Université de Montréal

Rethinking Continual Learning Approach and Study Out-Of-distribution Generalization Algorithms

par

Touraj Laleh

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

Mémoire présenté en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en Informatique

August 10, 2023

Université de Montréal

Faculté des arts et des sciences

Ce mémoire intitulé

Rethinking Continual Learning Approach and Study Out-Of-distribution Generalization Algorithms

présenté par

Touraj Laleh

a été évalué par un jury composé des personnes suivantes :

Gauthier Gidel

(président-rapporteur)

Irina Rish

(directeur de recherche)

Esma Aimeur

(membre du jury)

Résumé

L'un des défis des systèmes d'apprentissage automatique actuels est que les paradigmes d'IA standard ne sont pas doués pour transférer (ou exploiter) les connaissances entre les tâches. Alors que de nombreux systèmes ont été formés et ont obtenu des performances élevées sur une distribution spécifique d'une tâche, il est pas facile de former des systèmes d'IA qui peuvent bien fonctionner sur un ensemble diversifié de tâches qui appartiennent aux différentes distributions. Ce problème a été abordé sous différents angles dans différents domaines, y compris l'apprentissage continu et la généralisation hors distribution.

Si un système d'IA est formé sur un ensemble de tâches appartenant à différentes distributions, il pourrait oublier les connaissances acquises lors des tâches précédentes. En apprentissage continu, ce processus entraîne un oubli catastrophique qui est l'un des problèmes fondamentaux de ce domaine. La première projet de recherche dans cette thèse porte sur la comparaison d'un apprenant chaotique et d'un naïf configuration de l'apprentissage continu. La formation d'un modèle de réseau neuronal profond nécessite généralement plusieurs itérations, ou époques, sur l'ensemble de données d'apprentissage, pour mieux estimer les paramètres du modèle. La plupart des approches proposées pour ce problème tentent de compenser les effets de mises à jour des paramètres dans la configuration incrémentielle par lots dans laquelle le modèle de formation visite un grand nombre de échantillons pour plusieurs époques. Cependant, il n'est pas réaliste de s'attendre à ce que les données de formation soient toujours alimenté au modèle. Dans ce chapitre, nous proposons un apprenant de flux chaotique qui imite le chaotique comportement des neurones biologiques et ne met pas à jour les paramètres du réseau. De plus, il peut fonctionner avec moins d'échantillons par rapport aux modèles d'apprentissage en profondeur sur les configurations d'apprentissage par flux. Fait intéressant, nos expériences sur différents ensembles de données montrent que l'apprenant de flux chaotique a moins d'oubli catastrophique de par sa nature par rapport à un modèle CNN en continu apprentissage.

Les modèles d'apprentissage en profondeur ont une performance de généralisation hors distribution naïve où la distribution des tests est inconnue et différente de la formation. Au cours des dernières années, il y a eu eu de nombreux projets de recherche pour comparer les algorithmes hors distribution, y compris la moyenne et méthodes basées sur les scores. Cependant, la plupart des méthodes proposées ne tiennent pas compte du niveau de difficulté de tâches. Le deuxième projet de recherche de cette thèse, l'analyse de certains éléments logiques et pratiques les forces et les inconvénients des méthodes existantes de comparaison et de classement hors distribution algorithmes. Nous proposons une nouvelle approche de classement pour définir les ratios de difficulté des tâches afin de comparer les algorithmes de généralisation hors distribution. Nous avons comparé la moyenne, basée sur le score, et des classements basés sur la difficulté de quatre tâches sélectionnées du benchmark WILDS et cinq algorithmes hors distribution populaires pour l'expérience. L'analyse montre d'importantes changements dans les ordres de classement par rapport aux approches de classement actuelles.

Mots clés : Apprentissage en profondeur, Réseau Neuronal Convolutif, Apprentissage continu, Généralisation.

Abstract

One of the challenges of current machine learning systems is that standard AI paradigms are not good at transferring (or leveraging) knowledge across tasks. While many systems have been trained and achieved high performance on a specific distribution of a task, it is not easy to train AI systems that can perform well on a diverse set of tasks that belong to different distributions. This problem has been addressed from different perspectives in different domains including continual learning and out-of-distribution generalization.

If an AI system is trained on a set of tasks belonging to different distributions, it could forget the knowledge it acquired from previous tasks. In continual learning, this process results in catastrophic forgetting which is one of the core issues of this domain. The first research project in this thesis focuses on the comparison of a chaotic learner and a naive continual learning setup. Training a deep neural network model usually requires multiple iterations, or epochs, over the training data set, to better estimate the parameters of the model. Most proposed approaches for this issue try to compensate for the effects of parameter updates in the batch incremental setup in which the training model visits a lot of samples for several epochs. However, it is not realistic to expect training data will always be fed to the model. In this chapter, we propose a chaotic stream learner that mimics the chaotic behavior of biological neurons and does not update network parameters. In addition, it can work with fewer samples compared to deep learning models on stream learning setups. Interestingly, our experiments on different datasets show that the chaotic stream learner has less catastrophic forgetting by its nature in comparison to a CNN model in continual learning.

Deep Learning models have a naive out-of-distribution (OoD) generalization performance where the testing distribution is unknown and different from the training. In the last years, there have been many research projects to compare OoD algorithms, including average and score-based methods. However, most proposed methods do not consider the level of difficulty of tasks. The second research project in this thesis, analysis some logical and practical strengths and drawbacks of existing methods for comparing and ranking OoD algorithms. We propose a novel ranking approach to define the task difficulty ratios to compare OoD generalization algorithms. We compared the average, score-based, and difficulty-based rankings of four selected tasks from the WILDS benchmark and five popular OoD algorithms for the experiment. The analysis shows significant changes in the ranking orders compared with current ranking approaches.

Key Words : Deep Learning, Convolutional Neural Network, Continual Learning, Generalization.

Contents

List of tables

List of figures

List of Abbreviations

OoD	Out-Of-Distribution
DL	Deep Learning
CNN	Convolutional Neural Network
CIFAR	Canadian Institute for Advanced Research
FMoW	Functional Map of the World Dataset
CelebA	Large-scale CelebFaces Attributes Dataset

Remerciements

To you.

Acknowledgements

First, I express my sincere gratitude to my supervisor, Dr. Irina Rish, for providing guidance and feedback throughout my study. Being her student was an honor, and I learned many things from her. Second, I thank my primary collaborator and true friend Mojtaba Faramarzi. I also want to thank all my teammates in the Irina research team who participated in weekly meetings and for all their effort to make an excellent lab for research and a fun place to learn new things. I am also grateful to be part of Mila, the Quebec AI Institute. Finally, I want to acknowledge Compute Canada and Calcul Quebec for providing the computing resources used in this work. This thesis would not have been possible without the support of my friends and family!

Chapter 1

Introduction

Artificial intelligence (AI) is defined as the study and design of rational agents [63]. An intelligent system is a system that tries to maximize some expected notion of performance [4]. The long-standing goal of AI is to design a system that can imitate the behavior of humans or can demonstrate human-like general intelligence [76]. In the last decades, Deep Learning (DL), has achieved tremendous performance in different machine learning tasks, including object recognition [35], image classification [28, 35, 60], speech recognition [30], and natural language understanding [74, 79]. However, there are also some challenges, such as catastrophic forgetting and poor performance on the testing data when it has different distribution from training data. Therefore, enhancing the generalization ability of Machine Learning models is an important topic discussed in various research fields, such as Continual Learning, Lifelong Learning, Domain Generalization, and Out of distribution Generalization.

1.1. Machine Learning

Artificial Intelligence (AI) as a system that tries to reason and acts like humans. AI was first introduced in 1956 [49] and since that time, scientists from different domains such as cognitive science and computer science have made lots of advancements in understating intelligence and developing AI systems. In the early stage, AI systems were mainly rule-based systems. They were given a set of rules for reasoning. Later, they would do simple tasks such as search using rules [25]. However, some things could be improved with rule-based systems. First, these systems' performance depends on the number of rules given to the system for process. As a result, they do not do very well when the number of rules increases. Second, describing most real word systems using clean and well-organized rules was difficult. Machine learning is a subset of AI that studies problems and algorithms that automatically improve this notion of performance through experience using raw data. Machine learning systems hire classification, regression, and pattern recognition techniques to infer knowledge from data [72]. There are two machine learning systems types: parametric and non-parametric models. Parametric models (such as logistic regression and neural networks) infer knowledge in the given data using a set of parameters. On the other hand, non-parametric models (such as KNN) do not learn any parameters from the given data, though they infer some metrics and statistics such as mean for inference [72].

Learning algorithms are also categorized into supervised learning, unsupervised learning, semi-supervised learning, self-supervised learning, and reinforcement learning [23]. In supervised learning, the datasets that are given to the learner have the correct target clearly defined, and therefore the goal of the model is to find a relationship between the input and the learning target [72].

1.2. Deep Learning

Today, the most popular branch of machine learning that focuses on learning representation from given data is deep learning. The idea behind deep learning is to stack layers of computational units (called layers) to create robust architecture to build a parametric model. These layers facilitate data representation and hierarchical learning through sequential layers of abstraction [40, 67]. Deep learning models learn automatically and extract features from raw data using forward and backward propagation. These automatically extracted features in higher levels of abstraction are constructed from lower-level features in the hierarchy [41]. Features passing through subsequent layers could be transformed into more complex features. One example of deep learning models is convolutional neural networks (CNN). These networks are constructed by stacking convolutional and fully connected layers. CNN receives a tensor that can represent an image's raw data, often in three color channels (Red, Green, and Blue). These input data will be processed in CNN models one layer after another [41, 72].

1.3. Continual Learning

One key challenge of standard AI systems is that they are not good at transferring knowledge across different tasks. There have been many approaches to train a system that performs well in a specific task. However, it is not easy to train an AI system that performs well on a diverse set of tasks. Humans do not need to train over a stationary data distribution for multiple epochs. While they do not have perfect memory, they can incrementally acquire and update knowledge over their lifetime without catastrophically forgetting the knowledge relevant to the previous tasks. The challenge of training a system on a sequence of tasks is keeping the essential knowledge acquired from training previous tasks. If a system forgets the knowledge acquired from the previous task, it will show poor performance on previous tasks [50, 58]. This concept is referred to as catastrophic forgetting. Catastrophic forgetting affects parametric systems by forgetting the parameters of a system learn during training previous tasks [50, 72]. To make it clear, consider a system is trained on n different tasks $(t_1, t_2, ..., t_n)$ and the goal is to train the system on a new task which is t_{n+1}. The best situation for the system would be learning the new task and retaining its performance on the previous tasks. However, in practice, the system usually loses some of the previous knowledge and may be unable to learn new tasks. This situation is explained as **stability-plasticity dilemma** [72]. Plasticity refers to the fact that the system can retain previous knowledge without forgetting them [51]. Stability refers to learning new tasks [52]. Many research projects in machine learning focus on the stability-plasticity dilemma. Some approaches, like transfer learning, focus only on the plasticity aspect. In transfer learning, the focus is on learning the current task, even if it results in poor performance of the previous task [72]. Continual learning is a paradigm that focuses on both aspects: building a system that keeps acquiring new knowledge without forgetting the prior one. Continual learning can be applied on a single task and multi-task setup [72]. There are different setups in continual learning that categorizes problem from different perspectives. Each setup is based on specific assumptions, such as having access to the dataset beforehand and possibly changing the network architecture.

1.3.1. Continual Learning formulation

Consider there is a training data $D = \{(x_i, y_i)_{i=1}^{n}\}$, that includes n pairs of input samples $x_i \in \mathcal{X}$ and their corresponding target vectors $y_i \in \mathcal{Y}$. The goal of supervised learning is to learn a function $f : \mathcal{X} \to \mathcal{Y}$. This function can use an input sample $x \in \mathcal{X}$ and predict a target vector $y \in \mathcal{Y}$. Now the point is to calculate how much the prediction function differs from the ground truth. Therefore, a loss function is defined, and the risk associated with this function is [72]:

$$R(f) = \mathbb{E}_{x,y \sim P}[L(f(x), y)]\,, \tag{1.3.1}$$

where $P(x,y)$ is assumed as a fixed distribution, data is drowned. Therefore, using the Empirical Risk Minimization (ERM) principle, the optimal function that minimizes the empirical risk $\hat{R}$

$$\hat{R}(f) = \frac{1}{n}\sum_{i=1}^{n} L(f(x_i), y_i)\,, \tag{1.3.2}$$

$$\hat{f} = argmin_f \hat{R}(f)\,. \tag{1.3.3}$$

In continual learning setup, there is set of tasks (T_s) where each task includes a set of unique classes $C^{(t)}$. The tasks come in a sequence one by one and each task t comes with its set of data $D^{(t)} = \{(x_i, y_i)_{i=n}^{n_t+1}\}$, where $x_i \in \mathcal{X}$ and $y_i \subseteq \mathcal{C}^{(t)}$. Any time a new task is added to the sequence, the output set for function f will be updated. Applying empirical risk minimization, the risk function will be as follow [72]:

$$\hat{R}(f) = \frac{1}{-}\sum_{t=1} \frac{1}{|D^{(t)}|} \sum_{(x_i,y_i)\in D^{(t)}} L(f(x_i), y_i)\,, \tag{1.3.4}$$

$$\hat{f} = argmin_f \hat{R}(f)\,. \tag{1.3.5}$$

1.3.2. Different scenarios in Continual Learning

In studying continual learning, it is essential to understand different possible scenarios. Each scenario specifies clear and well-defined boundaries between the tasks to be learned during the training of the model [77]. These scenarios are Domain-incremental Learning, Task-incremental Learning, and Class-incremental Learning. These scenarios are defined based on whether the system knows about task identities during evaluation. Figure 1.1 depicts three scenarios that will be discussed in detail. In domain-incremental setup, there is no access to the task identity t during evaluation. Therefore, in domain-incremental, the input distribution is different while the output distribution is the same [72, 77]. For a and b from the set of whole numbers (W) we have :

$$\forall a, b \in W \text{ if } a \neq b : P(x^{(a)}) \neq P(x^{(b)}) \wedge \mathcal{C}^{(a)} = \mathcal{C}^{(b)}$$

In this setup, the model needs to infer which task it is and only focus on doing the task. A clear real-world An example of this setup is an agent that is required to learn to survive in different environments without explicitly identifying the environment it is confronted with [77].

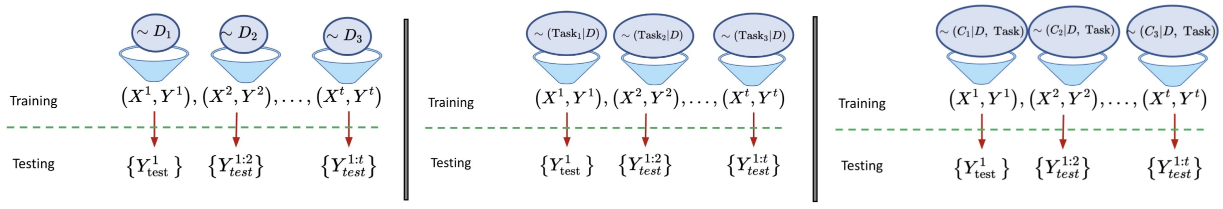
In the task-incremental scenario, the task identity is always known during train and evaluation. In this setup, the output is disjoint between tasks, and the evaluation of the model is based on the average performance across all tasks after training [72, 77].

$$\forall a, b \in W \text{ if } a \neq b : P(x^{(a)}) \neq P(x^{(b)}) \wedge P(y^{(a)}) \neq P(y^{(b)}) \wedge \; \mathcal{C}^{(a)} \cap \mathcal{C}^{(b)} = \Phi$$

In this scenario, it is possible to train task-specific model components. A typical network architecture in this scenario has a "multi-headed" output layer. It means each task has its output units, and other units of the network is shared among tasks [77].

Unlike the two discussed scenarios, in the class incremental learning scenario, the model has to infer task identity and classify tasks seen so far without a task-ID [72, 77]. The common real-world problem of incrementally learning new classes of objects.

$$\forall a, b \in W \text{ if } a \neq b : P(x^{(a)}) \neq P(x^{(b)}) \wedge P(y^{(a)}) \neq P(y^{(b)})$$



(a) Domain-Incremental **(b)** Task-Incremental **(c)** Class-Incremental

Fig. 1.1. Overview of the three Continual learning scenarios [72, 77]

1.3.3. Continual Learning strategies

Many approaches have been proposed for different scenarios in continual learning. Many approaches have been proposed for different scenarios in continual learning. Generally, continual learning methods are categorized into three categories: regularization-based, Memory-based, and architecture-based [72]. Architecture-based methods change the model's architecture by adding or removing a set of parameters. These methods specify a set of isolated parameters for each task. Memory-based methods keep some examples from previous tasks and revisit them while learning new tasks. The main feature of these approaches is *episodic memory* which is a subset of the observed examples from previous tasks. This set is used to minimize forgetting previous tasks by regularizing the learning of the upcoming tasks [72]. Regularization-based methods focus on regularizing the loss function to minimize parameter updates for new tasks. This issue comes from the fact that when a network is trained on a task, it tries to minimize objects on the current task. As a result, network parameters change across different tasks and reduce performance on previous tasks, called catastrophic forgetting. To avoid that, regularization-based methods penalize objective functions for limiting the changes in the model parameters when switching among different tasks. Regularization-based methods are categorized into four categories including : importance-based regularization, Bayesian-based regularization, distillation-based regularization, and optimization trajectory-based regularization [72].

1.4. Out-of-distribution generalization

Generalization is defined as a set of methods to reduce the gap in performance of an agent acting in seen training situations and the same agent acting in unknown test ones [4, 5]. The kind of generalization commonly addressed by statistical learning is in-distribution (i.i.d) when the data-generating process from the training examples is indistinguishable from that of the test examples. However, suppose the model is trained on a set of data but tested on a different set that is statistically different in any way. In that case, this is not an in-distribution generalization problem.

By definition, generalization problems that are not in-distribution are called out-of-distribution (OoD) generalization problems [4, 5, 69]. In most real-world cases, the goal is to build a system that performs well across many different tasks. In many high-risk real cases, such as healthcare, military, and autonomous driving, where training and test data are not identically and independently distributed, it is critical to generalize under distribution shift. In the OoD setup, the model has access to multiple datasets (or tasks) during training. Each of them includes examples collected from different environments or experimental conditions. The main goal of OoD generalization algorithms is to incorporate invariance across this training environment [26, 82].

1.4.1. Out-of-distribution generalization formulation

As we discussed in 1.3.1, classic supervised learning assumes data is drawn from a fixed distribution $P(X, Y)$. However, in real-world scenarios, the test distribution on which the model has been deployed may deviate from training distribution $P_{train}(X, Y) \neq P_{test}(X, Y)$. It means general out-of-distribution is a subset of supervised learning where the test distribution shifts from the training distribution and remains unknown during training. This is the result of a distribution shift that could happen due to reasons such as sample selection bias or data evaluation [69]. In most cases, it isn't easy to assume what types of shifts could happen during the test time. To make the problem feasible, there are some assumptions on distribution shifts that could occur in test data, such as covariant shift, diversity shift, and correlation shift [69]. Covariant shift happens when marginal distribution of X shifts from training phase $P_{train}(X) \neq P_{test}(X)$, but label generation mechanism $P_{train}(Y/X) = P_{test}(Y/X)$ [69]. Diversity shift usually happens when there are multiple domains in the detests. Each domain represents a certain spectrum of diversity. Train and test sets contain a specific group, which results in a diversity shift during the experiments. Finally, correlation shift focus on another type of shift caused by spurious correlations.

1.4.2. Out-of-distribution generalization methods

In recent years, many approaches have been proposed in the domain of out-of-distribution generalization. These approaches tackle three main challenges. First, a way to characterize and formalize distributional shift. Second, a method to design algorithms with good generalization performance. Third, a mechanism to evaluate out-of-distribution generalization algorithms, which require specifically designed datasets with distributional shift [69]. There are different branches of methods in literature trying to address these challenges, including domain generalization methods [21, 53], Casual learning methods [5, 9] and Stable learning methods [55, 68]. Domain generalization methods focus on data collected from different domains [70, 80]. Casual learning methods use casual structures and distribution shifts to formalize, train and test data. Finally, stable learning methods mostly use selection bias to model distributional shifts [69].

Domain generalization methods focus on real scenarios and utilize data collected from different domains. Data manipulating methods focus on ways to increase the diversity of existing training data. Two popular data manipulation techniques are data augmentation and data generation. Augmentation is one of the most helpful manipulation techniques, which usually includes flipping, rotation, scaling, cropping, and adding noise of training data [80]. They have been widely used in supervised learning to enhance the generalization performance of a model by reducing overfitting [70]. Data generation is another interesting technique that tries to increase the generalization capability of a model. The generative function for this approach is mostly implemented using approaches such as Variational Autoencoder(VAE) [32] and Generative Adversarial Networks (GAN) [24]. Casual learning methods are categorized into two groups, including domain-invariant representation learning and feature disentanglement. They try to learn features that remain invariant across different domains, which means the representations are general and transferable to other domains. The goal of domain-invariant representation learning methods is to reduce the representation discrepancy among different domains in a specific feature space to be domain invariant so that the learned model can have a generalizable capability to the unseen domain[80]. Invariant risk minimization (IRM) [4, 5] is another approach that considers another perspective on domain invariance representation for domain generalization. The main idea of IRM is to learn invariances across environments and find a data representation such that the optimal classifier on top of that representation matches for all environments.

Considering the abovementioned research, the research community has spent significant effort developing algorithms able to generalize out-of-distribution. These proposed algorithms try to consider and estimate different types of invariances from data. However, despite the enormous importance of out-of-distribution generalization, the literature is scattered [26]. Every year many different algorithms appear that are evaluated under different datasets and model selection criteria. In this situation, it is important to have a mechanism to evaluate and compare out-of-distribution generalization algorithms. Domainbed [26] is proposed as a framework to streamline rigorous and reproducible experimentation in domain generalization. In domainbed, different domain generalization algorithms, datasets, and some model selection criteria are implemented. In conclusion of all experiments, domainbed figured out that "When equipped with modern neural network architectures and data augmentation techniques, empirical risk minimization achieves state-of-the-art performance in domain generalization" [26].

1.4.3. Algorithms

In recent years, many efforts have been made to address the challenges of out-of-distribution generalization. In this section, we discuss some of the most well credited methods from different categories discussed in previous section including Empirical Risk Minimization, Invariant risk minimization (IRM) [4, 5].

1.4.3.1. **Empirical Risk Minimization**. Empirical risk minimization minimizes the average training loss across training points. Given a loss function $\ell(x, y; \theta) : X \times Y \times \Theta \to R$ (e.g., cross-entropy loss), ERM minimizes the following objective [5]:

$$J_{\text{ERM}}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(x_i, y_i; \theta). \tag{1.4.1}$$

1.4.3.2. **Invariant Risk Minimization (IRM)**. IRM is one approach that tries to keep feature representation invariant across different environments by reducing the representation discrepancy of multiple domains in a specific feature space [4, 69, 80]. Unlike most approaches that try to match the representation distribution of all domains, IRM enforces the optimal classifier on top of the representation space to be the same across all environments. The intuition behind this idea is the ideal representation for prediction is the only cause of the goal. It means the causal mechanism should not be affected by other factors [5, 80]. IRM can be formulated as follows:

$$\min_{\substack{g \in G, \\ f \in \bigcap_{i=1}^{M} \arg\min_{f' \in F} \epsilon^i(f' \circ g)}} \sum_{i=1}^{M} \epsilon^i(f \circ g) \tag{1.4.2}$$

for some function classes G of g and F of f. The constraint for f embodies the desideratum that all domains share the same representation-level classifier, and the objective function encourages f and g to achieve a low source domain risk. However, this problem is hard to solve as it involves an inner-level optimization problem in its constraint. The authors then develop a surrogate problem to learn the feature extractor g that is much more practical:

$$\min_{g \in G} \sum_{i=1}^{M} \epsilon^i(g) + \lambda \left\| \nabla_f \epsilon^i(f \circ g) \Big|_{f=1} \right\|^2, \tag{1.4.3}$$

where a dummy representation-level classifier $f = 1$ is considered, and the gradient norm term measures the optimality of this classifier. The work also presents a generalization theory under a strong linear assumption that the ground-truth invariant classifier can be identified for plenty of source domains [4, 5, 80].

Chapter 2

Chaotic Continual Learning

Will be submitted to ICML workshop of life-long learning 2023.

Contribution:

- I came up with the idea under the supervision of Prof. Irina Rish.
- Experiment setups and analysis were designed by me with valuable feedback from Mojtaba Faramarzi.
- I wrote all the code for algorithms and experiments reported in this thesis with valuable feedback from Mojtaba Faramarzi.
- I also wrote the entire paper with valuable help from Mojtaba Faramarzi.

Affiliation:

- Touraj Laleh: Mila - Quebec AI Institute, Université de Montréal.
- Mojtaba Faramarzi: Mila - Quebec AI Institute, Université de Montréal.
- Irina Rish: Mila - Quebec AI Institute, Université de Montréal, Canada CIFAR AI Chair.
- Sarath Chandar: Mila - Quebec AI Institute, École Polytechnique de Montréal, Canada CIFAR AI Chair.

2.1. Abstract

One of the challenges of current machine learning systems is that standard AI paradigms are not good at transferring (or leveraging) knowledge across tasks. While many systems have been trained and achieved high performance on a specific distribution of a task, it is not easy to train AI systems that can perform well on a diverse set of tasks that belong to different distributions. In addition, if an AI system is trained on a set of tasks belonging to different distributions, it could forget the knowledge it acquired from previous tasks. In continual learning, this process results in catastrophic forgetting, which is one of the core issues of this domain. This research focuses on the comparison of a chaotic learner and a naive continual learning setup. Training a deep neural network model usually requires multiple iterations, or epochs, over the training data set in order to estimate the parameters of the model better. Most proposed approaches for this issue try to compensate for the effects of parameter updates in the batch incremental setup in which the training model visits a lot of samples for several epochs. However, it is unrealistic to expect training data to always be fed to the model in a batch incremental setup. We propose a chaotic stream learner that mimics the chaotic behavior of biological neurons and does not update network parameters. In addition, it can work with fewer samples compared to deep learning models on stream learning setups. Interestingly, our experiments on different datasets show that the chaotic stream learner has less catastrophic forgetting by its nature in comparison to a CNN model in continual learning.

2.2. Introduction

Continual learning assumes that a learning agent is presented with a sequence of different ”tasks” (i.e., data coming from different probability distributions), where each task is a sequence of experiences from the same distribution [61]. The human brain can continuously learn different tasks without any of them negatively interfering with each other. However, learning a set of sequential tasks in the neural networks degrades the performance of the models. This is one of the biggest challenges in continual learning, which is known as catastrophic forgetting [11, 50] and affects all parametric AI systems. We want the knowledge to be transferred across tasks, while, in reality, the knowledge transfer is often uni-directional. It means the knowledge acquired from the previous tasks are often used to improve the performance on the current task (and only some of the tasks). Ideally, we would want the learning systems to be bi-directional. For example, training on the current task could improve performance on both previous and future tasks. This challenge comes from the fact that AI systems are designed in a way to be trained for every new task from scratch. As it is mentioned in [72], it is like a system that has to learn the alphabet every time it reads a book. This example shows unlike humans, who can incrementally acquire and update knowledge across new tasks, it is hard for AI systems to adapt to new tasks.

Continual Learning focuses on AI systems that keep accumulating new knowledge throughout their lifetime without forgetting the prior knowledge and use this accumulated knowledge to improve their performance on the different tasks.

Most approaches proposed to alleviate catastrophic forgetting are categorized into one of three main categories, including replay-based, regularization-based, and parameters isolation-based methods according to the task-specific information stored and used through sequential learning process [14].

Replay-based methods store exemplars in the replay buffer to alleviate the catastrophic forgetting while learning new tasks [10, 47, 59]. They apply gradient-based updates that facilitate a high-level transfer across different tasks through the examples from the past tasks that are simultaneously available while training on the current new task [72]. Incremental classifier and representation learning (ICARL) [59] is a replay-based method. During the training, ICARL selects a subset of examples per class that can best approximate the mean of each class in the feature space being learned. Then, each new example is classified by selecting the class whose exemplars are the most similar to it. Gradient Episodic Memory [47] uses episodic memory to store a subset of the observed examples from a given task and minimize catastrophic forgetting. GEM treats the losses on the episodic memories of tasks as inequality constraints, avoiding their increase but allowing their decrease. (A-GEM) [10] also project the gradients according to defined hard constraints

Since it is not always possible to keep exemplars, regularization-based methods propose extra regularization to consolidate previous knowledge when learning new tasks [33]. Some of these approaches regularize the loss function to minimize changes in the parameters important for previous tasks. These methods are called importance-based methods. In addition, there are distillation-based methods to preserve knowledge from previous tasks and transfer it to the model when it is being trained on the new tasks. Elastic weight consolidation (EWC) [33] was one of the first methods to establish this approach. It was inspired by the human brain in which synaptic consolidation enables continual learning by reducing the plasticity of synapses related to previously learned tasks [11]. Plasticity is the main cause of catastrophic forgetting since the weights learned in the previous tasks can be easily modified given a new task. More precisely, the plasticity of weights that are closely related to previous tasks is more prone to catastrophic forgetting than plasticity of weights that are loosely connected to previous tasks. This motivates [33] to quantify the importance of weights in terms of their impact on previous tasks' performance, and selectively decrease the plasticity of those important weights to previous tasks.

In the third approach, the capacity of the model is not restricted and the model architecture can be expanded [64, 81], copied [2] or frozen to alleviate catastrophic forgetting. These approaches are called architecture-based methods. Some of the solutions in this approach mask out the parameters or even neurons that are used for the previous tasks [19, 48]. There are also expansion-based methods that handle catastrophic forgetting by expanding the model capacity in order to adapt to new tasks [71].

Backpropagation is the main reason for catastrophic forgetting in continual learning, and most proposed approaches alleviate this issue by reducing the negative effects of backpropagation. In continual learning, the same way animals learn, training examples must be learned sequentially. A training sample can only appear once, and training samples are not identically distributed in each batch. In addition, the model could be evaluated at any time. Most above-mentioned approaches considered the input data stream as batch incremental learning. In batch incremental learning, a learner sequentially observes labeled training data, which is broken into different identically distributed batches. Each batch includes samples and labels from different classes, and the model is only permitted to learn from batches sequentially. However, the way we want to see continual learning as defined in [27] is stream learning. Stream learning is, in fact, a batch incremental in which each batch includes one sample, and the model may be evaluated anytime. Each sample could be visited only one time, and the agent needs to learn very fast. This approach is more close to few shot learning [3] that training samples can be streamed in stream learner approach and it does not require lots of samples to train the model.

Almost all above mentioned methods of continual learning use developed based on the batch incremental learning method which require the algorithm to go over the whole train data set samples for several epochs. This method is only feasible in some real-world continual learning scenarios, especially in the case of few-shot learning. In addition, most proposed approaches for catastrophic forgetting developed in a way to reduce the effects of backpropagation by retraining, regularization, or even changing the structure of the model. However, the main reason for catastrophic forgetting, parameter update during training, is still the same.

Chaos is defined as the phenomenon of complex, unpredictable , and random-like behavior arising from simple deterministic nonlinear systems. This is an interesting property of the brain to exhibit “Chaos” [39]. The neuronal system works in a way that small shifts in their internal functional parameters result to get the desired response to different influences. This attribute represents the dynamical properties of chaotic systems like the brain. Chaotic behavior is exhibited not only by brain networks which are composed of billions of neurons, but the dynamics at the neuronal level (cellular and subcellular) are also chaotic.

In [31] the first dynamical system's model for the interaction between the ion channels and axon membrane, which is capable of generating realistic action potentials, is proposed. And later, approaches such [20, 29] tried to simplify that approach. All these models exhibit chaotic behavior.

Inspired by the chaotic firing behavior of biological neurons, many approaches have been proposed to avoid backpropagation. ChaosNet [6] is one of those approaches that proposed a 1D chaotic dynamic using the Generalized Luröth Series (GLS) as a chaotic neuron [6]. It is a chaos-based neural network architecture using layers of neurons. The goal of ChaosNet is to learn classification tasks using limited training samples. This chaotic behavior of individual neurons could result in interesting properties of biological neural networks, which finally can even accomplish challenging classification tasks comparable to or better than conventional ANNs while requiring far less training However, ChaosNet can not compete with the deep neural network model in an image classification task since a deep learning model can visit training samples in several epochs. Adapting the GLS neuron in the continual few-shot learning and stream learning can be considered as an alternative approach since it suffers less catastrophic forgetting.

In this chapter, following the ChoasNet, we used a GLS neuron to simulate the chaotic behavior of a biological neuron to encode images with chaotic dynamics. We propose a GLS stream learner that uses a linear 1D ChaosNet neuron as a continual learner component. We also propose using some chaotic transformation as a data augmentation technique. Our results demonstrate that our chaotic learner has noticeable results in comparison to a CNN model in the batch incremental and stream learning setups for the image class incremental classification tasks.

2.3. GLS Stream Learner

Dendrites are exquisitely specialized cellular compartments that critically influence how neurons collect and process information. Retinal ganglion cell (RGC) dendrites receive synaptic inputs from bipolar and amacrine cells, thus allowing cell-to-cell communication and flow of visual information [1]. To be able to mimic this behavior, GLS Stream Learner

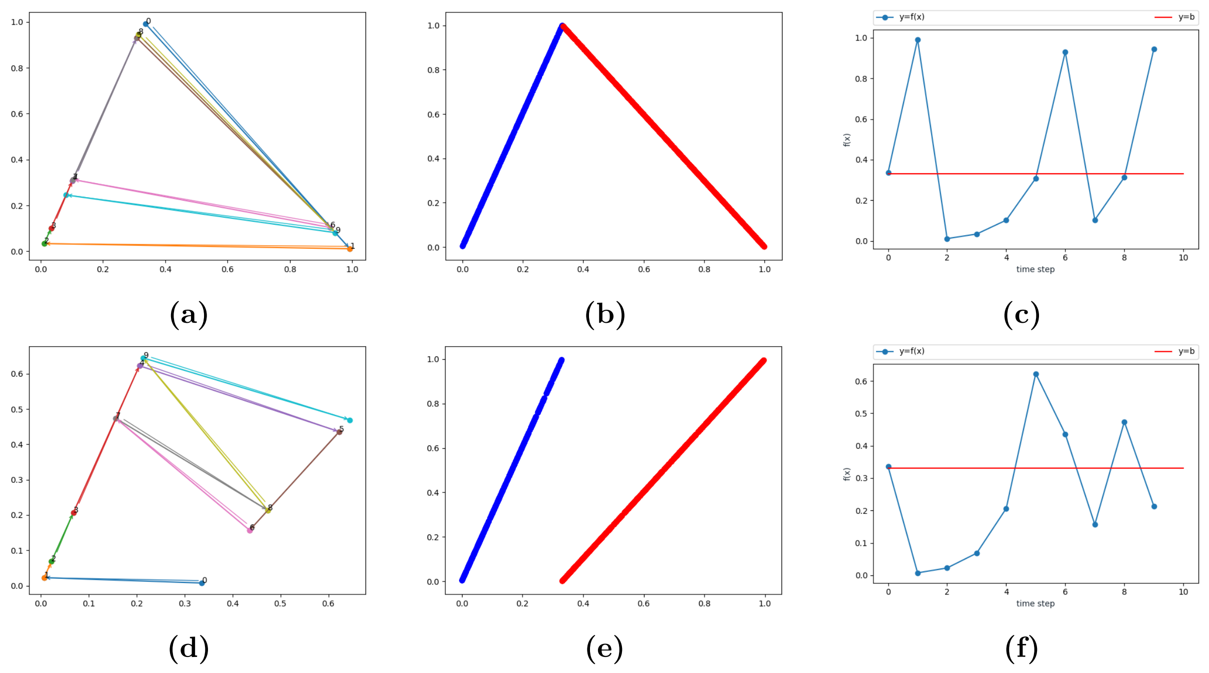


Fig. 2.1. The attractors of the Skew-Binary and Skew-Tent. (a) Skew-Tent sample for 10 steps. (b) Skew-Tent scatter plot for 1000 steps. (C) Skew-Tent sample for 10 steps . (d) Skew-Binary sample for 10 steps. (e) Skew-Binary scatter plot for 1000 steps. (f) Skew-Binary sample for 10 steps.

uses a linear 1D ChaosNet neuron as a continual learner component. Following the ChaosNet, GLS Stream Learner uses Generalized Luröth Series or GLS [6]. Generalized Luröth Series defines the time-series creation process as follows:

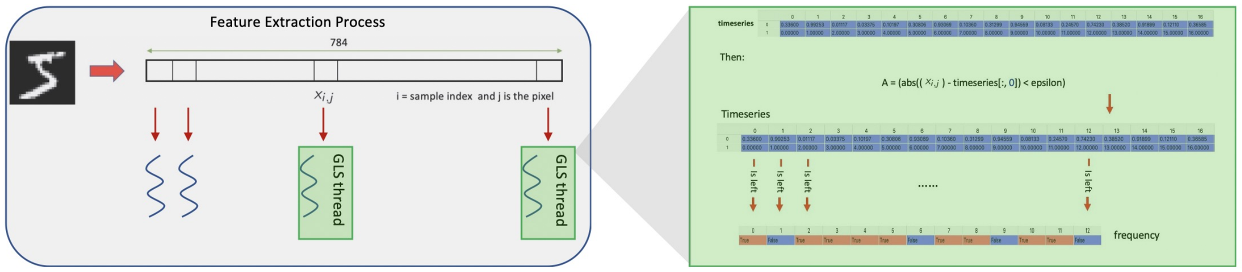
$$q \rightarrow f_1(q) \rightarrow f_2(q) \rightarrow f_3(q) \ldots \rightarrow f_i(q) \ldots \rightarrow f_N(q) \tag{2.3.1}$$

where q is the initialization of the dynamic. There are several chaotic ways to encrypt an image, and one of them is using skew-tent and skew-binary mapping [42]. Following are skew-tent and skew-binary mapping that are mathematically formulated as follows:

$$f_{Skew-Tent}(x) = \begin{cases} \frac{x}{b} & 0 \leq x < b \\ \frac{(1-x)}{(1-b)} & b \leq x < 1 \end{cases} \tag{2.3.2}$$

$$f_{Skew-\text{ Binary}}(x) = \begin{cases} \frac{x}{b} & 0 \leq x < b \\ \frac{(x-b)}{(1-b)} & b \leq x < 1 \end{cases} \tag{2.3.3}$$

Figure 2.1 illustrates the attractor of Generalized Luröth Series using the skew-tent and skew-binary mapping. In equations 2.3.2, b is a hyperparameter that has an important role in the attractor of the dynamic. Figure 2.1a and 2.1d show the ten steps movement in the


Feature Extraction Process
784
$x_{i,j}$
i = sample index and j is the pixel
GLS thread
GLS thread
Then:
A = (abs(($x_{i,j}$) - timeseries[:, 0]) < epsilon)
Is left
Is left
Is left
......
Is left
frequency

timeseries

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0.33600	0.99253	0.01117	0.03375	0.10197	0.30806	0.93069	0.10360	0.31299	0.94559	0.08133	0.24570	0.74230	0.38520	0.91899	0.12110	0.36585
1	0.00000	1.00000	2.00000	3.00000	4.00000	5.00000	6.00000	7.00000	8.00000	9.00000	10.00000	11.00000	12.00000	13.00000	14.00000	15.00000	16.00000

Timeseries

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0.33600	0.99253	0.01117	0.03375	0.10197	0.30806	0.93069	0.10360	0.31299	0.94559	0.08133	0.24570	0.74230	0.38520	0.91899	0.12110	0.36585
1	0.00000	1.00000	2.00000	3.00000	4.00000	5.00000	6.00000	7.00000	8.00000	9.00000	10.00000	11.00000	12.00000	13.00000	14.00000	15.00000	16.00000

0	1	2	3	4	5	6	7	8	9	10	11	12
True	False	True	True	True	True	False	True	True	False	True	True	False

Fig. 2.2. Feature extraction process that GLS stream learner apply to decode images information.

time-series using skew-tent and skew-binary. Repeating the process for several steps shows that the time-series numbers lie on the tent shape or two separated lines for skew-tent and skew-binary, respectively. Red and blue colors in scatter plots illustrated in Figure 2.1b and 2.1e can represent the active and passive status of a dendrite. It mimics either firing or not firing the response of a neuron corresponding to the input. Figure 2.1c and 2.1f are other points of view of showing the same behavior through time. All points above the red line, which is defined based on "b" hyperparameter, can be considered as true (firing) and the bellow points as false (not firing).

2.3.1. Feature extraction

GLS learner uses normalized images. So, the pixels are scalar numbers in $[0, 1]$. The feature extractor process runs several threads. And, each thread encodes the pixel j from x_i to a probability of firing rate count denoted as a $P_{i,j}$. The feature extraction process creates the GLS time-series with either skew-tent or skew-binary mapping. Then, the GLS thread tries to find the first ϵ-neighborhood point in the time-series to the pixel information. Then, it computes the firing rate count for each pixel as follows:

$$P_{i,j} = \frac{\text{False count}}{\text{length of frequency list}}, \tag{2.3.4}$$

The result is the decoded information vector P_i for x_i. Since we encode each pixel to a probability number, the size of P will be the same as the image size. Figure 2.2 shows the feature extraction process in detail.

2.3.2. Training

At the time T, a set of samples of new classes arrive in either batch incremental fashion or as a stream of data. In this case, we have $D_T \subseteq D_{train}$ where T is the task time for N

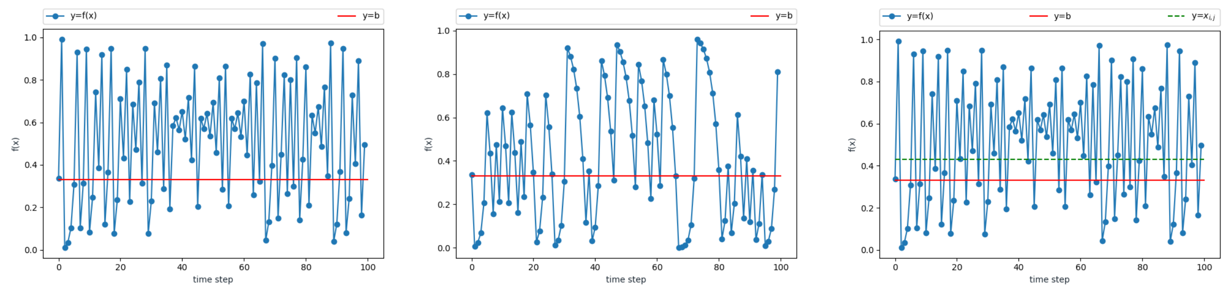


Fig. 2.3. (left) Skew-Tent sample for 100 steps. (middle) Skew-Binary sample for 100 steps. (right) feature extraction summary for a pixel information using skew-tent mapping (we showed only first 100 time steps).

samples from either i.i.d or non i.i.d observations of $(x_i,\, c_i)$ pairs where $c_i \in C$. And, C is the set of classes that the stream learner should learn at time T. For, each sample, x_i, the stream learner creates the vector P_i that is the extracted feature from x_i. Then, the stream learner computes the mean representation, $M^{SL}_{c_i}$, for m samples of class c_i that have seen at time T as follow:

$$M^{SL}_{c_i} = \frac{1}{m}\Big[\sum_{i=1}^{m} P^{c_i}_{i1}, \sum_{i=1}^{m} P^{c_i}_{i2}, \dots, \sum_{i=1}^{m} P^{c_i}_{in}\Big], \tag{2.3.5}$$

The batch incremental learner visits samples in several epochs. Therefore, GLS learner should compute a new vector, $M^{b_j}_{c_i}$, for the epoch j and appends it to the batch incremental representatives as follow:

$$M^{BSL}_{c_i} = \Big[M^{b_1}_{c_i}, M^{b_2}_{c_i}, \dots, M^{b_n}_{c_i}\Big], \tag{2.3.6}$$

where n is the number of epochs and $M^{b_j}_{c_i}$ is computed for the epoch j using 2.3.5.
Replay Buffer: GLS stream learner keeps either $M^{BSL}_{c_i}$ or $M^{SL}_{c_i}$ for the batch incremental or stream learning setup, respectively, in the replay buffer instead of keeping exemplars in the buffer.

2.3.3. Classification

Let assume $\varphi(x)$ extracts features for a sample $x \in X_{test}$ as described in 2.3.1. Then, GLS stream learner predicts the target as follows:

$$y^* = \underset{y=1,\dots,t}{\operatorname{argmin}} \left\|\varphi(x) - M_c\right\|, \tag{2.3.7}$$

Where $c \in C$ (all classes have learned so far) and $\|\varphi(x) - M_c\|$ is the cosine similarity distance of two $\varphi(x)$ and M_c vectors that can be either either $M^{BSL}_{c_i}$ or $M^{SL}_{c_i}$ for the batch incremental or stream learning, respectively.

2.3.4. Chaotic Data Augmentation

Augmentation techniques have been used to improve the deep learning model on unseen data at test time. Data augmentation techniques are classified as data-dependent regularization techniques. Similar to the deep learning model, chaotic based augmentation techniques can also improve the performance of the continual learner through time. In this work, we proposed using "baker's" transformation as a chaotic data augmentation that we can use to improve the GLS stream learner performance. The following formula is the mathematical definition of the "baker's" transformation dynamic mapping [17].

$$f_{baker's}(x_i, x_j) = \begin{cases} (2x_i, \lambda x_j) & \left(0 \leqslant x_i \leqslant \frac{1}{2}\right) \\ \left(2x_i - 1, \lambda x_j + \frac{1}{2}\right) & \left(\frac{1}{2} < x_i \leqslant 1\right) \end{cases} \tag{2.3.8}$$

where i and j are the row and column of each pixel in an image. It is called "baker's" transformation because it resembles the process of repeatedly stretching a piece of dough and folding it in two. To this end, it can work as stretching and folding data augmentation. Similarly, the dynamics of the hénon map may be decomposed into an area-preserving bend, followed by a contraction, followed by a reflection in the line $y = x$. The Hénon map $f : \mathbb{R}^2 \to \mathbb{R}^2$ as follow:

$$f_{Hénon}(x_i, x_j) = \left(x_j + 1 - ax_i^2, bx_i\right) \tag{2.3.9}$$

2.4. Experiments

There are three different data paradigms of stream learning for evaluating continual learning models based on the way the training data is organized [3, 27]. The model visits a limited number of samples in only one epoch in either one of the following setups in the stream learning context. In the first setup, the data stream is completely unordered. In the second setup, the data stream is ordered by the class and the models learns classes incrementally which results in catastrophic forgetting. In the third setup, data is organized on batches from specific instances of categories that can be revisited [3]. Diversely, the model incrementally learns new classes and it is allowed to revisits samples for several epochs while training a new task in the batch incremental setup. To evaluate our approach we follow the second paradigm that is more aligned with real world scenarios. Figure 2.4 shows the comparison of batch incremental learning and the stream learning setups [38].

For the experiment, we just compare the performance of the DLL model with stream learner in classs incremental setup. It is important to mention, here the goal is to evaluate

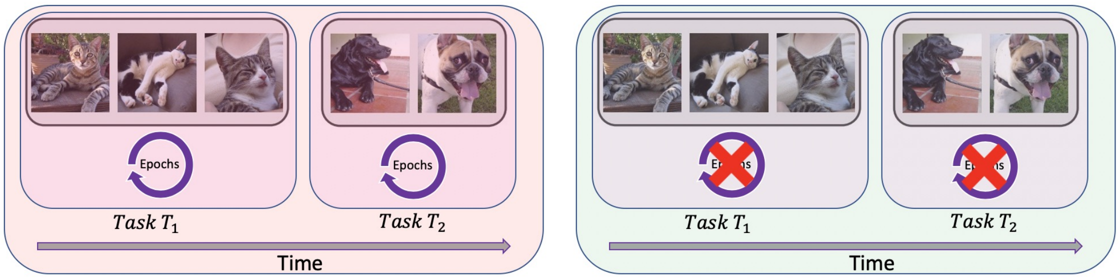


Fig. 2.4. Batch Incremental (left) v.s. Stream Learning (right) [38].

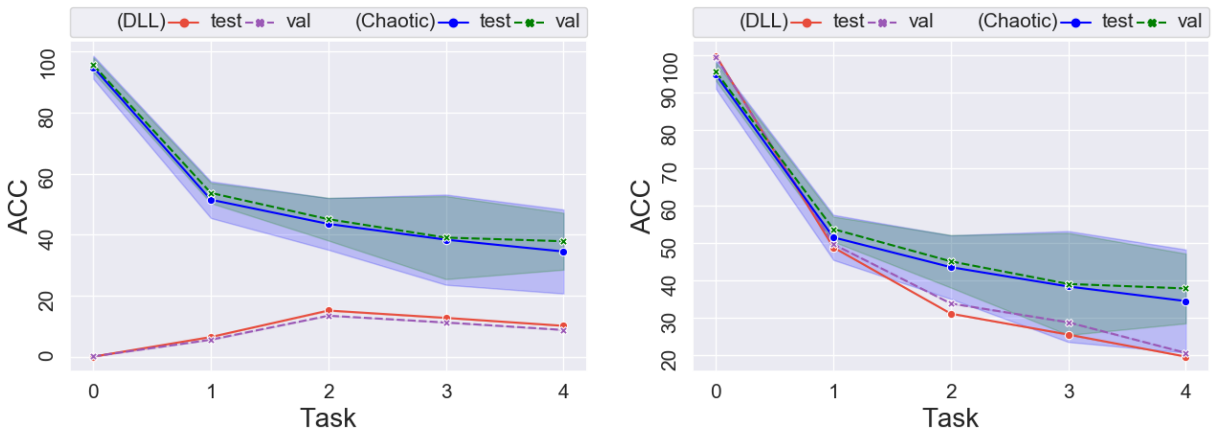


Fig. 2.5. Permuted-MNIST divided into 5 tasks with 2 classes per task.

the performance of GLS stream learner compared to a deep neural network which is working based on back propagation. As a result, we did not compare the stream learner with continual learning baseline. The CNN model that used in the experiments has the following architecture. It has 4 convolutional and 4 fully connected layers. In addition, the convolutional layers have 3, 10, 20 and 40 inputs and 10, 20, 40 and 64 output channels with 5, 5, 3, and 5 kernel size with stride of 1, respectively. The feature extractor part is followed by two fully connected layers that contain 680 and 280 neurons followed by a softmax module.

In our experiment, we observed that Skew-binary mapping achieves 3% more accuracy performance at each task in comparison to the Skew-tent. Therefore, all reported results for GLS stream learner are based on Skew-binary mapping with $\{b = 0.331, \epsilon = 0.01, q = 0.336, \text{time step} = 20000\}$ for MNIST, Permuted-MNIST and Omniglot datasets And $\{b = 0.331, \epsilon = 0.008, q = 0.136, \text{time step} = 30000\}$ for CIFAR-10.

MNIST and Permuted-MNIST:. We split the MNIST and Permuted-MNIST samples into 5 tasks with two classes per task. In the stream setup, we train models with 3 batches

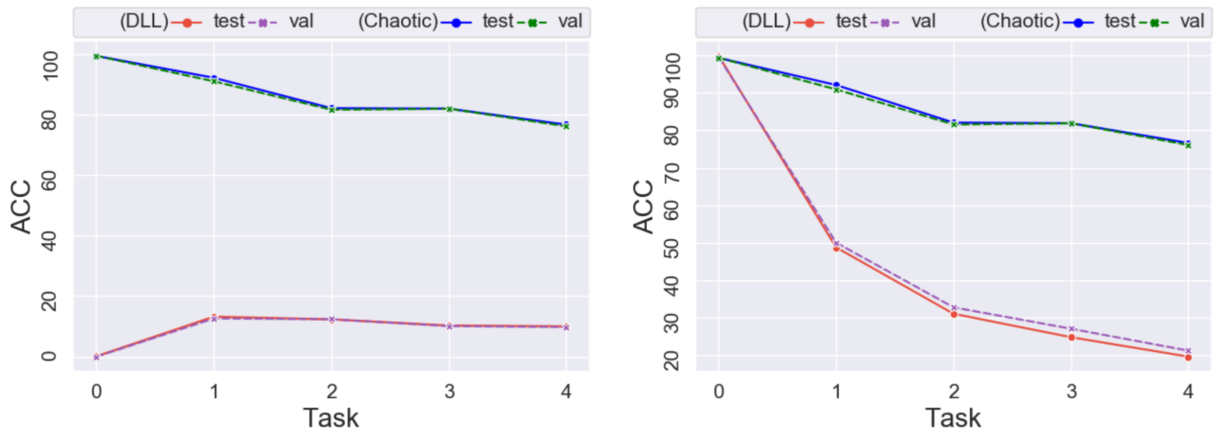


Fig. 2.6. MNIST divided into 5 tasks with 2 classes per task.

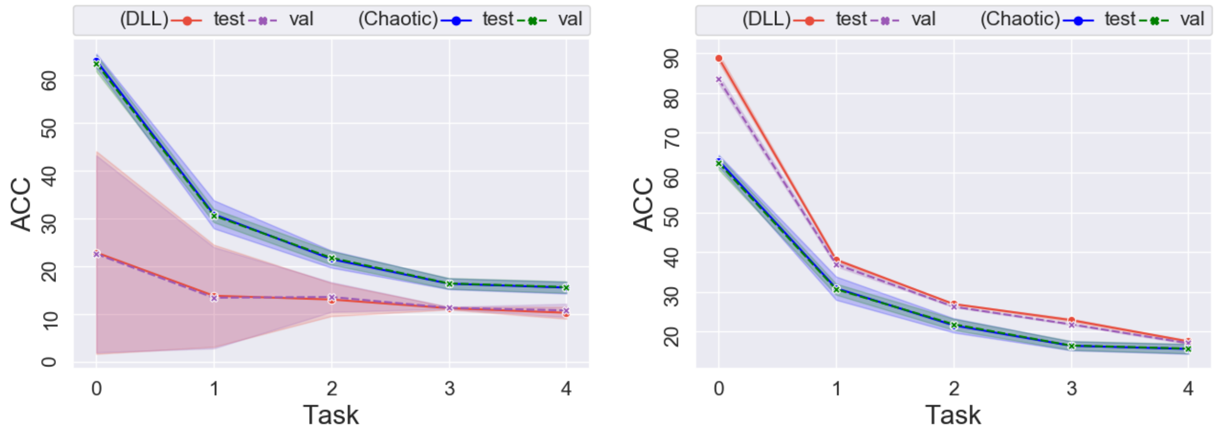


Fig. 2.7. CIFAR-10 divided into 5 tasks with 2 classes per task.

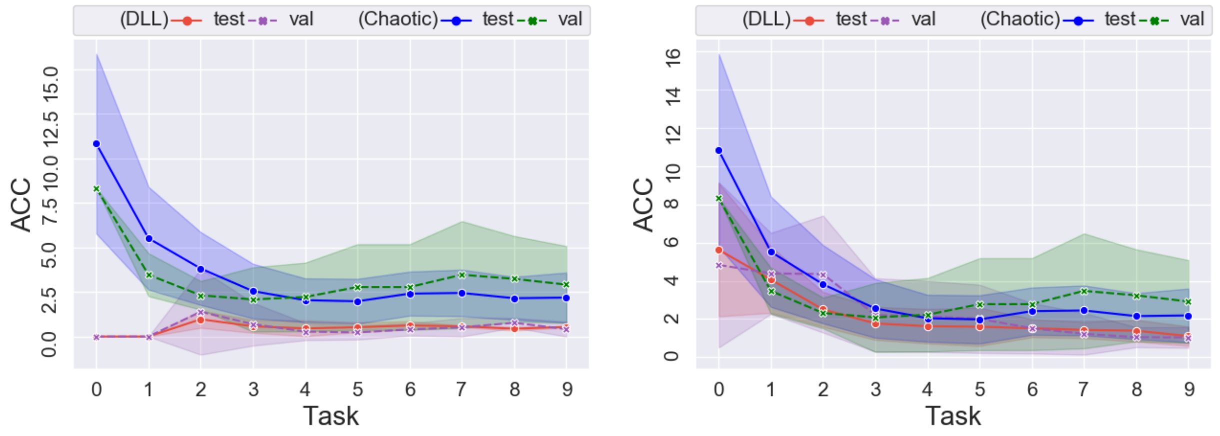
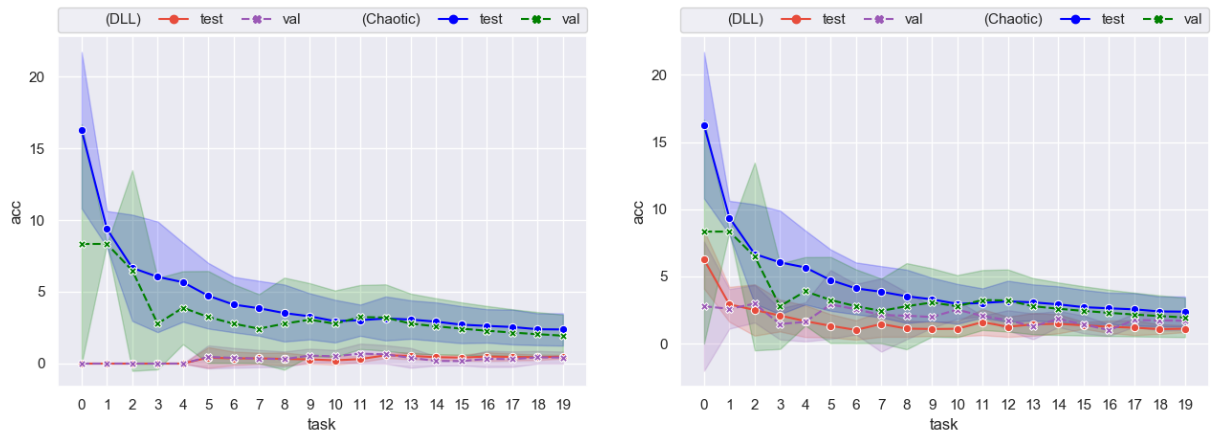


Fig. 2.8. Omniglot divided into 10 tasks with 20 classes per task.



.

Fig. 2.9. Omniglot divided into 20 tasks with 10 classes per task.

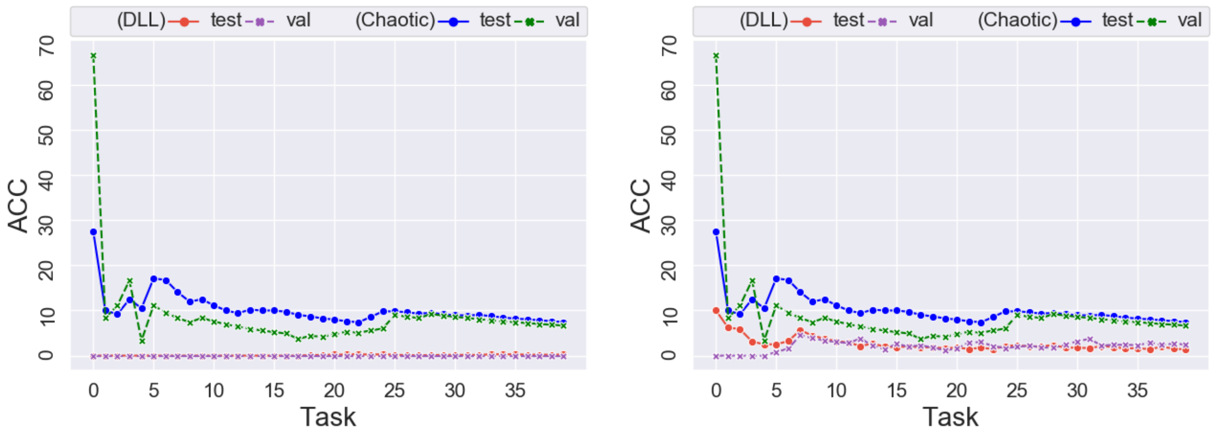


Fig. 2.10. Omniglot divided into 40 tasks and 5 classes per task.

and 32 samples per batch. Figures 2.6 and 2.5 illustrate the comparison results for MNIST and Permuted-MNIST. They show chaotic learner has significant marginal performance in comparison to the CNN model in both setups.

CIFAR-10: Figures 2.7 illustrate the comparison results on CIFAR-10. CIFAR-10 training data is split into the five tasks with two classes per task. For stream learner setup, we train models with four batches and 64 samples per batch. Unlike MNIST, the CNN model has better performance (figures 2.7 right) in batch incremental learning setup on CIFAR-10. However, the chaotic learner still leads to better performance in the stream setup approach.

Omniglot: To evaluate the performance on the Omniglot dataset [37], we designed two experimental setups. From 964 classes in the background TRUE set of the Omniglot dataset,

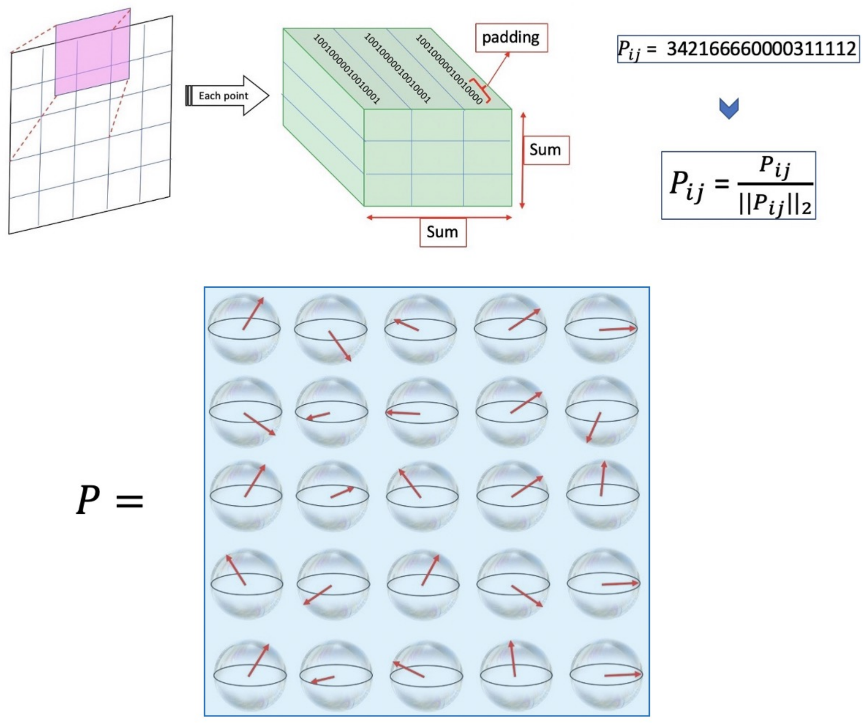


Fig. 2.11. The process of computing a higher-level abstraction for each pixel using normalized correlated extracted features.

we only chose 200 classes for this experiment. We split the selected data into 10, 20, and 40 tasks such that each task includes 20, 10, and 5 classes, respectively. We have 20 samples per class in this setup. We selected 60 percent of the samples for training. Therefore, we have a few samples for training in the batch incremental and stream learning setups (12 samples per class in the training set). Figure 2.8 represents the result of the Omniglot dataset that is divided into ten tasks. The figure contains further comparison results on Omniglot for the 20 and 40 tasks. The chaotic stream learner shows better results compared to the CNN model in both batch incremental and stream learner setups on Omniglot.

2.5. Future Work

To preserve the correlation between the extracted features, we can use a moving feature extractor block on the sample x_i, and calculate the correlated features for each pixel that lies in the moving block. Figure 2.11 briefly shows the process. First, we extract features associated with each pixel using the GLS feature extractor described in 2.3.1 for all pixels in the block. Adding a zero padding to the extracted firing rates helps to have the same size firing rate series. Applying an element-wise sum and then normalize the vector of the firing rates gives a higher-level abstraction of the correlated feature extracted for the pixel, P_{ij}, that is defined as follow:

$$P_{ij} = \frac{P_{ij}}{\|P_{ij}\|_2}, \tag{2.5.1}$$

All extracted P_{ij} have the same size but different angles. The angles can represent the higher-level abstraction of the correlated feature associated with each pixel and its neighbors. Our next step in this direction is experimenting with the effectiveness of using higher-level correlated abstraction instead of considering a standalone extracted futures using the GLS feature extractor in the stream learning setup.

2.6. Conclusion

In this work, GLS stream learner is proposed as a novel approach to alleviate catastrophic forgetting in the context of continual few-shot learning. This approach provides a mechanism based on the chaotic structure of a biological neuron that provides a different perspective from the most continual learning approaches. According to our experiment, this single chaotic neuron causes less forgetting in comparison to a deep learning model that needs a lot of time to train and more parameters to learn in batch incremental and stream learning setups. The deep learning model achieved profound performance in the image classification task. However, it suffers from catastrophic forgetting problems because of the backpropagation mechanism to update their parameters in the continual learning context. GLS stream learner can show the importance of thinking to find an alternative solution that is more suitable for stream and continual few-shot learning setups.

Chapter 3

Environment and Task difficulty in Out-of-distribution Generalization

3.0.1. Abstract

Deep Learning models have achieved exceptional performance in many machine learning tasks. However, they have a naive Out-of-Distribution (OoD) generalization performance where the testing distribution is unknown and different from the training. In the last years, there have been many research projects to compare OoD algorithms, including average and score-based methods. However, in most proposed methods for comparing and ranking OoD algorithms, all tasks considered have the same level of difficulty. This chapter analysis some logical and practical strengths and drawbacks of existing methods for comparing and ranking OoD algorithms. We propose a novel ranking approach to define the task difficulty ratios to compare OoD generalization algorithms. We compared the average, score-based, and difficulty-based rankings of four selected tasks from the WILDS benchmark and five popular OoD algorithms for the experiment. The analysis shows significant changes in the ranking orders compared with current ranking approaches.

3.1. Introduction

Many Machine learning systems that have been studied in the last ten years assume data in training and testing sets follow the same distributions. However, in the real world, this assumption does not hold. Out-of-Distribution (OoD) generalization focuses on the problems rising where testing distribution is different from training distribution, called distribution shift. This is one of the main challenges in the current machine learning systems to train deep learning models that can generalize to unknown shifted distributions at the test time [69]. This ability of models to generalize under distribution shifts is critical in real-world cases such as healthcare and autonomous cars, in which it is impossible to have robust models for all different types of distribution shifts at run time. Many datasets such as Waterbirds [65], CelebA [46], Camelyon17 [57], and FMoW[12] have been proposed to represent distributions shift from different aspects in realistic scenarios [80]. Each dataset is defined as a task, and each distribution shift is expressed as an environment in these benchmark setups. OoD generalization algorithms aim to leverage knowledge from previously trained environments to adapt and perform well in upcoming environments in the new setting [5]. Classic supervised learning methods cannot address the OoD generalization problem as their most fundamental assumption is that training and testing sets are identically and independently distributed. These approaches are not typically practical for OoD generalization since they mostly try to minimize errors in training data. In this situation, if there is a strong distributional shift in testing data, models optimized only with training data errors can even have worse performance than random guesses. [5, 13, 16].

Recently, the research community has spent significant effort developing many algorithms for OoD setup, including some well-known algorithms such as IRM [5], Group DRO [65], Deep CORAL [73] and VREx [36]. However, despite the enormous importance of OoD generalization, the literature in this domain is scattered. It means algorithms are evaluated in different setups using different datasets. As a result, it is not easy to have a clear comparison among proposed algorithms in this domain.

DomainBed [26] and WILDS [34], are two proposed frameworks for having fair and reproducible experimentation in OoD generalization setup. Both of these benchmarks include datasets from different domains, such as Natural language, speech processing, medicine, and healthcare which reflect a diverse range of distribution shifts in real-world applications [34]. Domainbed is a PyTorch testbed for domain generalization. The initial implementation includes nine algorithms, seven datasets, and three model selection methods, as well as the infrastructure to run all the experiments and generate the results. The main datasets in Domainbed are Colored MNIST [5], Rotated MNIST [22], PACS [18], VLCS [44], Terra Incognita [8] and DomainNet [54]. All these datasets are multi-domain image classification tasks. For example, Colored MNIST [5] contains 70,000 images and it is a variant version of the famous MNIST [15] dataset. In addition, Domainbed includes the implementation of around 20 different algorithms including Empirical Risk Minimization (ERM [78]), Group Distributionally Robust Optimization [65], Meta-Learning for Domain Generalization [43], Invariant Risk Minimization (IRM [5]). WILDS [34] is a curated benchmark that covers two common types of distribution shifts: domain generalization and subpopulation shift. In domain generalization, the training and test distributions comprise data from related but distinct domains arising from natural distribution shifts from different cameras, hospitals, and time periods. In subpopulation shift, test distributions are subpopulations of the training distribution and seek to perform well even in the worst-case subpopulation.

Domainbed compared the performance of different domain generalization algorithms on the task of out-of-distribution generalization. As a result, they run all algorithms on all datasets, and their results justify Emperial Rish Minimization (ERM) outperforms all previously published results. The fact domainbed tries to prove is When all conditions are equal, no algorithm outperforms ERM by a significant margin. Their finding is obtained by running every combination of the dataset, algorithm, and model selection criteria from scratch. To evaluate the performance of different algorithms, Domainbed proposed a ranking method to consider the average performance of each algorithm over all tasks as a criterion to compare different algorithms. Using this criterion, it reported that when equipped with modern neural network architectures and data augmentation techniques, Empirical Risk Minimization (ERM) achieves state-of-the-art performance in domain generalization [26].

One algorithm might marginally outperform only one task and perform poorly or leniently on the other. This situation could dominate the average result. In addition, different tasks could implement different types of distribution shifts. As a result, they could even have a different level of difficulty. Hence, ranking algorithms by only taking the average of their performance across all tasks is not fair and could be misleading.

To address the concept of difficulty, some approaches tried to understand the importance of example difficulty in Deep Neural network [7, 45]. They focus on the importance of data points with different amounts and types of example difficulty to understand deep learning models in a better way [75]. For example, difficulty from the perspective that the earlier training iteration (and all its subsequent iterations) the example will be predicted correctly, the easier it would be. Another approach is prediction depth which represents the number of hidden layers after which the network's final prediction is determined [7]. Prediction depth is consistent between architecture and ransom seeds. However, it only addresses the difficulty from the Examples' point of view. Sample or example difficulty is not the only important factor affecting a task's difficulty in the context of OoD. Environment difficulty results from distribution shifts among different environments of a single task.

The main contribution of this work is introducing a mechanism to quantify task and environment difficulty. We introduce the task difficulty ratio to understand how difficult a task is compared to other tasks and introduce our new ranking mechanism for OoD algorithms. Using that, we evaluate the performance of different algorithms for the task of out-of-distribution generalization.

3.2. Current state in evaluation of out-of-distribution generalization algorithms

Domainbed claims most of the learning algorithms for OoD generalization are almost at the same level of performance as the classic Empirical Risk Minimization (ERM). Domainbed considers the average performance of each algorithm over all tasks as a criterion to compare different algorithms across multiple tasks [26]. However, calculating the best performance across all datasets is unfair and has some drawbacks. One algorithm might marginally outperform only one task and perform poorly or leniently on the other tasks. This situation could dominate the average result. In addition, each task could have various distribution shifts with different intensities. Such a situation leads to different levels of difficulty among tasks. Hence, comparing and ranking algorithms by only taking the average of their performance across all tasks is not fair.

To address this issue, some approaches such as [82] tried to identify and measure different distribution shifts among a dataset. OoD-bench proposes a new comparison and ranking method for OoD algorithms based on the type of distribution shifts ubiquitous in various tasks. They grouped tasks due to the influences of two distinct kinds of distribution shifts, namely diversity shift and correlation shift [82]. In this way, each dataset has a certain level of diversity and correlation shift. For example, datasets such as VLCS, PACS, office-Home and DomainNet have multiple domains. Each of these domains represents a certain spectrum of diversity and training, and a test set of each dataset contains a specific group of them which results in a diversity shift during the experiments. However, Colored MNIST focus on another type of domain generalization problem caused by spurious correlations. In ColorMNIST [5], colored digits are arranged into training and test environments such that the labels and colors are strongly correlated, but the correlation relationship flips across the environments. This type of distribution shift is different from the type of distribution shift we have in LCS, PACS. To specify the level of each distribution in a dataset, a simple neural network is trained to learn a feature extractor that could compute diversity and correlation shift. OoD-bench demonstrated most of the existing OoD benchmarks dominated by one of two discussed kinds of shifts. For example, datasets such as PACS and Camelyon mostly reside on the y-axis, which shows they are more dominated by the diversity shift. However, Colored MNIST reside more on the x-axis, which is dominated by correlation shift. In addition to these specifically designed datasets, In addition, datasets such as ImageNet-A, ImageNet-R, and ImageNet-V2, which are under unknown distribution shifts, are in the middle of two different discussed distribution shifts [82].

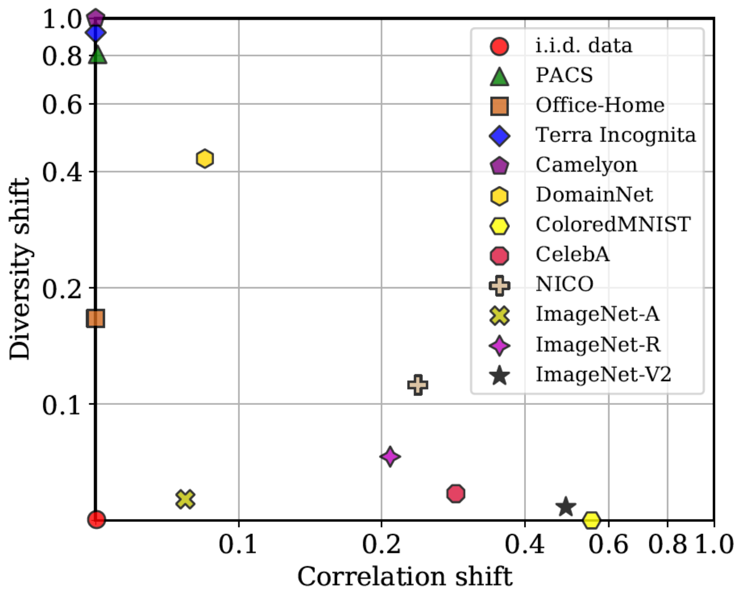


Fig. 3.1. Estimation of diversity and correlation shift in various datasets [82].

OoD-bench assigns scores -1, 0, or +1 depending on whether the attained accuracy is lower than, within, or higher than the standard error bar of ERM accuracy on the same dataset [82]. To rank algorithms, they added up all scores and sorted algorithms' based on accumulated scores. Although the OoD-bench ranking approach seems more reasonable, it does not provide a fair comparison. First, it assumes all datasets (tasks) in one category have the same difficulty level. Second, it gives a score of either +1 or -1 for any algorithm that beats or loses against ERM. As a result, it ignores how much an algorithm either outperforms or underperforms ERM on a specific task since it only considers one score point up or down.

To have a proper ranking method, finding a metric for the difficulty of tasks in OoD generalization setup is essential. However, Measuring the level of difficulty of tasks is a challenging topic. In OoD setup, multiple environments at training time have some distribution shifts from each other. We propose an approach to finding the difficulty ratio of a task based on comparing training environments. The most important factor that confuses the learner in OoD setup is extracted spurious correlated features and imposed gradients at the training time. This problem is known as gradient starvation [56]. From one environment to another environment, there are changes in sample distribution or in target distribution denoted as $p(x)$, $p(y|x)$, or $p(y)$. However, the classes that the learner should learn are the same. Assuming an almost mature feature extractor is frozen and followed by trainable classifier layers. We use each environment's data to train the classifiers in an isolated system. Then, we compare the models' performance with an unseen OoD validation set. It can show how much spurious correlated features dominate the trained models since the feature extractor is the same and frozen.

A set of environments with more Scattered OoD validation performances can construct more difficult tasks to train. This situation could happen since the deep learning model can stick with simple samples encapsulated in one environment, find a shortcut based on these samples, and ignore others. Staying with the shortcut leads to a poor generalization against a slight distribution shift in samples at testing time. Repeating this process at different prediction depths can give a better analysis of the environments' difficulty and task difficulty. Figure 3.2 illustrates this process for a task that contains four environments. The gray block shows the frozen feature extractor that has consecutive blocks. At the end of each block, a pink rectangular shape represents a trainable classifier model which will be trained in an isolated system. Then, we collect the model accuracy on an OoD validation set, an unseen shifted environment reserved for validating the models. Assume set E represents training environments for a task T as $E = \{E_1, E_2, ..., E_n\}$. For a feature extractor with k consecutive blocks, set A contains the models' accuracy corresponding to each environment. The task difficulty ratio for task T is denoted by D_T and computed as follows:

$$D_T = \frac{1}{B \cdot C_2^{|E|}} \sum_{k \in B} \frac{\sum_{i \in E} \sum_{j \in E} |A_{i,k} - A_{j,k}|}{Max(A_k) - Min(A_k)}, \tag{3.2.1}$$

where $A_{i,k}$ is the accuracy of classifier for environment E_i at the layer k. B represents the number of consecutive blocks that are considered prediction depths. $\sum_{i \in E} \sum_{j \in E} |A_{i,k} - A_{j,k}|$ is a possible pairwise accuracy differences at layer k. $Max(A_k)$ and $Max(A_k)$ also represent the maximum and minimum reported accuracies at layer k. It should be normalized by dividing it by the distance between the maximum and minimum accuracies at layer k. Dividing the results by the number of possible pairwise environments' differences can provide a fair comparison for tasks with more or fewer numbers of environments. And, the combination of the cardinality of the set E taken two at a time without repetition is denoted by $C_2^{|E|}$ in the Eq. 3.2.1. Then, the weighted average of a selected algorithm for all tasks is used for ranking where the weight of a task is the task difficulty.

3.3. Experiments

For experiments, we picked four popular image-based tasks including CelebA [46], Waterbirds [65], FMoW [12] and Camelyon17 [57] from WILDS [34]. WILDS is a benchmark

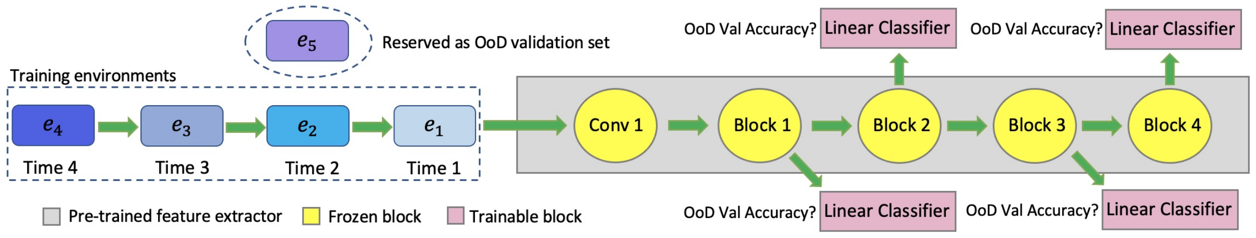


Fig. 3.2. Task difficulty ratio calculation process using a pre-trained feature extractor that is trained on the ImageNet dataset. This is a process for a task with four training environments and one reserved environment as an OoD validation set.

that includes data from different domains reflecting a diverse range of distribution shifts in real-world applications [34].

3.3.1. Datasets

We used WILDS [34], which is a benchmark of in-the-wild distribution shifts spanning diverse data modalities and applications. The benchmark includes data from different domains, including Medicine and healthcare, Genomics, Natural language, and speech processing, reflecting a diverse range of distribution shifts in real-world applications [34]. For our experiments, we used 4 datasets including CelebA [46], waterbirds [65], FMoW [12] and camelyon17 [57] that will be discussed in details.

3.3.1.1. Waterbirds. This dataset combines bird photographs from the Caltech-UCSD Birds-200-2011 (CUB) dataset [26]. The dataset includes two categories of birds, waterbirds and landbirds. Waterbirds are images of seabirds or waterfowls which are cropped from their original background and added onto a water background. The same situation for landbird which are cropped and added to the land. Table 3.1 shows the dataset has four different environments for training and validation.

Table 3.1. Train and validation environments in Waterbirds

Train Data	OoD Val Data
y = landbird, background = land: $n = 3498$	y = landbird, background = land: $n = 467$
y = landbird, background = water: $n = 184$	y = landbird, background = water: $n = 466$
y = waterbird, background = land: $n = 56$	y = waterbird, background = land: $n = 133$
y = waterbird, background = water: $n = 1057$	y = waterbird, background = water: $n = 133$

3.3.1.2. Camelyon17. Camelyon17 dataset [57] includes a 96x96 patch of a whole-slide image of a lymph node section from a patient with potentially metastatic breast cancer. The data is collected from five hospitals each of which represents a domain. The goal of this dataset is to find out whether the patch contains a tumor or not. As shown in Table 3.2, the dataset has five environments. There are three environments in training data and one environment in OoD validation set.

Table 3.2. Train and validation environments in Camelyon17

Train Data	OoD Val Data
hospital $= 0 : n = 53425$	hospital $= 0 : n = 0$
hospital $= 1 : n = 0$	hospital $= 1 : n = 34904$
hospital $= 2 : n = 0$	hospital $= 2 : n = 0$
hospital $= 3 : n = 116959$	hospital $= 3 : n = 0$
hospital $= 4 : n = 132052$	hospital $= 4 : n = 0$

3.3.1.3. CelebA. CelebFaces Attributes Dataset (CelebA) [46, 66] is a large-scale face attributes dataset. The dataset includes 200,000 images with large diversities and rich attribute annotations. The images in this dataset cover large pose variations and background clutter. Table 3.3 shows CelebA has four different environments. Classifying the hair color of celebrities as "blond" or "not blond" is considered a goal while the target is spuriously correlated with gender [45].

3.3.1.4. FMoW. This dataset includes satellite images over the years 2002t to 2018 [12].It can be used in different domains of global-scale monitoring such as sustainability, population density study, and forest monitoring 3.3. FMoW includes 1 million images from over 200 countries which are coming with rich metadata provided for each image. The metadata

Table 3.3. Train and Validation Set environments of CelebA

Train Data	OoD Val Data
$y =$ not blond, male $= 0 : n = 71629$	$y =$ not blond, male $= 0 : n = 8535$
$y =$ not blond, male $= 1 : n = 66874$	$y =$ not blond, male $= 1 : n = 8276$
$y =$ blond, male $= 0 : n = 22880$	$y =$ blond, male $= 0 : n = 2874$
$y =$ blond, male $= 1 : n = 1387$	$y =$ blond, male $= 1 : n = 182$

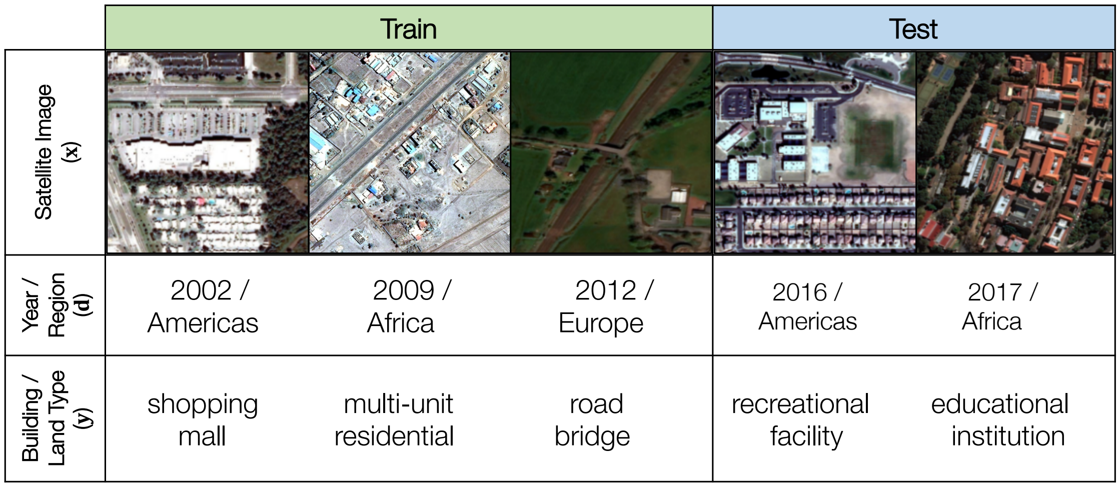


Fig. 3.3. FMoW contains satellite images of different geographical regions and at different times [34]

enables reasoning about location, time, sun angles, physical sizes, and other features when making predictions about objects in the image as described in WILDS [34]. Table 3.4 shows there are 16 environments in total in FMoW including 11 environments for the training set and 3 environments for OoD validation set.

3.3.2. Experiment setups and results

Table 3.5 shows the hyper-parameters and value metrics that are used for training classifiers for calculating the tasks' difficulty ratios. It worth mentioning that SGD is used as an optimizer with 0.9 Nesterov momentum and Adam with default hyper-parameters for the FMoW task.

In addition, we re-rank the performance of five important OoD generalization algorithms, including IRM [5], Group DRO [65], Deep CORAL [73], VREx [36], and Empirical Risk Minimization (ERM). We chose a pre-trained ResNet-50 model with the IamegNet dataset as a suitable feature extractor to find the difficulty ratio. Such a feature extractor can extract adequate essential features from samples since the model is trained on the dataset with complex images with 1K classes and one million examples [62]. As explained in section 3.2, we trained four classifiers for each environment on top of frozen feature extractor blocks in the ResNet-50 model. For each dataset, we repeat the experiment for five different seeds with a learning rate of 0.0001. For Waterbirds and CelebA, we trained classifiers for 120 epochs. Moreover, for FMoW and Camelyon17, we trained the model for 64 and 30 epochs, respectively. We also used SGD with 0.9 Nesterov momentum as an optimizer for Waterbirds,

Table 3.4. Train and Validation Set environments of FMoW

Train Data	OoD Val Data
year = 2002 : $n = 1455$	year = 2002 : $n = 0$
year = 2003 : $n = 1985$	year = 2003 : $n = 0$
year = 2004 : $n = 1545$	year = 2004 : $n = 0$
year = 2005 : $n = 2207$	year = 2005 : $n = 0$
year = 2006 : $n = 2765$	year = 2006 : $n = 0$
year = 2007 : $n = 1338$	year = 2007 : $n = 0$
year = 2008 : $n = 1975$	year = 2008 : $n = 0$
year = 2009 : $n = 6454$	year = 2009 : $n = 0$
year = 2010 : $n = 16498$	year = 2010 : $n = 0$
year = 2011 : $n = 19237$	year = 2011 : $n = 0$
year = 2012 : $n = 21404$	year = 2012 : $n = 0$
year = 2013 : $n = 0$	year = 2013 : $n = 3850$
year = 2014 : $n = 0$	year = 2014 : $n = 6192$
year = 2015 : $n = 0$	year = 2015 : $n = 9873$
year = 2016 : $n = 0$	year = 2016 : $n = 0$
year = 2017 : $n = 0$	year = 2017 : $n = 0$

Table 3.5. Hyper-parameters and value metrics that are used on each task in tasks' difficulty ration calculation process.

	Epoche	Batch Size	Optimizer	weight decay	Learning Rate	Value Metrics
Waterbirds	128	128	SGD	1.0	1e-5	Worst group ACC
CelebA	200	64	SGD	0.0	1e-3	Worst group ACC
FMoW	60	32	Adam	0.0	1e-4	Worst region ACC
Camelyon17	10	32	SGD	0.01	1e-3	Average ACC

CelebA, and Camelyon17. For FMoW, we used Adam as an optimizer. See More detail on the hyper-parameters in Appendix-Table 3.5.

Figure 3.4 shows the result of the ratio calculation process for Camelyon17 (left) and FMoW (right). ResNet-50 has four residual blocks represented on the x-axis. And the y-axis represents the OoD validation accuracy. Figure 3.4 clearly shows that the OoD validation accuracy in Camelyon17 task is more scattered in comparison to FMoW. Table 3.8 shows the task difficulty ratios that are calculated by Equation 3.2.1. We repeated the process five times and reported the ratios based on the average performance of environments in each task. Table 3.7 and 3.6 in the Appendix contain the detailed environments comparison for Waterbirds and CelebA.

Table 3.6 and 3.7 show the analysis of the environment used for task difficulty ratio calculation. Table 3.9 reports the selected algorithms' performance for the chosen tasks, including the average (Domainbed) and tasks' difficulty ranking (Ours) comparison. The R_2 (Score-based) failed since four out of five algorithms have a score of 0. To rank them, we have to consider the average again. R_3 represents a ranking result according to the tasks' difficulty.

Table 3.6. The accuracy of four classifiers trained for each environment on the Out-of-Distribution (OoD) validation set for the CelebA task with four environments.

Environment	Block 1	Block 2	Block 3	Block 4
E_1	84.61	84.61	84.61	84.61
E_2	84.61	84.61	84.61	84.61
E_3	15.38	15.38	15.38	15.38
E_4	15.38	15.38	15.38	15.38

Table 3.7. The accuracy of four classifiers trained for each environment on the Out-of-Distribution (OoD) validation set for the Waterbird task with four environments.

Environment	Block 1	Block 2	Block 3	Block 4
E_1	77.82	77.82	77.82	77.82
E_2	77.82	77.82	77.82	77.82
E_3	22.19	22.19	22.19	22.19
E_4	22.19	22.19	22.19	22.19

Table 3.8. The task difficulty ratio is calculated by Equation 3.2.1. The result is based on the average performance of environments in each task for five seeds.

Dataset	Task Difficulty Ratio
Camelyon17	0.80
FMoW	0.39
Waterbirds	0.66
CelebA	0.66

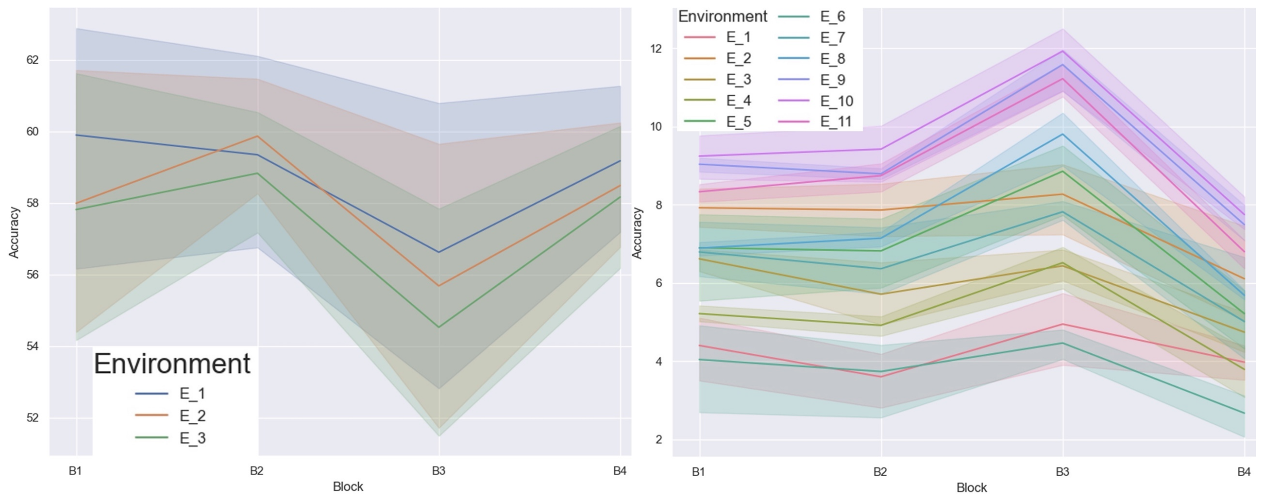


Fig. 3.4. The accuracy of four classifiers trained for each environment on the OoD validation set. Camelyon17 (left) and FMoW (right) environments.

Table 3.9. Ranking result for five algorithms on selected tasks from WILDS. R_1 represents the ranking based on average accuracy, R_2 expresses the score-based (OoD-Bench) ranking outcomes, and R_3 is the ranking according to the tasks' difficulty for the selected algorithms. We include results from the WILDS leaderboard[34]$*$ and [45]†.

	Waterbirds	CelebA	FMoW	Camelyon17	Average	Score	Ours	R_1	R_2	R_3
	(Worst-group Acc)	(Worst-group Acc)	(Worst-Reg Acc)	(Avg Acc)	(Domainbed)	(OoD-Bench)				
ERM	72.6†	47.2†	34.1*	70.8*	56.17	0	37.25	5	5	4
IRM	77.19	51.12	32.8*	64.2*	56.33	0	37.21	4	4	5
Group DRO	91.4†	88.9†	31.1*	68.4*	69.95	0	46.46	1	2	1
Deep COROL	88.3	89.44	32.8	59.5*	67.51	0	44.42	2	3	3
V-REx	87.3	64.2	33.64	82.84	66.98	2	44.83	3	1	2

3.4. Conclusion and future work

This work challenges the current approach for compare and ranking out-of-distribution generalization algorithms. Existing approaches, including average (Domainbed) and score-based (OoD-Bench) methods, have essential logical and practical drawbacks described in this work. To better understand each OoD generalization task, we presented a mechanism to define the task difficulty ratios and use them to compare tasks' difficulty on Out-of-Distribution generalization benchmarks. It helps to understand OoD generalization tasks better by comparing the tasks' environments. This analysis leads to a fair comparison between the OoD generalization algorithms using tasks' difficulty ratios. Studying the difficulty of the text-based task using the same approaches is a potential follow-up for this work.

Chapter 4

Conclusion

The ability to transfer or leverage knowledge across tasks poses a significant challenge for today's machine learning systems. Despite the success of AI systems trained on a specific task distribution, achieving high performance on diverse tasks with different distributions remains a challenging task. Forgetting previously acquired knowledge due to learning new tasks from different distributions is a common issue in AI systems, known as catastrophic forgetting. This issue is the result of backpropagation mechanism used to update their parameters. Addressing this problem has been attempted from different angles in various domains such as continual learning and Out-of-distribution (OOD).

Continual learning (also known as lifelong learning or incremental learning) is a machine learning paradigm where a system learns and adapts to new information over time, without forgetting previously learned knowledge. In other words, instead of learning a fixed set of tasks and then retraining the model from scratch when new data becomes available, a continual learning system is designed to continually learn from new data and build on its existing knowledge, while avoiding catastrophic forgetting of the previously learned knowledge.

The first project in this thesis aims to compare a chaotic learner with a naive continual learning approach. Traditional deep neural network models require multiple epochs to better estimate the model parameters from the training dataset. However, compensating for the effects of parameter updates in batch incremental learning setups is not always practical as it is not feasible to expect the training data will always be available. This chapter, addresses questions such as what are the alternative structures for neural networks to address those issues? Could we use a system to mimic the behavior of biological neurons and see how does it doing compared to neural networks? To address this issue, we propose a chaotic stream learner that mimics the behavior of biological neurons and does not update network parameters, requiring fewer samples compared to deep learning models in stream learning setups. The stream learner as a new method for mitigating catastrophic forgetting in continual few-shot learning. Unlike traditional approaches, this method leverages the chaotic structure of biological neurons for a unique perspective. In our experiments, we found that a single chaotic neuron used in the GLS stream learner leads to less forgetting compared to deep learning models, which require extensive training time and parameter learning for batch incremental and stream learning setups. Interestingly, our experiments on various datasets reveal that the chaotic stream learner naturally has less catastrophic forgetting than a CNN model in continual learning.

Out-of-distribution (OoD) generalization refers to the ability of a machine learning model to perform well on data that is different from the data it was trained on. In other words, when a model is tested on data that is not from the same distribution as its training data, it should still be able to make accurate predictions. OoD generalization is important because real-world data is often diverse and unpredictable, and it is impossible to train a model on all possible variations of the data. In addition, in some cases, the data distribution may change over time, making it important for a model to be able to generalize to new and different distributions. The second project in this thesis addresses some questions: firstly, whether current algorithm selection and ranking mechanisms are fair, and secondly, if they are not, what other factors need to be considered when comparing different algorithms for the task of Out-of-distribution generalization.

Existing methods for comparing and ranking OoD algorithms, such as average and score-based methods, have logical and practical drawbacks, and they do not consider the task difficulty level. In the second project of this thesis, we analyze the strengths and drawbacks of existing methods for comparing and ranking OoD algorithms. To address the issue of task difficulty level, we propose a novel ranking approach that defines the task difficulty ratios to compare OoD generalization algorithms. By analyzing the environments of the OoD generalization tasks, we can compare the tasks more accurately and achieve a fairer comparison between different OoD generalization algorithms based on the difficulty ratios of the tasks. We conducted experiments by comparing the average, score-based, and difficulty-based rankings of four selected tasks from the WILDS benchmark and five popular OoD algorithms. The analysis reveals significant changes in the ranking orders compared to current ranking approaches. Our work suggests that future algorithms should be more comprehensively evaluated considering metrics that measure or relates to distribution shift. It is still an open problem whether there exist algorithms that perform well under different types of distribution shifts. In the absence of such an algorithm, metrics such as task difficulty could be employed to rank and select the suitable algorithms according to the task.

References

[1] Jessica AGOSTINONE et Adriana DI POLO : Retinal ganglion cell dendrite pathology and synapse loss: Implications for glaucoma. In Progress in brain research, volume 220, pages 199–216. Elsevier, 2015.

[2] Rahaf ALJUNDI, Punarjay CHAKRAVARTY et Tinne TUYTELAARS : Expert gate: Lifelong learning with a network of experts. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jul 2017.

[3] Antreas ANTONIOU, Massimiliano PATACCHIOLA, Mateusz OCHAL et Amos STORKEY : Defining benchmarks for continual few-shot learning. arXiv preprint arXiv:2004.11967, 2020.

[4] Martin ARJOVSKY : Out of distribution generalization in machine learning. Thèse de doctorat, New York University, 2020.

[5] Martin ARJOVSKY, Léon BOTTOU, Ishaan GULRAJANI et David LOPEZ-PAZ : Invariant risk minimization. arXiv preprint arXiv:1907.02893, 2019.

[6] Harikrishnan Nellippallil BALAKRISHNAN, Aditi KATHPALIA, Snehanshu SAHA et Nithin NAGARAJ : Chaosnet: A chaos based artificial neural network architecture for classification. Chaos: An Interdisciplinary Journal of Nonlinear Science, 29(11):113125, Nov 2019.

[7] Robert JN BALDOCK, Hartmut MAENNEL et Behnam NEYSHABUR : Deep learning through the lens of example difficulty. arXiv preprint arXiv:2106.09647, 2021.

[8] Sara BEERY, Grant VAN HORN et Pietro PERONA : Recognition in terra incognita. In Proceedings of the European conference on computer vision (ECCV), pages 456–473, 2018.

[9] Peter BÜHLMANN : Invariance, causality and robustness. Statistical Science, 35(3):404–426, 2020.

[10] Arslan CHAUDHRY, Marc'Aurelio RANZATO, Marcus ROHRBACH et Mohamed ELHOSEINY : Efficient lifelong learning with a-gem. arXiv preprint arXiv:1812.00420, 2018.

[11] Zhiyuan CHEN et Bing LIU : Lifelong machine learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, 12(3):1–207, 2018.

[12] Gordon Christie, Neil Fendley, James Wilson et Ryan Mukherjee : Functional map of the world. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 6172–6180, 2018.

[13] Elliot Creager, Jörn-Henrik Jacobsen et Richard Zemel : Environment inference for invariant learning. In International Conference on Machine Learning, pages 2189–2200. PMLR, 2021.

[14] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh et Tinne Tuytelaars : Continual learning: A comparative study on how to defy forgetting in classification tasks. arXiv preprint arXiv:1909.08383, 2019.

[15] Li Deng : The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE Signal Processing Magazine, 29(6):141–142, 2012.

[16] John Duchi et Hongseok Namkoong : Learning models with uniform performance via distributionally robust optimization. arXiv preprint arXiv:1810.08750, 2018.

[17] Kenneth Falconer : Fractal geometry: mathematical foundations and applications. John Wiley & Sons, 2004.

[18] Chen Fang, Ye Xu et Daniel N Rockmore : Unbiased metric learning: On the utilization of multiple datasets and web images for softening bias. In Proceedings of the IEEE International Conference on Computer Vision, pages 1657–1664, 2013.

[19] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A. Rusu, Alexander Pritzel et Daan Wierstra : Pathnet: Evolution channels gradient descent in super neural networks. CoRR, abs/1701.08734, 2017.

[20] Richard FitzHugh : Impulses and physiological states in theoretical models of nerve membrane. Biophysical journal, 1(6):445–466, 1961.

[21] Yaroslav Ganin et Victor Lempitsky : Unsupervised domain adaptation by backpropagation. In International conference on machine learning, pages 1180–1189. PMLR, 2015.

[22] Muhammad Ghifary, W. Bastiaan Kleijn, Mengjie Zhang et David Balduzzi : Domain generalization for object recognition with multi-task autoencoders. CoRR, abs/1508.07680, 2015.

[23] Ian Goodfellow, Yoshua Bengio et Aaron Courville : Deep learning. MIT press, 2016.

[24] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville et Yoshua Bengio : Generative adversarial nets. Advances in neural information processing systems, 27, 2014.

[25] Leo GUGERTY : Newell and simon's logic theorist: Historical background and impact on cognitive modeling. In Proceedings of the human factors and ergonomics society annual meeting, volume 50, pages 880–884. SAGE Publications Sage CA: Los Angeles, CA, 2006.

[26] Ishaan GULRAJANI et David LOPEZ-PAZ : In search of lost domain generalization. arXiv preprint arXiv:2007.01434, 2020.

[27] Tyler L HAYES, Ronald KEMKER, Nathan D CAHILL et Christopher KANAN : New metrics and experimental paradigms for continual learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 2031–2034, 2018.

[28] Kaiming HE, Xiangyu ZHANG, Shaoqing REN et Jian SUN : Deep residual learning for image recognition. CoRR, abs/1512.03385, 2015.

[29] James L HINDMARSH et RM ROSE : A model of neuronal bursting using three coupled first order differential equations. Proceedings of the Royal society of London. Series B. Biological sciences, 221(1222):87–102, 1984.

[30] G. HINTON, L. DENG, D. YU, G. E. DAHL, A. MOHAMED, N. JAITLY, A. SENIOR, V. VANHOUCKE, P. NGUYEN, T. N. SAINATH et B. KINGSBURY : Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine, 29(6):82–97, Nov 2012.

[31] Alan L HODGKIN et Andrew F HUXLEY : A quantitative description of membrane current and its application to conduction and excitation in nerve. The Journal of physiology, 117(4):500, 1952.

[32] Diederik P KINGMA et Max WELLING : Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.

[33] James KIRKPATRICK, Razvan PASCANU, Neil RABINOWITZ, Joel VENESS, Guillaume DESJARDINS, Andrei A RUSU, Kieran MILAN, John QUAN, Tiago RAMALHO, Agnieszka GRABSKA-BARWINSKA et al. : Overcoming catastrophic forgetting in neural networks. Proc. of the national academy of sciences, 2017.

[34] Pang Wei KOH, Shiori SAGAWA, Henrik MARKLUND, Sang Michael XIE, Marvin ZHANG, Akshay BALSUBRAMANI, Weihua HU, Michihiro YASUNAGA, Richard Lanas PHILLIPS, Irena GAO et al. : Wilds: A benchmark of in-the-wild distribution shifts. In International Conference on Machine Learning, pages 5637–5664. PMLR, 2021.

[35] Alex KRIZHEVSKY, Ilya SUTSKEVER et Geoffrey E HINTON : Imagenet classification with deep convolutional neural networks. In F. PEREIRA, C. J. C. BURGES, L. BOTTOU et K. Q. WEINBERGER, éditeurs : Advances in Neural Information Processing Systems

25, pages 1097–1105. Curran Associates, Inc., 2012.

[36] David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Remi Le Priol et Aaron Courville : Out-of-distribution generalization via risk extrapolation (rex). In International Conference on Machine Learning, pages 5815–5826. PMLR, 2021.

[37] Brenden M Lake, Ruslan Salakhutdinov et Joshua B Tenenbaum : Human-level concept learning through probabilistic program induction. Science, 350(6266):1332–1338, 2015.

[38] Touraj Laleh, Mojtaba Faramarzi, Irina Rish et Sarath Chandar : Chaotic continual learning. In 4th Lifelong Machine Learning Workshop at ICML 2020, 2020.

[39] GC Layek : An introduction to dynamical systems and chaos, volume 449. Springer, 2015.

[40] Yann LeCun, Yoshua Bengio et al. : Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks, 3361(10):1995, 1995.

[41] Yann LeCun, Yoshua Bengio et Geoffrey Hinton : Deep learning. nature, 521(7553): 436–444, 2015.

[42] Chengqing Li : When an attacker meets a cipher-image in 2018: A year in review. CoRR, abs/1903.11764, 2019.

[43] Da Li, Yongxin Yang, Yi-Zhe Song et Timothy Hospedales : Learning to generalize: Meta-learning for domain generalization. In Proceedings of the AAAI conference on artificial intelligence, volume 32, 2018.

[44] Da Li, Yongxin Yang, Yi-Zhe Song et Timothy M Hospedales : Deeper, broader and artier domain generalization. In Proceedings of the IEEE international conference on computer vision, pages 5542–5550, 2017.

[45] Evan Z Liu, Behzad Haghgoo, Annie S Chen, Aditi Raghunathan, Pang Wei Koh, Shiori Sagawa, Percy Liang et Chelsea Finn : Just train twice: Improving group robustness without training group information. In International Conference on Machine Learning, pages 6781–6792. PMLR, 2021.

[46] Ziwei Liu, Ping Luo, Xiaogang Wang et Xiaoou Tang : Large-scale celebfaces attributes (celeba) dataset. Retrieved August, 15(2018):11, 2018.

[47] David Lopez-Paz et Marc'Aurelio Ranzato : Gradient episodic memory for continual learning, 2017.

[48] Arun Mallya et Svetlana Lazebnik : Packnet: Adding multiple tasks to a single network by iterative pruning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 7765–7773, 2018.

[49] John McCarthy, Marvin L. Minsky, Nathaniel Rochester et Claude E. Shannon : A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. AI Magazine, 27(4):12, Dec. 2006.

[50] Michael McCloskey et Neal J Cohen : Catastrophic interference in connectionist networks: The sequential learning problem. In Psychology of learning and motivation, volume 24, pages 109–165. Elsevier, 1989.

[51] Martial Mermillod, Aurélia Bugaiska et Patrick Bonin : The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects, 2013.

[52] Seyed Iman Mirzadeh, Mehrdad Farajtabar et Hassan Ghasemzadeh : Dropout as an implicit gating mechanism for continual learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pages 232–233, 2020.

[53] Krikamol Muandet, David Balduzzi et Bernhard Schölkopf : Domain generalization via invariant feature representation. In International Conference on Machine Learning, pages 10–18. PMLR, 2013.

[54] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko et Bo Wang : Moment matching for multi-source domain adaptation. In Proceedings of the IEEE/CVF international conference on computer vision, pages 1406–1415, 2019.

[55] Jonas Peters, Peter Bühlmann et Nicolai Meinshausen : Causal inference by using invariant prediction: identification and confidence intervals. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 78(5):947–1012, 2016.

[56] Mohammad Pezeshki, Oumar Kaba, Yoshua Bengio, Aaron C Courville, Doina Precup et Guillaume Lajoie : Gradient starvation: A learning proclivity in neural networks. Advances in Neural Information Processing Systems, 34:1256–1272, 2021.

[57] Nicolas Pinchaud et Martin Hedlund : Camelyon17 grand challenge. Submission results Camelyon17 challange. https://camelyon17. grand-challenge. org/evaluation/results/. Accessed, 18, 2019.

[58] Roger Ratcliff : Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. Psychological review, 97(2):285, 1990.

[59] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl et Christoph H Lampert : icarl: Incremental classifier and representation learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2001–2010, 2017.

[60] Shaoqing Ren, Kaiming He, Ross Girshick et Jian Sun : Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence,

D. D. Lee, M. Sugiyama et R. Garnett, éditeurs : Advances in Neural Information Processing Systems 28, pages 91–99. Curran Associates, Inc., 2015.

[61] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu et Gerald Tesauro : Learning to learn without forgetting by maximizing transfer and minimizing interference. arXiv preprint arXiv:1810.11910, 2018.

[62] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg et Li Fei-Fei : Imagenet large scale visual recognition challenge, 2015.

[63] Stuart J Russell : Artificial intelligence a modern approach. Pearson Education, Inc., 2010.

[64] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu et Raia Hadsell : Progressive neural networks, 2016.

[65] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto et Percy Liang : Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. arXiv preprint arXiv:1911.08731, 2019.

[66] Shiori Sagawa, Aditi Raghunathan, Pang Wei Koh et Percy Liang : An investigation of why overparameterization exacerbates spurious correlations. In International Conference on Machine Learning, pages 8346–8356. PMLR, 2020.

[67] Jürgen Schmidhuber : Deep learning in neural networks: An overview. Neural networks, 61:85–117, 2015.

[68] Zheyan Shen, Peng Cui, Tong Zhang et Kun Kunag : Stable learning via sample reweighting. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 5692–5699, 2020.

[69] Zheyan Shen, Jiashuo Liu, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu et Peng Cui : Towards out-of-distribution generalization: A survey. arXiv preprint arXiv:2108.13624, 2021.

[70] Connor Shorten et Taghi M Khoshgoftaar : A survey on image data augmentation for deep learning. Journal of big data, 6(1):1–48, 2019.

[71] Shagun Sodhani, Sarath Chandar et Yoshua Bengio : Toward training recurrent neural networks for lifelong learning. Neural computation, 32(1):1–35, 2020.

[72] Shagun Sodhani, Mojtaba Faramarzi, Sanket Vaibhav Mehta, Pranshu Malviya, Mohamed Abdelsalam, Janarthanan Rajendran et Sarath Chandar : An introduction to lifelong supervised learning. arXiv preprint arXiv:2207.04354, 2022.

[73] Baochen SUN, Jiashi FENG et Kate SAENKO : Return of frustratingly easy domain adaptation. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 30, 2016.

[74] Ilya SUTSKEVER, Oriol VINYALS et Quoc V. LE : Sequence to sequence learning with neural networks. In Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, NIPS'14, page 3104–3112, Cambridge, MA, USA, 2014. MIT Press.

[75] Mariya TONEVA, Alessandro SORDONI, Remi Tachet des COMBES, Adam TRISCHLER, Yoshua BENGIO et Geoffrey J. GORDON : An empirical study of example forgetting during deep neural network learning. CoRR, abs/1812.05159, 2018.

[76] Alan M TURING : Computing machinery and intelligence. In Parsing the turing test, pages 23–65. Springer, 2009.

[77] Gido M. van de VEN et Andreas S. TOLIAS : Three scenarios for continual learning, 2019.

[78] Vladimir N VAPNIK : An overview of statistical learning theory. IEEE transactions on neural networks, 10(5):988–999, 1999.

[79] Ashish VASWANI, Noam SHAZEER, Niki PARMAR, Jakob USZKOREIT, Llion JONES, Aidan N. GOMEZ, Lukasz KAISER et Illia POLOSUKHIN : Attention is all you need. CoRR, abs/1706.03762, 2017.

[80] Jindong WANG, Cuiling LAN, Chang LIU, Yidong OUYANG, Tao QIN, Wang LU, Yiqiang CHEN, Wenjun ZENG et Philip YU : Generalizing to unseen domains: A survey on domain generalization. IEEE Transactions on Knowledge and Data Engineering, 2022.

[81] Ju XU et Zhanxing ZHU : Reinforced continual learning, 2018.

[82] Nanyang YE, Kaican LI, Lanqing HONG, Haoyue BAI, Yiting CHEN, Fengwei ZHOU et Zhenguo LI : Ood-bench: Benchmarking and understanding out-of-distribution generalization datasets and algorithms. arXiv preprint arXiv:2106.03721, 2021.