Université de Montréal

Advances in Deep Learning Methods for Speech Recognition and Understanding

par Dmitriy Serdyuk

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

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

octobre, 2020

Résumé

Ce travail expose plusieurs études dans les domaines de la reconnaissance de la parole et compréhension du langage parlé. La compréhension sémantique du langage parlé est un sous-domaine important de l'intelligence artificielle. Le traitement de la parole intéresse depuis longtemps les chercheurs, puisque la parole est une des charactéristiques qui definit l'être humain. Avec le développement du réseau neuronal artificiel, le domaine a connu une évolution rapide à la fois en terme de précision et de perception humaine. Une autre étape importante a été franchie avec le développement d'approches bout en bout. De telles approches permettent une coadaptation de toutes les parties du modèle, ce qui augmente ainsi les performances, et ce qui simplifie la procédure d'entrainement. Les modèles de bout en bout sont devenus réalisables avec la quantité croissante de données disponibles, de ressources informatiques et, surtout, avec de nombreux développements architecturaux innovateurs. Néanmoins, les approches traditionnelles (qui ne sont pas bout en bout) sont toujours pertinentes pour le traitement de la parole en raison des données difficiles dans les environnements bruyants, de la parole avec un accent et de la grande variété de dialectes.

Dans le premier travail, nous explorons la reconnaissance de la parole hybride dans des environnements bruyants. Nous proposons de traiter la reconnaissance de la parole, qui fonctionne dans un nouvel environnement composé de différents bruits inconnus, comme une tâche d'adaptation de domaine. Pour cela, nous utilisons la nouvelle technique à l'époque de l'adaptation du domaine antagoniste. En résumé, ces travaux antérieurs proposaient de former des caractéristiques de manière à ce qu'elles soient distinctives pour la tâche principale, mais non-distinctive pour la tâche secondaire. Cette tâche secondaire est conçue pour être la tâche de reconnaissance de domaine. Ainsi, les fonctionnalités entraînées sont invariantes vis-à-vis du domaine considéré. Dans notre travail, nous adoptons cette technique et la modifions pour la tâche de reconnaissance de la parole dans un environnement bruyant.

Dans le second travail, nous développons une méthode générale pour la régularisation des réseaux génératif récurrents. Il est connu que les réseaux récurrents ont souvent des difficultés à rester sur le même chemin, lors de la production de sorties longues. Bien qu'il soit possible d'utiliser des réseaux bidirectionnels pour une meilleure traitement de séquences pour l'apprentissage des charactéristiques, qui n'est pas applicable au cas génératif. Nous avons développé un moyen d'améliorer la cohérence de la production de longues séquences avec des réseaux récurrents. Nous proposons un moyen de construire un modèle similaire à un réseau bidirectionnel. L'idée centrale est d'utiliser une perte L2 entre les réseaux récurrents génératifs vers l'avant et vers l'arrière. Nous fournissons une évaluation expérimentale sur une multitude de tâches et d'ensembles de données, y compris la reconnaissance vocale, le sous-titrage d'images et la modélisation du langage.

Dans le troisième article, nous étudions la possibilité de développer un identificateur d'intention de bout en bout pour la compréhension du langage parlé. La compréhension sémantique du langage parlé est une étape importante vers le développement d'une intelligence artificielle de type humain. Nous avons vu que les approches de bout en bout montrent des performances élevées sur les tâches, y compris la traduction automatique et la reconnaissance de la parole. Nous nous inspirons des travaux antérieurs pour développer un système de bout en bout pour la reconnaissance de l'intention.

Mots clés: apprentissage profond, apprentissage automatique, reconnaissance de la parole, réseaux de neurones, adaptation de domaine, reconnaissance de la parole bruyante, apprentissage antogoniste, réseaux de neurones récurrents, génération de séquences, compréhension du langage vocal, apprentissage de bout en bout.

Abstract

This work presents several studies in the areas of speech recognition and understanding. The semantic speech understanding is an important sub-domain of the broader field of artificial intelligence. Speech processing has had interest from the researchers for long time because language is one of the defining characteristics of a human being. With the development of neural networks, the domain has seen rapid progress both in terms of accuracy and human perception. Another important milestone was achieved with the development of end-to-end approaches. Such approaches allow co-adaptation of all the parts of the model thus increasing the performance, as well as simplifying the training procedure. End-to-end models became feasible with the increasing amount of available data, computational resources, and most importantly with many novel architectural developments. Nevertheless, traditional, non end-to-end, approaches are still relevant for speech processing due to challenging data in noisy environments, accented speech, and high variety of dialects.

In the first work, we explore the hybrid speech recognition in noisy environments. We propose to treat the recognition in the unseen noise condition as the domain adaptation task. For this, we use the novel at the time technique of the adversarial domain adaptation. In the nutshell, this prior work proposed to train features in such a way that they are discriminative for the primary task, but non-discriminative for the secondary task. This secondary task is constructed to be the domain recognition task. Thus, the features trained are invariant towards the domain at hand. In our work, we adopt this technique and modify it for the task of noisy speech recognition.

In the second work, we develop a general method for regularizing the generative recurrent networks. It is known that the recurrent networks frequently have difficulties staying on same track when generating long outputs. While it is possible to use bi-directional networks for better sequence aggregation for feature learning, it is not applicable for the generative case. We developed a way improve the consistency of generating long sequences with recurrent networks. We propose a way to construct a model similar to bi-directional network. The key insight is to use a soft L2 loss between the forward and the backward generative recurrent networks. We provide experimental evaluation on a multitude of tasks and datasets, including speech recognition, image captioning, and language modeling.

In the third paper, we investigate the possibility of developing an end-to-end intent recognizer for spoken language understanding. The semantic spoken language understanding is an important step towards developing a human-like artificial intelligence. We have seen that the end-to-end approaches show high performance on the tasks including machine translation and speech recognition. We draw the inspiration from the prior works to develop an end-to-end system for intent recognition.

Key words: deep learning, machine learning, speech recognition, neural networks, domain adaptation, noisy speech recognition, adversarial learning, recurrent neural networks, sequence generation, spoken language understanding, end-to-end learning.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This general theme of this document is the development of methods to recognize spoken signal and understanding the human speech using novel deep learning architectures. The ability to exchange information via sounds is an essential aspect of human intelligence. Therefore, I believe that studying the mechanisms of spoken interaction is one of the most important building bricks for hypothetical general artificial intelligence as well a useful tool for many practical applications.

Despite the fact that there was several decades of research progress in the areas of speech processing and machine learning, there are many unresolved questions. One of the most important milestones in the field was the development and adoption of deep learning. It allowed to significantly increase the quality of speech processing, including speech recognition and synthesis. Nevertheless, there is still space for improvement.

First, many speech recognition systems are not enough robust to noisy environments, accented speech, and variable recording hardware. It is known that speech recognition rapidly degrades in challenging environments. Such environments are crowded places, where many persons talk simultaneously; airports, where some fragments of speech are completely obstructed by background noise; roads and factories, where background noise constantly interferes with the speech.

The second shortcoming originates from an architecture which is widely used in deep learning systems dealing with sequentual inputs or outputs. This is the case for speech processing, where a single utterance can be as long as several thousands frames long. Dealing with the information flow through the long sequences is generally hard. Despite the numerous attempts to tackle the issue, long sequences generated with state of the art models at the time were unable to keep the consistency throughout the whole length.

The third issue concerns understanding the spoken language. A usual way to perform such a task is to have a pipeline of a recognizer followed by a natural language processing component. Meanwhile, it has been shown that end-to-end approach is superior in many cases because it allows to optimize all the components in tandem and allows them to co-adapt. Therefore, building an end-to-end systems for understanding spoken language is an important task.

Attempting to solve three issues defined above, this work presents three papers:

1. *Invariant Representations for Noisy Speech Recognition*, published at Neural Information Processing Systems 2016 workshop on End-to-end Learning for Speech and Audio Processing by Dmitriy Serdyuk, Kartik Audhkhasi, Philémon Brakel, Bhuvana Ramabhadran, Samuel Thomas, Yoshua Bengio (see Chapter 3);

2. *Twin Networks: Matching the Future for Sequence Generation*, published at International Conference on Learning Representations 2018 by Dmitriy Serdyuk, Nan Rosemary Ke, Alessandro Sordoni, Adam Trischler, Chris Pal, Yoshua Bengio (see Chapter 4);

3. *Towards End-to-end Spoken Language Understanding*, published at International Conference on Acoustics, Speech and Signal Processing by Dmitriy Serdyuk, Yongqiang Wang, Christian Fuegen, Anuj Kumar, Baiyang Liu, Yoshua Bengio (see Chapter 5).

1.1 Presented Papers

The subsections below give a brief reference for each paper as well as the statement of contribution.

1.1.1 Invariant Representations for Noisy Speech Recognition

Modern speech recognition systems are very powerful when run with clean and predictable signal. Unfortunately, this is not the case when the environment is noisy. The quality rapidly degrades with the background noise. This effect is especially pronounced when the speech recognition system encounters an environmental condition is was not trained for.

A recent work [42] proposed to use adversarial training methods to perform model adaptation. In the nutshell, they train the model to perform the task at hand while at the same time injecting a signal in the middle which dictates *not* to distinguish domains. Hence, this is called adversarial training – the part of the model is optimized to perform badly an auxiliary task. The injected signal makes the model to learn features that are similar across different domains but are suited well for the main task.

In our publication Invariant Representation for Noisy Speech Recognition, my co-authors and me developed a method based on [42]. The method is using an adversarial adaptation training to improve speech recognition in noisy environments. Reformulating the task as domain adaptation helped us to improve the baseline system and to better understand the extents of applicability of the adversarial adaptation technique.

We performed numerous rigorous experiments testing a hybrid MLP-HMM system using a well-benchmarked Aurora-4 noisy speech corpus. In our experiments we tested the performance of the system when trained on a small subset of noise conditions and tested on a different subset of conditions. We tested the simulated noise, as well the genuine one. Furthermore, we tested the adaptation to different recording conditions (utterances captured with a different microphone).

Statement of Contribution

The author of this thesis was was the leading author for this paper. My contributions of the first author to this work are:

1. He found the primary idea to use the adversarial domain adaptation to learn invariant representations;
2. He proposed model for learning invariant representations;
3. He proposed the set of experiments to test the performance of the model;
4. He conducted the majority of the experiments.

The other authors provided me with invaluable help with the dataset preparation, consultation, brain storming, and help with the experiments.

1.1.2 Twin Networks: Matching the Future for Sequence Generation

Generative recurrent networks are notoriously hard to train. One of the problems is that the generated text samples are not consistent throughout long sequences. This can be observed with unconditional generation, when a generative model is trained to produce text given previous piece of text. In such a scenario, recurrent generative networks tend to go off the rails when asked to generate sufficiently long texts. The conditional generative neural models are also affected by this problem, although to smaller extent.

This work considers a family of recurrent networks that have some hidden state. This includes simple Elman recurrent networks, recurrent networks composed of Long Short-Term Memory (LSTM) cells, or Gated Recurrent Unit (GRU) cells. The recurrent hidden state is connected to all the previous inputs through the previous hidden states. Furthermore, the hidden state is what is used to generate the following outputs. Therefore, in this work we hypothesized that the representation contained in the recurrent hidden state is the summarization of all the previous time-steps which is necessary for producing the following time-steps.

In this work, we developed a method to help a recurrent network to generate more consistent samples. Because of the nature of the factorization of the total probability of the output sequence, the generation has to be performed in a single direction. Although, bi-directional networks are well-suited for sequence representation learning, it is hard to use them for sequence generation. Therefore, the first motivation for this work is to develop a way to use bi-directional network for generation.

The second motivation comes from the fact that the hidden state summarizes all the previous states. Now, let us consider a network that generates the same sequence in the opposite direction. The hidden state of the backward-running network is the summarization of all the future states. It means that learning to predict the backward hidden state has to help to generate more consistent samples.

Driven by these motivations we develop a method to regularize generative recurrent networks. In the nutshell, we train an extra recurrent network to generate the same sequence backwards. Then, the co-temporal states are tied together via an L2 loss. This loss is only used during the training. Then, during the testing, the backward-running network is discarded and the generation is performed as usual.

This regularization method yielded consistent performance for a multitude of tasks and datasets. We performed the experiments with speech recognition, image captioning, language modelling, and pixel-by-pixel recurrent image generation tasks.

Statement of Contribution

The author of this thesis was leading the research effort on the Twin Networks project. My contributions are:

1. Together with the second author he developed the idea of Twin Networks;

2. He conducted the experiments on speech recognition;

3. He conducted a half of the experiments on image captioning.

The shared first author contributed in the development of the twin networks, conducted several experiments in image captioning and language modeling. The third author conducted the majority of the experiments for language modelling and the image generation. The other authors contributed in the discussion, brainstorming and planning the research effort.

1.1.3 Towards End-to-end Spoken Language Understanding

Spoken language understanding is a task that is familiar to most people through the digital assistants. The user utters a command, such as “What is the weather today?”, then the assistant is expected to respond to the command, for example looking up and reporting the weather. Modern spoken language understanding systems work in several stages. Usually, in order to understand the user’s query, the assistant transcribes it first into a text input, then feeds the text into the natural language understanding model. In other words, spoken language understanding requires training a speech recognizer as well as a natural language understanding model.

With the development of end-to-end learning systems, new breakthroughs were achieved in a diverse set of machine learning tasks. These tasks include machine translation, speech recognition, image captioning, and speech synthesis. The end-to-end paradigm allows the components of the system to co-adapt. Optimizing all the components for the required task loss (or its surrogate) helps to achieve state of the art performance in many cases.

In this work we investigate the possibility to train in conjunction the recognizer and the language understanding parts. To make the first step toward end-to-end learning, we train the intent recognizer without an intermediate text transcription.

We employ the multitude of the tricks from the literature on end-to-end speech recognition. Our experimental results show a great potential in this line of research. Although, we do not reach the state of the art for this task, we show similar performance using drastically less computation. We analyse the model we trained and observe that it is capable of capturing the semantic attention directly from the audio features.

Contribution

The author of this thesis was the sole leading author for this work and performed the majority of the hand-on work for this project. The contributions of the first author are:

1. He developed the experimentation plan and the ideas to test for this project;
2. He conducted the baseline experiments for two-stage spoken language understanding;
3. He conducted the experiments for the end-to-end spoken language understanding.

The second collaborator collected the dataset, performed the data cleanup, and provided consultation for the text-based natural language understanding. The third author contributed in the dataset collection as well in the brainstorming and the global project planning. The fourth and the fifth authors provided consultation and the baselines for the traditional natural language understanding systems. The last author provided support in project planning and discussions.

1.2 Artificial Intelligence

While performing day-to-day tasks humans were always thinking of ways to optimize them, reduce the effort. Since the time when our predecessors learned to use simple tools, throughout the period of domestication of agricultural plants, and up to industrial revolution and modern age, the humanity was developing ways to reduce labor amount needed to perform more and more complex work. Development of tools, automatization, distribution are the factors that helped to progress our world. While humans are able to optimize hand labor, scientists started to wonder if it is possible to improve on mental labor. In other words, is it possible to create a thinking machine?

Although, humans intuitively understand the concept of intelligence ("I know when I see it"), it is difficult to define it rigorously. In the broad terms, intelligence is the ability to perceive information, retain it as knowledge and skills, and apply later towards adaptive behavior [108]. Alan Turing made an attempt [155] to define intelligence by comparing it to a human via a message exchange session. A judge has to communicate with a pretender. After the message exchange, the judge is answering a question if the interlocutor is a human or not. While this approach sets up a certain baseline, it has many limitations. Such that, it tests only certain aspects of intelligence and susceptible to the choice of the judge.

When it is challenging to define intelligence in humans, it is even more challenging to define it across species. Some form of intelligence was observed in many multi-cellular organisms. Generally, simpler organisms tend to have weaker form intelligence, therefore it is thought to be correlated with the number of connections in the brain.

Inspired by the advanced functions of the central nervous system in living creatures, the artificial intelligence researchers proposed a concept of the artificial neural network. Some of the first works in this directions, was *perceptron* [124], an array of 400 photocells randomly connected to the "hidden neurons", implemented by hubs of wiring. Each connection to the hidden neurons had a tunable potentiometer. In order to tune the potentiometers, the team of Rosenblatt used a perceptron rule: after each demonstration the decision boundary was moved to classify correctly as many samples, as possible. This machine was trained to recognize digit shapes and other pattern recognition tasks. The perceptron machine, the first demonstration of the artificial neural network, was implemented in hardware. Now we use software implementations.

Another important milestone in the field of artificial intelligence was the adoption of the *error back-propagation* algorithm (back-propagation, or even backprop for short). The algorithm uses dynamic programming methods to compute the gradient of the loss with respect to the parameters of a given multi-layer model.

For some time, researchers in the field considered that it is necessary to pre-train very deep artificial neural networks. Unsupervised algorithms for pre-training were developed. One of the most prominent ones is the pre-training using *deep belief networks* [68]. Such pre-training algorithms allowed to use deeper networks, meaning more hidden layers. It turned out that deep networks better generalize to unseen examples and are able to perform more complex tasks. With the development of new methods to construct neural networks, the need for pre-training has disappeared. Nevertheless, the development of unsupervised and semi-supervised algorithms is still being pursued by many researchers.

One of the properties of multi-layer neural networks is that it can be examined which activations respond most to which input. Therefore, it is possible to talk about the *hidden representation* of the input signal. The hidden representation is a vector of all neuron activations from a given layer for a given input. This representation is a condensed information about the input. Along the depth of the artificial neural network, the representations discard the information which is irrelevant for the task at hand and preserve and transform useful information into a form that is easier to aggregate. All this makes it important to study the hidden representation. Therefore, many researchers use the visualizations of the hidden representations to reason about the model at hand, the data, and the algorithms.

Furthermore, the *representation learning* became a relevant subfield of machine learning. The ability to compress and represent in a form that is easily understandable by machine made learnable representations a crucial tool for many applications. One of the most frequent uses of learning representations is text embedding. Each word in vocabulary is transformed into a real vector. The transformation is trained in a way that the resulting vector can be used in a number of applications, such as but not limited to part of speech tagging, parse tree building, text summarization, understanding, and text composing.

1.3 The Role of Speech Processing in the Development of Artificial Intelligence

The verbal skills is one of the definitive feature of human intelligence. Without the development of complex communication via making sounds and primitive forms of spoken language it would be hardly possible to sustain large societies and, later, civilizations. Verbal communication enabled the rapid experience exchange, simplified teaching and learning, and allowed more sophisticated reflection of the surrounding world. Early humans were using their language skills to teach their peers. Compare this to learning form example, where an individual might need to perform a dangerous activity in order to acquire the same knowledge. Furthermore, early humans used the verbal communication to raise their children faster. Again, it is easier and less dangerous to explain a danger rather than to show it. These two aspects gave early humans amazing evolutionary advantage. The third aspect is the development of the sophisticated analysis and meta-analysis. This allowed to kick-start early forms of sciences and research.

Humans developed verbal skills long before than they started to write. Even after the development of writing, it was accessible only to a small fraction of literate people. Usually, literacy was available to the elite, secular and religious, and to the professions like accountants. The rest of the people usually were unable to write. This leads to a conclusion that the spoken communication was the major part of human experience exchanve for very long time. Furthermore, the verbal communication defined various languages (in a sense of "English language"), and the written form was secondary.

Therefore, when studying how to automatically understand language, it is important to study the spoken language. I believe that the progress in the automatic spoken language recognition and understanding is crucial. Not only would it allow us to develop better commercial systems for respective tasks, but it will provide us better understanding of languages.

Since the written form was utilized by a small minority of the educated, in some cases, the written language developed to become sufficiently different from the oral form. The difference and variability in the pronunciation and spelling makes it more challenging to construct algorithms working with speech. Furthermore, many local accents developed inside languages. Such a great variability is a great challenge for modern recognition algorithms.

1.4 Audio and Speech Applications

With the development of technology *automatic speech recognition* (ASR) systems become a daily part of our lives and used in portable devices, call centers, for automatic meeting transcription and many other fields. After several decades of research in the area of speech recognition, a complicated pipeline was developed, it is called the hybrid ASR system. One feature of the hybrid system is that the components are optimized separately and are compiled into a single system after. With the development of machine learning and deep feature representation learning we are investigating the end-to-end approach for a challenging task of ASR. The end-to-end means that that the machine learning model is optimized as a whole, having several levels of representation as opposed to manual feature extraction and separately training several models with handcrafted objectives. Historically, end-to-end approach showed its advantages in the context of convolutional neural networks and now they show much better performance than the handcrafted techniques like key-point extraction.

The ASR task is a challenging task due to the fact that the spectrograms are not interpretable by a human and the speech signal is high volume data having length hard to fit into memory and optimize for a whole utterance. The speech recognition field has several sub-fields apart from the ASR itself. The tasks solved by modern speech related systems include speaker recognition and identification; speech separation in multi-speaker environments; speech enhancement; low resource, noisy, or accented speech recognition. Some of these tasks are classification tasks and some of them are regression tasks making them even more challenging due to the output space complexity.

The rest of this work is structured in the following way. Chapter 2 introduces the basic tools used throughout the papers presented in Chapters 3, 4, 5. This chapter is a brief overview of machine learning (Section 2.1), deep learning (Section 2.2), and some particular architectures important for the following exposition. This chapter also includes an overview of traditional speech recognition (Section 2.3) with hybrid systems and more modern end-to-end approaches (Section 2.4). Chapters 3, 4, 5 are dedicated to the three papers presented. Chapter 6 concludes the thesis.

Chapter 2

Background

The field of machine learning is relatively new and fast moving. This makes the terminology vary from one source to another. This section briefly describes common techniques used throughout this work.

Section 2.1.1 starts with common description of a machine learning problem, Section 2.1.2 discusses ways to prevent the model from over specialization on the training set, or regularize, Section 2.2 introduces neural networks, a machine learning model used in this work and Section 2.2.1 continues with an efficient algorithm to optimize a neural network and Section 2.2.2 introduces the methods to regularize neural networks.

Then we discuss the tools specific to speech recognition models. In Section 2.2.4 we introduce a method to model sequences of data and we continue with a description of a sequence-to-sequence learning problems in Section 2.2.6, namely the attention-based sequence generator.

2.1 Machine Learning

2.1.1 General Setup

We start by defining a common problem setup. For a supervised *learning task* every data point is a pair of (x,y), where $x \in X$ is an input and $y \in Y$ is an output. The X set is referred as the input space and the Y set is the output space. Both sets might be finite, infinite, countable, or uncountable and depending on the nature of the output set, the learning task is called classification, regression, or structured prediction task.

- Classification task corresponds to the finite output set Y of a number of classes. For example, a task of MNIST digit recognition [88] has a 784-dimensional input set $X = \{0, 1, \dots, 255\}^{784}$ where each dimension is a gray scale pixel color for 28×28 image; and the output space is $Y = 0, \dots, 9$ – a digit on the input picture.

- Regression task is a task with a real-valued output space. An example of this kind of task is a linear curve fitting: the input and the output are both real values, meaning $X = Y = \mathbf{R}$ and the output is a linear function of the input $y = ax + b$ with the unknown parameters a and b.

- Structured prediction task involves more sophisticated output spaces Y, such as a set of sets, graphs, trees, etc. One particular example is interesting to us: a case when the output space is a set of sequences under finite alphabet $V : |V| = m$, $Y = V^*$. This is a task of sequence prediction like language modelling, speech recognition, machine translation, caption generation. The output text is split into tokens such as words, sub-words, or characters and the learning system is asked to produce a sequence of these tokens which is a translation of the input to another language, a transcription of the speech signal, or a caption for an input image.

The last component of a learning task is the *task loss* $L(\hat{y}, y)$ is a discrepancy between the candidate answer $\hat{y}$ and the *ground-truth* y. The lower the loss is – the better the model. Sometimes, people use the *score* (the higher the better) interchangeably with the loss meaning that the loss is the negation of the score and other way around. The form the loss depends on the output space Y. For example, for classification tasks, the loss might be categorical cross-entropy, miss-classification error, precision, recall, or F1 score; for the regression a commonly used loss is the mean squared error (MSE); and for sequence prediction tasks it is usually word error rate (WER), character error rate (CER), BLEU score [111], METEOR [12] and others.

The life-cycle of a machine learning task is divided into the training phase and the test phase. During the training phase, we have access to a set of pairs $\{(x_i, y_i)\}_{i=0}^{N}$ which is usually called the *training set*, we are asked to provide a *prediction function* $f(\cdot) \in F \subset X \to Y$ which maps inputs x to the outputs y. A process of mapping the training set to a prediction function (finding the most appropriate function in the prediction function space F) is called *training*. The function space F is often parametrized, we denote a function parametrized by a parameter θ as $f_\theta(\cdot)$, while θ belongs to a set of acceptable parameters Θ, this space is defined by the prediction function space structure – model architecture and regularization which will be discussed later in Section 2.1.2.

Once we obtained the prediction function, we would like to check how it performs on unseen examples. For this purpose, a subset of the examples is selected and fixed before training. Most of the datasets are usually provided with a test set. During the test phase, the prediction function is *evaluated* on an another set, which is called *test* or frequently in signal processing literature, *evaluation* set. The function is fed by an input to compute the average loss like

$$\frac{1}{N_T} \sum_{x,y \in T} L(f(x), y). \tag{2.1}$$

The distribution from which the data is sampled $p(x,y)$ is commonly referred as data *manifold*.

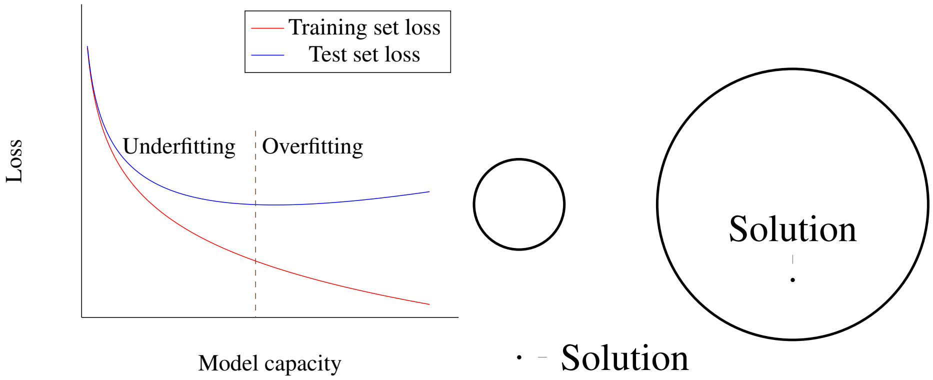
2.1.2 Bias-Variance trade-off: Underfitting and Overfitting

While we define the learning task to optimize the training objective, the real world application would to use the prediction from the machine learning system to analyze new, unseen examples and this kind of generalization is essentially necessary for the practical use. A good model should have two properties: it should be powerful enough to be able to model the train data dependencies and it should not be too flexible capture “extra” training set dependencies which are not present in the test set. The first property implies that the model should have reasonably well score on the train set while the second one tells that the model should not be overspecialized to the train set.

The ability of the model to represent different configuration is the model’s *capacity* and in simple cases it is just the number of the parameters $|\Theta|$, computed up to all possible parameter symmetry. Additionally, capacity reduces with any kind of constraint, soft or a hard one. The bigger capacity, the richer the space of prediction functions representable by the model. It means that a model with small capacity does not have enough flexibility to represent the training data, assign the right class for the majority of the data for classification task. As soon as the model capacity is too high, it specializes on the training data which leads to the degradation of the test score. This behaviour is depicted on the Figure 2.1a: the low capacity regime is called *underfitting* and the high capacity regime is *overfitting*. The ideal model is one on the dashed line, with minimal test error rate. This is connected to the bias and variance of the statistical estimator, and thus called bias-variance trad-off, it is illustrated on the Figure 2.1b, the left picture demonstrates a small family of functions which corresponds to low capacity, low variance and high bias; and the right picture demonstrates the case of high capacity and high variance.

For this purpose we maintain a separate test set which is used for scoring. Unfortunately, the process of searching a model and its configuration (also called hyper-parameter search) is also a learning task which can be prone to overfitting. To solve this issue, we introduce one more set: cross-validation set which should be used for the hyper-parameter tuning and model search. Ideally, in order to avoid overfitting on the test set, the model should be evaluated on the test set only once.

Some of the methods for capacity control for the model, the regularization techniques discussed in the Section 2.2.2.



(a) Overfitting illustration.

(b) Bias-variance trade-off.

Figure 2.1: Overfitting example. This is an illustration of a model accuracy on the training and the test sets depending on the model capacity. With a limited capacity model is not able to capture the data dependencies and its performance is bad on both test and training sets. With an increase of capacity the performance increases then the test loss stops improving while the training loss continues to go towards zero. The first region is the *underfitting*, the model is not able to fit the data; and the second is *overfitting*, the model is so powerful that it fits the particular characteristics of the training data which does not exist in the test set.

2.2 Deep Learning

When the field is field of machine learning is new and fast moving, *deep learning* is even more so.

One of a successful models nowadays are variations of neural networks. The neural networks are compositional functions

$$f(x) = f_K(f_{K-1}(\cdots f_1(x))), \tag{2.2}$$

which consist of K differentiable functions $f_1, \ldots, f_K$, or in some cases almost everywhere differentiable (like rectified linear nonlinearities, which are introduced below).

The simplest example of a neural network is a logistic regression. It is mistakenly called “regression”, though it solves a classification task. The model is very simple: first, the input features are linearly transformed

$$h = Wx, \tag{2.3}$$

where the matrix of parameters W has the dimensionality of $|Y| \times |X|$. Then the transformed features are put through the softmax function

$$\hat{y} = \text{softmax}(h) = \frac{\exp h}{\sum_j \exp h_j}. \tag{2.4}$$

The softmax function is a generalization of a sigmoid function

$$\sigma(h) = \frac{1}{1 + \exp(-h)}, \tag{2.5}$$

which is applicable for a scalar h and is used to represent how probable the positive class is. The $\hat{y}$ can be considered as an estimated probability distribution over the output classes. Finally, the cross-entropy with the ground truth answer is computed as

$$L(\hat{y}, y) = \sum_j y_j \ln \hat{y}_j. \tag{2.6}$$

Here we assume that the ground truth is represented as a one-hot vector, meaning that y is a vector with a number of elements equals to a number of classes containing zeros in all positions except for a one in a position of the ground truth class.

The neural network terminology often uses layers as building blocks, in the multiclass logistic regression example there is only two layers: a linear layer and a softmax layer. Often a linear and a succeeding nonlinearity are referred as a single layer. More complicated networks stack more layers on top of each other.

Feed-forward networks or multi-layer perceptrons (MLP) are very similar to the logistic regression but include more layers of linear transformation followed by nonlinearities before applying the final softmax layer. Figure 2.2 demonstrates a computation flow graph for a two layer MLP. More complicated models may involve more complicated computation flow and the requirement is that the computation graph should be a directed acyclic graph (DAG). In this case the output can be computed given inputs. The more layers network has the *deeper* it is, the effect of the depth on the ability to represent diverse function and the regularization will be discussed below.

The nonlinearities is an essential part of deep neural networks since stacking more than one linear layer is equivalent to having a single one. Historically, a sigmoid nonlinearity was very popular due to several factors, it can be interpreted as a probability of having some hidden feature; it is biologically plausible in some sense; the sigmoid networks can pre-trained in layer-wise manner using restricted Boltzmann machines (RBM) with binary units. Nowadays, the importance of pretraining it not so high due to development of computational resources such as graphical processing units (GPU), availability of the optimized implementations of linear algebra operations (cuBLAS, cuDNN), and innovations in the optimization which will be discussed in the Section 2.2.1. These and many other factors resulted into number of research papers on investigation of different kinds of nonlinearities. The most frequently used ones are

- Sigmoid element-wise nonlinearity is difficult to train due to saturation effects.

- It is used to ensure that the output is constrained to be in the region $[0,1]$ or to model the Bernoulli distribution

$$g(x) = \sigma(x). \tag{2.7}$$

- Rectified linear units (ReLU) are successfully used in the context of convolutional neural networks (CNN) for image processing tasks and fully connected networks, but there is some work concerning applications of ReLUs for RNNs which concludes that a careful initialization is needed

$$g(x) = \max\{0, x\}. \tag{2.8}$$

- MaxOut units [48] is a generalization of rectified units and basically separate the output space into several linear regions

$$g_i(x) = \max_{j \in [1,k]} \{x^T W_{\cdot ij} + b_{ij}\} \tag{2.9}$$

- The hyperbolic tangent (tanh) nonlinearity mostly used for the recurrent neural networks, see Section 2.2.4

$$g(x) = \tanh x. \tag{2.10}$$

Constructing deeper networks help learning better representations and many studies show that lower layers learn simple features like linear edge detectors when higher layers combine the features received from previous layers and construct more complex representations like circle detectors and even face detectors or animal detectors [83]. This is shown[1] for a toy task in Figure 2.3. The task is two class separation where the fist class data points are sampled uniformly in the inner circle and the second class is sampled in the ring outside the circle. This task cannot be solved using linear classifiers since there is no linear boundary separating the classes and here we used a two layer MLP with 4 hidden units in each layer and tanh nonlinearities everywhere. The graph shows the decision boundary reprojected to the input space for every hidden neuron and the final decision boundary for the whole classifier. It is clear that the first layer (Figure 2.3a) learns simple linear boundaries, the second layer (Figure 2.3b) learns more complicated representations with simple curves, and the softmax output (Figure 2.3c) is able to combine it to produce the high quality decision boundary.

This makes deep neural networks to become a step towards end-to-end learning since the feature representations are trained from data and no handcrafting is required.

2.2.1 Optimization

In this section we briefly discuss optimization with focus for neural networks.

The neural networks were developed to be easily optimized using gradient methods, the main idea is using the chain rule

$$\frac{\partial f_i(g(x))}{\partial x_j} = \sum_k \frac{\partial f_i(g(x))}{\partial g_k(x)} \frac{\partial g_k(x)}{\partial x_j}, \tag{2.11}$$

[1] Special credits to the tensorflow team and the tensorflow playground: https://playground.tensorflow.org/ which was used to produce these images.

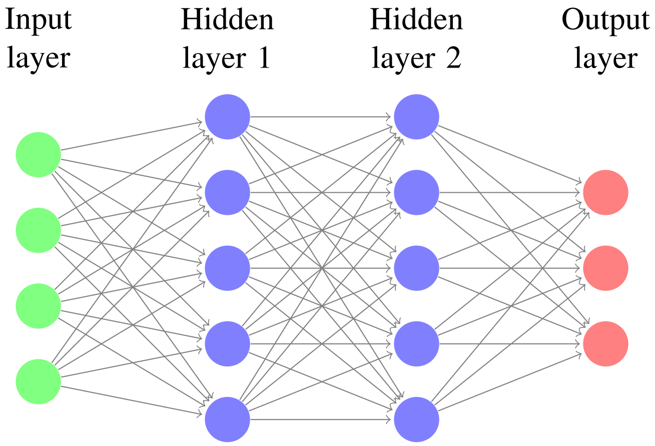


Figure 2.2: Example of a two hidden layer neural network. The input size is 4, both hidden layers have size 5 and the output is 3 (which is the number of output classes). This is a computation flow directed acyclic graph (DAG). More complicated neural network models may have more complicated computation flow as far as it can be represented with and a graph without cycles.

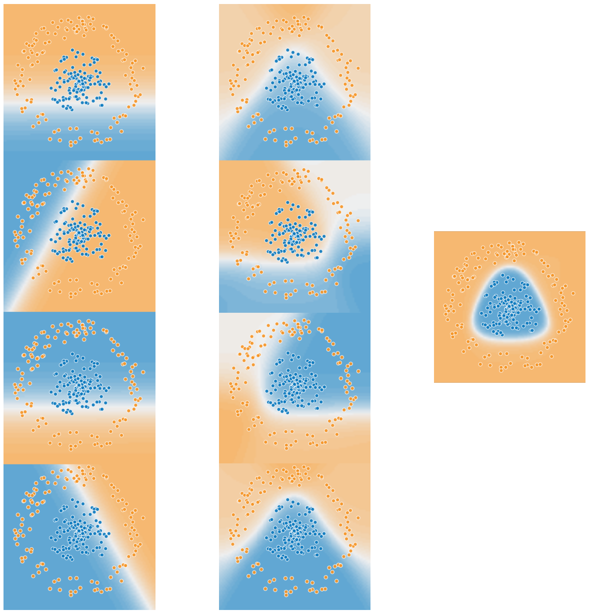
where f and g are vector functions of a vector argument x. Since a neural network is a composition of many differentiable functions the chain rule can be applied for every composition recursively. The efficient algorithm to compute the gradient is an obvious case of dynamic programming: during forward pass we compute all the activations (intermediate function results) then in a backward pass we compute the gradient using the chain rule and the stored activation from the forward pass. Notice, that this algorithm can be applied to a matrix input x, where the matrix is the batch of inputs.

Now, we are able to compute the gradient and the easiest way to optimize a network is using stochastic gradient descent (SGD). A mini-batch of examples is sampled from the training set, the gradient of the loss is computed using the back-propagation algorithm and the parameters are updated as

$$\theta_i^{(t+1)} = \theta_i^{(t)} - \alpha \frac{\partial L(\hat{y}, y)}{\partial \theta_i^{(t)}}, \tag{2.12}$$

where α is the learning rate.

There exist more sophisticated optimization methods which estimate the Hessian diagonal such as RMSProp, AdaGrad, ADADELTA [173], Adam [78].



(a) Layer 1. (b) Layer 2. (c) Final.

Figure 2.3: Decision boundaries for a two layer network trained on a toy data. Points from the first class are surrounded by points from the other class which makes this task impossible to solve using linear classifiers but can be easily solved after around 200 iterations of the gradient descend.

2.2.2 Regularization

The overfitting problem discussed in the Section 2.1.2 requires to use regularization to reduce the capacity of the model.

One of the most successful methods for neural network models is dropout. It randomly turns off neurons of the network with some constant probability, the common choice is 50% for the hidden neurons and 80% for the input ones, but it may be task dependent. The dropout prevents co-adaptation of the neurons and enforces over-representation making the network more robust to small perturbations of the input.

In our work we also used a weight matrix norm constraint regularization method. This was crucial to use it sophisticated recurrent models (Section 2.2.4) for large vocabulary speech recognition models reported in the Section 2.4.

The weight noise, and in particular, the adaptive weight noise [50] is a method which uses the variational inference ideas for regularizing recurrent neural networks was used to achieve the state of the art on small speech corpora like TIMIT. This method sets up the Gaussian prior over the weight matrices elements and performs sampling. In other words, random Gaussian noise is added to the weights. In the case of the adaptive weight noise, the mean and the variance of the Gaussian are trainable parameters.

2.2.3 Autoregressive Models

Distributions modelling complex real world systems generally require a lot of factors. When every such factor corresponds to a variable in the distribution, the dimensionality of the input is substantial. This especially true for streams of data, strings, time series. These structures are potentially infinite therefore they require infinite dimensional input. However, an intuition tells us that every observation in the series does not depend on the future observations. In other words, when modelling a given observation, we need to take into consideration only previous observations. Models that work this way are called autoregressive models.

Putting into math the intuition in the previous paragraph, we look at a distribution of a sequence of random variables $y_1, \dots, y_T$. Any multi-variable distribution $p(y_1, \dots, y_T)$ can be factorized as

$$p(y_1, \dots, y_T) = p(y_1)p(y_2|y_1)\dots p(y_T|y_1, \dots, y_{T-1}) = p(y_1)\prod_{t=2}^{T} p(y_t|y_1, \dots, y_{t-1}), \tag{2.13}$$

or using notation $y_{<i} = \{y_1, \ldots, y_{i-1}\}$,

$$p(y_1, \ldots, y_T) = p(y_1) \prod_{t=2}^{T} p(y_t | y_{<t}). \tag{2.14}$$

This is also true for a model distribution $q(y_1, \ldots, y_T)$

$$q(y_1, \ldots, y_T) = q(y_1) \prod_{t=2}^{T} q(y_t | q_{<t}). \tag{2.15}$$

Writing down the Kullback-Leibler divergence between distributions p and q

$$D_{KL}(p||q) = -H(p) - \mathbb{E}_p \log q(y_1) + \sum_{t=2}^{T} \log q(y_t | q_{<t}), \tag{2.16}$$

where $H(p)$ is the entropy of the data distribution p that is constant, therefore does not affect optimization. Minimization of the Kullback-Leibler divergence corresponds to minimization of the second term which is called the cross-entropy between p and q. An important observation is that the cross-entropy of the factorized distribution is the sum of conditional cross-entropies

$$H(p,q) = H\left(p(y_1), q(y_1)\right) + \sum_{t=2}^{T} H\left(p(y_t | y_{<t}), q(y_t | y_{<t})\right). \tag{2.17}$$

In practice, it is intractable to perform the summation over all the inputs. Furthermore, data distribution p is unknown in most practical cases, we are given only samples from this distribution. To tackle these two issues, we make a single sample approximation of the expectation

$$\mathbb{E}_p \log q(y_1, \ldots, y_T) \approx \log q(y'_1, \ldots, y'_T), \tag{2.18}$$

where $y'_1, \ldots, y'_T \sim p(y_1, \ldots, y_T)$ a sample from the data distribution p. Basically, this means that we can have an estimate of the cross-entropy simply sampling a single point from the dataset. Combining this approximation with the Equation 2.17

$$H(p,q) \approx -\log q(y'_1) - \sum_{t=2}^{T} \log q(y'_t | y'_{<t}). \tag{2.19}$$

In other words, we use a sample from the dataset to guide the prediction. Then, at every t we make a one step ahead prediction. This procedure is know as *teacher* *forcing* algorithm because the sequence we condition on $y'_{<t}$ is forced to be a sample from the dataset.

When performing inference, most commonly we use ancestral sampling from the learned distribution q. In other words, we sample the y'_1 from $q(y_1)$, then every y_t from $q(y_t|y'_{<t})$, where the conditioning part notation means that the all variables before t are clamped to previously sampled $y'_1, \ldots, y'_{t-1}$.

While this is straightforward on the paper, in practice it can lead to problems. This class of problems is a kind of over-fitting (see Section 2.2.2) and is called *exposure bias* [121]. The exposure bias problem might affect any autoregressive model, but first was discovered in the context of recurrent neural networks (see Section 2.2.4). The origin of the exposure bias is that the training procedure is substantially different from the inference procedure. During the training, we see only the correct samples from the dataset, while during the inference, the model is conditioned on the samples from the model itself. Even for very accurate models the probability of making a mistake in a long sequence is growing exponentially. In other words, even if the probability of making a mistake at any given time-step is as low as ε, the probability of making a mistake in a sequence of length T is $1-(1-\varepsilon)^T$. After making a single mistake during the inference, the model can find itself in the state which it has never encountered during the training. Because of this, the model starts to wander off the familiar path and produce worse and worse mistake.

The exposure bias problem has been studied from different aspects. The proposed solutions range from approaches based on reinforcement learning based *data as demonstrator* [160], SEARN [35], *dataset aggregation* [125]. Other works apply REINFORCE algorithm [163] to sequence prediction problems [54]. A downside of the reinforcement learning approaches is that, usually, the gradient has high variance. Another set of approaches tries to bring closer the training stage to the inference. The examples are *scheduled sampling* [16], where randomly chosen tokens are sampled from the model distribution instead of the dataset; *professor forcing* [87], where a generative adversarial network is used to close the gap between the training and testing.

Besides the exposure bias, autoregressive models frequently have difficulties generating long consistent sequences. This is discussed in Section 2.2.5 in the context of recurrent neural networks.

2.2.4 Recurrent Neural Networks

For the tasks which have temporal structure we need to deal with the sequences which potentially have variable length. This type of networks is referred as the recurrent neural networks (RNNs). The simplest RNN takes the input and the previous state to produce a new state

$$\mathbf{h}_t = g(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h), \tag{2.20}$$

where g is a nonlinearity function, often a hyperbolic tangent; x_t is the input at the time step t, h_{t-1} is the previous hidden states; and $\mathbf{W}_{xh}$, $\mathbf{W}_{hh}$, $\mathbf{b}_h$ are the parameters, input-to-hidden, hidden-to-hidden matrices and the bias respectively. The RNNs can be used in two scenarios: whether for generation or for feature representation. A generative RNNs models a distribution

$$p(y_1, \dots, y_T) = p(y_1)p(y_2|y_1)\cdots p(y_T|y_1, \dots, y_{T-1}) \tag{2.21}$$

aggregating the information required for the conditioning in the state variable. The second scenario is a feature representation for the temporal information. In this case the representation and the output is the tensor of the hidden states.

In both these cases the function computed by the RNN is differentiable, so the derivative with respect to the parameters can be computed using the backpropagation algorithm on a unfolded RNN, this algorithm is referred as *back-propagation through time* (BPTT).

2.2.5 Long Short-Term Memory recurrent networks

A particularly successful recurrent neural network architecture is the Long Short-Term Memory (LSTM) [71]. The LSTM is designed to handle long-term dependencies by gating the information that enters and leaves its so-called memory cells using multiplicative gating units. The hidden states of a commonly used LSTM variant [52] are computed using the following set of equations:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{W}_{ci}\mathbf{c}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{W}_{cf}\mathbf{c}_{t-1} + \mathbf{b}_f) \\
\mathbf{c}_t &= \mathbf{f}_t \otimes \mathbf{c}_{t-1} + \mathbf{i}_t \otimes \tanh(\mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{W}_{co}\mathbf{c}_t + \mathbf{b}_f) \\
\mathbf{h}_t &= \mathbf{o}_t \otimes \tanh(\mathbf{c}_t)
\end{aligned}$$

where **i**, **f** and **o** represent the input, forget and output gates and **c** contains the values of the memory cells. The symbol $\otimes$ signifies element-wise multiplication. The matrices $\mathbf{W}_{ci}$, $\mathbf{W}_{cf}$ and $\mathbf{W}_{co}$ are constrained to be diagonal.

Another variant is Gated Recurrent Units (GRU, [33]) which was designed to be more computationally efficient than LSTM units as it has a simpler architecture [31]. The hidden states $\mathbf{h}_t$ are computed using the following equations:

$$\begin{aligned}
\mathbf{z}_t &= \sigma(\mathbf{W}_{xz}\mathbf{x}_t + \mathbf{U}_{hz}\mathbf{h}_{t-1}), \\
\mathbf{r}_t &= \sigma\left(\mathbf{W}_{xr}\mathbf{x}_t + \mathbf{U}_{hr}\mathbf{h}_{t-1}\right), \\
\tilde{\mathbf{h}}_t &= \tanh\left(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{U}_{rh}(\mathbf{r}_t \otimes \mathbf{h}_{t-1})\right), \\
\mathbf{h}_t &= (1 - \mathbf{z}_t)\mathbf{h}_{t-1} + \mathbf{z}_t\tilde{\mathbf{h}}_t,
\end{aligned}$$

The networks introduced above can only aggregate the information in one direction. Sometimes the features at time step t depend on the future information and for this purpose two recurrent networks can be run in opposite directions and their states concatenated. This type of recurrent networks is often referred as the *bidirectional* RNNs (BiRNN). Obviously, not only can simple RNNs be bidirectional, one can construct bidirectional LSTMs and GRUs using their hidden states.

Frequently, one more trick is used to obtain a better representation: stacking several RNNs on top of each other like it is done in deep neural networks. It has been shown that one can pass the data through several layers of RNNs to obtain better performance for speech recognition in [56]. It is done simply considering the output sequence $h_1, \ldots, h_T$ of the states of the first layer as the input to the second one. The Figure 2.4 illustrates two simple bidirectional networks stacked on top of each other.

2.2.6 Sequence Generation with Recurrent Neural Networks

The tasks like speech recognition involve inputs and outputs having variable length. Therefore the input should be *aligned* to the output, which is a complicated chicken-and-egg type of learning problem. The model may need a good alignment to produce good classification results but there is no way to figure the alignment without a classification model. For speech recognition models the Hidden Markov Model is usually used and will be discussed in the Section 2.3.1.

The idea to train the model to align and perform its task simultaneously originated to the machine translation community in a context of encoder-decoder networks [27, 150]. The encoder network is used to generate an intermediate rep-

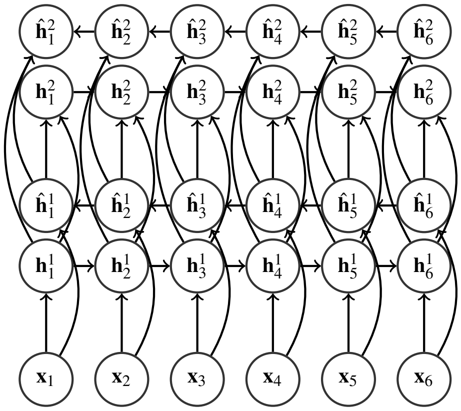


Figure 2.4: Two Bidirectional Recurrent Neural Networks stacked on top of each other.

resentation which is passed to the decoder network which is typically an RNN working as a generative one as described in Section 2.2.4.

A simple encoder-decoder is supposed to save a whole input sequence to a fixed dimension vector to pass it to the decoder. This is not feasible for some applications having long sequences as the handwritten text generating. This was addressed in [52] and a kind of attention was proposed which used several Gaussian windows averaging over the time dimension to select the information needed for every generation step.

The Attention-based Recurrent Sequence Generators (ARSG [8]) uses a more general approach to the attention. The ARSG produces the output sequence $y_1, \ldots, y_T$ one element at a time using the state information and simultaneously aligning the generated element to the input sequence $h_1, \ldots, h_L$ produced by the encoder. The encoder usually is a stack of bidirectional recurrent networks (LSTMs or GRUs). The elements are "selected" from the input sequence like

$$\mathbf{c}_t = \sum_l \alpha_{tl} \mathbf{h}_l, \tag{2.22}$$

where the α_{tl} are the attention weights produced by an attention mechanism. See Figure 2.5 for a schematic representation of an ARSG.

A particular attention mechanism used throughout most of our works is work-

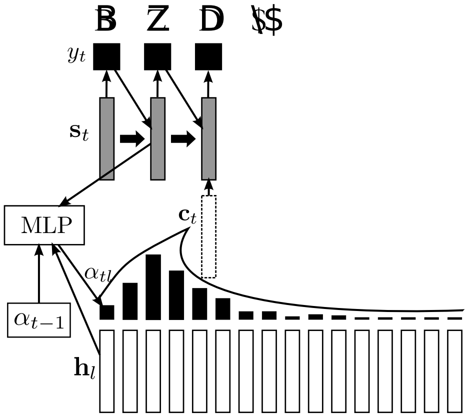


Figure 2.5: A schematic illustration of an attention mechanism network. Every frame in the encoded input h_l is weighted by an MLP which depends on this frame, previous hidden state of the generator and the previous time step attention weights. This makes this particular attention content and position based attention. The output is sampled conditioned on the ground truth previous character during training and on the previously sampled character during testing.

ing as follows

$$\mathbf{F} = \mathbf{Q} * \alpha_{t-1} \tag{2.23}$$

$$e_{tl} = \mathbf{w}^\top \tanh(\mathbf{W}\mathbf{s}_{t-1} + \mathbf{V}\mathbf{h}_l + \mathbf{U}\mathbf{f}_l + \mathbf{b}) \tag{2.24}$$

$$\alpha_{tl} = \frac{\exp(e_{tl})}{\sum\limits_{l=1}^{L} \exp(e_{tl})}. \tag{2.25}$$

where **W**, **V**, **U**, **Q** are parameter matrices, **w** and **b** are parameter vectors, $*$ denotes convolution, $\mathbf{s}_{t-1}$ stands for the previous state of the RNN component of the ARSG. We explain how it works starting from the end: (2.25) shows how the weights α_{tl} are obtained by normalizing the scores e_{tl}. As illustrated by (2.24), the score depends on the previous state $\mathbf{s}_{t-1}$, the content in the respective location $\mathbf{h}_l$ and the vector of so-called convolutional features $\mathbf{f}_l$. The name “convolutional” comes from the convolution along the time axis used in (2.23) to compute the matrix **F** that comprises all feature vectors $\mathbf{f}_l$.

Different types of attention were used for machine translation [8], caption generation [164] and phoneme-level speech recognition [29].

2.3 Automatic Speech Recognition

This section briefly summarizes previous work on *automatic speech recognition* (ASR). We start with description of the hybrid speech recognizes in Section 2.3.1 and continue with the discussion of the end-to-end systems in the Section 2.3.2.

2.3.1 Hybrid Systems

Widely used type of ASR systems is hybrid speech recognition systems, they combine several components into one pipeline and every component is optimized separately [41]. First step, feature extraction, is performed using signal processing techniques is discussed in Section 2.3.1. The processed features are the inputs for the acoustic model represented by some kind of a phone classifier (Gaussian Mixture model, deep neural network [67], or recurrent neural network [56]), it is trained to assign a phone for every time frame. It is discussed more in the Section 2.3.1. Then a pronunciation model which is frequently a hidden Markov model is trained to transform the phonemes to the output sequence, it is explained in the Section 2.3.1. Finally, the acoustic model is combined with an external language model, see Section 2.3.1.

An ASR system is modelling the probability of a sequence of words $y_1, \dots, y_L$ given an input sequence of acoustic vectors $x_1, \dots, x_T$ [21] which can be factorized in the following way

$$p(y|x) \propto p(x|y)p(y) = p(x|s)p(s|y)p(y), \tag{2.26}$$

where s is the pronunciation (or sounding). And in order to solve the speech recognition task one has to find the most probable sequence of outputs

$$\hat{y} = \arg\max_{y,s} p(x|s)p(s|y)p(y), \tag{2.27}$$

maximizing over all possible pronunciation and the outputs. Three multiplier at the right hand side represent three main parts of an ASR system $p(y)$ is the language model; $p(x|s)$ is the pronunciation model and $p(x|s)$ is the acoustic model.

Feature Extraction

The most popular way to preform the feature processing of the raw audio signal is to extract so-called mel-frequency cepstral coefficients or MFCCs and its first and second derivatives. MFCCs can be computed using following procedure [1]:

- First, the Fourier transform of the raw signal divided into intersecting windows is taken.
- The powers of the spectrum obtained after the Fourier transform is mapped onto the mel scale using triangular overlapping windows.
- The logarithms of the powers at each of the mel frequencies are computed.
- The discrete cosine transform of the list of mel log powers is performed, as if it were a signal.
- The MFCCs are computed as the amplitudes of the resulting spectrum.

Phoneme Recognition

The acoustic model perhaps is the hardest part in terms of training difficulty of the ASR system. The initial alignment is obtained running Baum-Welch algorithm with the GMM-HMM model (the Hidden Markov models for pronunciation are discussed in Section 2.3.1). This algorithm is a particular case of the expectation-maximization (EM) algorithm. Then a more complicated DNN or RNN acoustic model is trained with this pre-trained alignment.

The acoustic model is trained to model the $p(x|s)$ distribution from the Equation (2.27). And the pronunciation is supposed to consist from the phones (atomic sounds in the language), which construct phonemes, constrained by the lexicon.

Hidden Markov Models

A popular for many tasks hidden Markov model is used for the pronunciation model in hybrid ASR systems. This model is basically is stochastic finite automaton and received its name due to that the observable stochastic process is modelled with an assumption of having some hidden states. The inference in the HMM is performed using the efficient dynamic programming Viterbi algorithm.

Hidden Markov models used in ASR are used in a concatenation of several final state transducers (FST). First, the context-dependency transducer is constructed. This FST is concatenated with the hidden Markov model, then with the pronunciation FST which maps the words to their pronunciations. This FST

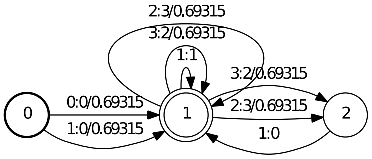


Figure 2.6: A simple weighted finite state transducer. It has three states, 0 is the initial state and 1 is the accepting state. Both the input and the output has the vocabulary of $\{0, 1, 2, 3\}$. Every transition is marked as input:output/weight.

is concatenated with the grammar transducer (or the language model, see Section 2.3.1). The hidden Markov model is trained to map from the transition-ids to the context-dependent frames.

Language Models

Most commonly used language models in ASR systems are N-gram language models, although the state of the art was achieved with a deep RNN language model in many tasks [100].

An efficient implementation of the N-gram language models uses the *weighted finite state transducers* (wFST; [3, 102]) and it makes the N-gram LM compatible with the pronunciation HMM represented as a stochastic finite state automaton. See Figure 2.6 for an example of a simple wFST. These two models can be combined using simple wFST operation of concatenation. Other standard operations are used to minimize the resulting transducer and to push the weights toward the initial state to help the beam search.

In the case when the RNN LM is used it is not possible to use the wFST machinery to get the concatenated acoustic and the LM transducer. A common approach in this case and sometimes with the N-gram models as well is to produce a *lattice* and re-weight it with the LM. Basically, the lattice is the augmented result of the beam search algorithm run on the decoding wFST, it is the set of the paths obtained from the beam search after merging the same states.

2.3.2 End-to-end Systems

End-to-end systems recently proposed for the ASR. The connectionist temporal classification [53] uses the dynamic programming algorithm to construct a differentiable loss which integrates out the alignment of the input and the output.

This is performed adding a new token to the output vocabulary for the blank output and this token is ignored when constructing the output. Later work [54] uses the same CTC cost to construct a model which optimizes the task loss again, using the dynamic programming and a type of the REINFORCE algorithm. One of the problems of the CTC-based models that it does not learn the internal language model due to the absence of the output RNN. This was solved with the neural transducers [51] which use the same idea of dynamic programming but include the output RNN. The disadvantage of the neural transducer is that it is computationally expensive to train.

2.4 End-to-end Large Vocabulary Speech Recognition

This section summarizes work on adaptation of attention-based architectures which brings us closer to the end-to-end speech recognition on large vocabulary datasets, published in [9]. This model is one of the crucial components for the experiments outlined in Chapter 4 and an inspiration for the work presented in Chapter 5.

2.4.1 Introduction

Several successful works on end-to-end large vocabulary automatic speech recognition include [98, 64, 63], which used CTC-based architectures for LVSR tasks like Wall Street Journal dataset and Switchboard dataset.

The work [9] builds up on the works [28, 29] on investigation ARSGs 2.2.6 for speech recognition and provides experimentation on a bigger Wall Street Journal speech corpora. The tree main contributions of this work are the following. First, the paper shows how training on long sequences can be made feasible by limiting the area explored by the attention to a range of most promising locations. This reduces the total training complexity from quadratic to linear, largely solving the scalability issue of the approach. This has already been proposed [29] under the name “windowing”, but was used only at the decoding stage in that work. Second, in the spirit of the Clockwork RNN [82] and hierarchical gating RNN [33], the paper introduces a recurrent architecture that successively reduces source sequence length by pooling frames neighboring in time. This mechanism has been independently proposed in [23].

Finally, the paper shows how a character-level ARSG 2.2.6 and $N-$gram word-level language model can be combined into a complete system using the *weighted finite finite transducers* (wFST) framework.

2.4.2 Model for Large Vocabulary Speech Recognition

The work uses an attention-based sequence generation model introduced in the Section 2.2.6. The encoder network is a multilayer bidirectional RNN (see Section 2.2.5) and uses one more trick to speed up the computation: pooling along the time dimension. In a nutshell, the paper provides only every k frames to the succeeding layer of a recurrent network, see the Figure 2.7 for the details. In the

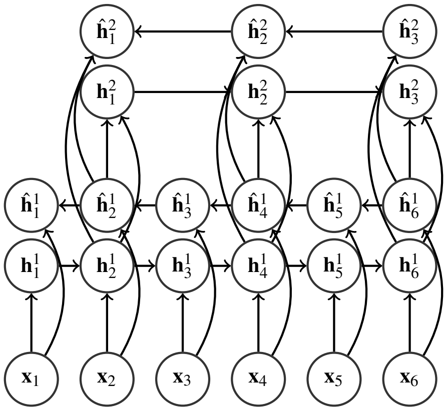


Figure 2.7: A pooling over time BiRNN: the upper layer runs twice slower then the lower one. It can average, or subsample (as shown in the figure) the hidden states of the layer below it.

experiments they choose k equals 2 and the performance does not change much comparing to the full version without pooling.

The decoder is a single layer GRU network equipped with an attention mechanism described in the Section 2.2.6. It was crucial to use the norm constrain regularization which also happened to help optimization preventing weights from blowing up.

2.4.3 Integration with a Language Model

The paper shows hot to combine the attention-based acoustic model with the n-gram language model. Such language models are provided with Wall Street Journal dataset. It is also possible to use a stronger language model trained with an external corpus. The paper innovatively converts the language model into a *weighted finite state transducer* (wFST). Then, the paper proposes to concatenate the obtained wFST with a spelling transducer 2.3.1 and to use the wFST machinery to minimize and determinize the final transducer. The spelling transducer is a simple transducer which spells out letter-by-letter every word from the vocabulary. To simplify the work for the beam search the paper proposes to push the weights towards the initial state of the obtained transducer.

Using this combined wFST, the paper uses the beam search for decoding. The loss minimized is

$$L = -\log p_{ED}(y|x) - \beta \log p_{LM}(y) - \gamma T, \tag{2.28}$$

where y is the transcript, p_{ED} is the encoder-decoder cost (acoustic model cost), p_{LM} is the language model cost obtained with the wFST. The last term prevents the network from generating too short sequences. This criterion was also proposed in [64]. The hyper-parameters β and γ are tuned on the validation set.

The experiments were conducted on the Wall Street Journal (WSJ) corpus (available at the Linguistic Data Consortium as LDC93S6B and LDC94S13B). The paper used 123 dimentional features that consist of 40 mel-scale filter-bank coefficients with the engery combined with their first and second order derivatives. The paper used the same text pre-processing as in [64]. The 32 distinct labels left are: 26 characters, apostrophe, period, dash, space, noise and end-of-sequence tokens.

The precise architecture used for the encoder network consisted of 4 layers of 250 forward and 250 backward GRU units. The top two layers used the temporal subsampling of 2 (see Figure 2.7). Therefore, the encoder reduced the utterance length by the factor of 4. A centered convolution filter of width 200 was used in the attention mechanism to extract a single feature from the previous step alignment as described in the Section 2.2.6.

The AdaDelta algorithm [173] with gradient clipping was used for optimization.

The intial alignment was initialized as described above. After that the training was restarted with the windowing described in the same section. The window parameters were $w_L = w_R = 100$, which corresponds to considering a large 8 second long span of audio data at each step, taking into account the pooling done between layers. Training with the AdaDelta hyper-parameters $\rho = 0.95$, $\varepsilon = 10^{-8}$ was continued until log-likelihood stopped improving. Finally, the best model was annealed in terms of log-likelihood by restarting the training with $\varepsilon = 10^{-10}$.

We found regularization necessary for the best performance. The column norm constraint 1 was imposed on all weight matrices [69]. This corresponds to constraining the norm of the weights of all the connections incoming to a unit.

As explained above, the paper used the beam search to minimize the combined cost L defined by (2.28). A sequence was considered terminated when it ended with the special end-of-sequence token, which the network was trained to generate in the end of each transcript.

In order to compare with the model that will be introduced in Chapter 4, it is important to summarize the experimental results from [9]. These results are gathered in Table 2.1. The proposed model outperforms the CTC 2.3.2 systems when no external language model is used. The improvement from adding an external language model is however much larger for CTC-based systems. The final

Table 2.1: Character Error Rate (CER) and Word Error Rate (WER) scores for our setup on the Wall Street Journal Corpus in comparison with other results from the literature. Note that our results are not directly comparable with those of networks predicting phonemes instead of characters, since phonemes are easier targets.

Model	CER%	WER%
Encoder-Decoder	6.7	19.3
Encoder-Decoder + bi-gram LM	5.4	13.0
Encoder-Decoder + trigram LM	4.8	11.3
Graves and Jaitly (2014)		
CTC	9.2	30.1
CTC, expected transcription loss	8.4	27.3
Hannun et al. (2014)		
CTC	10.0	35.8
CTC + bi-gram LM	5.7	14.1
Miao et al. (2015),		
CTC for phonemes + lexicon	-	26.9
CTC for phonemes + trigram LM	-	7.3
CTC + trigram LM	-	9.0

performance of the proposed model is better than the one reported in [64] (13.0% vs 14.1%), but worse than the the one from [98] (11.3% vs 9.0%) when the same language models are used.

Chapter 3

Adversarial Training of Invariant Features for Speech Recognition

The following section presents paper D. Serdyuk, K. Audhkhasi, P. Brakel, B. Ramabhadran, S. Thomas, and Y. Bengio. Invariant representations for noisy speech recognition. *Neural Information Processing Systems End-to-end Learning for Speech and Audio Processing Workshop*, 2016.

3.1 Context

Recent advances in domain adaptation allow us to construct networks that are invariant to certain labeled factors. The reverse gradient algorithm uses an adversarial learning approach to train a network to produce the desired label as well as to deceive a second classifier that is trained on adaptation labels. This is tightly connected to the ideas behind generative adversarial networks (GANs). Therefore, some insights for training GANs are needed for the reverse gradient's successful application. We adapt this approach to train an acoustic system that is invariant to undesirable factors such as the recording environment noise type. Our experiments show small improvement on the Aurora-4 dataset. We explore this approach further and conclude that the adversarial training is beneficial in the case of unseen conditions during evaluation.

Unsupervised learning of disentangled and interpretable representations from sequential data A work [72] explores Bayesian ways to help learning disentangled representation from sequential data. In the contrast to GAN-like approach taken in the above work, [72] uses *variational auto-encoder* (VAE, [79]) machinery for approximate learning of the features. The authors evaluate their approach on two datasets: TIMIT [43] and Aurora-4 [113]. The improvements on both datasets show that the VAE-style adaptation is a viable way to learn disentangled representations.

More specifically, the work proposes several different encoder-decoder architectures, where the best one behaves as follows. Two latent variables summarize the input using a stochastic recurrent network. Then, the input is reconstructed back from the latent variables by another stochastic neural network.

One of the downsides of training a VAE is that a decoder is needed. An accurate decoder for speech signal is usually complicated because it requires to synthesize speech. The task of speech synthesis is known to be hard due to extremely long sequences and complex long term interactions. Therefore, the decoder is usually generates some simplified feature of the signal.

Adversarial multi-task learning of deep neural networks for robust speech recognition In this work [145] apply adversarial learning domain adaptation to an in-house speech recognition dataset which is based on the Wall Street Journal [116] corpus corrupted by noise. The work demonstrated that the system is able to adapt to eight new noises and various SNR levels.

Data augmentation Another tangent way of dealing with noise in speech signal is data augmentation. Several approaches emerged in recent years. One notable work [81] uses point noise sources to augment their training data. Another prominent work [114] corrupts the spectrum with artifacts used in computer vision literature, such as masking rectangular blocks in time-frequency domain, and time warping.

3.2 Introduction

One of the most challenging aspects of *automatic speech recognition* (ASR) is the mismatch between the training and testing acoustic conditions. During testing, a system may encounter new recording conditions, microphone types, speakers, accents and types of background noises. Furthermore, even if the test scenarios are seen during training, there can be significant variability in their statistics. Thus, it's important to develop ASR systems that are invariant to unseen acoustic conditions.

Several model and feature based adaptation methods such as Maximum Likelihood Linear Regression (MLLR), feature-based MLLR and i-vectors [133] have been proposed to handle speaker variability; and Noise Adaptive Training (NAT; [74]) and Vector Taylor Series (VTS; [156]) to handle environment variability. With the increasing success of Deep Neural Network (DNN) acoustic models for ASR investigated in [67, 136, 127], and in more recent works [98, 128] complicated structure of acoustic conditions is modeled within a single network. This allows us to take advantage of the network's ability to learn highly non-linear feature transformations, with greater flexibility in constructing training objective functions that promote learning of noise invariant representations. At the same time, the more complex relation between the parameters of deep neural networks and the functions or distributions they represent, makes it impossible to simply use the same types of adaptation and robustness methods that have been developed for GMM-based systems.

The main idea[1] of this work is to force the acoustic model to learn a representation which is invariant to noise conditions, instead of explicitly using noise robust acoustic features (Section 3.5), similar to [146]. This type of noise-invariant training requires noise-condition labels during training only. It is related to the idea of generative adversarial networks (GAN) and the gradient reverse method proposed by [47] and [42] respectively (discussed with other related work in Section 3.4). We connect the GAN training and the gradient reverse method for domain adaptation in Section 3.3. Contributions of this paper are following:

- We use the insights from GAN literature to improve domain adaptation for noise robust training on Aurora-4 speech dataset (Section 3.6);

- We explore the boundaries of applicability of the adversarial training using different subsets of noise conditions. We conclude that this approach is beneficial when the evaluation set contains unseen noise conditions.

3.3 Background

Generative adversarial networks are data generating neural networks that have been especially successful at producing small images. Unlike most generative models, they are not trained by optimizing the likelihood of the data, but by the

[1] This work is a continuation of our preliminary results presented as a workshop paper [140].

optimization of a so-called *adversarial* objective. The training of generative adversarial networks involves two neural networks: the generator and the discriminator.

The generator network G has an input of randomly-generated feature vectors $\boldsymbol{z}$ and is asked to produce a sample $\boldsymbol{x}$, e.g. an image, similar to the images in the training set. The discriminator network D can either receive a generated image from the generator G or an image from the training set. Its task is to distinguish between the "counterfeit" generated image and the "genuine" image taken from the dataset. Thus, the discriminator is just a classifier network with a sigmoid output layer and can be trained with gradient back-propagation. This gradient can be propagated further to the generator network (assuming that the output of the generator is continuous).

The two networks in the GAN setup are competing with each other: the generator is trying to deceive the discriminator network, while the discriminator tries to do its best to recognize if there was a deception, similar to adversarial game-theoretic settings. Formally, the objective function of GAN training is $\min_G \max_D V(D,G)$ with

$$\begin{aligned} V(D,G) = &\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \\ &\qquad \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]. \end{aligned} \tag{3.1}$$

The maximization over the discriminator D forms a usual cross-entropy objective, the gradients are computed with respect to the parameters of D. An important property of this objective is that the gradient of the composition $D(G(\cdot))$ is well defined. Therefore the generator G can be trained with the back-propagation algorithm using the gradient signal from the discriminator D. The generator G is minimizing the classification objective using the gradients propagated through the second term. The minimization over G makes it produce samples which D classifies as originating from the train data.

Several practical guidelines were proposed for optimizing GANs by [118] and further explored by [130]. The second term of Eq. 3.1 is frequently exchanged with a term $-\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(D(G(\boldsymbol{z})))]$. The gradient of this term has the same sign but it has better properties in some cases. We experimented with several losses for our discriminator and chose one similar to Eq. 3.1 but we maintain the negative sign in our notation to demonstrate the general direction of the gradient of the discriminator loss.

Optimization of the GAN objective corresponds to a minimization of the Jensen-Shannon divergence between the data distribution and the distribution represented by the generator. This divergence is zero if and only if the two distributions are identical. From this perspective, we can say that the discriminator provides a training signal which allows one to match distributions in general. In this work we are interested in matching feature distributions of different noise conditions.

Prior work by [42] proposed a method for training a network which can be adapted to new domains. The training data consists of the images labeled with classes of interest and separate domain (image background) labels. The network has a Y-shaped structure: the image is fed to the first network which produces a hidden representation h. Then this representation h is input to two separate networks: a domain classifier network (D) and a target classifier network (R). The goal of training is to learn a hidden representation that is invariant to the domain labels and still performs well on the target classification task, so that the domain information doesn't interfere with the target classifier at test time. Similar to the GAN objective, which forces the generation distribution be close to the data distribution, the *gradient reverse method* makes domain specific distributions similar to each other.

The network is trained with three goals: the hidden representation h should be helpful for the target classifier, harmful for the domain classifier, and the domain classifier should have a good classification accuracy. More formally, the authors define the loss function as

$$L = L_1(\hat{y}, y; \theta_R, \theta_E) + \alpha L_2(\hat{d}, d; \theta_D) - \beta L_3(\hat{d}, d; \theta_E), \tag{3.2}$$

where y is the ground truth class, d is the domain label, corresponding hat variables are the network predictions, and θ_E, θ_R and θ_D are the subsets of parameters for the encoder, recognizer and the domain classifier networks respectively, the L_* are the costs that are discussed in detail in Section 3.5. The hyper-parameters α and β denote the relative influence of the loss functions terms.

3.4 Related Work

Neural networks display some robustness towards different recording conditions and speaker by themselves and the effectiveness of representations produced by a neural network for internal noise reduction is discussed by [170]. This work sets a baseline for experiments on the Aurora-4 dataset.

To be even more robust with respect to different recording conditions and speakers, one can either aim to adapt the model parameters to these new situations or to learn representations which are invariant to them. Most approaches so far, are based on adaptation. Unfortunately, many of the adaptation methods which have been designed for GMM-HMM systems cannot be applied to DNN-based systems. For this reason, a large body of recent work has investigated new adaptation methods for neural networks.

Some of the linear and affine transformations used for GMM adaptation can still be applied to neural networks when one limits these adaptations to the very last layer, as was shown in [166]. The improvements seemed to rely mostly on the adaptation of the bias parameters. A clear downside of this approach is that it doesn't utilize the representational power provided by the non-linear multi-layer structure of deep neural networks.

Many neural network speaker adaptation methods are based on i-vectors. The most common way to exploit i-vectors is by providing them as additional inputs [138, 133]. This means that i-vectors also need to be available during testing and this may not always be practical.

In the specific case of noise robustness, one can use speech enhancement to obtain more robust features and neural networks have been used for this for decades [80]. A further step in this direction is to train systems to recognize speech and perform speech enhancement jointly [106]. We argue that speech enhancement is a more difficult task to learn than invariance to certain noise conditions and more likely to suffer from overfitting when the amount of available noisy data is limited. Another downside of enhancement approaches is that there is no reason to expect them to generalize well to noise conditions that were not available during training.

Another adaptation strategy is to retrain the model parameters using a very small number of utterances from the new domain while making sure that the model doesn't stray too far away from the parameter values that were obtained from the train data [171]. The adaptation to new domains can also be kept under control by limiting the number of parameters which are adapted to the new data or by limiting the adaptation to a rescaling of the hidden unit activations [151]. The most important difference between these approaches and our work is that we don't adapt the model parameters at all after training. In many situations this may not be possible and retraining of models may require more from the hardware on which the speech recognition system is implemented.

Recently, in a work by [146] a multi-layer sigmoidal network was trained in an adversarial fashion on an in-house transcription task corrupted by noise. This work is very similar to our approach but we evaluate our methods on more challenging benchmark, where we find that the method of [146] does not work out of the box. Therefore we draw parallels with the GAN literature and improve the adversarial cost. While investigating different numbers of noise conditions, our work also differs due to the use of more modern rectifier activation based classification networks. Finally, in other very recent work [132] a form of adversarial training with a network that predicts an i-vector using the mean square error loss was used.

3.5 Invariant Representations for Speech Recognition

Most ASR systems are DNN-HMM hybrid systems. The context dependent (CD) HMM states (acoustic model) are the class labels of interest. The recording conditions, speaker identity, or gender represent the domains in GANs. The task is to make the hidden layer representations of the HMM state classifier network invariant with respect to these domains. We hypothesize that this adversarial method of training helps the HMM state classifier to generalize better to unseen domain conditions and requires only a small additional amount of supervision, i.e., the domain labels.

Figure 3.1 depicts the model, which is same as the model for the gradient reverse method. It is a feed-forward neural network trained to predict the CD-HMM state, with a branch that predicts the domain (noise condition). This branch is discarded in the testing phase. In our experiments we used the noise condition as the domain label, merging all noise types into one label and defining ‘clean’ as the other label. Our training loss function is

$$\begin{aligned} L = L_1(\hat{y}, y; \theta_R, \theta_E) + \alpha L_2(\hat{d}, d; \theta_D) - \\ \beta[d \log(1-\hat{d}) + (1-d)\log(\hat{d})], \end{aligned} \tag{3.3}$$

where d is the domain label and $\hat{d}$ is the predicted domain. The L_3 term from Eq. 3.2 is defined according to the standard GAN objective for stability during training. This term maximizes the probability of an incorrect domain classification in contrast to the gradient reverse where the correct classification is minimized. The terms L_1 and L_2 are regular cross-entropies which are minimized with corresponding parameters θ_E and θ_D. For simplicity, we use only a single hyper-parameter – the weight of the third term.

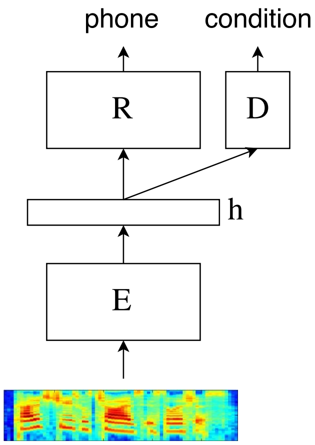
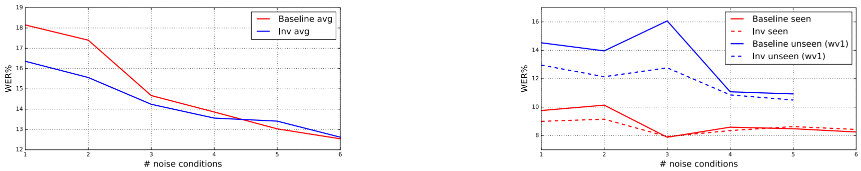


Figure 3.1: The model consists of three neural networks. The encoder E produces the intermediate representation h which used in the recognizer R and in the domain discriminator D. The hidden representation h is trained to improve the recognition and minimize the domain discriminator accuracy. The domain discriminator is a classifier trained to maximize its accuracy on the noise type classification task.



(a)

(b)

Figure 3.2: **(a)** Average performance of the baseline multi-condition and invariance model varying with the number of noise conditions used for training. **(b)** Average performance on seen versus unseen noise conditions. Testing was performed on all wv1 conditions (Sennheiser microphone).

3.6 Experiments

We experimentally evaluated our approach on the well-benchmarked Aurora-4 [113] noisy speech recognition task. Aurora-4 is based on the WSJ0 corpus. It contains noises of six categories which were added to the clean data. Every clean and noisy utterance has been filtered to simulate the phone quality recording using P.341 [112]. The training data contains 4400 clean utterances and 446 utterances for each noise condition, i.e., a total of 2676 noisy utterances. The test set consists

Table 3.1: Average word error rates (WER%) on Aurora-4 dataset on all test conditions, including seen and unseen noise and unseen microphone. The first column specifies the number of noise conditions used for the training. The results in the last row are from a preliminary experiment with layer-wise pre-training, close to state-of-the-art model and a corresponding invariance training starting with a pre-trained model. 'BL' and 'Inv' are the baseline and our model respectively. Last row summarizes the results reported in prior work by Yu et al. (2013a)

	All		A		B		C		D	
	Inv	BL	Inv	BL	Inv	BL	Inv	BL	Inv	BL
1	16.36	18.14	6.54	7.57	12.71	14.09	11.45	13.10	22.47	24.80
2	15.56	17.39	5.90	6.58	11.69	13.28	11.12	13.51	21.79	23.96
3	14.24	14.67	5.45	5.08	10.76	12.44	9.75	9.84	19.93	19.30
4	13.61	13.84	5.08	5.29	9.73	9.97	9.49	9.56	19.49	19.90
5	13.41	13.02	5.12	5.34	9.52	9.42	9.55	8.67	19.33	18.65
6	12.62	12.60	4.80	4.61	9.04	8.86	8.76	8.59	18.16	18.21
6*	11.85	11.99	4.52	4.76	8.76	8.76	7.79	8.57	16.84	16.99
$6'$	13.4		5.6		8.8		8.9		20.0	

of clean data, data corrupted by 6 noise types, and data recorded with a different microphone for both the clean and noisy conditions.

For both the clean and noisy data, we extracted 40-dimensional mel-filter-bank features with their deltas and delta-deltas spliced over ±5 frames, resulting in 1320 input features that were subsequently mean and variance normalized. The baseline acoustic model was a 6-layer DNN with 2048 rectified linear units at every layer. It was trained using momentum-accelerated SGD for 15 epochs with new-bob annealing (the learning rate is halved if no improvement on the validation set, as in [105, 127]).

To evaluate the impact of our method on generalization to *unseen* noises (the most typical situation in practice), we performed 6 experiments with different sets of noises seen during training. The networks were trained on clean data, with each noise condition added one-by-one in the following order: airport, babble, car, restaurant, street, and train. The last training group included all noises and therefore matched the standard multi-condition training setup. For every training group, we trained the baseline and the invariance model, where we branched out at the 4^{th} layer to a binary classifier predicting clean versus noisy data. Due to the imbalance between amounts of clean and noisy utterances, we had to over-sample noisy frames to ensure that every mini-batch contained equal number of clean and noisy speech frames.

Table 3.1 summarizes the results. Figure 3.2 visualizes the word error rate (WER) for the baseline multi-condition training and invariance training as the number of seen noise types varies. We conclude that the best performance gain is achieved when a small number of noise types are available during training. It can be seen that invariance training is able to generalize better to unseen noise types compared with multi-condition training. In practice, it's only possible to train on a small fraction of all the possible noise conditions in the world, so this apparent ability to generalize to unseen conditions is a promising result.

We note that our experiments did not use layer-wise pre-training, commonly used for small datasets. The baseline WERs reported are very close to the state-of-the-art. Our preliminary experiments on a pre-trained network (better overall WER) when using all noise types (last row of Table 3.1) for training show the same trend as the non-pretrained networks.

3.7 Conclusion

This work shows that the invariance training helps the ASR system to generalize better to unseen noise conditions and improves the word error rate when a small number of noise types are seen during training. Our experiments show that, relative to the image recognition tasks reported by [42], the task of the domain classification (D) network is harder for speech recognition. Therefore, the gradient of the L_3 term in Eq. 3.2 is noisier. Future research includes enhancements to the domain adaptation network while exploring alternative network architectures and invariance-promoting loss functions.

Chapter 4

Twin Networks: Matching the Future for Sequence Generation

This chapter presents paper D. Serdyuk, N. R. Ke, A. Sordoni, A. Trischler, C. Pal, and Y. Bengio. Twin networks: Matching the future for sequence generation. In *International Conference on Learning Representations*, 2018.

4.1 Context

After this paper was published, several works were published following similar line of research or applying methods developed here for other tasks.

Relevant Literature

The idea of improving sequence generation with bi-directional recurrent networks gained a lot of interest in the machine learning community. Methods to approach this problem range from refinement during the inference to Bayesian methods. Below, we list several prominent publications that are similar in spirit to ours and publications that followed the original in later years.

Future vectors A paper [92] proposes so-called *future vectors* to enhance training of language model. This augmented language model is used for inference in a large vocabulary speech recognition task. Similarly to the work presented in Section 4.3, [92] extracts the future vectors by running a recurrent model backwards in time.

More precisely, the future vectors are pre-activations of a backward running LSTM 2.2.5 language model,

$$z_i = \mathrm{LSTM}(x_i, z_{i+1}), \tag{4.1}$$

where x_i is an embedding of the i-th word, z_i and z_{i+1} are the i-th and $(i+1)$-th future vectors accordingly, and the LSTM cell is omitted for simplicity. This model is trained until convergence to model inverted sequences $(x_N \dots x_1)$.

In the next stage of the method proposed, another auto-regressive LSTM model is trained to model future vectors z and the input sequence x simultaneously. This model runs forward and uses MSE criterion to predict z and regular cross-entropy for x, so that the total loss is

$$L = \lambda \mathrm{MSE}(\widehat{z}, z) + \mathrm{CE}(\widehat{x}, x), \tag{4.2}$$

with λ set to 1. The paper makes an architectural choice to run two separate LSTM networks to predict x and z respectively.

While this co-temporal work introduces similar ideas, there are several substential differences to contributions of our work. First, our experimental evaluation is much more substantial. We evaluate on a high range of diverse tasks, such as speech recognition, image captioning, image generation, and language modelling. In the contrast, [92] generally focuses on speech recognition. Then, we train both networks – forward and backward running – at the same time. This helps them to co-adapt and simplifies the setup. Finally, we share almost all the parameters for the backward running network.

Variational methods for bi-directional sequence generation The works listed in this paragraph utilize stochastic recurent networks in the VAE framework [79]. Stochastic models help to encode variability in observed data. While it is prohibitively expensive to make inference in bi-directional generative model, VAE allows to simply sample from prior during inference.

The evidence lower bound can be rewritten as

$$\mathcal{F} = \mathbb{E}_{q(z|x;\phi)} \left[\log p(x|z;\theta)\right] - \mathrm{KL}\left(q(z|x;\phi)||p(z|\theta)\right). \tag{4.3}$$

Simply put, first, the input x is encoded into the latent space z with an encoder $q(z|x)$; then z is decoded back with the decoder $p(x|z;\theta)$. To propagate the gradient through the sampling process $z \sim q(z|x;\phi)$, the reparametrization trick is applied – the distribution q is reparametrized as some deterministic function $z = f(x, \varepsilon, \phi)$ of a random variable ε sampled from some fixed distribution $\varepsilon \sim u(\varepsilon)$.

The uniting idea of the publication highlighted below is to use a bi-directional network for the encoder $q(z|x;\phi)$. Similarly to the Twin Networks, the encoder is used only for training. During the inference, the prior $p(z|\theta)$ is used to sample from $p(x|\theta) = p(x|z;\theta)p(z|\theta)$.

In [2] authors couple two recurrent networks – one is running forward, second is running backward – with a number of latent variables z_t for every time-step t. More formally, an auto-regressive generative model is trained with the following evidence lower bound

$$L(x) = \sum_t \mathbb{E}_{q(z_t|x;\phi)} [\log p(x_{t+1}|x_{1:\,t}, z_{1:\,t};\theta)] - KL(q(z_t|x;\phi)||p(z_t|x_{1:\,t-1}, z_{1:t-1})), \tag{4.4}$$

where $1\colon t$ decodes a sub-sequence from 1 up to t. In addition, authors introduce an auxiliary cost

$$L_{aux} = \log p(b_t|z_t;\xi), \tag{4.5}$$

where b_t is t-th hidden state of the backward running network. Just alike Twin Networks approach, the backward running network is discarded when performing inference.

While discarding the backward network is easy and straightforward, it may be desirable to use information captured by it. To this end, [143] introduces bi-directional variational LSTM 2.2.5. This approach extends the approach of [2]. Although, the backward network is still discarded during the inference, the proposed approach encourages the forward network to learn dependencies in the future time-steps. More specifically, the authors introduce latent variables $\widetilde{b}_t$ that is tasked to learn from the deterministic states b_t of the backward network. Again, the approach itroduces two auxiliary costs to ensure that latent variables $\widetilde{h}_t$ and $\widetilde{b}_t$ stay close to hidden states of two corresponding recurrent networks h_t and b_t

$$L_{aux} = \sum_t \alpha \log p(b_t|z_t;\psi) + \beta \log p(h_{t-1}|z_t;\xi). \tag{4.6}$$

To summarize, several variational approaches that aim to learn future for sequence generation. Probabilistic models are capable of learning complex multimodal distributions such as for speech synthesis. This comes at a cost of complex and unstable training procedure. On the other hand, Twin Networks is a deterministic approach that has difficulties modeling multi-model distributions (as noted in Section 4.6). But the training procedure is stable (see Section 4.5) and the implementation is straightforward.

Follow-up works

Twin Networks for Online Speech Recognition Usually, the acoustic model for hybrid ASR (see Section 2.3) does not have a requirement to be uni-directional. Therefore, the best reported results usually use bi-directional networks [55]. However, it is not the case for the online speech recognition. In the task of online speech recognition we want to aggregate the audio input as soon as possible and output transcribed speech while the speaker is finishing a phrase. This is a setup close to how humans understand speech. Additionally, the online ASR is used for time sensitive situations, for example, to generate closed captions for TV programs on-the-fly.

One straightforward approach for online ASR is to reuse state of the art bi-directional models. In the nutshell, in order to use a bi-directional RNN, one needs to wait to read some extra audio. Generally, this is up to several hundreds milliseconds, which corresponds to several tens of frames. Then, the network is trained to produce the transcript up to time $t-t_d$, where is t_d is the introduced delay. This necessarily introduces certain latency t_d. The lower the latency, the fewer frames in the future can we aggregate with the bi-directional recurrent network, the less benefit from it.

Applying Twin Networks technique, [122] allows to decrease the latency down to zero. Just like we cannot access future when we generate sequences, we cannot access future in zero latency online recognition. Therefore, it makes sense to train a Twin for the uni-directional recurrent acoustic model. After training, the Twin is discarded so that the inference is again strictly uni-directional. This way, we can introduce some of benefits of bi-directional networks into the online recognizer. The experiments show that indeed a Twin Network performs better than the baseline system. However, there is still the gap between the online and the offline system that uses all the future frames.

The main difference between this work and the Twin Networks paper is that here the bi-directional RNN is used for the encoder. More specifically, the backward RNN is trained for the acoustic model. On the contrast, the original publication proposed to use the backward RNN only for the decoder, which plays the role of the linguistic model and, partially, the language model.

One of the shortcomings of this approach is that it cannot be extended for the end-to-end sequence to sequence models. The main reason is that the encoder in the end-to-end model is interconnected with the decoder. It means that when removing the backward running RNN, there is no easy way to detach it from the decoder.

Twin Networks for Sound Source Separation The work [39] is focusing on the task of singing voice separation.[1] In this task, the input is the recording of a given song with music. The desired output is the voice of the singer, a single component of the song mixture. This work focuses on a monaural source separation, meaning that the input has a single channel. In the contrast to multi-channel audio, it is generally harder, for a reason that there is no direction encoded in the signal. Therefore, it is not possible to apply beam forming or other methods of determining the location of each instrument in a given recording.

Traditional source separation usually works by predicting the mask in time-frequency domain [126]. After applying this mask, the resulting audio signal is treated as one of the components for separation. Therefore, optimizing the binary mask, it is possible to perform separation of two sources. It is possible to extend this to the case of separation several sources by using a multinary mask. A prior work [101] combined the masking approach with a denoiser. The two main components for source separation employed here are the recurrent masker and the denoiser.

This work was further extended [38] for the harmonic/percussive source separation. Similar methods were applied to a task of separating the musical mixture into two parts: the drum section, and the rest.

Stronger Sequence Generation Models The presented here work on Twin Networks aims to improve sequence generation and planning ahead when generating long sequences. While this was a problem at the time, some recently published stronger models have generation consistency problems to much lesser extent. The family of these models is based on the research of the *transformer*, an attention-based generative model [159]. The transformer model consists of multiple layers of self-attention only connections. This means that there are no recurrent connections. The main advantage of using self-attention is that the model is able to get access to any previous time-step without intermediate hidden state. These “shortcuts” not only simplify access to previous information, but also help training and convergence similarly to skip connections [65].

After the success of the transformer networks for translation, other works based on similar architectures started to emerge. One of the highlights is *bidirectional encoder representations from transformer*, BERT [36], a model for text embedding. Consequent works use BERT to improve text generation [119]. The resulting models, GPT and GPT-2, achieved state of the art performance on many

[1] Demo is available at http://arg.cs.tut.fi/demo/mad-twinnet/.

datasets for language modelling and produced samples of text that seem to be written by a human.

With the development of such strong language models, the problem of consistent text generation is less relevant than before. Nevertheless, the tasks that require smaller models or explicit recurrent state still can benefit from Twin Network model. Furthermore, the Twin Networks is not specific to the particular architecture, therefore can be used with the transformer models after some small changes.

4.2 Twin Networks: Introduction

Recurrent neural networks (RNNs) are the basis of state-of-art models for generating sequential data such as text and speech. RNNs are trained to generate sequences by predicting one output at a time given all previous ones, and excel at the task through their capacity to remember past information well beyond classical *n*-gram models [18, 71]. More recently, RNNs have also found success when applied to conditional generation tasks such as speech-to-text [30, 23], image captioning [164] and machine translation [150, 7].

RNNs are usually trained by *teacher forcing*: at each point in a given sequence, the RNN is optimized to predict the next token given all preceding tokens. This corresponds to optimizing one-step-ahead prediction. As there is no explicit bias toward planning in the training objective, the model may prefer to focus on the most recent tokens instead of capturing subtle long-term dependencies that could contribute to global coherence. Local correlations are usually stronger than long-term dependencies and thus end up dominating the learning signal. The consequence is that samples from RNNs tend to exhibit local coherence but lack meaningful global structure. This difficulty in capturing long-term dependencies has been noted and discussed in several seminal works [70, 18, 71, 115].

Recent efforts to address this problem have involved augmenting RNNs with external memory [37, 49, 59], with unitary or hierarchical architectures [5, 139], or with explicit planning mechanisms [60]. Parallel efforts aim to prevent overfitting on strong local correlations by regularizing the states of the network, by applying dropout or penalizing various statistics [103, 172, 40, 84, 97].

In this paper, we propose *TwinNet*,[2] a simple method for regularizing a recurrent neural network that encourages modeling those aspects of the past that

[2] The source code is available at https://github.com/dmitriy-serdyuk/twin-net/.

are predictive of the long-term future. Succinctly, this is achieved as follows: in parallel to the standard forward RNN, we run a "twin" backward RNN (with no parameter sharing) that predicts the sequence in reverse, and we encourage the hidden state of the forward network to be close to that of the backward network used to predict the same token. Intuitively, this forces the forward network to focus on the past information that is useful to predicting a specific token and that is *also* present in and useful to the backward network, coming from the future (Fig. 4.1).

In practice, our model introduces a regularization term to the training loss. This is distinct from other regularization methods that act on the hidden states either by injecting noise [84] or by penalizing their norm [85, 97], because we formulate explicit auxiliary targets for the forward hidden states: namely, the backward hidden states. The activation regularizer (AR) proposed by [97], which penalizes the norm of the hidden states, is equivalent to the TwinNet approach with the backward states set to zero. Overall, our model is driven by the intuition (a) that the backward hidden states contain a summary of the future of the sequence, and (b) that in order to predict the future more accurately, the model will have to form a better representation of the past. We demonstrate the effectiveness of the TwinNet approach experimentally, through several conditional and unconditional generation tasks that include speech recognition, image captioning, language modelling, and sequential image generation. To summarize, the contributions of this work are as follows:

- We introduce a simple method for training generative recurrent networks that regularizes the hidden states of the network to anticipate future states (see Section 4.3);

- The paper provides extensive evaluation of the proposed model on multiple tasks and concludes that it helps training and regularization for conditioned generation (speech recognition, image captioning) and for the unconditioned case (sequential MNIST, language modelling, see Section 4.5);

- For deeper analysis we visualize the introduced cost and observe that it negatively correlates with the word frequency (more surprising words have higher cost).

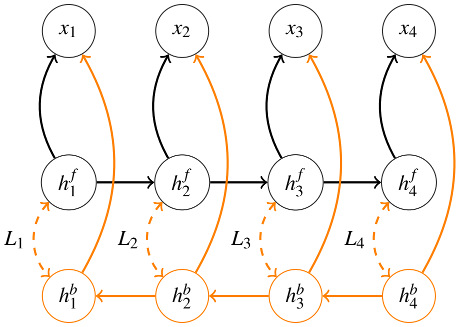


Figure 4.1: The forward and the backward networks predict the sequence $s = \{x_1,...,x_4\}$ independently. The penalty matches the forward (or a parametric function of the forward) and the backward hidden states. The forward network receives the gradient signal from the log-likelihood objective as well as L_t between states that predict the same token. The backward network is trained only by maximizing the data log-likelihood. During the evaluation part of the network colored with orange is discarded. The cost L_t is either a Euclidean distance or a learned metric $||g(h_t^f) - h_t^b||_2$ with an affine transformation g. Best viewed in color.

4.3 Model

Given a dataset of sequences $\mathcal{S} = \{s^1,\ldots,s^n\}$, where each $s^k = \{x_1,\ldots,x_{T_k}\}$ is an observed sequence of inputs $x_i \in \mathcal{X}$, we wish to estimate a density $p(s)$ by maximizing the log-likelihood of the observed data $\mathcal{L} = \sum_{i=1}^n \log p(s^i)$. Using the chain rule, the joint probability over a sequence $x_1,\ldots,x_T$ decomposes as:

$$p(x_1,\ldots,x_T) = p(x_1)p(x_2|x_1)\ldots = \prod_{t=1}^{T} p(x_t|x_1,\ldots,x_{t-1}). \tag{4.7}$$

This particular decomposition of the joint probability has been widely used in language modeling [17, 99] and speech recognition [11]. A recurrent neural network is a powerful architecture for approximating this conditional probability. At each step, the RNN updates a hidden state h_t^f, which iteratively summarizes the inputs seen up to time t:

$$h_t^f = \Phi_f(x_{t-1}, h_{t-1}^f), \tag{4.8}$$

where f symbolizes that the network reads the sequence in the forward direction, and Φ_f is typically a non-linear function, such as a LSTM cell [71] or a GRU [27]. Thus, h_t^f forms a representation summarizing information about the sequence's past. The prediction of the next symbol x_t is performed using another non-linear transformation on top of h_t^f, i.e. $p_f(x_t|x_{<t}) = \Psi_f(h_t^f)$, which is typically a linear or affine transformation (followed by a softmax when x_t is a symbol). The basic idea of our approach is to encourage h_t^f to contain information that is useful to predict x_t and which is also compatible with the upcoming (future) inputs in the sequence. To achieve this, we run a twin recurrent network that predicts the sequence in reverse and further require the hidden states of the forward and the backward networks to be close. The backward network updates its hidden state according to:

$$h_t^b = \Phi_b(x_{t+1}, h_{t+1}^b), \tag{4.9}$$

and predicts $p_b(x_t|x_{>t}) = \Psi_b(h_t^b)$ using information only about the future of the sequence. Thus, h_t^f and h_t^b both contain useful information for predicting x_t, coming respectively from the past and future. Our idea consists in penalizing the distance between forward and backward hidden states leading to the same prediction. For this we use the Euclidean distance (see Fig. 4.1):

$$L_t(s) = \|g(h_t^f) - h_t^b\|_2, \tag{4.10}$$

where the dependence on x is implicit in the definition of h_t^f and h_t^b. The function g adds further capacity to the model and comes from the class of parametrized affine transformations. Note that this class includes the identity transformation. As we will show experimentally in Section 4.5, a learned affine transformation gives more flexibility to the model and leads to better results. This relaxes the strict match between forward and backward states, requiring just that the forward hidden states are predictive of the backward hidden states.[3]

The total objective maximized by our model for a sequence s is a weighted sum of the forward and backward log-likelihoods minus the penalty term, computed at each time-step:

$$\mathcal{F}(s) = \sum_t \log p_f(x_t|x_{<t}) + \log p_b(x_t|x_{>t}) - \alpha L_t(s), \tag{4.11}$$

where α is an hyper-parameter controlling the importance of the penalty term. In order to provide a more stable learning signal to the forward network, we only

[3] Matching hidden states is equivalent to matching joint distributions factorized in two different ways, since a given state contains a representation of all previous states for generation of all later states and outputs. For comparison, we made several experiments matching outputs of the forward and backward networks rather than their hidden states, which is equivalent to matching $p(x_t|x_{<t})$ and $p(x_t|x_{>t})$ separately for every t. None of these experiments converged.

propagate the gradient of the penalty term through the forward network. That is, we avoid co-adaptation of the backward and forward networks. During sampling and evaluation, we discard the backward network.

The proposed method can be easily extended to the conditional generation case. The forward hidden-state transition is modified to

$$h_t^f = \Phi_f \left(x_{t-1}, \left[h_{t-1}^f, c \right] \right), \tag{4.12}$$

where c denotes the task-dependent conditioning information, and similarly for the backward RNN.

4.4 Related Work

Bidirectional neural networks [135] have been used as powerful feature extractors for sequence tasks. The hidden state at each time step includes both information from the past and the future. For this reason, they usually act as better feature extractors than the unidirectional counterpart and have been successfully used in a myriad of tasks, e.g. in machine translation [10], question answering [24] and sequence labeling [95]. However, it is not straightforward to apply these models to sequence generation [174] due to the fact that the ancestral sampling process is not allowed to look into the future. In this paper, the backward model is used to regularize the hidden states of the forward model and thus is only used during training. Both inference and sampling are strictly equivalent to the unidirectional case.

Gated architectures such as LSTMs [71] and GRUs [32] have been successful in easing the modeling of long term-dependencies: the gates indicate time-steps for which the network is allowed to keep new information in the memory or forget stored information. A number of Works [57, 37, 49] effectively augment the memory of the network by means of an external memory. Another solution for capturing long-term dependencies and avoiding gradient vanishing problems is equipping existing architectures with a hierarchical structure [139]. Other works tackled the vanishing gradient problem by making the recurrent dynamics unitary [5]. In parallel, inspired by recent advances in “learning to plan” for reinforcement learning [147, 152], recent efforts try to augment RNNs with an explicit planning mechanism [60] to force the network to commit to a plan while generating, or to make hidden states predictive of the far future [90].

Regularization methods such as noise injection are also useful to shape the learning dynamics and overcome local correlations to take over the learning process. One of the most popular methods for neural network regularization is dropout [149]. Dropout in RNNs has been proposed in [103], and was later extended in [137, 40], where recurrent connections are dropped at random. Zoneout [84] modifies the hidden state to regularize the network by effectively creating an ensemble of different length recurrent networks. [85] introduce a "norm stabilization" regularization term that ensures that the consecutive hidden states of an RNN have similar Euclidean norm. Recently, [97] proposed a set of regularization methods that achieve state-of-the-art on the Penn Treebank language modeling dataset. Other RNN regularization methods include the weight noise [50], gradient clipping [115] and gradient noise [107].

4.5 Experimental Setup and Results

We now present experiments on conditional and unconditional sequence generation, and analyze the results in an effort to understand the performance gains of TwinNet. First, we examine conditional generation tasks such as speech recognition and image captioning, where the results show clear improvements over the baseline and other regularization methods. Next, we explore unconditional language generation, where we find our model does not significantly improve on the baseline. Finally, to further determine what tasks the model is well-suited to, we analyze a sequential imputation task, where we can vary the task from unconditional to strongly conditional.

Speech Recognition

We evaluated our approach on the conditional generation for character-level speech recognition, where the model is trained to convert the speech audio signal to the sequence of characters. The forward and backward RNNs are trained as conditional generative models with soft-attention [30]. The context information c is an encoding of the audio sequence and the output sequence s is the corresponding character sequence. We evaluate our model on the Wall Street Journal (WSJ) dataset closely following the setting described in [9]. We use 40 mel-filter bank features with delta and delta-deltas with their energies as the acoustic inputs to the model, these features are generated according to the Kaldi s5 recipe [117]. The resulting input feature dimension is 123.

We observe the Character Error Rate (CER) for our validation set, and we early stop on the best CER observed so far. We report CER for both our validation

Table 4.1: Average character error rate (CER, %) on WSJ dataset decoded with the beam size 10. We compare the attention model for speech recognition [“Baseline,” 9]; the regularizer proposed by [85] (“Stabilizing norm”); penalty on the L2 norm of the forward states [97] (“AR”), which is equivalent to TwinNet when all the hidden states of the backward network are set to zero. We report the results of our model (“TwinNet”) both with $g = I$, the identity mapping, and with a learned g.

Model	**Test CER**	**Valid CER**
Baseline	6.8	9.0
Baseline + Gaussian noise	6.9	9.1
Baseline + Stabilizing Norm	6.6	9.0
Baseline + AR	6.5	8.9
Baseline + TwinNet ($g = I$)	6.6	8.7
Baseline + TwinNet (learnt g)	6.2	8.4

and test sets. For all our models and the baseline, we follow the setup in [9] and pre-train the model for 1 epoch, within this period, the context window is only allowed to move forward. We then perform 10 epochs of training, where the context window looks freely along the time axis of the encoded sequence, we also perform annealing on the models with 2 different learning rates and 3 epochs for each annealing stage. We use the AdaDelta optimizer for training. We perform a small hyper-parameter search on the weight α of our twin loss, $\alpha \in \{2.0, 1.5, 1.0, 0.5, 0.25, 0.1\}$, and select the best one according to the CER on the validation set.[4]

Results We summarize our findings in Table 4.1. Our best performing model shows relative improvement of 12% comparing to the baseline. We found that the TwinNet with a learned metric (learnt g) is more effective than strictly matching forward and hidden states. In order to gain insights on whether the empirical usefulness comes from using a backward recurrent network, we propose two ablation tests. For “Gaussian Noise,” the backward states are randomly sampled from a Gaussian distribution, therefore the forward states are trained to predict white noise. For “AR,” the backward states are set to zero, which is equivalent to penalizing the norm of the forward hidden states [97]. Finally, we compare the model with the “Stabilizing Norm” regularizer [85], that penalizes the difference

[4] The best hyper-parameter was 1.5.

of the norm of consecutive forward hidden states. Results shows that the information included in the backward states is indeed useful for obtaining a significant improvement.

Analysis The training/validation curve comparison for the baseline and our network is presented in Figure 4.2a.[5] The TwinNet converges faster than the baseline and generalizes better. The L2 cost raises in the beginning as the forward and backward network start to learn independently. Later, due to the pressure of this cost, networks produce more aligned hidden representations. Figure 4.3 provides examples of utterances with L2 plotted along the time axis. We observe that the high entropy words produce spikes in the loss for such words as "uzi." This is the case for rare words which are hard to predict from the acoustic information. To elaborate on this, we plot the L2 cost averaged over a word depending on the word frequency. The average distance decreases with the increasing frequency. The histogram comparison (Figure 4.2b) for the cost of rare and frequent words reveal that the not only the average cost is lower for frequent words, but the variance is higher for rare words. Additionally, we plot the dependency of the L2 cost cross-entropy cost of the forward network (Figure 4.2c) to show that the conditioning also plays the role in the entropy of the output, the losses are not absolutely correlated.

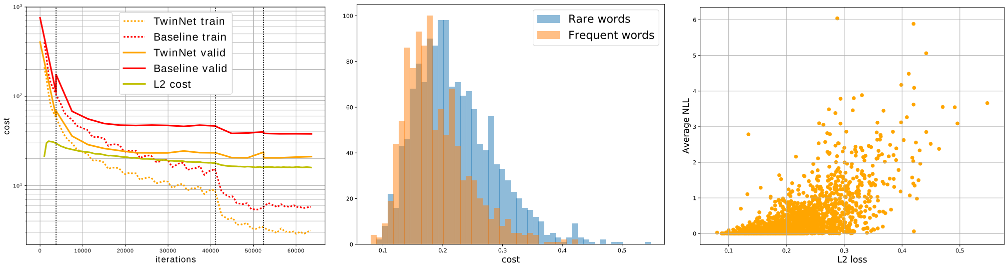
Image Captioning

We evaluate our model on the conditional generation task of image captioning task on Microsoft COCO dataset [91]. The MS COCO dataset covers 82,783 training images and 40,504 images for validation. Due to the lack of standardized split of training, validation and test data, we follow Karpathy's split [75, 164, 162]. These are 80,000 training images and 5,000 images for validation and test. We do early stopping based on the validation CIDEr scores and we report BLEU-1 to BLEU-4, CIDEr, and Meteor scores. To evaluate the consistency of our method, we tested TwinNet on both encoder-decoder ["Show&Tell," 161] and soft attention ["Show, Attend and Tell," 164] image captioning models.[6]

We use a residual network [65] with 101 and 152 layers pre-trained on ImageNet for image classification. The last layer of the Resnet is used to extract

[5] The saw tooth pattern of both training curves corresponds to shuffling within each epoch as was previously noted by [20].

[6] Following the setup in https://github.com/ruotianluo/neuraltalk2.pytorch.



(a) (b) (c)

Figure 4.2: Analysis for speech recognition experiments. **(a)**: Training curves comparison for TwinNets and the baseline network. Dotted vertical lines denote stages of pre-training, training, and two stages of annealing. The L2 cost is plotted alongside. The TwinNet converges to a better solution as well as provides better generalization. **(b)**: Comparison of histograms of the cost for rare words (first 1500) versus frequent words (all other). The cost is averaged over characters of a word. The distribution of rare words is wider and tends to produce higher L2 cost. **(c)**: L2 loss vs. average cross-entropy loss.

2048 dimensional input features for the attention model [164]. We use an LSTM with 512 hidden units for both “Show & Tell” and soft attention. Both models are trained with the Adam [78] optimizer with a learning rate of 10^{-4}. TwinNet showed consistent improvements over “Show & Tell” (Table 4.2). For the soft attention model we observe small but consistent improvements for majority of scores.

Unconditional Generation: Sequential MNIST and Language Modeling

We investigate the performance of our model in pixel-by-pixel generation for sequential MNIST. We follow the setting described by [87]: we use an LSTM with 3-layers of 512 hidden units for both forward and backward LSTMs, batch size 20, learning rate 0.001 and clip the gradient norms to 5. We use Adam [78] as our optimization algorithm and we decay the learning rate by half after $5, 10$, and 15 epochs. Our results are reported at the Table 4.3. Our baseline LSTM implementation achieves 79.87 nats on the test set. We observe that by adding the TwinNet regularization cost consistently improves performance in this setting by about 0.52 nats. Adding dropout to the baseline LSTM is beneficial. Further gains were observed by adding both dropout and the TwinNet regularization cost. This

Table 4.2: Results for image captioning on the MS COCO dataset, the higher the better for all metrics (BLEU 1 to 4, METEOR, and CIDEr). We re-implement both Show&Tell [161] and Soft Attention [164] in order to add the twin cost. We use two types of images features extracted either with Resnet-101 or Resnet-152.

Models	**B-1**	**B-2**	**B-3**	**B-4**	**M**	**C**
DeepVS [75]	62.5	45.0	32.1	23.0	19.5	66.0
ATT-FCN [169]	70.9	53.7	40.2	30.4	24.3	—
Show & Tell [161]	—	—	—	27.7	23.7	85.5
Soft Attention [164]	70.7	49.2	34.4	24.3	23.9	—
Hard Attention [164]	71.8	50.4	35.7	25.0	23.0	—
MSM [168]	73.0	56.5	42.9	32.5	25.1	98.6
Adaptive Attention [93]	74.2	58.0	43.9	33.2	26.6	108.5
No attention, Resnet101						
Show&Tell (Our impl.)	69.4	51.6	36.9	26.3	23.4	84.3
+ TwinNet	71.8	54.5	39.4	28.0	24.0	87.7
Attention, Resnet101						
Soft Attention (Our impl.)	71.0	53.7	39.0	28.1	24.0	89.2
+ TwinNet	72.8	55.7	41.0	29.7	25.2	96.2
No attention, Resnet152						
Show&Tell (Our impl.)	71.7	54.4	39.7	28.8	24.8	93.0
+ TwinNet	72.3	55.2	40.4	29.3	25.1	94.7
Attention, Resnet152						
Soft Attention (Our impl.)	73.2	56.3	41.4	30.1	25.3	96.6
+ TwinNet	73.8	56.9	42.0	30.6	25.2	97.3

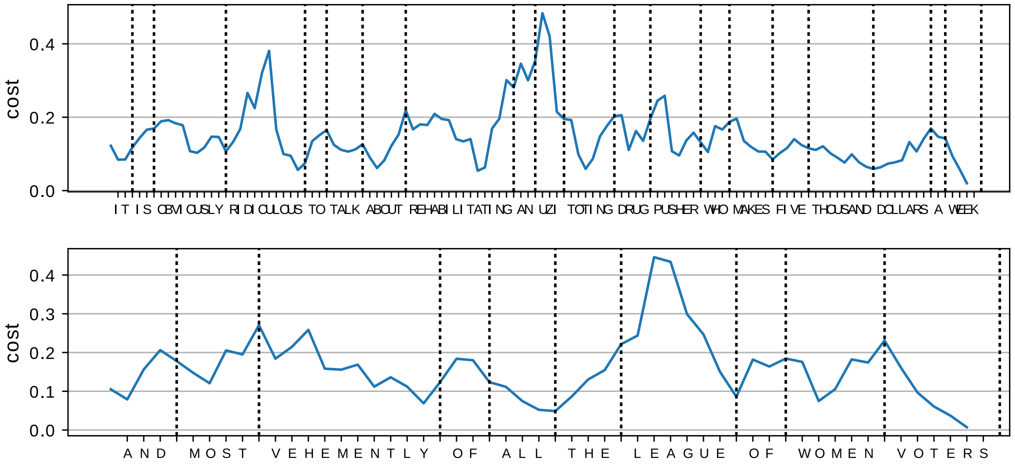


Figure 4.3: Example of the L2 loss plotted along the time axis. Notice that spikes correspond to rare words given the acoustic information where the entropy of the prediction is high. Dotted vertical lines are plotted at word boundary positions.

last model achieves 79.12 nats on test set. Note that this result is competitive with deeper models such as PixelRNN [110] (7-layers) and PixelVAE [61] which uses an auto-regressive decoder coupled with a deep stochastic auto-encoder.

As a last experiment, we report results obtained on a language modelling task using the PennTree Bank and WikiText-2 datasets [97]. We augment the state-of-the-art AWD-LSTM model [97] with the proposed TwinNet regularization cost. The results are reported in Table 4.4.

4.6 Discussion

In this paper, we presented a simple recurrent neural network model that has two separate networks running in opposite directions during training. Our model is motivated by the fact that states of the forward model should be predictive of the entire future sequence. This may be hard to obtain by optimizing one-step ahead predictions. The backward path is discarded during the sampling and evaluation process, which makes the sampling process efficient. Empirical results show that the proposed method performs well on conditional generation for several tasks. The analysis reveals an interpretable behaviour of the proposed loss.

One of the shortcomings of the proposed approach is that the training process

Table 4.3: Test set negative log-likelihood for binarized sequential MNIST, where $^\blacktriangledown$ denotes lower performance of our model with respect to the baselines.

Model	**MNIST**
DBN 2hl [45]	$\approx$84.55
NADE [157]	88.33
EoNADE-5 2hl [120]	84.68
DLGM 8 [131]	$\approx$85.51
DARN 1hl [58]	$\approx$84.13
DRAW [58]	$\leq$80.97
P-Forcing$_{\text{(3-layer)}}$ [87]	79.58
PixelRNN$_{\text{(1-layer)}}$ [110]	80.75
PixelRNN$_{\text{(7-layer)}}$ [110]	79.20
PixelVAE [61]	79.02$^\blacktriangledown$
MatNets [6]	78.50$^\blacktriangledown$
Baseline LSTM$_{\text{(3-layers)}}$	79.87
+ TwinNet$_{\text{(3-layers)}}$	79.35
Baseline LSTM$_{\text{(3-layers)}}$ + dropout	79.59
+ TwinNet$_{\text{(3-layers)}}$	79.12

doubles the computation needed for the baseline (due to the backward network training). However, since the backward network is discarded during sampling, the sampling or inference process has the exact same computation steps as the baseline. This makes our approach applicable to models that requires expensive sampling steps, such as PixelRNNs [110] and WaveNet [109]. One of future work directions is to test whether it could help in conditional speech synthesis using WaveNet.

We observed that the proposed approach yield minor improvements when applied to language modelling with PennTree bank. We hypothesize that this may be linked to the amount of entropy of the target distribution. In these high-entropy cases, at any time-step in the sequence, the distribution of backward states may be highly multi-modal (many possible futures may be equally likely for the same past). One way of overcoming this problem would be to replace the proposed L2 loss (which implicitly assumes a unimodal distribution of the backward states) by a more expressive loss obtained by either employing an inference network [79] or distribution matching techniques [47]. We leave that for future investigation.

Table 4.4: Perplexity results on WikiText-2 and Penn TreeBank [97]. AWD-LSTM refers to the model of [97] trained with the official implementation at `http://github.com/salesforce/awd-lstm/`.

Penn Treebank	**Valid**	**Test**
LSTM [172]	82.2	78.4
4-layer LSTM [96]	67.9	65.4
5-layer RHN [96]	64.8	62.2
AWD-LSTM	61.2	58.8
+ TwinNet	**61.0**	**58.3**

WikiText-2	**Valid**	**Test**
5-layer RHN [96]	78.1	75.6
1-layer LSTM [96]	69.3	65.9
2-layer LSTM [96]	69.1	65.9
AWD-LSTM	68.7	65.8
+ TwinNet	**68.0**	**64.9**

Chapter 5

Towards End-to-end Spoken Language Understanding

This section presents paper D. Serdyuk, Y. Wang, C. Fuegen, A. Kumar, B. Liu, and Y. Bengio. Towards end-to-end spoken language understanding. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5754–5758. IEEE, 2018. The paper explores applicability of modern techniques for a task of intent recognition from audio.

Spoken language understanding system is traditionally designed as a pipeline of a number of components. First, the audio signal is processed by an automatic speech recognizer for transcription or n-best hypotheses. With the recognition results, a natural language understanding system classifies the text to structured data as domain, intent and slots for down-streaming consumers, such as dialog system, hands-free applications. These components are usually developed and optimized independently. In this paper, we present our study on an end-to-end learning system for spoken language understanding. With this unified approach, we can infer the semantic meaning directly from audio features without the intermediate text representation. This study showed that the trained model can achieve reasonable good result and demonstrated that the model can capture the semantic attention directly from the audio features.

5.1 Context

Relation to keyword spotting. Keyword spotting is a task where the input is the audio features and the model is asked to find one of the words in some predefined set [15, 77, 76]. The keyword spotting is usually used as a certain filter that triggers further action. For example, some smart assistants monitor ambient sounds and wake up when a pre-programmed word or a phrase is spotted. Recently there were several attempts to train a keyword spotting model in end-to-end fashion [144, 89] with CTC or attention-based models.

The intent recognition is similar to the keyword spotting. When the keyword spotting task is a classification task between classes 'Keyword A', 'Keyword B', . . . , 'Keyword Z', 'No keywords'; the intent recognition is a classificaiton task between intent classes 'Intent A', . . . , 'Intent Z'. In many cases, recognizing the intent is similar to spotting several keywords. For example, recognizing the intent 'forecast' may imply spotting such words as 'weather', 'temperature', 'wind speed', etc. The crucial difference is that in the typical intent recognition, the set of the related words or phrases is not closed nor defined. Therefore, an intent recognition model has to learn from examples to infer an appropriate set of keywords.

Nevertheless, the multitude of end-to-end approaches to the keyword spotting makes the intent recognition the next promising direction.

Later developments There was substantial amount of work after the publication presented above. A work [26] proposes an approach for the intent recognition. The key ingredient to make the system work was the fine-tuning of the speech recognition model.

A similar approach is taken in [94]. The model is pre-trained to classify words and classify phonemes. Then, the auxiliary classifiers are dropped in the favor of the intent classifier. The paper uses a newly developed dataset Fluent Speech Commands. In addition to presenting the dataset, authors propose a way to pretrain the spoken language understanding model. First, an end-to-end automatic speech recognition model is trained. The authors include the phoneme recognizer and the word recognizer as two targets simultaneously. More specifically, the encoder aggregates the input audio features, then a phoneme recognizer branches out. After this, a second stage of the enoder is applied. Finally, the features are fed to the word recognizer. After training such a model, the parameters of the two stages of the encoder are used to initialize the encoder of the end-to-end spoken language understanding model. In order to ensure that the training process is stable, this work recommends to use a schedule to unfreeze the updating of the encoder parameters.

Some of end-to-end spoken language understanding approaches are discussed in an overview paper [62]. Furthermore, this paper proposes several sequence-to-sequence models of their own to tackle all three problems of the spoken language understanding. These problems are domain classification, intent recognition, and slot filling. The paper investigates four types of architectures. In the direct model, where the intent is extracted directly from audio, which is similar to our paper. In the joint model, the intent and the words are extracted simultaneously. In the multi-task model, the intent and the word extractors share a common encoder and then branch out into two separate sub-models. Finally, in the multi-state model, the word extractor is applied first, then the words are used in the intent recognizer. The work finds experimentally that the joint and multi-task models have best performance.

Another prominent work building upon our study is [46]. It concerns the named entity recognition for spoken language understanding. The work build an end-to-end system for named entity recognition and extraction. This includes names of cities, places, persons, businesses and such. The work proposes to augment the output produced by the neural network. More specifically, special tags ('<org', '<pers', etc, and a closing tag '>') are added as extra outputs. The Connectionist Temporal Classification is trained with this augmented output. The model proposed is based on the Deep Speech 2 [4].

5.2 Introduction

With the growing demand of voice interfaces for mobile and virtual reality (VR) devices, spoken language understanding (SLU) has received many researchers' attention recently [153, 165, 167, 19, 123, 134, 154]. Given a spoken utterance, a typical SLU system performs three main tasks: domain classification, intent detection and slot filling [153]. Standard SLU systems are usually designed as a pipeline structure. As shown in Figure 5.1, recorded speech signals are converted by an automatic speech recognition (ASR) module into the spoken format text, followed by an optional inverse text normalization module to translate the spoken domain text to the written domain; then a natural language understanding (NLU) module is used to determine intent and extract slots accordingly. Although there are some works (e.g. [104]) that take the possible ASR errors into consideration when designing the NLU module, the pipeline approach is widely adopted. One arguable limitation of this pipelined architecture is that each module is optimized separately under different criteria: the ASR module is trained to minimize the word error rate (WER) criterion, which typically weights each word equally;

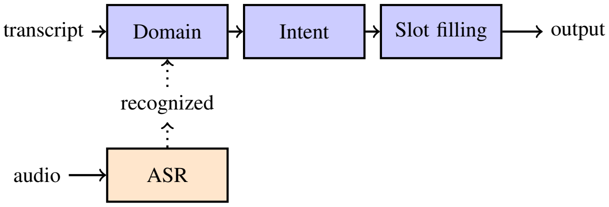


Figure 5.1: Traditional SLU system. The NLU part (in blue) is trained on transcripts and consists of a domain classifier, intent classifier and a slot filling model. The speech recognition part (in orange) is trained independently. During the evaluation time the NLU system uses the recognized text from the ASR system, as denoted by the dotted arrows.

obviously not every word has the same impact on the intent classification or slot filling accuracy. On the other hand, the NLU module is typically trained on the clean text (transcription) without ASR errors, but during evaluation, it has to use recognized hypotheses outputted by the ASR module as the input: errors in ASR module, especially in noisy conditions, will be propagated to SLU degrading its performance. Second consideration is how human process speech signal to extract intent level concepts. Intuitively, when asked to perform intent detection tasks, humans do not understand speech by recognizing word by word; instead, humans interpret and understand speech directly, while attention is given to the high level concepts which are directly related with the task. It is thus desirable to train an SLU system in an end-to-end fashion.

End-to-end learning has been widely used in several areas, such as machine translation [150, 7, 44] and image-captioning [164]. It has also been investigated for speech synthesis [109] and ASR tasks [4, 22, 148]. For example, the CTC loss function is used to train an ASR system to map the feature sequences directly to the word sequences in [148] and it has been shown to perform similarly to the traditional ASR systems; in [29, 9], encode-decoder models with attention have been used for ASR tasks, including large vocabulary ASR. End-of-end learning of memory networks is also used for knowledge carryover in multi-turn spoken language understanding [25].

Inspired by these success stories, we explore the possibility to extend the end-to-end ASR learning to include NLU component and optimize the whole system for SLU purpose. As the first step towards an end-to-end SLU system, in this work, we focus on maximizing the single-turn intent classification accuracy using log-Mel filter-bank feature directly. Contributions of this work are following:

1. We demonstrate the possibility of training a language understanding model

from audio features (see Section 5.4). To the best knowledge of authors this is the first work on this topic;

2. We show that the performance of an SLU degrades when evaluating on the ASR output. The details of our experiments are described in Section 5.5, and results in Section 5.6; Additionally, we perform experiments on noise-corrupted data. We artificially add noise to the audio data to demonstrate the degradation of the performance of a standard SLU evaluation and test our system.

5.3 Standard ASR and NLU systems

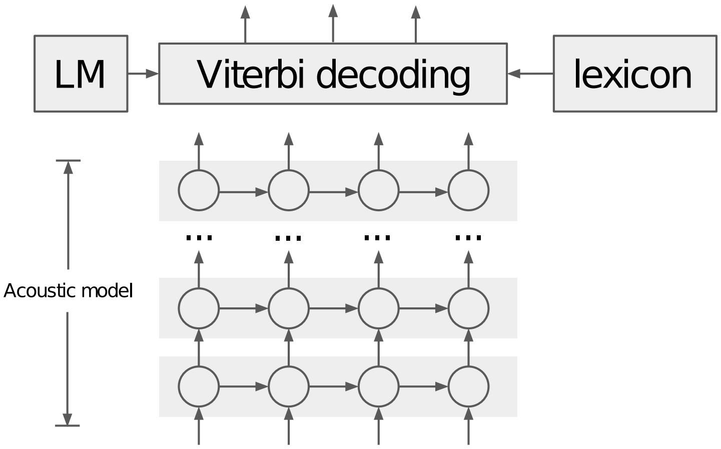


Figure 5.2: A schematic picture of a traditional ASR system. The recurrent acoustic model predicts the states of an HMM. It is decoded with Viterbi algorithm using the language model (LM).

Given a sequence of feature vectors $X = (x_1, \dots, x_T)$, an ASR system is trained to find the most likely word sequences $W^* = (w_1, \dots, w_n)$ using the chain rule:

$$W^* = \arg\max_W p(W|X) = \arg\max_W p(X|W)p(W).$$

Therefore, the ASR system is usually divided into two models: an acoustic model $p(X|W)$ and a language model $p(W)$ (AM and LM respectively). CD-HMM-

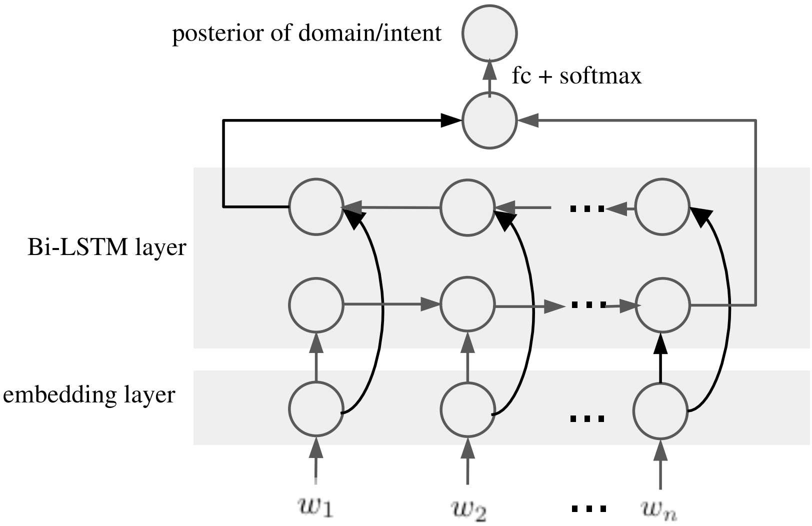


Figure 5.3: A schematic diagram of a standard NLU system for domain/intent classification. "`fc`" stands for a fully-connected layer.

LSTM [129] is widely used as an AM, in which the feature vector sequence is converted to the likelihood vectors of context-dependent HMM states for each acoustic frame. Together with the LM ($p(W)$ is usually a statistical n-gram model) and a dictionary, a Viterbi decoder is used to search for the most likely word sequence. Figure 5.2 depicts the described standard ASR architecture: the core part of the AM is a multi-layer LSTM [71] network, which predicts the probability of CD-HMM states for each frame. Since most of the SLU systems requires spontaneous response, usually only uni-directional LSTM is used.

Given the word sequence output by the ASR module, an NLU module is used to perform domain and intent classification, and to fill slots for different intents accordingly. Following [123], we use LSTM-based utterance classifier, in which the input words are first embedded in a dense representation, and then the LSTM network is used to encode the word sequence. We found 2-layer bi-directional LSTM encoder performs better. Fig. 5.3 demonstrates how a standard NLU system predicts the domain/intent for a given word sequence. Note that since the NLU system incurs much smaller latency compared with the ASR system and the classification only starts after the entire word sequence become available, it is possible to use bi-directional RNNs for NLU.

In the pipelined approach to SLU, ASR, and NLU modules are usually trained

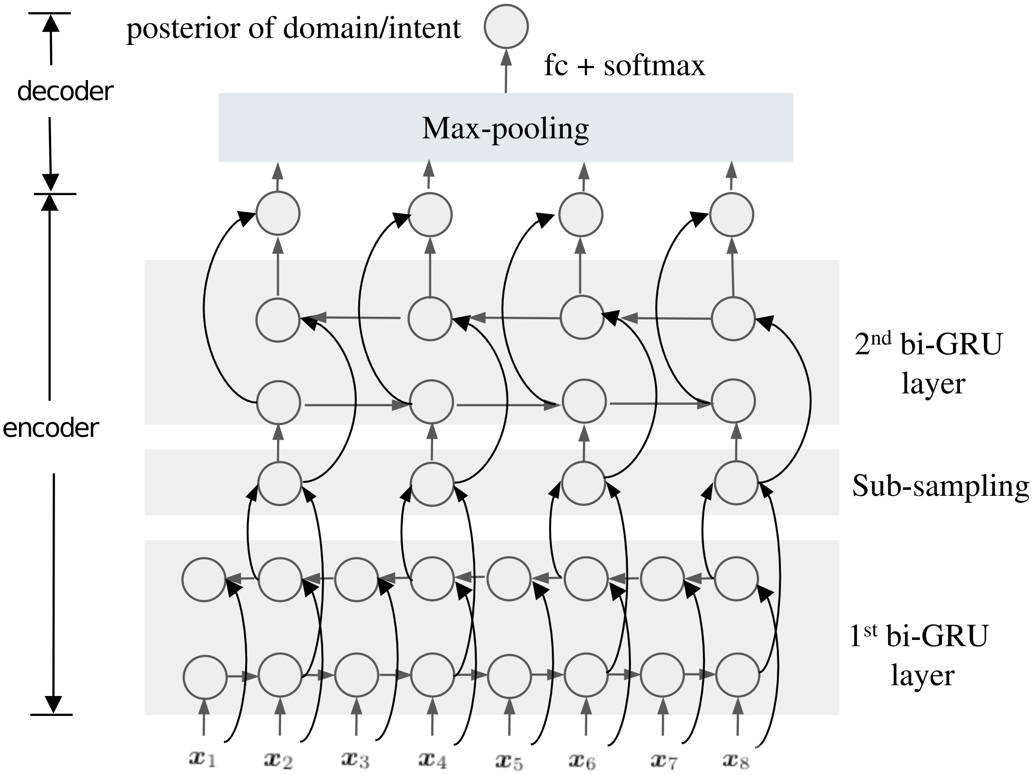


Figure 5.4: End-to-end SLU system. Sub-sampling is performed at every layer to reduce the output sequence length. Our best model uses 4 layer bidirectional GRU.

independently, where the NLU module is trained using human transcription as the input. During the evaluation phase, the ASR output is piped into the NLU module.

5.4 End-to-end Spoken Language Understanding

As the first step towards the end-to-end spoken language understanding, we focus on two tasks: speech-to-domain and speech-to-intent. Both tasks are sequence classification problems where the input is log-Mel filter-bank feature vectors.

The task of end-to-end SLU is close to speech recognition with a difference that the structure of the output is simpler but the transformation is more complicated and the data is noisy and non-unimodal (close utterances may have sufficiently different intents). For this reason our model was inspired by works in end-to-end speech recognition [51, 7, 22, 14]. We use an encoder-decoder framework. The input log-Mel filter-bank features are processed by an encoder which is a multi-layer bidirectional *gated recurrent unit* (GRU, [31]) network. A potential issue of using log-Mel filter-bank feature is that it is generated every 10 ms: the 10-ms frame rate is good for classifying sub-phone unit like CD-HMM states, while it may not be suitable for utterance-level classification as GRUs may forget most of the speech content when it arrives at the end of the utterance end due to gradient vanishing. In order to reduce the sequence length processed by GRUs, we sub-sample [7, 22] the hidden activations along the time domain for every bi-direction GRU layer (see Fig. 5.4). This allowed us to extract a representation roughly at a syllable level in a given utterance. On the other hand, this significantly reduced the computational time for both training and prediction, which enables us to use bi-directional GRU for real time intent and/or domain classification. Given the encoder output, a max-pooling layer along the time axis is used to compress it into a fixed dimension vector. This is followed by a fully-connected feed-forward layer. Finally, a softmax layer is used to compute the posterior probability of intents or domains.

5.5 Experiments

We train and evaluate our models on an in-house dataset containing VR spoken commands collected for that purpose. The dataset is close in spirit to ATIS corpus [66]. The dataset contains about 320 hours of near field annotated data collected from a diverse set of more than 1000 de-identified speakers. It was recorded in two scenarios: scripted and free speech. The scripted half is read speech with a fixed script. For the free speech part, the participants were asked to achieve certain goal using any phrasing, then these utterances were transcribed. Every utterance has transcription as well as meta information including a domain label and an intent label. After cleaning data we extracted 5 domains and 35 distinct intents. Roughly 11,000 utterances, totaling 10 hours audio, are used as the evaluation set.

For all our experiments we used log-Mel filter-bank features. We used an encoder-decoder architecture [150]. The encoder is a 4 layer bidirectional GRU network with 256 units every layer. The output was sub-sampled with a stride of 2 at every layer to reduce the length of the representation. The decoder network takes the output of the encoder and aggregates them with a max-pooling layer and puts it through a 1-layer feed-forward network (hidden size 1024) to produce either the domain or intent class. The network was optimized with Adam [78] algorithm until convergence on the validation set. We used the *batch normalization* [73] for every feed-forward connection to speed up the convergence of this network.

The baseline NLU model for our experiments was a recurrent network similar to our model with an exception that we did not use sub-sampling. The input word was represented as one-hot vector and a trainable embedding was used. This model is close to state of the art models and ones used in production [34, 86]. This network was evaluated in two regimes: using the transcript text, and using the recognized text. The former shows the upper bound for our models and corresponds to the perfect speech recognition. The later regime was transcribed by a speech recognizer: the core part of AM is a four-layer CD-HMM-LSTM network[129] with 800 memory cells in each layer, trained on the same 320 hours training data with a cross-entropy criterion to predict posterior probabilities of 6,133 clustered content-dependent states; the vocabulary size is 80,000 and the LM is trained on the transcribed 320 hours speech and interpolated with a large background LM which is trained on other sources; there are roughly 200M n-grams in the LM. Our ASR system achieved 3.5% word error rate on the evaluation set. Each utterance from the evaluation set was recognized with this ASR and inputed to the baseline NLU network. In order to provide the fair comparison we did not use any external data in both cases. No pretrained embeddings or dictionary look-up was used.

We also emulate the real-world situation where the input to the SLU system is noisy. Both training and evaluation datasets were corrupted by convolving with recorded room impulse responses (RIRs) whose T60 times ranges from 200ms to 1 second. Background noise was added as well: for training data, the SNR ranges from 5 to 25dB, while for evaluation data, the SNR ranges from 0 to 20dB. Every training utterance is distorted 2 times by using different RIRs, sources of background noise and SNRs. This results in a 600 hours noise-corrupted training set. Utterances in evaluation set are only distorted once. Both the ASR system and our end-to-end system were retrained on this noise-corrupted training set. Due to the reverberation and relatively strong background noise, the ASR has considerably higher word error rate (28.6% WER).

Figure 5.5: Filter-bank features and corresponding saliency maps for our speech-to-domain model. It can be seen that the model responds to "what's the weather" part and "play" or "spotify" from the second example. Top image is "weather" domain, bottom image is "music" domain.

5.6 Results and discussion

We first present results of domain classification. Five domains in this corpus are "music," "weather," "news," "sports," "opt-in, opt-out." The results for domain recognition are summarized in Table 5.1. The performance in "Transcript text" row is evaluated on perfect transcriptions that correspond to human transcription and such performance is not achievable with the current ASR system. For qualitative analysis of our end-to-end model, we visualize the saliency map (gradients w.r.t. input) in Figure 5.5 for a selected utterance. We notice that the position of the network response corresponds to meaningful positions in the utterance. As it can be seen in Table 5.1, the accuracy is close to perfect on this dataset. Therefore we continue with a more challenging task of the intent classification.

Table 5.1: Results for domain classification. The first row corresponds to evaluation on clean transcripts, the maximum performance achievable with this model.

Input	**Accuracy, %**
Transcript text	99.2
Recognized text	98.1
Audio	97.2

Examples of intents are "Learn about the weather in your location," "Learn about the weather in different location," "Learn the results of a completed sports match, game, tournament," "Learn the status of an ongoing sports match, game, tournament." These classes are more fine grained and require deeper understanding of a given utterance. For the ablation study we report the performance of variations of our end-to-end model: initially, we use last-layer hidden activations at the end of both left-to-right and right-to-left directions as the encoder output; this is compared with using max pooling to aggregate all the hidden activations in the last layers in Table 5.2, which shows that using max-pooling activations increases performance. We hypothesize that due to the long input speech sequence (ranging from 100 to 1,000 input frames), the activations in the last time step may not sufficiently summarize the utterance. This is contrary to the NLU model using text as input, whereas using hidden activations from the last time step usually suffice, as the input length is relatively small (a majority of them are less than 20 words). We also observe that using batch normalization (BN) in both encoder and decoder significantly improves the classification accuracy, which indicates that our end-to-end model is difficult to optimize due to the long input sequence. The results for the noise-corrupted data are also reported in the bottom part of Table 5.2. The performance of both models degrades significantly. A potential advantage of our end-to-end approach to SLU is that since the model is optimized under the same criterion, the model can be made very compact; this is demonstrated in the same table by comparing the number of parameters of neural networks in standard SLU system and our system. Note that the parameters in the LM are not included, although it is usually two or three magnitude more than AM in ASR. Due to the compactness of our end-to-end model, we are able to predict the intent or domain with a real time factor around 0.002, which makes the use of bi-GRU in our model possible for real time applications.

With this work we hope to start a discussion on the topic of the audio-based SLU. Although, the end-to-end approach does not show superior performance, it provides a promising research direction. With significantly less parameters our system is able to reach 10% relatively worse accuracy. One of the future directions for this work is to encompass the slot filling task into this framework. It requires simultaneous prediction of a word and a slot. This can be tackled with the attention-based networks. Other directions are to explore different architectures for the decoder, such as using different pooling strategies, using deeper networks and incorporating convolutional transformations.

Table 5.2: Results for intent classification. As mentioned above, the first row is the maximum performance achievable with this model. We report the number of parameters for the models used to demonstrate that a much smaller end-to-end model achieves reasonable performance.

Input	**Parameters**	**Accuracy, %**
Text input		
Transcript	15.5M	84.0
Recognized	15.5M	80.9
Clean audio features		
no BN, no max-pool	0.4M	71.3
no BN, max-pool	0.4M	72.5
BN, max-pool	0.4M	74.1
Noisy audio features		
Recognized text	15.4M	72.0
Audio	0.4M	64.9

Chapter 6

Conclusion

Modern deep learning models enabled rapid advancement in many areas. In particular, the research in recurrent neural networks allowed to achieve new breakthroughs in many tasks that involve sequential processing. This work focuses on a family of sequential tasks that concern speech processing. The papers presented in this work discussed several issues with modern deep learning systems for speech processing. The first issue is the robustness to the environmental conditions of the input audio in speech recognition. The second issue concerns regularizing recurrent neural networks for speech recognition as well as several other tasks. Finally, the third issue is the applicability of the popular end-to-end methods to the task of speech understanding. The presented works investigate and address these issues.

In the work on the adversarial adaptation we developed a method to train a speech recognition model that is robust to unseen environmental noises. We built upon the method of the adversarial adaptation that was introduced in a prior work. We reformulated the robustness training as the domain adaptation. Every noise type was considered as a separate domain. Then we trained a network to classify which domain is the input. This network branched out off the acoustic model network. Then, we trained *not* to classify the domain correctly in the main network while performing the primary task of recognition. In the result, the feature at the point where the acoustic model branches into two sub-networks is discriminative towards the primary task and not discriminative towards the domain recognition task. Therefore, this feature is invariant with respect to the domain.

We benchmarked the proposed adversarial invariant training model with a popular Aurora-4 dataset. We constructed a series of six experiments adding each environmental noise condition at a time. This allowed us to construct a diverse set to test the applicability of the proposed method. We found that the invariance training is the most helpful in the cases when the number of noise types is small.

One of the benefits of using the adversarial invariance training is that it does not introduce any overhead at the inference time. The domain classifier is discarded and the adversarial loss is not computed. Therefore, the adversarial domain adaptation can be thought as a form of regularization. A model trained with the adversarial domain adaptation is a drop-in replacement for any existent model in production. On the other side, the adversarial loss significantly complicates training. We observed that the training is less stable, requires a careful hyper-parameter fine-tuning, and a careful selection of the particular loss.

The second paper presented in this work discusses how to generate long consistent sequences using recurrent neural networks. Indeed, the widely-used networks such as simple Elman recurrent networks, LSTMs, and GRUs frequently struggle to keep consistency throughout a long generated output.

We proposed to improve generative networks that contain some recurrent hidden state. In the nutshell, our method helps the generation by introducing a regularizer that is a second recurrent neural network (we call it twin network) trained to produce the same sequence in the opposite direction. The primary and the twin network were tied together via the L2 loss. Therefore, the primary network is incentivised to have a hidden state that is predictable of the essential information for generating all the future outputs.

We tested this method on a multitude of tasks. We found that the twin network method performs consistently good across various tasks and datasets. First, we tested our proposed system on a toy task of pixel-by-pixel generation of images. Inspired by the progress, we conducted our experiments sequence generation tasks such as end-to-end speech recognition, image captioning, language modelling. For further insight, we analysed the L2 loss for the training system. We found that this loss indicates inputs that are surprising for the network.

As a regularization method, the training with the twin network is a drop-in replacement for any potential generative recurrent network. This method does not have any overhead during the inference time. In order to minimize the overhead during the training time, we explored the possibility to share parameters between the primary network and the twin. We found that our approach works best for the conditional generation tasks such as speech recognition and image captioning. For the tasks of language modelling and image generation, the improvement margins are smaller. We connected this to the entropy present in the generated sequence. The entropy is much smaller for the conditional tasks compared to the unconditional ones.

The third work in this thesis concerns spoken language understanding. Speech recognition technology is rarely used in isolation. Most commonly the recognized speech is then used for some down-stream task. One of such tasks is the language understanding. A typical system for spoken language understanding consists of two components: the speech recognizer which converts audio to text, and the natural language processing part that aggregates the text. Chaining two separate systems optimized for two different losses may result into aggregation and amplification of errors. In our work we investigated the feasibility of using end-to-end approach to spoken language understanding. Inspired by the ideas from previous works on end-to-end speech recognition, we implemented a system for intent recognition from audio features. We conducted our experiments on a large-scale in-house dataset. The experiments showed a promising direction in this work. We were able to achieve results comparable to two-stage system. Furthermore, we analysed the trained system and visualized the trained features. We found that the trained model directs the semantic attention to the appropriate regions of the input audio signal.

The work on the end-to-end intent recognition is a foundation for the future research on end-to-end systems for spoken language understanding. One of the promising future research directions is the end-to-end slot filling. Our proposed system can be extended for slot filling by introducing a recurrent pointer network that would fill the slot step-by-step. Other approaches include the architectures based on the memory networks or reinforcement learning.

While each of the papers described above is a small step for the machine learning research, I believe that they play a role in the bigger picture. The methods proposed help to improve the accuracy of recognising speech, improve the consistency of generated sequences, and potentially improve the performance of intent recognition. Ultimately, this will help to advance the field of artificial intelligence. It is crucial to have a natural interface for an advanced enough artificial intelligence agent. I strongly believe that such an interface has to be a voice interface. The reasons for my belief include that the voice interface allows faster information exchange than the text input, as well as shorter times for every exchange.

Unfortunately, our technology for speech recognition and understanding is not advanced enough for fluent communication with existent weak digital assistants. It is crucial to improve the existing technology of speech processing to accelerate the adoption and development of artificial intelligence.

Acknowledgments

I would like to thank the developers of Theano deep learning framework [13], Blocks library for Theano and Fuel data management library [158], speech recognition toolkit Kaldi [117].

I thank Samsung for support of this research. I thank IBM Research, Facebook AI, and Element AI for support of my research and fruitful collaborations during my internships. I especially thank Compute Canada and Calcul Québec for providing computational resources for a part of this work.

I thank following people for insightful conversations throughout my studies in no particular order: Jose Sotelo, Alexandre de Brébisson, Kundan Kumar, Samy Bengio, Asja Fischer, Mathias Berglund, Çaglar Gulçehre, Pascal Lamblin, Roland Memisevic, Karan Grewal, Tim Cooijmans, Loïc Barrault, Junyoung Chung, Inchul Song, Jörg Bornschein, Christopher Beckham, Martin Arjovsky, Sungjin Ahn, Nicolas Ballas, Jason Yosinski, Henry Choi, David Vázquez, Kelvin Xu, David Warde-Farley, Francesco Visin, Samira Shabanian, Negar Rostamzadeh, Adriana Romero, Soroush Mehri, Daniel Im, Konstantinos Drossos, Felix Hill, Ian Goodfellow, Orhan Firat, Mélanie Ducoffe, Yann Dauphin, Aaron Courville, Christopher Pal, Devon Hjelm, Hugo Larochelle, Mathieu Germain, Margaux Luck, Joao Felipe Santos, Olivier Breuleux, Maxime Chevalier-Boisvert, Cem Subakan, Eugene Belilovsky, Kris Sankaran, Florian Golemo, Jason Jo, Devansh Arpit, Joseph Paul Cohen, Harm De Vries, Dendi Suhubdy, Xavier Bouthillier, Sandeep Subramanian, Sarath Chandar, Guillaume Alain, Sai Rajeshwar, Yaroslav Ganin, Anirudh Goyal, Tegan Maharaj, Akram Erraqabi, Ying Zhang, Saizheng Zhang, Li Yao, Eugene Vorontsov, Bart van Merriënboer, Chiheb Trabelsi, Iulian Vlad Serban, Mohammad Pezeshki, Vincent Michalski, William Fedus, Zhouhan Lin, César Laurent, David Scott Krueger, Kyle Kastner, Sina Honari, Sherjil Ozair, Giancarlo Kerg, Rithesh Kumar, Faruk Ahmed, Michael Noukhovitch, Olexa Bilaniuk, Shawn Tan, Olivier Mastropietro, Alex Lamb, Evan Racah, Tristan Deleu, Loren Lugosch, Andrea Eunbee Jang, Carolyne Pelletier, Yasmeen

Hitti, Ines Moreno, Jakub Sygnowski, Stanisław Jastrzębski, Konrad Żołna, Chen Xing, Titouan Parcollet, Marcin Moczulski.

Bibliography

[1] K. Aizawa, Y. Nakamura, and S. Satoh. *Advances in Multimedia Information Processing-PCM 2004: 5th Pacific Rim Conference on Multimedia, Tokyo, Japan, November 30-December 3, 2004, Proceedings*. Springer, 2004.

[2] A. G. ALIAS PARTH GOYAL, A. Sordoni, M.-A. Côté, N. R. Ke, and Y. Bengio. Z-Forcing: Training stochastic recurrent networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6713–6723. Curran Associates, Inc., 2017.

[3] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri. OpenFst: A general and efficient weighted finite-state transducer library. In J. Holub and J. Žd'árek, editors, *Implementation and Application of Automata*, number 4783 in Lecture Notes in Computer Science, pages 11–23. Springer Berlin Heidelberg, Jan. 2007.

[4] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, et al. Deep Speech 2: End-to-end speech recognition in English and Mandarin. In *ICML*, 2016.

[5] M. Arjovsky, A. Shah, and Y. Bengio. Unitary evolution recurrent neural networks. In *ICML*, 2016.

[6] P. Bachman. An architecture for deep, hierarchical generative models. In *NIPS*, 2016.

[7] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[8] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the ICLR 2015*, 2015.

[9] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio. End-to-end attention-based large vocabulary speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume abs/1508.04395, pages 4945–4949. IEEE, 2016.

[10] D. Bahdanau, D. Serdyuk, P. Brakel, N. R. Ke, J. Chorowski, A. C. Courville, and Y. Bengio. Task loss estimation for sequence prediction. 2015.

[11] L. R. Bahl, F. Jelinek, and R. L. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE transactions on pattern analysis and machine intelligence*, (2), 1983.

[12] S. Banerjee and A. Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, volume 29, pages 65–72, 2005.

[13] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. Theano: new features and speed improvements. 2012.

[14] E. Battenberg, J. Chen, R. Child, A. Coates, Y. Gaur, Y. Li, H. Liu, S. Satheesh, D. Seetapun, A. Sriram, and Z. Zhu. Exploring neural transducers for end-to-end speech recognition. *arXiv preprint arXiv:1707.07413*, 2017.

[15] Y. Benayed, D. Fohr, J. P. Haton, and G. Chollet. Confidence measures for keyword spotting using Support Vector Machines. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, volume 1, pages I–I. IEEE, 2003.

[16] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *arXiv:1506.03099 [cs, stat]*, 2015.

[17] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *JMLR*, 2003.

[18] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 1994.

[19] A. Bhargava, A. Celikyilmaz, D. Hakkani-Tür, and R. Sarikaya. Easy contextual intent prediction and slot detection. In *ICASSP*, 2013.

[20] L. Bottou. Curiously fast convergence of some stochastic gradient descent algorithms. In *Proceedings of the symposium on learning and data science, Paris*, 2009.

[21] H. Bourlard and N. Morgan. Hybrid HMM/ANN systems for speech recognition: Overview and new research directions. In *Adaptive Processing of Sequences and Data Structures*, pages 389–417. Springer, 1998.

[22] W. Chan, N. Jaitly, Q. Le, and O. Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *ICASSP*, 2016.

[23] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals. Listen, attend and spell. *arXiv:1508.01211 [cs, stat]*, Aug. 2015.

[24] D. Chen, A. Fisch, J. Weston, and A. Bordes. Reading Wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*, 2017.

[25] Y.-N. Chen, D. Hakkani-Tür, G. Tür, J. Gao, and L. Deng. End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding. In *Interspeech*, 2016.

[26] Y.-P. Chen, R. Price, and S. Bangalore. Spoken language understanding without speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6189–6193. IEEE, 2018.

[27] K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Empirical Methods of Natural Language Processing*, oct 2014.

[28] J. Chorowski, D. Bahdanau, K. Cho, and Y. Bengio. End-to-end continuous speech recognition using attention-based recurrent NN: First results. *arXiv:1412.1602 [cs, stat]*, Dec. 2014.

[29] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio. Attention-based models for speech recognition. *CoRR*, abs/1506.07503, 2015.

[30] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio. Attention-based models for speech recognition. In *NIPS*. 2015.

[31] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv:1412.3555 [cs, stat]*, 2014.

[32] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv:1412.3555*, 2014.

[33] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Gated feedback recurrent neural networks. In *ICML-15*, 2015.

[34] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *JMLR*, 2011.

[35] H. Daumé, J. Langford, and D. Marcu. Search-based structured prediction. *Machine learning*, 2009.

[36] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[37] A. B. Dieng, C. Wang, J. Gao, and J. Paisley. TopicRNN: A recurrent neural network with long-range semantic dependency. *arXiv preprint arXiv:1611.01702*, 2016.

[38] K. Drossos, P. Magron, S. I. Mimilakis, and T. Virtanen. Harmonic-percussive source separation with deep neural networks and phase recovery. In *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*, pages 421–425. IEEE, 2018.

[39] K. Drossos, S. I. Mimilakis, D. Serdyuk, G. Schuller, T. Virtanen, and Y. Bengio. MaD TwinNet: Masker-denoiser architecture with twin networks for monaural sound source separation. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2018.

[40] Y. Gal and Z. Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *NIPS*, 2016.

[41] M. Gales and S. Young. The application of hidden markov models in speech recognition. *Foundations and Trends in Signal Processing*, 1(3):195–304, 2008.

[42] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015.

[43] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett. DARPA TIMIT acoustic-phonetic continous speech corpus CD-ROM. NIST speech disc 1-1.1. *NASA STI/Recon technical report n*, 93, 1993.

[44] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. *arXiv:1705.03122*, 2017.

[45] M. Germain, K. Gregor, I. Murray, and H. Larochelle. MADE: Masked autoencoder for distribution estimation. In *ICML*, 2015.

[46] S. Ghannay, A. Caubrière, Y. Estève, N. Camelin, E. Simonnet, A. Laurent, and E. Morin. End-to-end named entity and semantic concept extraction from speech. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 692–699. IEEE, 2018.

[47] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[48] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio. Maxout networks. *ICML (3)*, 28:1319–1327, 2013.

[49] E. Grave, A. Joulin, and N. Usunier. Improving neural language models with a continuous cache. *arXiv preprint arXiv:1612.04426*, 2016.

[50] A. Graves. Practical variational inference for neural networks. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, editors, *NIPS*, pages 2348–2356. Curran Associates, Inc., 2011.

[51] A. Graves. Sequence transduction with recurrent neural networks. *arXiv:1211.3711 [cs, stat]*, 2012.

[52] A. Graves. Generating sequences with recurrent neural networks. *arXiv:1308.0850 [cs, stat]*, Aug. 2013.

[53] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *ICML-06*, 2006.

[54] A. Graves and N. Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1764–1772, 2014.

[55] A. Graves, N. Jaitly, and A.-r. Mohamed. Hybrid speech recognition with deep bidirectional LSTM. In *ASRU*, 2013.

[56] A. Graves, A.-R. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *ICASSP*, pages 6645–6649. IEEE, 2013.

[57] A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[58] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra. DRAW: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[59] C. Gulcehre, S. Chandar, and Y. Bengio. Memory augmented neural networks with wormhole connections. *arXiv preprint arXiv:1701.08718*, 2017.

[60] C. Gulcehre, F. Dutil, A. Trischler, and Y. Bengio. Plan, attend, generate: Planning for sequence-to-sequence models. *In Proc. of NIPS*, 2017.

[61] I. Gulrajani, K. Kumar, F. Ahmed, A. A. Taiga, F. Visin, D. Vazquez, and A. Courville. PixelVAE: A latent variable model for natural images. *arXiv preprint arXiv:1611.05013*, 2016.

[62] P. Haghani, A. Narayanan, M. Bacchiani, G. Chuang, N. Gaur, P. Moreno, R. Prabhavalkar, Z. Qu, and A. Waters. From audio to semantics: Approaches to end-to-end spoken language understanding. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 720–726. IEEE, 2018.

[63] A. Y. Hannun, C. Case, J. Casper, B. C. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng. Deep Speech: Scaling up end-to-end speech recognition. *CoRR*, abs/1412.5567, 2014.

[64] A. Y. Hannun, A. L. Maas, D. Jurafsky, and A. Y. Ng. First-pass large vocabulary continuous speech recognition using bi-directional recurrent dnns. *arXiv:1408.2873 [cs, stat]*, 2014.

[65] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[66] C. T. Hemphill, J. J. Godfrey, G. R. Doddington, et al. The ATIS spoken language systems pilot corpus. In *DARPA speech and natural language workshop*, 1990.

[67] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97, 2012.

[68] G. E. Hinton and R. R. Salakhutdinov. A better way to pretrain deep Boltzmann machines. In *Advances in Neural Information Processing Systems*, pages 2447–2455, 2012.

[69] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[70] S. Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91, 1991.

[71] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[72] W.-N. Hsu, Y. Zhang, and J. Glass. Unsupervised learning of disentangled and interpretable representations from sequential data. In *Advances in neural information processing systems*, pages 1878–1889, 2017.

[73] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[74] O. Kalinli, M. L. Seltzer, J. Droppo, and A. Acero. Noise adaptive training for robust automatic speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 2010.

[75] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, 2015.

[76] J. Keshet, D. Grangier, and S. Bengio. Discriminative keyword spotting. *Speech Communication*, 51(4):317–329, 2009.

[77] H. Ketabdar, J. Vepa, S. Bengio, and H. Bourlard. Posterior based keyword spotting with a priori thresholds. In *Ninth International Conference on Spoken Language Processing*, 2006.

[78] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980 [cs, stat]*, 2014.

[79] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[80] W. G. Knecht, M. E. Schenkel, and G. S. Moschytz. Neural network filters for speech enhancement. *IEEE Transactions on Speech and Audio Processing*, 1995.

[81] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur. A study on data augmentation of reverberant speech for robust speech recognition. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5220–5224. IEEE, 2017.

[82] J. Koutnik, K. Greff, F. Gomez, and J. Schmidhuber. A clockwork RNN. In *ICML-14*, 2014.

[83] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. 2012.

[84] D. Krueger, T. Maharaj, J. Kramár, M. Pezeshki, N. Ballas, N. R. Ke, A. Goyal, Y. Bengio, H. Larochelle, A. Courville, and C. Pal. Zoneout: Regularizing RNNs by randomly preserving hidden activations. 2016.

[85] D. Krueger and R. Memisevic. Regularizing RNNs by stabilizing activations. *arXiv:1511.08400*, 2015.

[86] S. Lai, L. Xu, K. Liu, and J. Zhao. Recurrent convolutional neural networks for text classification. In *AAAI*, 2015.

[87] A. M. Lamb, A. G. A. P. Goyal, Y. Zhang, S. Zhang, A. C. Courville, and Y. Bengio. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pages 4601–4609, 2016.

[88] Y. LeCun, C. Cortes, and C. J. Burges. The MNIST database of handwritten digits. *Available: http://yann.lecun.com/exdb/mnist*, 1998.

[89] C. Lengerich and A. Hannun. An end-to-end architecture for keyword spotting and voice activity detection. *arXiv preprint arXiv:1611.09405*, 2016.

[90] J. Li, W. Monroe, and D. Jurafsky. Learning to decode for future success. 2017.

[91] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *European conference on computer vision*, 2014.

[92] Q. Liu, Y. Qian, and K. Yu. Future vector enhanced LSTM language model for LVCSR. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 104–110. IEEE, 2017.

[93] J. Lu, C. Xiong, D. Parikh, and R. Socher. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. *In Proc. of CVPR 17*, 2017.

[94] L. Lugosch, M. Ravanelli, P. Ignoto, V. S. Tomar, and Y. Bengio. Speech model pre-training for end-to-end spoken language understanding. *arXiv preprint arXiv:1904.03670*, 2019.

[95] X. Ma and E. Hovy. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. *arXiv preprint arXiv:1603.01354*, 2016.

[96] G. Melis, C. Dyer, and P. Blunsom. On the state of the art of evaluation in neural language models. *arXiv preprint arXiv:1707.05589*, 2017.

[97] S. Merity, N. S. Keskar, and R. Socher. Regularizing and optimizing LSTM language models. *arXiv preprint arXiv:1708.02182*, 2017.

[98] Y. Miao, M. Gowayyed, and F. Metze. EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding. *arXiv:1507.08240 [cs, stat]*, 2015.

[99] T. Mikolov. Recurrent neural network based language model. 2010.

[100] T. Mikolov, S. Kombrink, L. Burget, J. Černockỳ, and S. Khudanpur. Extensions of recurrent neural network language model. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011.

[101] S. I. Mimilakis, K. Drossos, J. F. Santos, G. Schuller, T. Virtanen, and Y. Bengio. Monaural singing voice separation with skip-filtering connections and recurrent inference of time-frequency mask. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 721–725. IEEE, 2018.

[102] M. Mohri, F. Pereira, and M. Riley. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88, Jan. 2002.

[103] T. Moon, H. Choi, H. Lee, and I. Song. RNNDROP: A novel dropout for RNNs in ASR. In *ASRU*, 2015.

[104] F. Morbini, K. Audhkhasi, R. Artstein, M. Van Segbroeck, K. Sagae, P. Georgiou, D. R. Traum, and S. Narayanan. A reranking approach for recognition and classification of speech input in conversational dialogue systems. In *SLT*, 2012.

[105] N. Morgan and H. Bourlard. Continuous speech recognition. *IEEE signal processing magazine*, 1995.

[106] A. Narayanan and D. Wang. Joint noise adaptive training for robust automatic speech recognition. In *ICASSP*, 2014.

[107] A. Neelakantan, L. Vilnis, Q. V. Le, I. Sutskever, L. Kaiser, K. Kurach, and J. Martens. Adding gradient noise improves learning for very deep networks. 2015.

[108] U. Neisser, G. Boodoo, T. J. Bouchard Jr, A. W. Boykin, N. Brody, S. J. Ceci, D. F. Halpern, J. C. Loehlin, R. Perloff, R. J. Sternberg, et al. Intelligence: Knowns and unknowns. *American psychologist*, 51(2):77, 1996.

[109] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv:1609.03499*, 2016.

[110] A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

[111] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[112] N. Parihar and J. Picone. Signal processing for client/server applications in next generation cellular telephony. 1998.

[113] N. Parihar and J. Picone. Aurora working group: Dsr front end lvcsr evaluation au/385/02. *Inst. for Signal and Information Process, Mississippi State University, Tech. Rep*, 2002.

[114] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le. Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*, 2019.

[115] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *ICML*, 2013.

[116] D. B. Paul and J. M. Baker. The design for the wall street journal-based csr corpus. In *Proceedings of the workshop on Speech and Natural Language*, pages 357–362. Association for Computational Linguistics, 1992.

[117] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, et al. The Kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, 2011.

[118] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv:1511.06434*, 2015.

[119] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8), 2019.

[120] T. Raiko, Y. Li, K. Cho, and Y. Bengio. Iterative neural autoregressive distribution estimator nade-k. In *NIPS*, 2014.

[121] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. Sequence level training with recurrent neural networks. 2015.

[122] M. Ravanelli, D. Serdyuk, and Y. Bengio. Twin regularization for online speech recognition. *arXiv preprint arXiv:1804.05374*, 2018.

[123] S. V. Ravuri and A. Stolcke. Recurrent neural network and LSTM models for lexical utterance classification. In *Interspeech*, 2015.

[124] F. Rosenblatt. Perceptron simulation experiments. *Proceedings of the IRE*, 48(3):301–309, 1960.

[125] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

[126] S. T. Roweis. One microphone source separation. In *Advances in neural information processing systems*, pages 793–799, 2001.

[127] T. N. Sainath, B. Kingsbury, B. Ramabhadran, P. Fousek, P. Novak, and A.-r. Mohamed. Making deep belief networks effective for large vocabulary continuous speech recognition. In *ASRU*, 2011.

[128] T. N. Sainath, R. J. Weiss, A. Senior, K. W. Wilson, and O. Vinyals. Learning the speech front-end with raw waveform CLDNNs. In *Interspeech*, 2015.

[129] H. Sak, A. Senior, and F. Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Interspeech*, 2014.

[130] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training GANs. *arXiv:1606.03498*, 2016.

[131] T. Salimans, D. P. Kingma, and M. Welling. Markov chain monte carlo and variational inference: Bridging the gap. *arXiv preprint arXiv:1410.6460*, 2014.

[132] G. Saon, G. Kurata, T. Sercu, K. Audhkhasi, S. Thomas, D. Dimitriadis, X. Cui, B. Ramabhadran, M. Picheny, L.-L. Lim, et al. English conversational telephone speech recognition by humans and machines. *arXiv:1703.02136*, 2017.

[133] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny. Speaker adaptation of neural network acoustic models using i-vectors. In *ASRU*, 2013.

[134] R. Sarikaya, G. E. Hinton, and A. Deoras. Application of deep belief networks for natural language understanding. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 2014.

[135] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 1997.

[136] F. Seide, G. Li, and D. Yu. Conversational speech transcription using context-dependent deep neural networks. In *Interspeech*, 2011.

[137] S. Semeniuta, A. Severyn, and E. Barth. Recurrent dropout without memory loss. 2016.

[138] A. Senior and I. Lopez-Moreno. Improving DNN speaker independence with i-vector inputs. In *ICASSP*, 2014.

[139] I. V. Serban, A. Sordoni, R. Lowe, L. Charlin, J. Pineau, A. C. Courville, and Y. Bengio. A hierarchical latent variable encoder-decoder model for generating dialogues. 2017.

[140] D. Serdyuk, K. Audhkhasi, P. Brakel, B. Ramabhadran, S. Thomas, and Y. Bengio. Invariant representations for noisy speech recognition. *Neural Information Processing Systems End-to-end Learning for Speech and Audio Processing Workshop*, 2016.

[141] D. Serdyuk, N. R. Ke, A. Sordoni, A. Trischler, C. Pal, and Y. Bengio. Twin networks: Matching the future for sequence generation. In *International Conference on Learning Representations*, 2018.

[142] D. Serdyuk, Y. Wang, C. Fuegen, A. Kumar, B. Liu, and Y. Bengio. Towards end-to-end spoken language understanding. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5754–5758. IEEE, 2018.

[143] S. Shabanian, D. Arpit, A. Trischler, and Y. Bengio. Variational bi-LSTMs. *arXiv preprint arXiv:1711.05717*, 2017.

[144] C. Shan, J. Zhang, Y. Wang, and L. Xie. Attention-based end-to-end models for small-footprint keyword spotting. *arXiv preprint arXiv:1803.10916*, 2018.

[145] Y. Shinohara. Adversarial multi-task learning of deep neural networks for robust speech recognition. In *INTERSPEECH*, pages 2369–2372. San Francisco, CA, USA, 2016.

[146] Y. Shinohara. Adversarial multi-task learning of deep neural networks for robust speech recognition. *Interspeech*, 2016.

[147] D. Silver, H. van Hasselt, M. Hessel, T. Schaul, A. Guez, T. Harley, G. Dulac-Arnold, D. Reichert, N. Rabinowitz, A. Barreto, et al. The predictron: End-to-end learning and planning. *arXiv preprint arXiv:1612.08810*, 2016.

[148] H. Soltau, H. Liao, and H. Sak. Neural speech recognizer: acoustic-to-word LSTM model for large vocabulary speech recognition. *arXiv:1610.09975*, 2016.

[149] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 2014.

[150] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112, 2014.

[151] P. Swietojanski and S. Renals. Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models. In *SLT Workshop*, 2014.

[152] A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel. Value iteration networks. In *NIPS*, 2016.

[153] G. Tur and R. De Mori. *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons, 2011.

[154] G. Tur, L. Deng, D. Hakkani-Tür, and X. He. Towards deeper understanding: Deep convex networks for semantic utterance classification. In *ICASSP*, 2012.

[155] A. M. Turing. Computing machinery and intelligence. In *Parsing the Turing Test*, pages 23–65. Springer, 2009.

[156] C. K. Un, N. S. Kim, et al. Speech recognition in noisy environments using first-order vector Taylor series. *Speech Communication*, 1998.

[157] B. Uria, M.-A. Côté, K. Gregor, I. Murray, and H. Larochelle. Neural autoregressive distribution estimation. *JMLR*, 17(205), 2016.

[158] B. van Merriënboer, D. Bahdanau, V. Dumoulin, D. Serdyuk, D. Warde-Farley, J. Chorowski, and Y. Bengio. Blocks and Fuel: Frameworks for deep learning. *arXiv:1506.00619 [cs, stat]*, June 2015.

[159] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[160] A. Venkatraman, M. Hebert, and J. A. Bagnell. Improving multi-step prediction of learned time series models. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[161] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *CVPR*, 2015.

[162] C. Wang, H. Yang, C. Bartz, and C. Meinel. Image captioning with deep bidirectional LSTMs. In *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 2016.

[163] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[164] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML-15*, 2015.

[165] P. Xu and R. Sarikaya. Contextual domain classification in spoken language understanding systems using recurrent neural network. In *ICASSP*, 2014.

[166] K. Yao, D. Yu, F. Seide, H. Su, L. Deng, and Y. Gong. Adaptation of context-dependent deep neural networks for automatic speech recognition. In *SLT Workshop*, 2012.

[167] K. Yao, G. Zweig, M.-Y. Hwang, Y. Shi, and D. Yu. Recurrent neural networks for language understanding. In *Interspeech*, 2014.

[168] T. Yao, Y. Pan, Y. Li, Z. Qiu, and T. Mei. Boosting image captioning with attributes. *arXiv preprint arXiv:1611.01646*, 2016.

[169] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo. Image captioning with semantic attention. In *CVPR*, 2016.

[170] D. Yu, M. L. Seltzer, J. Li, J.-T. Huang, and F. Seide. Feature learning in deep neural networks – studies on speech recognition tasks. *arXiv:1301.3605*, 2013.

[171] D. Yu, K. Yao, H. Su, G. Li, and F. Seide. KL-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition. In *ICASSP*, 2013.

[172] W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent neural network regularization. 2014.

[173] M. D. Zeiler. ADADELTA: An adaptive learning rate method. *arXiv:1212.5701 [cs, stat]*, 2012.

[174] X. Zhang, J. Su, Y. Qin, Y. Liu, R. Ji, and H. Wang. Asynchronous bidirectional decoding for neural machine translation. *arXiv preprint arXiv:1801.05122*, 2018.