Yang, S. J. Hwang, and Y. Yang. Learning to propagate labels: Transductive propagation network for few-shot learning. *arXiv preprint arXiv:1805.10002*, 2018.
- Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *arXiv preprint arxiv:1907.11692*, 2019b.
- Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv:2103.14030*, 2021b.
- S. Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.
- B. London, B. Huang, and L. Getoor. Stability and generalization in structured prediction. *Journal of Machine Learning Research*, 17(222):1–52, 2016.
- J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.
- J. Lu, D. Batra, D. Parikh, and S. Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems*, pages 13–23, 2019.
- H. Luo, L. Ji, B. Shi, H. Huang, N. Duan, T. Li, X. Chen, and M. Zhou. Univilm: A unified video and language pre-training model for multimodal understanding and generation. *arXiv preprint arXiv:2002.06353*, 2020.
- L. Ma, X. Jia, Q. Sun, B. Schiele, T. Tuytelaars, and L. Van Gool. Pose guided person image generation. *Advances in neural information processing systems*, 30, 2017.
- O. Mañas, A. Lacoste, X. Giro-i Nieto, D. Vazquez, and P. Rodriguez. Seasonal contrast: Unsupervised pre-training from uncurated remote sensing data. *ICCV*, 2021.

-
- L. E. Margulieux, M. Guzdial, and R. Catrambone. Subgoal-labeled instructional material improves performance and transfer in learning to develop mobile applications. In *Conference on International Computing Education Research*, 2012.
- J. Marin, A. Biswas, F. Ofli, N. Hynes, A. Salvador, Y. Aytar, I. Weber, and A. Torralba. Recipe1m+: A dataset for learning cross-modal embeddings for cooking recipes and food images. *IEEE transactions on pattern analysis and machine intelligence*, 2019.
- A. Mathiasen and F. Hvilshøj. Backpropagating through fr\`echet inception distance. *arXiv preprint arXiv:2009.14075*, 2020.
- L. Metz, N. Maheswaranathan, B. Cheung, and J. Sohl-Dickstein. Learning unsupervised learning rules. *arXiv preprint 1804.00222*, 2018.
- L. Mi, W. Zhang, X. Gu, and Y. Wang. Variational wasserstein clustering. In *European Conference on Computer Vision*, 2018a.
- L. Mi, W. Zhang, and Y. Wang. Regularized wasserstein means based on variational transportation. *arXiv preprint 1812.00338*, 2018b.
- A. Miech, D. Zhukov, J.-B. Alayrac, M. Tapaswi, I. Laptev, and J. Sivic. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2630–2640, 2019.
- A. Miech, J.-B. Alayrac, L. Smaira, I. Laptev, and A. Z. Josef Sivic. End-to-end learning of visual representations from uncurated instructional videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020.
- T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3, 2010.
- T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv:1301.3781*, 2013.
- G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.
- T. M. Mitchell and T. M. Mitchell. *Machine learning*, volume 1. McGraw-hill New York, 1997.
- S. Mohamed and B. Lakshminarayanan. Learning in implicit generative models. *arXiv:1610.03483*, 2016.
- K. Moon and A. Hero. Multivariate f-divergence estimation with confidence. In *Advances in Neural Information Processing Systems*, pages 2420–2428, 2014.

-
- Y. Mroueh, T. Sercu, and V. Goel. McGan: Mean and covariance feature matching GAN. In *ICML*, 2017.
- H. Müller and A. Holzinger. Kandinsky patterns. *Artificial Intelligence*, 300:103546, 2021.
- J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957.
- F. Murtagh and P. Contreras. Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1): 86–97, 2012.
- N. Nauata, K.-H. Chang, C.-Y. Cheng, G. Mori, and Y. Furukawa. House-gan: Relational generative adversarial networks for graph-constrained house layout generation. In *European Conference on Computer Vision*, pages 162–177. Springer, 2020.
- A. Newell and J. Deng. How useful is self-supervised pretraining for visual tasks? In *CVPR*, 2020.
- X. Nguyen, M. J. Wainwright, and M. I. Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, 2010.
- A. Q. Nichol and P. Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.
- W. Nie, Z. Yu, L. Mao, A. B. Patel, Y. Zhu, and A. Anandkumar. Bongard-logo: A new benchmark for human-level concept learning and reasoning. *Advances in Neural Information Processing Systems*, 33:16468–16480, 2020.
- M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016.
- S. Nowozin, B. Cseke, and R. Tomioka. f-GAN: Training generative neural samplers using variational divergence minimization. In *NIPS*, 2016.
- C. O’Neil-Hart. Why you should lean into how-to content in 2018. www.thinkwithgoogle.com/advertising-channels/video/self-directed-learning-youtube/, 2018. Accessed: 2019-09-03.
- A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *ICML*, 2016.
- A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv:1807.03748*, 2018.

-
- B. N. Oreshkin, P. Rodriguez, and A. Lacoste. Tadam: task dependent adaptive metric for improved few-shot learning. In *NeurIPS*, 2018.
- A. Osokin and P. Kohli. Perceptually inspired layout-aware losses for image segmentation. In *ECCV*, 2014.
- A. Osokin, F. Bach, and S. Lacoste-Julien. On structured prediction theory with calibrated convex surrogate losses. In *NIPS*, 2017.
- E. Oyallon, S. Zagoruyko, G. Huang, N. Komodakis, S. Lacoste-Julien, M. Blaschko, and E. Belilovsky. Scattering networks for hybrid representation learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2208–2221, 2018.
- B. Pan, H. Cai, D.-A. Huang, K.-H. Lee, A. Gaidon, E. Adeli, and J. C. Niebles. Spatio-temporal graph for video captioning with knowledge distillation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020.
- K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.
- D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.
- H. Petzka, A. Fischer, and D. Lukovnikov. On the regularization of wasserstein GANs. In *International Conference on Learning Representations*, 2018. URL <https://openreview.net/forum?id=B1hYRMbCW>.
- P. O. Pinheiro, A. Almahairi, R. Y. Benmalek, F. Golemo, and A. C. Courville. Unsupervised learning of dense visual representations. In *NeurIPS*, 2020.
- B. A. Pires, C. Szepesvari, and M. Ghavamzadeh. Cost-sensitive multiclass classification risk bounds. In *ICML*, 2013.
- L. Qiao, Y. Shi, J. Li, Y. Wang, T. Huang, and Y. Tian. Transductive episodic-wise adaptive metric for few-shot learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3603–3612, 2019.
- L. Qiao, Y. Zhao, Z. Li, X. Qiu, J. Wu, and C. Zhang. Defrcn: Decoupled faster r-cnn for few-shot object detection. In *ICCV*, 2021.
- A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016.
- A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. *Preprint*, 2018.

-
- A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. *arXiv:2103.00020*, 2021.
- C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *ArXiv*, abs/1910.10683, 2019.
- A. Raghu, M. Raghu, S. Bengio, and O. Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. *arXiv preprint arXiv:1909.09157*, 2019.
- S. Rahman, S. Khan, and N. Barnes. Improved visual-semantic alignment for zero-shot object detection. In *AAAI*, 2020.
- K. Rakelly, E. Shelhamer, T. Darrell, A. Efros, and S. Levine. Conditional networks for few-shot semantic segmentation. In *ICLR*, 2018.
- A. Ramdas, N. Trillos, and M. Cuturi. On wasserstein two-sample testing and related families of nonparametric tests. *Entropy*, 19(2):47, 2017.
- A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.
- A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.
- J. C. Raven. Standardization of progressive matrices, 1938. *British Journal of Medical Psychology*, 1941.
- S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2016.
- A. Razavi, A. van den Oord, and O. Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *NeurIPS*, 2019.
- S. J. Reddi, A. Ramdas, B. Póczos, A. Singh, and L. Wasserman. On the decreasing power of kernel and distance based nonparametric hypothesis tests in high dimensions. In *AAAI*, 2015.
- J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv:1804.02767*, 2018.

-
- J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.
- S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. In *International conference on machine learning*, pages 1060–1069. PMLR, 2016.
- M. D. Reid and R. C. Williamson. Information, divergence and risk for binary experiments. *Journal of Machine Learning Research*, 12(Mar):731–817, 2011.
- M. Ren, E. Triantafillou, S. Ravi, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel. Meta-learning for semi-supervised few-shot classification. *arXiv preprint 1803.00676*, 2018.
- S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NeurIPS*, 2015.
- J. Requeima, J. Gordon, J. Bronskill, S. Nowozin, and R. E. Turner. Fast and flexible multi-task classification using conditional neural adaptive processes. In *Advances in Neural Information Processing Systems*, pages 7957–7968, 2019.
- P. Rodriguez, I. Laradji, A. Drouin, and A. Lacoste. Embedding propagation: Smoother manifold for few-shot classification. In *ECCV*. Springer, 2020.
- B. Romera-Paredes and P. Torr. An embarrassingly simple approach to zero-shot learning. In *International Conference on Machine Learning*, pages 2152–2161, 2015.
- O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.
- A. Ruderman, M. Reid, D. García-García, and J. Petterson. Tighter variational representations of f-divergences via restriction to probability measures. *arXiv preprint arXiv:1206.4664*, 2012.
- O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.
- T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training GANs. In *NIPS*, 2016.
- R. Sanabria, O. Caglayan, S. Palaskar, D. Elliott, L. Barrault, L. Specia, and F. Metze. How2: A large-scale dataset for multimodal language understanding. *arXiv preprint arXiv:1811.00347*, 2018.

-
- V. Sanh, A. Webson, C. Raffel, S. H. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler, T. L. Scao, A. Raja, et al. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*, 2021.
- T. Schick and H. Schütze. Exploiting cloze questions for few shot text classification and natural language inference. *arXiv preprint arXiv:2001.07676*, 2020.
- J. Schmidhuber, S. Hochreiter, et al. Long short-term memory. *Neural Comput*, 9(8):1735–1780, 1997.
- R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.
- B. Shi, L. Ji, Y. Liang, N. Duan, P. Chen, Z. Niu, and M. Zhou. Dense procedure captioning in narrated instructional videos. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6382–6391, 2019.
- A. Shin, M. Ishii, and T. Narihira. Perspectives and prospects on transformer architecture for cross-modal tasks with language and vision. *International Journal of Computer Vision*, pages 1–20, 2022.
- M. Siam, N. Doraiswamy, B. N. Oreshkin, H. Yao, and M. Jagersand. Weakly supervised few-shot object segmentation using co-attention with visual and semantic embeddings. In *IJCAI*, pages 860–867, 7 2020.
- S. Smith, M. Patwary, B. Norick, P. LeGresley, S. Rajbhandari, J. Casper, Z. Liu, S. Prabhunoye, G. Zerveas, V. Korthikanti, et al. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*, 2022.
- J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, 2017a.
- J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Neural Information Processing Systems*, 2017b.
- R. Socher, M. Ganjoo, C. D. Manning, and A. Ng. Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*, pages 935–943, 2013.
- J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.
- K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*, 2019.

-
- D. Soudry, E. Hoffer, M. S. Nacson, S. Gunasekar, and N. Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.
- B. K. Sriperumbudur, K. Fukumizu, A. Gretton, B. Schölkopf, G. R. Lanckriet, et al. On the empirical estimation of integral probability metrics. *Electronic Journal of Statistics*, 6:1550–1599, 2012.
- W. Su, X. Zhu, Y. Cao, B. Li, L. Lu, F. Wei, and J. Dai. Vi-bert: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*, 2019.
- C. Sun, F. Baradel, K. Murphy, and C. Schmid. Contrastive bidirectional transformer for temporal representation learning. *arXiv preprint arXiv:1906.05743*, 2019a.
- C. Sun, A. Myers, C. Vondrick, K. Murphy, and C. Schmid. Videobert: A joint model for video and language representation learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7464–7473, 2019b.
- Z. Sun, S. Cao, Y. Yang, and K. Kitani. Rethinking transformer-based set prediction for object detection. *arXiv:2011.10881*, 2020.
- F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. In *CVPR*, 2018a.
- F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208, 2018b.
- I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *NeurIPS*, 2014.
- C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.
- H. Tan and M. Bansal. Lxmert: Learning cross-modality encoder representations from transformers. *arXiv preprint arXiv:1908.07490*, 2019.
- P. Tang, X. Wang, X. Bai, and W. Liu. Multiple instance detection network with online instance classifier refinement. In *CVPR*, 2017.
- P. Tang, X. Wang, S. Bai, W. Shen, X. Bai, W. Liu, and A. Yuille. Pcl: Proposal cluster learning for weakly supervised object detection. *T-PAMI*, 42(1):176–191, 2018.

-
- A. Tarvainen and H. Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *arXiv:1703.01780*, 2017.
- B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *NIPS*, 2003.
- L. Theis, A. v. d. Oord, and M. Bethge. A note on the evaluation of generative models. In *ICLR*, 2016.
- R. Thoppilan, D. De Freitas, J. Hall, N. Shazeer, A. Kulshreshtha, H.-T. Cheng, A. Jin, T. Bos, L. Baker, Y. Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.
- H. Touvron, M. Cord, M. Douze, F. Massa, and A. Sablay-rolles. Hjeou, “training data-efficient image transformers & distillation through attention,”. *arXiv preprint arXiv:2012.12877*, 2020.
- E. Triantafillou, T. Zhu, V. Dumoulin, P. Lamblin, U. Evci, K. Xu, R. Goroshin, C. Gelada, K. Swersky, P.-A. Manzagol, and H. Larochelle. Meta-dataset: A dataset of datasets for learning to learn from few examples. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=rkgAGAVKPr>.
- H.-Y. Tseng, H.-Y. Lee, J.-B. Huang, and M.-H. Yang. Cross-domain few-shot classification via learned feature-wise transformation. *arXiv preprint arXiv:2001.08735*, 2020.
- M. Tsimpoukelli, J. L. Menick, S. Cabi, S. Eslami, O. Vinyals, and F. Hill. Multimodal few-shot learning with frozen language models. *Advances in Neural Information Processing Systems*, 34:200–212, 2021.
- J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *IJCV*, 104(2):154–171, 2013.
- A. Van Den Oord, O. Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.
- A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.
- R. Vedantam, C. Lawrence Zitnick, and D. Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575, 2015.

-
- R. Vedantam, A. Szlam, M. Nickel, A. Morcos, and B. M. Lake. Curi: A benchmark for productive concept learning under uncertainty. In *International Conference on Machine Learning*, pages 10519–10529. PMLR, 2021.
- P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008.
- O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. *NeurIPS*, 2016a.
- O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In *Neural Information Processing Systems*, 2016b.
- K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl, et al. Constrained k-means clustering with background knowledge. In *International Conference on Machine Learning*, 2001.
- C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.
- F. Wan, C. Liu, W. Ke, X. Ji, J. Jiao, and Q. Ye. C-mil: Continuation multiple instance learning for weakly supervised object detection. In *CVPR*, 2019.
- J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine-grained image similarity with deep ranking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1386–1393, 2014.
- T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018a.
- X. Wang, Y. Ye, and A. Gupta. Zero-shot recognition via semantic embeddings and knowledge graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6857–6866, 2018b.
- X. Wang, J. Wu, J. Chen, L. Li, Y. fang Wang, and W. Y. Wang. Vatex: A large-scale, high-quality multilingual dataset for video-and-language research. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4580–4590, 2019a.
- X. Wang, T. Huang, J. Gonzalez, T. Darrell, and F. Yu. Frustratingly simple few-shot object detection. In *ICML*, 2020a.
- X. Wang, R. Zhang, C. Shen, T. Kong, and L. Li. Dense contrastive learning for self-supervised visual pre-training. In *CVPR*, 2021.

-
- Y. Wang, W.-L. Chao, K. Q. Weinberger, and L. van der Maaten. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. *arXiv preprint arXiv:1911.04623*, 2019b.
- Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)*, 53(3):1–34, 2020b.
- Y.-X. Wang, D. Ramanan, and M. Hebert. Meta-learning to detect rare objects. In *CVPR*, 2019c.
- F. Wei, Y. Gao, Z. Wu, H. Hu, and S. Lin. Aligning pretraining for detection via object-level contrastive learning. *arXiv:2106.02637*, 2021a.
- J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021b.
- S. Weir, J. Kim, K. Z. Gajos, and R. C. Miller. Learnersourcing subgoal labels for how-to videos. In *CSCW*, 2015.
- Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision*, 2016.
- J. Wu, S. Liu, D. Huang, and Y. Wang. Multi-scale positive sample refinement for few-shot object detection. In *ECCV*, 2020a.
- X. Wu, D. Sahoo, and S. Hoi. Meta-rcnn: Meta learning for few-shot object detection. In *ACMMM*, 2020b.
- Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2251–2265, 2018.
- T. Xiao, X. Wang, A. A. Efros, and T. Darrell. What should not be contrastive in contrastive learning. In *ICLR*, 2020.
- T. Xiao, C. J. Reed, X. Wang, K. Keutzer, and T. Darrell. Region similarity representation learning. *arXiv:2103.12902*, 2021.
- Y. Xiao and R. Marlet. Few-shot object detection and viewpoint estimation for objects in the wild. In *ECCV*, 2020.
- J. Xie, R. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, 2016.

-
- S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 305–321, 2018.
- H. Xu, G. Ghosh, P.-Y. Huang, P. Arora, M. Aminzadeh, C. Feichtenhofer, F. Metze, and L. Zettlemoyer. Vlm: Task-agnostic video-language model pre-training for video understanding. *arXiv preprint arXiv:2105.09996*, 2021a.
- M. Xu, Z. Zhang, H. Hu, J. Wang, L. Wang, F. Wei, X. Bai, and Z. Liu. End-to-end semi-supervised object detection with soft teacher. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3060–3069, 2021b.
- X. Yan, Z. Chen, A. Xu, X. Wang, X. Liang, and L. Lin. Meta r-cnn: Towards general solver for instance-level low-shot learning. In *CVPR*, 2019.
- C. Yang, Z. Wu, B. Zhou, and S. Lin. Instance localization for self-supervised detection pretraining. In *CVPR*, 2021a.
- J. Yang, D. Parikh, and D. Batra. Joint unsupervised learning of deep representations and image clusters. In *CVPR*, 2016.
- Z. Yang, Z. Gan, J. Wang, X. Hu, F. Ahmed, Z. Liu, Y. Lu, and L. Wang. Crossing the format boundary of text and boxes: Towards unified vision-language modeling. *arXiv preprint arXiv:2111.12085*, 2021b.
- H.-J. Ye, H. Hu, D.-C. Zhan, and F. Sha. Few-shot learning via embedding adaptation with set-to-set functions. In *Computer Vision and Pattern Recognition (CVPR)*, 2020.
- M. Yin, G. Tucker, M. Zhou, S. Levine, and C. Finn. Meta-learning without memorization. *arXiv preprint arXiv:1912.03820*, 2019.
- YouTubeBlog. You know what’s cool? a billion hours. <https://youtube.googleblog.com/2017/02/you-know-whats-cool-billion-hours.html>, 2017. Accessed: 2020-06-23.
- J. Yu, Y. Xu, J. Y. Koh, T. Luong, G. Baid, Z. Wang, V. Vasudevan, A. Ku, Y. Yang, B. K. Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2022.
- S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee. A survey of modern deep learning based object detection models. *arXiv:2104.11892*, 2021.
- ZenithMedia. Online video viewing to reach 100 minutes a day in 2021. <https://www.zenithmedia.com/online-video-viewing-to-reach-100-minutes-a-day-in-2021/>, 2019. Accessed: 2020-06-23.

-
- X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer. Scaling vision transformers. *arXiv:2106.04560*, 2021.
- F. Zhang, T. Pan, and B. Wang. Semi-supervised object detection with adaptive class-rebalancing self-training. *arXiv preprint arXiv:2107.05031*, 2021a.
- G. Zhang, Z. Luo, K. Cui, and S. Lu. Meta-detr: Few-shot object detection via unified image-level meta-learning. *arXiv:2103.11731*, 2021b.
- R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *ECCV*, 2016.
- R. Zhang, P. Isola, and A. A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *CVPR*, 2017.
- Z. Zhang, Y. Shi, C. Yuan, B. Li, P. Wang, W. Hu, and Z. Zha. Object relational graph with teacher-recommended learning for video captioning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020.
- Q. Zheng, C. Wang, and D. Tao. Syntax-aware action targeting for video captioning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020a.
- Y. Zheng, R. Huang, C. Han, X. Huang, and L. Cui. Background learnable cascade for zero-shot object detection. In *ACCV*, 2020b.
- Y. Zheng, J. Wu, Y. Qin, F. Zhang, and L. Cui. Zero-shot instance segmentation. In *CVPR*, 2021.
- L. Zhou, C. Xu, and J. J. Corso. YouCookII dataset. http://youcook2.eecs.umich.edu/static/YouCookII/youcookii_readme.pdf, 2018a. Accessed: 2020-06-23.
- L. Zhou, C. Xu, and J. J. Corso. Towards automatic learning of procedures from web instructional videos. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018b.
- L. Zhou, Y. Zhou, J. J. Corso, R. Socher, and C. Xiong. End-to-end dense video captioning with masked transformer. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8739–8748, 2018c.
- X. Zhou, D. Wang, and P. Krähenbühl. Objects as points. *arXiv:1904.07850*, 2019.
- J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, pages 2223–2232, 2017.
- L. Zhu and Y. Yang. Actbert: Learning global-local video-text representations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020.

X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *ICLR*, 2021.

A

Appendix for: “Exploring Properties of GAN and VAE Losses”

In this appendix, we present the following supplementary material:

- Additional generated samples for the Thin-8 task are in Section [1.1](#).
- Generated samples for the Sum-25 task are in Section [1.2](#).

1 Experimental results

1.1 Additional Samples for VAE and GAN

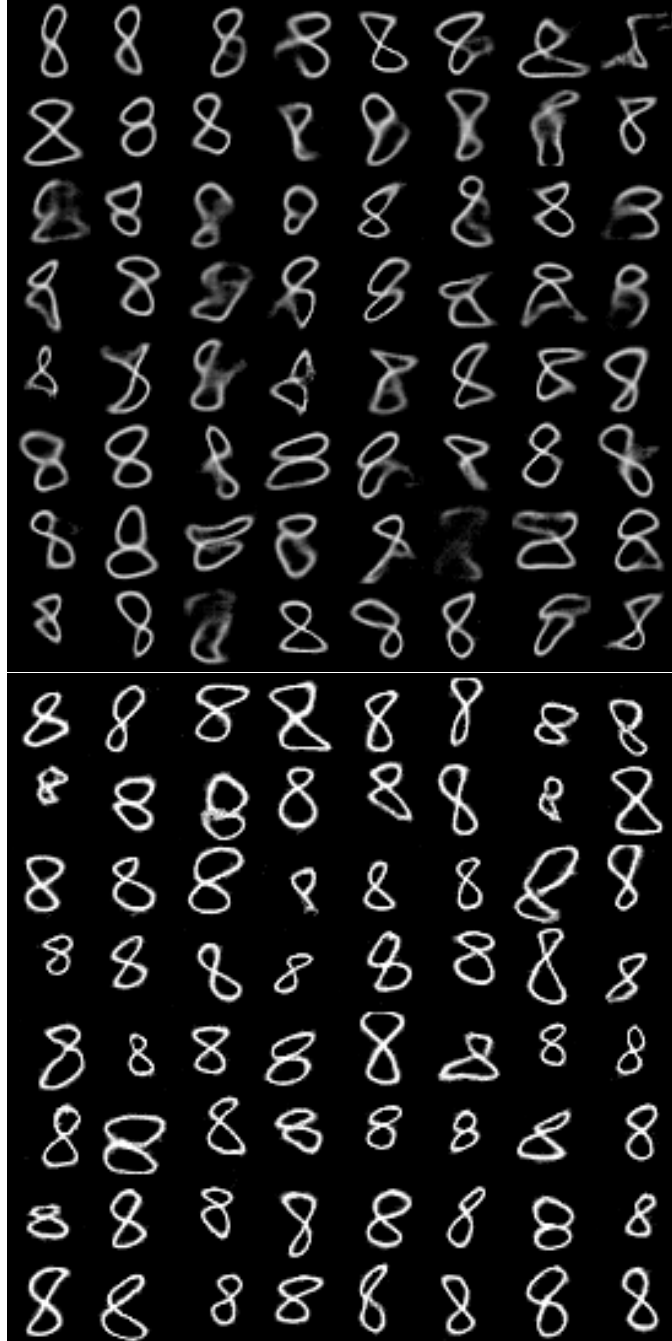


Figure A.1: VAE (top) and GAN (bottom) samples with 16 latent variables and 32×32 resolution.

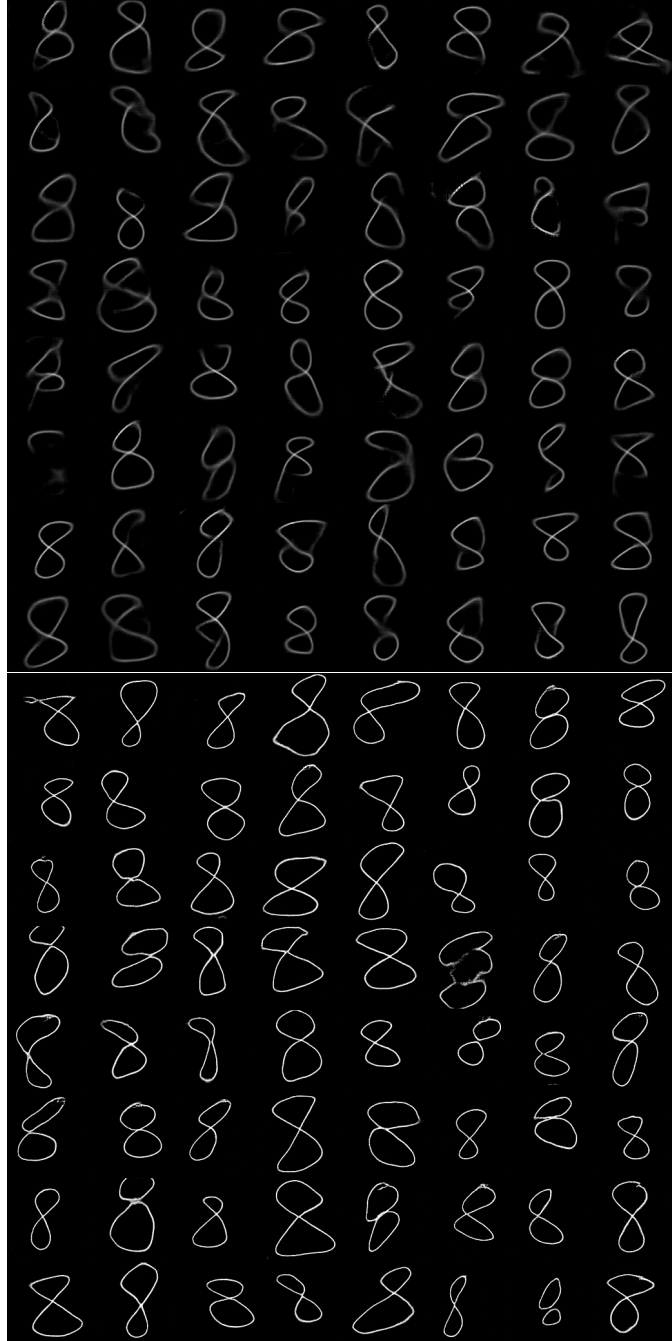


Figure A.2: VAE (**top**) and GAN (**bottom**) samples with 16 latent variables and 128×128 resolution.

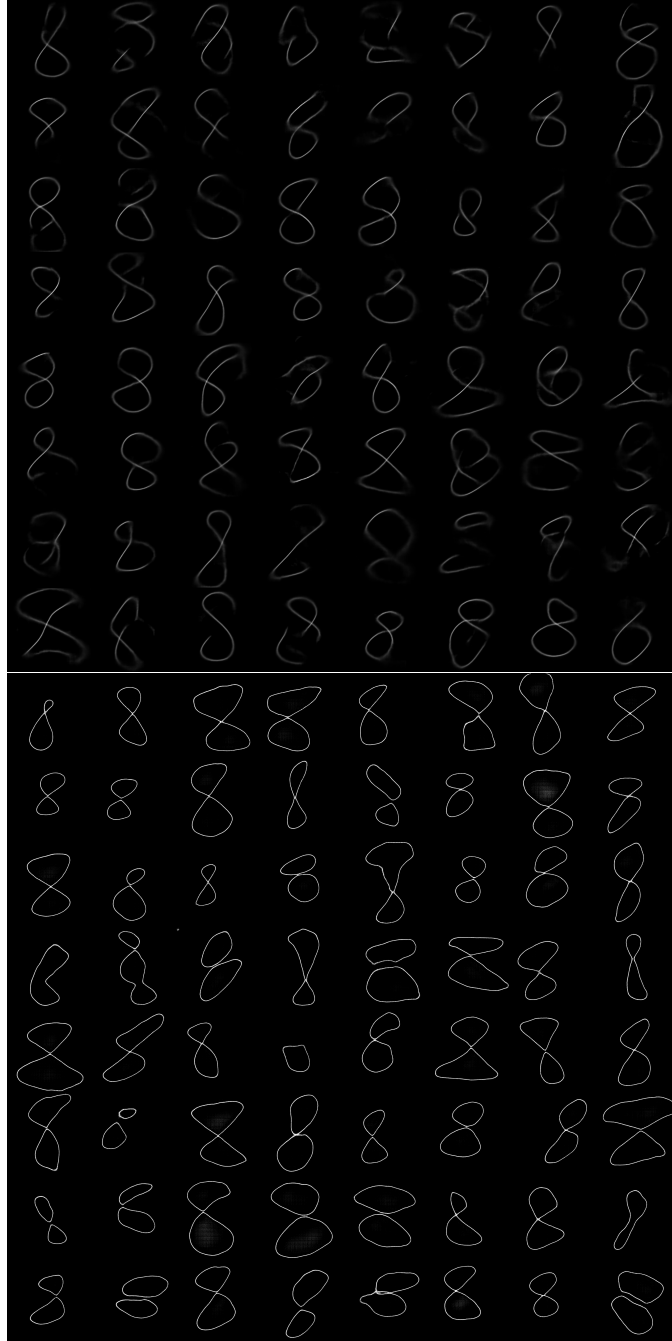


Figure A.3: VAE (**top**) and GAN (**bottom**) samples with 16 latent variables and 512×512 resolution.

1.2 Sum-25: Generated samples

Figure A.4 shows some additional samples from the VAE, WGAN-GP, and WGAN-GP with side-task, trained on the Sum-25. All three generators have 200 latent variables and the same architecture.



Figure A.4: Samples from generators with similar architecture and 200 latent variables, trained with the following objectives: VAE (**left**), WGAN-GP (**middle**), and WGAN-GP with side-task (**right**). Each row represents a sample (combination of 5 digits) generated by the model.

B

Appendix for “Are Few-Shot Learning Benchmarks too Simple ? Solving them without Test-Time Labels”

1 Links to the Code

The code for same-domain experiments on *miniImageNet*, *tieredImageNet*, CUB, and cross-domain experiments on *miniImageNet*→CUB is forked from the original FEAT code¹ and is available at <https://anonymous.4open.science/r/7fd48c5c-1a56-426d-a980-6bdc1c03f06a/>.

The code for cross-domain experiments on Meta-Dataset is forked from the original Meta-Dataset code² and is available at <https://anonymous.4open.science/r/cccb838f-8401-44f7-bfed-8da9c8c69aef/>.

The code for few-shot clustering experiments and same-domain experiments on Omniglot is forked from the original ProtoNet implementation³ and is available at <https://anonymous.4open.science/r/cbedf355-0158-4cbd-93f0-854d6af33cf4/>.

2 Additional Related Work from Clustering Literature

Supervised clustering. Supervised clustering is defined in [Finley and Joachims \[2005\]](#) as “learning how to cluster future sets of items [...] given sets of items and complete clusterings over these sets”. They use structured SVM to learn a similarity-metric between pairs of items, then run a fixed clustering algorithm which optimizes the sum of similarities of pairs in the same cluster. In follow-up work [[Finley and Joachims, 2008](#)], they use K-Means as the clustering algorithm. A main difference with our work is that we learn a nonlinear embedding function, whereas they assume linear embeddings. The work of [Awasthi and Zadeh \[2010\]](#) is also called supervised clustering, although they solve a very different problem. They propose a clustering algorithm which repetitively presents candidate clusterings to a “teacher” and actively requests feedback (supervision).

¹<https://github.com/Sha-Lab/FEAT>

²<https://github.com/google-research/meta-dataset>

³<https://github.com/jakesnell/prototypical-networks>

Learning to cluster. Recent deep learning literature has preferred the term “learning to cluster” to “supervised clustering”. Although the task is still the same, the main difference is the learning of a similarity metric using deep networks. Because of this aspect, these works are often classified as falling in the “metric learning” literature. Hsu et al. [2017, 2019] propose a Constrained Clustering Network (CCN) for learning to cluster based on two distinct steps: learning a similarity metric to predict if two examples are in the same class, and optimizing a neural network to predict cluster assignments which tend to agree with the similarity metric. CCNs obtained the state-of-the-art results when compared against other supervised clustering algorithms, we will thus use CCN as a strong baseline. In our experiments, Centroid Networks improve over CCN on their benchmarks, while being simpler to train and computationally much cheaper.

Semi-supervised & constrained clustering. Semi-supervised clustering consists of clustering data with some supervision in the form of “this pair of points should be/not be in the same cluster”. Some methods take the pairwise supervision as hard constraints [Wagstaff et al., 2001], while others (including CCN) learn metrics which tend to satisfy those constraints [Bilenko et al., 2004]. See the related work sections in Finley and Joachims [2005], Hsu et al. [2017].

Sinkhorn K-Means. The idea of formulating clustering as minimizing a Wasserstein distance between empirical distributions has been proposed several times in the past [Mi et al., 2018a]. Canas and Rosasco [2012] explicit some theoretical links between K-Means and the Wasserstein-2 distance. The most similar work to Sinkhorn K-Means is Regularized Wasserstein-Means [Mi et al., 2018b], but they use another method for solving optimal transport. Specifically using Sinkhorn distances (regularized Wasserstein distances) for clustering has even been suggested in Genevay et al. [2018]. However, as we could not find an explicit description of the Sinkhorn K-Means anywhere in the literature, we coin the name and explicitly state the algorithm in Section 4.2. To our knowledge, we are the first to use Sinkhorn K-Means in the context of learning to cluster and to scale it up to more complex datasets like *mini*ImageNet. Note that our work should not be confused with Wasserstein K-Means and similar variants, which consist in replacing the squared L_2 base-distance in K-Means with a Wasserstein distance.

3 Additional information on Sinkhorn K-Means.

Sinkhorn Distances. The *Wasserstein-2* distance is a distance between two probability masses p and q . Given a base distance $d(x, x')$, we define the cost of transporting one unit of mass from x to x' as $d(x, x')^2$. The Wasserstein-2 distance is defined as the cheapest cost for transporting all mass from p to q . When the transportation plan is regularized to have large entropy, we obtain Sinkhorn

distances, which can be computed very efficiently for discrete distributions [Cuturi, 2013, Cuturi and Doucet, 2014] (entropy-regularization makes the problem strongly convex). Sinkhorn distances are the basis of the Sinkhorn K-Means algorithm, which is the main component of Centroid Networks. In Algorithm 1, we describe the Sinkhorn algorithm in the particular case where we want to transport mass from the weighted data points (x_i, R_j) to the weighted centroids (c_j, C_j) , where R_j and C_j are the weights of the data points and centroids, respectively. In practice, we leverage the log-sum-exp trick in computing the Sinkhorn distances to avoid numerical underflows.

Optimization problem Both of Sinkhorn and Regular K-Means can be formulated as a joint minimization in the centroids $c_j \in \mathbb{R}^d$ (real vectors) and the assignments $p_{i,j} \geq 0$ (scalars) which specify how much of each point x_i is assigned to centroid c_j :

- **K-Means.** Note that compared to the usual convention, we have normalized assignments $p_{i,j}$ so that they sum up to 1.

$$\begin{aligned} & \text{minimize} && \min_{p,c} \sum_{i=1}^N \sum_{j=1}^K p_{i,j} \|x_i - c_j\|^2 \\ & \text{subject to} && \sum_{j=1}^K p_{i,j} = \frac{1}{N}, \quad i \in 1:N \\ & && p_{i,j} \in \{0, \frac{1}{N}\}, \quad i \in 1:N, j \in 1:K \end{aligned}$$

- **Sinkhorn K-Means.**

$$\begin{aligned} & \text{minimize} && \min_{p,c} \sum_i \sum_j p_{i,j} \|x_i - c_j\|^2 - \gamma \underbrace{H(p)}_{\text{entropy}} \\ & \text{subject to} && \sum_{j=1}^K p_{i,j} = \frac{1}{N}, \quad i \in 1:N \\ & && \sum_{i=1}^N p_{i,j} = \frac{1}{K}, \quad j \in 1:K \\ & && p_{i,j} \geq 0 \quad i \in 1:N, j \in 1:K \end{aligned}$$

where $H(p) = -\sum_{i,j} p_{i,j} \log p_{i,j}$ is the entropy of the assignments, and $\gamma \geq 0$ is a parameter tuning the entropy penalty term.

Differences between Sinkhorn vs. Regular K-Means. The first difference is that K-Means only allows hard assignments $p_{i,j} \in \{0, \frac{1}{N}\}$, that is, each point x_i is assigned to exactly one cluster c_j . On the contrary, the Sinkhorn K-Means formulation allows soft assignments $p_{i,j} \in [0, \frac{1}{N}]$, but with the additional constraint that the clusters have to be balanced, i.e., the same amount of points are soft-assigned to each cluster $\sum_i p_{i,j} = \frac{1}{K}$. The second difference is the penalty term

$-\gamma H(p)$ which encourages solutions of high-entropy, i.e., points will tend to be assigned more uniformly over clusters, and clusters more uniformly over points. Adding entropy-regularization allows us to compute $p_{i,j}$ very efficiently using the work of Cuturi [2013]. Note that removing the balancing constraint $\sum_i p_{i,j} = \frac{1}{K}$ in the Sinkhorn K-Means objective would yield a regularized K-Means objective with coordinate update steps identical to EM in a mixture of Gaussians (with $p_{i,j}$ updated using softmax conditionals).

Why is Sinkhorn K-means expected to improve performance ? The ablation study in Section 6 shows that using Sinkhorn K-Means instead of K-Means is the most decisive factor in improving performance. There are mainly two possible explanations :

1. Sinkhorn K-Means is particularly well adapted to the few-shot clustering and unsupervised few-shot classification problems because it strictly enforces the number of images per cluster, whereas K-Means does not.
2. Sinkhorn K-Means is likely to converge better than K-means due to the entropy-regularization factor of the Sinkhorn distance.

To illustrate the second point, consider the limit case where the regularization factor of Sinkhorn distance goes to infinity ($\gamma \rightarrow \infty$). Then, the assignments in Sinkhorn K-Means become uniform (each cluster is assigned equally to all points), and all the centroids converge – in one step – to the average of all the points, reaching global minimum. This is by no means a rigorous proof, but the limit case suggests that Sinkhorn K-Means converges well for large enough γ . This behavior is to be contrasted with K-means, for which convergence is well known to depend largely on the initialization.

What is the effect of using weighted vs. unweighted averages ? One could argue that comparing CentroidNets with ProtoNets is unfair because using Sinkhorn K-Means leads to centroids which are weighted averages, whereas ProtoNet prototypes are restricted to be unweighted averages. Therefore, we run Centroid Networks on *mini*Imagenet, but under the constraint that centroids to be unweighted averages of the data points. To do so, starting from the soft weights, we reassign each data point only to its closest centroid, and compute the unweighted averages. The comparison between ProtoNets and CentroidNets is now fair in the sense that both prototypes and centroids use unweighted averages. (Numbers below based on official ProtoNet implementation [Snell et al., 2017a])

- Unsupervised accuracy on *mini*Imagenet is 0.5508 ± 0.0072 for weighted average and 0.5497 ± 0.0072 for unweighted average. The difference is not significant.

-
- Clustering accuracy on *miniImageNet* is 0.6421 ± 0.0069 for weighted average and 0.6417 ± 0.0069 for unweighted average. The difference is also not significant.

This experiment suggests that using weighted averages does not bring an unfair advantage, and therefore does not invalidate our comparison. More generally, instead of trying to tune ProtoNets and CentroidNets as well as possible, we try to make ProtoNets and CentroidNets more comparable by using the same architectures and representation.

4 Implementation Details

Splits. For Omniglot which has 1623 classes of handwritten characters, we consider the “Vinyals” splits [Vinyals et al., 2016a]. For *miniImageNet* which has 100 object classes, we consider the “Ravi” splits [Ravi and Larochelle, 2016] of 64 training, 16 validation, 20 testing classes. For CUB which has 200 bird species, we use the same split as Ye et al. [2020] which consists of 100 training, 50 validation, and 50 testing classes. For Meta-Dataset, we consider the official splits and sampling scheme, which features variable numbers of ways and shots [Triantafillou et al., 2020]. For Omniglot-CCN, we consider the same splits as Hsu et al. [2017] : 30 background alphabets are used for training and 20 evaluation alphabets are used for validation (there is no testing set).

Backbones. We consider two backbones throughout the experiments: the Conv-4 classically used in few-shot learning [Snell et al., 2017a, Finn et al., 2017, Vinyals et al., 2016a] and the ResNet-12 [Ye et al., 2020, Lee et al., 2019]. We also consider the CCN architecture [Hsu et al., 2017, 2019, 2018] for the few-shot clustering experiments, and the ResNet-18 architecture for Meta-Dataset [Triantafillou et al., 2020].

Reference implementations. All our CentroidNet implementations are derived from specific reference ProtoNet implementations [Snell et al., 2017a, Triantafillou et al., 2020, Ye et al., 2020]. Unless otherwise specified, we always use exactly the same training procedures and hyperparameters as the reference implementations. We have grouped our experiments below based on their reference implementations.

Code based on Ye et al. [2020]. For same-domain experiments on *miniImageNet*, *tieredImageNet*, CUB (Section 5.1), cross-domain experiments on *miniImageNet*→CUB (Section 5.2), and transductive experiments on *miniImageNet*, *tieredImageNet* (Section 5.3), we derive CentroidNet from the implementation of Ye et al. [2020] using the Conv-4 and ResNet-12 architecture. We denote

ProtoNet* [Ye et al., 2020] the accuracies reported from their paper and ProtoNet* (repro) the accuracies reproduced from their code.⁴ We adopt exactly the same training strategy for ProtoNet and CentroidNet using the default hyperparameters of their implementation. We start from the provided pretrained checkpoints, which are given by training a classifier on all training classes (non-episodic training) with data augmentation. Then, we finetune the models on 5-way 5-shot LT few-shot classification episodes using Adam optimizer with initial learning rates of 0.0001 for Conv-4 and 0.0002 for ResNet-12. We do not use a center loss, and use a temperature of 32 for Conv-4 and 64 for ResNet-12 to rescale the squared Euclidean distances between prototypes/centroids and data embeddings before feeding them to the softmax layer/Sinkhorn K-Means. We run a hyperparameter search over Sinkhorn regularization values 0.03, 0.1, 0.3, 1, 3, 10, 30 (values given for squared Euclidean distances rescaled by the temperature). The optimal value is always $\gamma = 3$ except for CUB ($\gamma = 1$), *miniImageNet*→CUB ($\gamma = 1$), *miniImageNet* with ResNet-12 ($\gamma = 10$), and transductive *tieredImageNet* ($\gamma = 1$, value given for Sinkhorn on logit values).

Code based on Snell et al. [2017a]. For same-domain experiments on Omniglot (Section 5.1) and the ablation study (Section 6), we use the Conv-4 architecture from the official Prototypical Networks [Snell et al., 2017a] implementation.⁵ This results in a 64-dimensional embedding for Omniglot and 1600-dimensional embedding for *miniImageNet*. For *miniImageNet*, we pretrain the embedding function using prototypical networks to solve 30-way problems instead of 5, which is the recommended trick in the paper [Snell et al., 2017a]. For Omniglot, we train from scratch on 5-way 5-shot and 20-way 5-shot episodes (number of ways for training and testing match). We use a center loss of 1 and Sinkhorn regularization of $\gamma = 1$, on unnormalized squared Euclidean distances.

Code based on Snell et al. [2017a] and Hsu et al. [2017] For few-shot clustering (Section 5.3), we start from the official ProtoNet implementation and train both Conv-4 and CCN architectures on the CCN splits of Omniglot. We use a center loss of 0.1 and Sinkhorn regularization of $\gamma = 1$, on unnormalized squared Euclidean distances.

Code based on Triantafillou et al. [2020] For cross-domain experiments on Meta-Dataset (Section 5.2) we derive CentroidNet from their implementation of ProtoNet⁶ which uses the ResNet-18 architecture. We train with a center loss of 0.01 and Sinkhorn regularization $\gamma = 0.1$ after rescaling the squared Euclidean distances by the number of dimensions.

⁴<https://github.com/Sha-Lab/FEAT>

⁵<https://github.com/jakesnell/prototypical-networks>

⁶<https://github.com/google-research/meta-dataset>

5 Additional Few-Shot Clustering Results

Table B.1: Clustering accuracies for Centroid Networks and K-Means (raw and ProtoNet features) on Omniglot 5-way 5-shot, Omniglot 20-way 5-shot, and *miniImageNet* 5-way 5-shot.

FSC Method dataset→	Clustering Accuracy			
	Omniglot-5	Omniglot-20	<i>miniImageNet</i> -5	<i>tieredImageNet</i> -5
K-Means (Raw)	45.2 ± 0.5	30.7 ± 0.2	41.4 ± 0.4	-
K-Means (Conv)	83.5 ± 0.8	76.8 ± 0.4	48.7 ± 0.5	-
CentroidNet (Conv)	99.6 ± 0.1	99.1 ± 0.1	64.5 ± 0.7	-
CentroidNet (ResNet)	-	-	77.3	82.49

6 Ablation Study

	Clustering	Train Cond.	Eval Cond.	Center Loss	ClusteringAcc	+/-	UnsuperAcc	+/-
O1	K-means++	Softmax	Softmax	No	83.80%	0.80%	85.20%	0.90%
O2	K-means++	Softmax	Sinkhorn	No	87.32%	0.80%	84.00%	0.80%
O3	K-means++	Sinkhorn	Sinkhorn	Yes	89.80%	0.70%	86.00%	0.90%
O4	Sinkhorn K-means	Sinkhorn	Sinkhorn	No	99.20%	0.20%	98.50%	0.20%
O5	Sinkhorn K-means	Softmax	Softmax	No	99.40%	0.16%	98.90%	0.20%
O6	Sinkhorn K-means	Softmax	Softmax	Yes	99.50%	0.10%	99.00%	0.10%
O7	Sinkhorn K-means	Softmax	Sinkhorn	Yes	99.50%	0.10%	99.00%	0.10%
O8	Sinkhorn K-means	Sinkhorn	Sinkhorn	Yes	99.60%	0.10%	99.10%	0.10%

Figure B.1: Omniglot 5-way 5-shot Ablation Study

	Clustering	Train Cond.	Eval Cond.	Center Loss	ClusteringAcc	+/-	UnsuperAcc	+/-
M1	K-means++	Softmax	Softmax	No	50.50%	0.50%	39.40%	0.70%
M2	K-means++	Softmax	Sinkhorn	No	57.40%	0.60%	38.60%	0.70%
M3	K-means++	Softmax	Softmax	Yes	50.60%	0.30%	39.50%	0.40%
M4	K-means++	Softmax	Sinkhorn	Yes	56.90%	0.30%	38.70%	0.40%
M5	K-means++	Sinkhorn	Softmax	Yes	48.60%	0.60%	36.70%	0.70%
M6	K-means++	Sinkhorn	Sinkhorn	Yes	56.60%	0.60%	35.80%	0.70%
M7	Sinkhorn K-means	Sinkhorn	Sinkhorn	No	62.30%	0.70%	52.50%	0.80%
M8	Sinkhorn K-means	Softmax	Softmax	No	63.70%	0.70%	54.80%	0.80%
M9	Sinkhorn K-means	Softmax	Softmax	Yes	64.50%	0.80%	55.30%	0.80%
M10	Sinkhorn K-means	Softmax	Sinkhorn	Yes	64.50%	0.80%	55.40%	0.80%
M11	Sinkhorn K-means	Sinkhorn	Sinkhorn	Yes	63.10%	0.70%	53.20%	0.80%

Figure B.2: *miniImageNet* 5-way 5-shot Ablation Study

[Figures B.1, B.2] We conduct an ablation study on Omniglot (5-way 5-shot) and *miniImageNet* (5-way 5-shot) to determine the effect and importance of the various proposed tricks and components. Implementations in this section are based on the official ProtoNet implementation [Snell et al., 2017a], thus numbers might differ slightly from the main paper.

- **K-Means vs. Sinkhorn K-Means.** From comparing O3 to O4, O1 to O5, M6 to M7, M1 to M8, it appears that using Sinkhorn K-Means instead of K-Means++ is the most beneficial and important factor.
- **Center Loss.** From comparing O2 to O3, O5 to O6, O4 to O8, M7 to M11, M8 to M9, center loss seems to be beneficial (although the significance is at the limit of the confidence intervals). It is the second most influential factor.
- **Softmax vs. Sinkhorn conditionals** (at train and test time). For training, it is not clear whether using Sinkhorn or Softmax conditionals is beneficial or not. For evaluation, from comparing M1 to M2, M3 to M4, M5 to M6, it seems that Sinkhorn conditionals are better if the metric is clustering accuracy, while Softmax conditionals might be better if the metric is unsupervised accuracy, although the effect seems to be negligible (see how the color patterns are inverted).

7 Confidence Intervals

We give 95% confidence intervals for the accuracies reported in the experimental section.

Table B.2: Confidence Intervals for Same-Domain Benchmarks

<i>miniImageNet</i>		
LT Methods	LT Accuracy	
backbone \rightarrow	Conv	ResNet
MatchNet [Vinyals et al., 2016a]	51.09±0.71	-
MAML [Finn et al., 2017]	63.11±0.92	-
RelationNet [Sung et al., 2018b]	67.07±0.69	-
ProtoNet [Snell et al., 2017a]	68.20±0.66	-
FEAT [Ye et al., 2020]	71.61±0.16	-
TADAM [Oreshkin et al., 2018]	-	76.70±0.30
MetaOptNet [Lee et al., 2019]	-	78.63±0.46
SimpleShot [Wang et al., 2019b]	-	80.02±0.14
CTM [Li et al., 2019b]	-	80.51±0.13
ProtoNet* [Ye et al., 2020]	71.33±0.16	80.53±0.14
FEAT [Ye et al., 2020]	-	82.05±0.14
NLT Methods	NLT Accuracy	
backbone \rightarrow	Conv	ResNet
CentroidNet (ours)	57.57±0.94	69.86±0.94

<i>tieredImageNet</i>		
LT Methods	LT Accuracy	
backbone \rightarrow	ResNet	
ProtoNet [Snell et al., 2017a]	72.69±0.74	
RelationNet [Sung et al., 2018b]	71.32±0.78	
MetaOptNet [Lee et al., 2019]	81.56±0.63	
CTM [Li et al., 2019b]	84.28±1.73	
SimpleShot [Wang et al., 2019b]	84.58±0.16	
ProtoNet* [Ye et al., 2020]	84.03±0.16	
FEAT [Ye et al., 2020]	84.79±0.16	
NLT Methods	NLT Accuracy	
backbone \rightarrow	ResNet	
CentroidNet (ours)	75.36±1.04	

Omniglot		
LT Methods	5-way	20-way
backbone \rightarrow	ConvNet	
SiameseNet [Koch et al., 2015]	98.4	97.0
MatchNet [Vinyals et al., 2016a]	98.9	98.5
NeuralStat [Edwards and Storkey, 2016]	99.5	98.1
MemoryMod [Kaiser et al., 2017]	99.6	98.6
ProtoNet* [Snell et al., 2017a]	99.7	98.9
MAML [Finn et al., 2017]	99.9	98.9
NLT Methods	5-way	20-way
backbone \rightarrow	ConvNet	
CentroidNet (ours)	99.1±0.1	98.1±0.1

CUB		
LT Methods	LT Accuracy	
backbone \rightarrow	ConvNet	
MatchNet [Vinyals et al., 2016a]	72.86±0.70	
MAML [Finn et al., 2017]	72.09±0.76	
ProtoNet [Snell et al., 2017a]	70.77±0.69	
ProtoNet* (repro) [Ye et al., 2020]	75.33±0.71	
RelationNet [Sung et al., 2018b]	76.11±0.69	
MatchNet [Ye et al., 2020]	79.00±0.16	
ProtoNet [Ye et al., 2020]	81.50±0.15	
FEAT [Ye et al., 2020]	82.90±0.15	
NLT Methods	NLT Accuracy	
backbone \rightarrow	ConvNet	
CentroidNet (ours)	66.13±1.08	

Table B.3: Confidence Intervals for Cross-Domain Benchmarks

Meta-Dataset								
Test Dataset method →	Train on ILSVRC				Train on all datasets			
	Proto	LT		NLT	Proto	LT		NLT
		CNAPs	SUR	Centro		CNAPs	SUR	Centro
ILSVRC	44.12	50.6	56.3	26.40±0.88	41.79	52.3	56.3	23.84±0.82
Omniglot	53.40	45.2	67.5	36.83±1.20	81.93	88.4	93.1	66.25±1.12
Aircraft	45.29	36.0	50.4	24.15±0.72	69.43	80.5	85.4	57.50±1.01
Birds	63.59	60.7	71.7	41.08±1.05	64.73	72.2	71.4	43.56±1.03
Textures	61.78	67.5	70.2	39.63±0.70	66.35	58.3	71.5	43.50±0.76
QuickDraw	49.58	42.3	52.4	31.04±0.95	67.74	72.5	81.3	46.96±1.04
Fungi	35.27	30.1	39.1	18.11±0.71	38.94	47.4	63.1	21.76±0.76
VGG Flower	78.09	70.7	84.3	47.98±0.96	84.45	86.0	82.8	55.11±0.95
Traffic Sign	46.08	53.3	63.1	22.03±0.66	49.91	60.2	70.4	22.71±0.66
MSCOCO	35.63	45.2	52.8	18.19±0.69	36.64	42.6	52.4	17.60±0.77

MiniImageNet → CUB		
LT Methods backbone →	LT Accuracy	
	ConvNet	ResNet
MAML [Chen et al., 2019a]	-	51.34±0.72
MatchNet [Chen et al., 2019a]	-	53.07±0.74
RelationNet [Chen et al., 2019a]	-	57.71±0.73
ProtoNet* (repro)	62.52±0.73	61.38±0.76
ProtoNet [Chen et al., 2019a]	-	62.02±0.70
Baseline [Chen et al., 2019a]	-	65.57±0.70
GNN-FT [Tseng et al., 2020]	-	66.32±0.80
Neg-Softmax [Liu et al., 2020]	-	69.30±0.73
NLT Methods backbone →	NLT Accuracy	
	ConvNet	ResNet
CentroidNet (ours)	47.01±0.91	44.62±0.90



Appendix for “Multimodal Pretraining for Dense Video Captioning”

1 The ViTT dataset

Sampling video for annotation. The goal of the ViTT dataset design is to mirror topic distribution in the “wild”. Therefore, instead of starting from specific how-to instructions and searching for corresponding videos, we sampled videos from the validation set of the YouTube-8M dataset [Abu-El-Haija et al., 2016], a large-scale collection of YouTube videos with topical labels, subject to YouTube policies.

Exclusion criteria were lack of English ASR and the topic label “Game”. The latter was motivated by the fact that in this type of videos, the visual information predominantly features video games, while the ViTT dataset was intended to contain only videos with real-world human actions. Cooking videos can be easily identified by sampling videos that came with “Cooking” or “Recipe” topic labels. Given the convenience and the fact that much of prior work in this area had focused on cooking videos, approximately half of the dataset was designed to include cooking videos only, while the remaining videos would be randomly sampled non-cooking videos, as long as they were verified as instructional by human annotators.

Annotation process Annotators were presented with a video alongside its timestamped, automatic transcription shown in sentence-length paragraphs. They were asked to watch the video and first judge whether the video was instructional. For the purpose of our dataset, we determine that a video is instructional if it focuses on real-world human actions that are accompanied by procedural language explaining what is happening on screen, in reasonable details. Also for our purposes, instructional videos need to be grounded in real life, with a real person in the video exemplifying the action being verbally described.

For videos judged to be instructional, annotators were then asked to:

- Delimit the main segments of the video.
- Determine their start time if different from the automatically suggested start time (explained below).
- Provide a label summarizing or explaining the segment.

Annotation guidelines Annotators were instructed to identify video segments with two potential purposes:

- Allow viewers to jump straight to the start of a segment for rewatch.
- Present viewers with an index to decide whether to watch the video in full or directly skip to the segment of interest.

Our guidelines suggested a range of five to ten segments as long as the the structure and content of the video permitted. For short videos, the direction was to prioritize quality over quantity and to only define those segments that formed the narrative structure of the video, even if the resulting number of segments was below 5.

To help annotators determine segment start times, transcriptions were shown in “sentences” — we expected that sentence start times might be good candidates for segment start times. We obtained sentence boundaries automatically as follows. Given the stream of timestamped ASR tokens for a video, we first separated them into blocks by breaking two consecutive tokens whenever they were more than 2 seconds apart. We then used a punctuation prediction model to identify sentence boundaries in each resulting block. Each sentence was shown with the timestamp corresponding to its first token. Annotators were advised that transcriptions had been automatically divided into paragraphs that may or may not correspond to a video segment — if they decided that a segment started from a particular sentence, they could choose to use the start time of the sentence as the start time for the segment, or, if needed, they could put in an adjusted start time instead.

Once the start time had been identified, annotators were asked to provide a free-text label to summarize each segment. We instructed the annotators to use nouns or present participles (-ing form of verbs) to write the labels for the video segments, whenever possible. Additionally, we asked that the labels be succinct while descriptive, using as few words as possible to convey as much information as possible.

Data statistics and post-processing The resulting dataset consists of 8,169 instructional videos that received segment-level annotations, of which 3,381 are cooking-related. Overall there are an average of 7.1 segments per video (max: 19). Given our instructions, the descriptions are much shorter in lengths compared to a typical captioning dataset: on average there are 2.97 words per description (max: 16); 20% of the captions are single-word, 22% are two-words, and 25% are three words. We refer to these descriptions as “tags” given how short they are.

When possible, annotators were also asked to start and end the video with an opening and closing segment. As a result, most annotations start with an introduction segment: this accounts for roughly 11% of the 88455 segments in the dataset (“intro”: 8%, “introduction”: 2.3%). Note that while “intro” and “introduction” are clearly

paraphrases of each other, an automatic metric will penalize a model predicting “intro” when the groundtruth is “introduction”. Similarly, the ending segment was described in several varieties: “outro”: 3.4%, “closing”: 1%, “closure”, “conclusion”, “ending”, “end of video”: each under 1%. Penalizing paraphrases of the ground truth is an inherent weakness of automatic metrics. To mitigate this, we decided to reduce the chance of this happening for the most frequent tags in the dataset. That is, in our experiments, we identified three groups of tags among the top-20 most frequent tags, and standardized them as follows.

intro	intro, introduction, opening
outro	outro, closing, closure, conclusion, ending, end of video, video closing
result	finished result, final result, results

Table C.1: Standardization of top tags

Note that this does not mean we can solve this problem as a classification task like in visual question answering (VQA): overall, there are 56,027 unique tags with a vocabulary size of 12,509 for the 88,455 segments; 51,474 tags appeared only once in the dataset, making it infeasible to reduce the segment-level captioning problem into a pure classification task. Table C.2 shows the top 10 most frequent tags after standardization.

Estimate of human performance. A subset of the candidate videos were given to three annotators¹, to help us understand variations in human annotations. 5,840 videos received dense captioning from exactly one annotator and were used as

¹A small set were unintentionally given to six annotators.

Tag	% of segments
intro	11.4
outro	6.6
result	0.9
ingredients	0.8
listing ingredients	0.2
supplies	0.2
mixing ingredients	0.2
materials	0.1
what you’ll need	0.1
lining the eyes	0.1

Table C.2: 10 most frequent tags after standardization.

training data. Videos with more than one annotation were used as validation / test data. Note that not all the videos with multiple timeline annotations have exactly three sets of them — in fact, 1368 videos received 3-way segment-level annotations. This is because not all annotators agreed on whether a video was instructional. Computing annotator agreement for the annotated timelines is non-trivial. Here we focus on an estimate of tagging agreement when a pair of annotators agreed over the segment start time. Specifically, we go through each video that received multiple segment-level annotations. For each segment where two annotators chose the same ASR sentence as its starting point, we take the tags they produced for this segment and consider one of them as groundtruth, the other as prediction, and add that into our pool of (groundtruth, prediction) pairs. We can then compute standard automatic evaluations metrics over this pool. The results are as follows.

BLEU-1	METEOR	ROUGE-L	CIDEr
43.34	33.56	41.88	1.26

Table C.3: Estimate of human performance for the segment-level captioning on ViTT-All (computed over 7528 pairs).

BLEU-1	METEOR	ROUGE-L	CIDEr
41.61	32.50	41.59	1.21

Table C.4: Estimate of human performance for the segment-level captioning on ViTT-Cooking (computed over 2511 pairs).

Note that METEOR, and CIDEr scores are both penalized by the lack of n-grams for higher n. That is, when both groundtruth and prediction are single-word, say, “intro”, this pair will not receive a full score from any of these metrics. But the ROUGE-L score is in the same ballpark as estimate of human performance in prior work [Hessel et al. \[2019\]](#). One might note that perhaps this pool of label pairs contains a higher share of “intro”, since annotators might be more likely to agree over where an opening segment starts. Indeed, 20% of the time, one of the tags is “intro”. Interestingly, in spite of standardization of top tags, 14% of the time one tag is “intro”, the other tag is *not* “intro”: they can be less frequent paraphrases (e.g., “welcoming”, “greeting”, “opening and welcoming”) or something semantically different (e.g., “using dremel tool”).

2 Separated vs. Concatenated-Modality Architecture

Prior work has explored both concatenating different modalities and feeding them into the same multimodal Transformer encoder [Sun et al., 2019b, Hessel et al., 2019], as well as separating them into unimodal transformers [Sun et al., 2019a, Lu et al., 2019]. We opt for the separated architecture because it offers more flexibility. First, the concatenated architecture requires embedding the text and video features into the same space. When the video features are projected using a simple network, there is no guarantee that we can meaningfully project them into the text embedding space. VideoBERT [Sun et al., 2019b] gives more flexibility to the video embeddings by quantizing video features and learning an embedding for each codeword. However, the quantization step has subsequently been claimed to be detrimental [Sun et al., 2019a]. Moreover, the concatenated architecture uses the same sets of forward and attention weights to process text and video, and performs layer normalization jointly between the two modalities, which is not necessarily meaningful. Finally, the separated architecture makes it easy to switch between variable length text-only, video-only, or text+video modalities, whereas concatenated architectures might rely on separating tokens, modalities embeddings, and using fixed sequence lengths [Luo et al., 2020].

3 Additional Implementation Details

We optimize all models on a nVidia v100 GPU using the Adam optimizer with inverse square root schedule, batch size 32, warm-up period of 4,000 iterations, and maximum learning rate of 0.0001, following MASS [Song et al., 2019]. The positional embeddings are initialized randomly. We use dropout and attention dropout with probabilities 0.1. With E2vidD6, pretraining takes 3-6 days depending on the objective and bidirectional finetuning takes up to 1.5 days, however those times could be improved by optimizing the data pipeline.

4 Example Predictions

We show examples of **good** and **bad** predictions on YouCook2 (Figure C.1 and ViTT-All (Figure C.2 and C.3). The captions are generated by E2vidD6-BiD (no pretraining) and E2vidD6-MASS-BiD (text-only MASS pretraining).

Sample Frame	Ground Truth	Ø-Pretraining	MASS-Pretraining	Comments
	<i>crush and grate the garlic</i>	<i>grate garlic and add to bowl (good)</i>	<i>crush ginger and garlic (good)</i>	ginger is correct despite not appearing in ground truth.
	<i>crimp shut with fork</i>	<i>place the filling on the wrapper (ok)</i>	<i>seal the edges of the wrapper (good)</i>	pretrained model is more specific
	<i>place wings on the baking sheet and cook flipping</i>	<i>bake the pizza in the oven (bad)</i>	<i>cook the wings on the grill (good)</i>	only pretrained model predicted correct food
	<i>add the pork back into the hot oil</i>	<i>add the rice to the pot (bad)</i>	<i>place the meat on the pan (good)</i>	Ø model hallucinates the rice and pot
	<i>add thyme bay leaves onion and clam juice and boil the mixture</i>	<i>add diced tomatoes tomato puree and mix well (bad)</i>	<i>add thyme thyme onion and clam juice to the pot and stir (ok)</i>	Ø hallucinates a lot of nonexistent ingredients
	<i>cook bacon in a pot with oil and pepper</i>	<i>add chopped tomatoes to pan and stir (bad)</i>	<i>add bacon and stir (ok)</i>	both models missed oil and pepper (not mentioned in ASR)
	<i>pour dressing on top of the salad and toss</i>	<i>add dressing to the bowl (good)</i>	<i>serve the soup over the salad (bad)</i>	pretrained model referred to dressing as “soup”
	<i>slice the ginger into pieces</i>	<i>slice a celery (bad)</i>	<i>slice the chicken (bad)</i>	both models had wrong ingredients (ASR segment does not mention what is being sliced)

Figure C.1: Example good and bad predictions on YouCook2. The pretrained model is generally but not always better. Note that there are no “intro” or “outro”-like labels on YouCook2 because the dataset was specifically curated to only contain actual recipe steps.

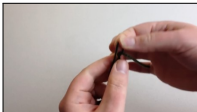





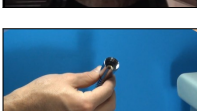
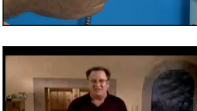




Sample Frame	Ground Truth	Ø-Pretraining	MASS-Pretraining	Comments
	<i>tightening extra loop</i>	<i>tightening the loop (good)</i>	<i>tightening the loop (good)</i>	both models perform well
	<i>adding eye-shadow</i>	<i>blending eye shadow (good)</i>	<i>applying eye shadow (good)</i>	both models perform well
	<i>showcasing the finished look</i>	<i>showing finished look (good)</i>	<i>showing finished look (good)</i>	both models perform well
	<i>rolling and folding the clay</i>	<i>rolling and blending (ok)</i>	<i>rolling and folding the clay (good)</i>	MASS is a bit more specific
	<i>highlighting brow bone</i>	<i>applying eye shadow (ok)</i>	<i>brushing on the brows (good)</i>	MASS is a bit more specific
	<i>covering the chicken and cooking</i>	<i>cooking the bread (bad)</i>	<i>cooking the chicken (good)</i>	only MASS got the right ingredient
	<i>connecting spray hose and sprayer</i>	<i>connecting the new cover (ok)</i>	<i>connecting the valve (good)</i>	spray hose is more specific than valve
	<i>implementing second layer</i>	<i>showing finished product (ok)</i>	<i>showing second layer (good)</i>	MASS is more specific
	<i>making decorative trim</i>	<i>cutting the edges (good)</i>	<i>cutting the fabric (good)</i>	both models yield good predictions
	<i>checking bleach container</i>	<i>outro (bad)</i>	<i>checking the container (good)</i>	MASS is a bit more specific
	<i>demonstrating the flip</i>	<i>checking the battery (bad)</i>	<i>flipping the board (good)</i>	Ø model got influenced by car mechanics tutorials
	<i>tilting board</i>	<i>setting up the oven (bad)</i>	<i>turning the board (good)</i>	Ø overfitted on cooking videos

Figure C.2: Example **good** predictions on ViTT-All (Part 1). The pretrained model is generally but not always better.

Sample Frame	Ground Truth	Ø-Pretraining	MASS-Pretraining	Comments
	<i>securing the bar in place</i>	<i>removing the cover (bad)</i>	<i>checking for the other side (bad)</i>	predictions are not specific enough
	<i>starting with unlocking bars</i>	<i>opening the box (bad)</i>	<i>pulling the car on (bad)</i>	predictions are incorrect or not specific enough
	<i>demonstrating technique</i>	<i>attaching paper (bad)</i>	<i>stamping paper (good)</i>	the technique is about stamping the paper
	<i>spritzing in additional water</i>	<i>pouring water into the water (ok)</i>	<i>adding water to water (ok)</i>	understandable but ungrammally
	<i>checking for leaks</i>	<i>checking for the new new new new new new new new new (bad)</i>	<i>checking the process (ok)</i>	Ø got into a loop, MASS not specific enough
	<i>displaying materials needed</i>	<i>intro (bad)</i>	<i>removing paste (ok)</i>	prediction makes sense because narrator is displaying thermal paste remover
	<i>sketching on the swirls</i>	<i>drawing the lines (good)</i>	<i>drawing on the eyes (bad)</i>	pretrained model overfitted on makeup tutorials
	<i>crimping wire and completing project</i>	<i>attaching the screws (bad)</i>	<i>attaching the wire to the wire (ok)</i>	both models have trouble with the concept of crimping a wire
	<i>cutting with guide line</i>	<i>cutting the top of the top of the top of the top of the top (bad)</i>	<i>explaining process (ok)</i>	Ø model got into a loop, MASS model is not specific enough

Figure C.3: Example **ok** and **bad** predictions on ViTT (Part 2). The pretrained model is generally but not always better.

5 Full result tables

We present tables with all the ablation results that we run. There are two main takeaway messages from the results involving the pretraining approach: (a) the accuracy improvements, as measured across all the metrics we use, indicate the value of using a pretraining approach to this problem, specifically one that is capable of leveraging the ASR signals at both pretraining and finetuning stages, and (b) the training speedup achieved from pretraining is impressive, as a pretrained model converges much faster than training from scratch. This is especially visible on ViTT-All where finetuning after MASS pretraining reaches best ROUGE-L score at epoch 2, whereas it takes around 11 epochs to converge when training from scratch.

Method	Input	Pretraining	BLEU-4	METEOR	ROUGE-L	CIDEr
Constant Pred [Hessel et al., 2019]	-	-	2.70	10.30	21.70	0.15
MART Lei et al. [2020]	Video	-	8.00	15.90	-	0.36
DPC Shi et al. [2019]	Video + ASR	-	2.76	18.08	-	-
EMT Zhou et al. [2018c]	Video	-	4.38	11.55	27.44	0.38
CBT Sun et al. [2019a]	Video	Kinetics + HowTo100M	5.12	12.97	30.44	0.64
AT [Hessel et al., 2019]	ASR	-	8.55	16.93	35.54	1.06
AT+Video [Hessel et al., 2019]	Video + ASR	-	9.01	17.77	36.65	1.12
UniViLM #1 [Luo et al., 2020]	Video	-	6.06	12.47	31.48	0.64
UniViLM #2 [Luo et al., 2020]	Video + ASR	-	8.67	15.38	35.02	1.00
UniViLM #5 [Luo et al., 2020]	Video + ASR	HowTo100M	10.42	16.93	38.02	1.20
<i>∅ Pretraining</i>						
E2D2-UniD	ASR	-	7.42	15.15	33.26	0.85
E2D6-UniD	ASR	-	7.88	15.29	34.10	0.87
E2D2-BiD	ASR	-	6.85	15.64	34.26	0.91
E2D6-BiD	ASR	-	7.90	15.70	34.86	0.93
E2vidD2-UniD	Video + ASR	-	7.47	15.11	34.77	0.90
E2vidD6-UniD	Video + ASR	-	7.61	15.57	34.28	0.89
E2vidD2-BiD	Video + ASR	-	8.39	15.36	34.54	0.91
E2vidD6-BiD	Video + ASR	-	8.01	16.19	34.66	0.91
E2vidD2-BiDalt	Video + ASR	-	8.12	15.83	34.83	0.93
E2vid,D6-BiDalt	Video + ASR	-	7.70	16.11	34.78	0.91
E2vidD2-BiD (S3D)	Video + ASR	-	8.04	16.17	36.01	0.96
E2vidD6-BiD (S3D)	Video + ASR	-	7.91	16.28	35.23	0.93
<i>Text Pretraining</i>						
E2D2-MASS-UniD	ASR	YT8M-cook + Recipe1M	10.52	17.14	37.39	1.14
E2D6-MASS-UniD	ASR	YT8M-cook + Recipe1M	10.72	17.74	37.85	1.17
E2D2-MASS-BiD	ASR	YT8M-cook + Recipe1M	10.84	17.44	37.20	1.13
E2D6-MASS-BiD	ASR	YT8M-cook + Recipe1M	10.60	17.42	38.08	1.20
E2vidD2-MASS-UniD	Video + ASR	YT8M-cook + Recipe1M	10.84	17.39	38.24	1.16
E2vidD6-MASS-UniD	Video + ASR	YT8M-cook + Recipe1M	11.39	18.00	38.71	1.22
E2vidD2-MASS-BiD	Video + ASR	YT8M-cook + Recipe1M	11.38	18.04	38.67	1.19
E2vidD6-MASS-BiD	Video + ASR	YT8M-cook + Recipe1M	11.47	17.70	38.80	1.25
E2vid,D2-MASS-BiDalt	Video + ASR	YT8M-cook + Recipe1M	11.49	17.85	38.60	1.18
E2vid,D6-MASS-BiDalt	Video + ASR	YT8M-cook + Recipe1M	11.07	17.68	38.43	1.22
E2vidD2-MASS-BiD (S3D)	Video + ASR	YT8M-cook + Recipe1M	11.13	17.71	38.57	1.12
E2vidD6-MASS-BiD (S3D)	Video + ASR	YT8M-cook + Recipe1M	11.64	18.04	38.75	1.24
<i>Multimodal Pretraining</i>						
E2vidD2-MASSalign-BiD	Video + ASR	YT8M-cook + Recipe1M	11.54	17.57	37.70	1.15
E2vidD6-MASSalign-BiD	Video + ASR	YT8M-cook + Recipe1M	11.53	17.62	39.03	1.22
E2vidD2-MASSvid-BiD	Video + ASR	YT8M-cook + Recipe1M	11.17	17.71	38.32	1.17
E2vidD6-MASSvid-BiD	Video + ASR	YT8M-cook + Recipe1M	12.04	18.32	39.03	1.23
E2vidD2-MASSdrop-BiD	Video + ASR	YT8M-cook + Recipe1M	11.21	17.99	38.72	1.23
E2vidD6-MASSdrop-BiD	Video + ASR	YT8M-cook + Recipe1M	10.45	17.74	38.82	1.22
Human [Hessel et al., 2019]	Video + ASR	-	15.20	25.90	45.10	3.80

Table C.5: Video Captioning Results on YouCook2. We use YT8M-cook/Recipe1M for pretraining. All video features are Compact 2D [Wang et al., 2014] except when marked as S3D [Xie et al., 2018].

Method	Input	Pretraining	BLEU-1	METEOR	ROUGE-L	CIDEr
Constant baseline (“intro”)	-	-	1.42	3.32	11.15	0.28
<i>∅ Pretraining</i>						
E2D2-UniD	ASR	-	17.94	8.55	27.06	0.64
E2D6-UniD	ASR	-	18.91	8.96	27.80	0.67
E2D2-BiD	ASR	-	18.81	8.82	27.63	0.65
E2D6-BiD	ASR	-	19.60	9.12	27.88	0.68
E2vidD2-UniD	Video + ASR	-	18.94	8.99	28.05	0.67
E2vidD6-UniD	Video + ASR	-	19.29	9.15	27.97	0.69
E2vidD2-BiD	Video + ASR	-	19.37	9.21	28.56	0.69
E2vidD6-BiD	Video + ASR	-	19.49	9.23	28.53	0.69
<i>Text Pretraining</i>						
E2D2-MASS-UniD	ASR	HowTo100M + WikiHow	21.53	10.24	29.95	0.77
E2D6-MASS-UniD	ASR	HowTo100M + WikiHow	22.09	10.58	30.67	0.79
E2D2-MASS-BiD	ASR	HowTo100M + WikiHow	20.73	10.20	30.15	0.76
E2D6-MASS-BiD	ASR	HowTo100M + WikiHow	21.93	10.60	30.45	0.79
E2vidD2-MASS-UniD	Video + ASR	HowTo100M + WikiHow	21.46	10.45	30.56	0.78
E2vidD6-UniD	Video + ASR	HowTo100M + WikiHow	22.21	10.75	30.86	0.81
E2vidD2-MASS-BiD	Video + ASR	HowTo100M + WikiHow	21.78	10.64	30.72	0.79
E2vidD6-MASS-BiD	Video + ASR	HowTo100M + WikiHow	22.44	10.83	31.27	0.81
<i>Multimodal Pretraining</i>						
E2vidD2-MASSalign-BiD	Video + ASR	HowTo100M + WikiHow	22.07	10.33	30.60	0.77
E2vidD6-MASSalign-BiD	Video + ASR	HowTo100M + WikiHow	22.31	10.66	31.13	0.79
E2vidD2-MASSvid-BiD	Video + ASR	HowTo100M + WikiHow	22.15	10.75	31.06	0.80
E2vidD6-MASSvid-BiD	Video + ASR	HowTo100M + WikiHow	22.45	10.76	31.49	0.80
E2vidD2-MASSdrop-BiD	Video + ASR	HowTo100M + WikiHow	21.84	10.55	31.10	0.79
E2vidD6-MASSdrop-BiD	Video + ASR	HowTo100M + WikiHow	22.37	11.00	31.40	0.82
Human estimate	Video + ASR	-	43.34	33.56	41.88	1.26

Table C.6: Video captioning results on ViTT-All. We use HowTo100M/WikiHow for pretraining. We also estimate human performance (details in Appendix 1; Tables C.3,C.4).

Method	Input	Pretraining	BLEU-1	METEOR	ROUGE-L	CIDEr
Constant baseline (“ <i>intro</i> ”)	-	-	1.16	2.93	10.21	0.25
<i>∅ Pretraining</i>						
E2D2-UniD	ASR	-	19.73	9.43	27.95	0.69
E2D6-UniD	ASR	-	20.24	9.93	28.59	0.71
E2D2-BiD	ASR	-	19.73	9.72	27.92	0.68
E2D6-BiD	ASR	-	20.77	10.08	28.63	0.72
E2vidD2-UniD	Video + ASR	-	19.97	9.75	28.30	0.69
E2vidD6-UniD	Video + ASR	-	20.46	9.93	28.62	0.69
E2vidD2-BiD	Video + ASR	-	20.60	10.08	29.45	0.71
E2vidD6-BiD	Video + ASR	-	20.45	9.88	28.88	0.69
<i>Text Pretraining</i>						
E2D2-MASS-UniD	ASR	YT8M-cook + Recipe1M	22.89	11.53	31.62	0.84
E2D6-MASS-UniD	ASR	YT8M-cook + Recipe1M	24.47	12.22	32.51	0.90
E2D2-MASS-BiD	ASR	YT8M-cook + Recipe1M	22.75	11.63	31.54	0.84
E2D6-MASS-BiD	ASR	YT8M-cook + Recipe1M	24.79	12.25	32.40	0.88
E2vidD2-MASS-UniD	Video + ASR	YT8M-cook + Recipe1M	23.86	11.85	32.32	0.86
E2vidD6-MASS-UniD	Video + ASR	YT8M-cook + Recipe1M	24.32	12.32	32.90	0.90
E2vidD2-MASS-BiD	Video + ASR	YT8M-cook + Recipe1M	22.93	11.68	32.15	0.87
E2vidD6-MASS-BiD	Video + ASR	YT8M-cook + Recipe1M	24.22	12.22	32.60	0.89
<i>Multimodal Pretraining</i>						
E2vidD2-MASSalign-BiD	Video + ASR	YT8M-cook + Recipe1M	24.02	11.91	32.73	0.86
E2vidD6-MASSalign-BiD	Video + ASR	YT8M-cook + Recipe1M	24.92	12.25	33.09	0.90
E2vidD2-MASSvid-BiD	Video + ASR	YT8M-cook + Recipe1M	24.15	12.10	32.96	0.88
E2vidD6-MASSvid-BiD	Video + ASR	YT8M-cook + Recipe1M	24.87	12.43	32.97	0.90
E2vidD2-MASSdrop-BiD	Video + ASR	YT8M-cook + Recipe1M	23.70	12.01	32.71	0.88
E2vidD6-MASSdrop-BiD	Video + ASR	YT8M-cook + Recipe1M	24.48	12.22	33.10	0.89
Human estimate	Video + ASR	-	41.61	32.50	41.59	1.21

Table C.7: Video captioning results on ViTT-Cooking. We use YT8M-cook and Recipe1M for optional pretraining.