A Study of Adaptive Enhancement Methods for Improved Distant Speech Recognition

by

Andrew Richard Titus

S.B., Massachusetts Institute of Technology (2017)

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degree of

Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2018

Author .....

Department of Electrical Engineering and Computer Science

May 10, 2018

Certified by .....

James Glass

Senior Research Scientist

Thesis Supervisor

Certified by .....

Hao Tang

Postdoctoral Associate

Thesis Supervisor

Accepted by .....

Katrina LaCurts

Chair, Master of Engineering Thesis Committee

A Study of Adaptive Enhancement Methods for Improved Distant Speech Recognition

by

Andrew Richard Titus

Submitted to the Department of Electrical Engineering and Computer Science
on May 10, 2018, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Computer Science and Engineering

Abstract

Automatic speech recognition systems trained on speech data recorded by microphones placed close to the speaker tend to perform poorly on speech recorded by microphones placed farther away from the speaker due to reverberation effects and background noise. I designed and implemented a variety of machine learning models to improve distant speech recognition performance by adaptively enhancing incoming speech to appear as if it was recorded in a close-talking environment, regardless of whether it was originally recorded in a close-talking or distant environment. These were evaluated by passing the enhanced speech to acoustic models trained on only close-talking speech and comparing error rates to those achieved without speech enhancement. Experiments conducted on the AMI, TIMIT and TED-LIUM datasets indicate that decreases in error rate on distant speech of up to 33% relative can be achieved by these with only minor increases (1% relative) on clean speech.

Thesis Supervisor: James Glass
Title: Senior Research Scientist

Thesis Supervisor: Hao Tang
Title: Postdoctoral Associate

Acknowledgments

I am extremely grateful to my advisor, Jim Glass, for providing me with the resources and guidance necessary to complete this thesis in the fast-paced M.Eng. program. Thank you for the opportunity to work with you in Spoken Language Systems.

I am also very grateful to my co-advisor, Hao Tang, who helped me numerous times with debugging my code, providing suggestions and new ideas, and editing my thesis. Thank you for taking such an interest in my work and making time to meet with me on a regular basis.

I also thank everyone who has helped me to improve my skills in software development and machine learning over the course of my studies, including my labmates in Spoken Language Systems and coworkers and fellow interns at Lawrence Livermore National Laboratory, Bloomberg LP and Apple, Inc. Thank you for teaching me both technical and non-technical skills in both the academic and corporate worlds.

Thank you to Tony Eng and the rest of the 6.UAT course staff for financially supporting me through my M.Eng. studies as a Teaching Assistant. I am very thankful for the opportunity to not only help improve the communication skills of my students, but also my own.

Last but certainly not least, thank you to Miranda, my friends and my family for providing me with love and support throughout my studies. Thank you for being there for me in both good times and bad times. I couldn't have done it without you.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Acoustic models for automatic speech recognition (ASR) systems have been significantly improved by advances in neural networks, including deep neural networks (Hinton et al., 2012), long short-term memory (LSTM) networks (Hsu et al., 2016; Graves et al., 2013) and convolutional neural networks (Sainath et al., 2015). Despite their successes, these models can still perform poorly on speech from domains that were not well-represented in the data with which they were trained (Yoshioka and Gales, 2015). As ASR systems become more prevalent in everyday life, it becomes increasingly important for them to perform well, even in these adverse conditions.

One such set of adverse conditions is referred to as distant speech recognition, in which the microphone is situated far from the speaker. ASR systems trained on speech data recorded by microphones placed close (within several inches) to the speaker tend to perform poorly in the distant speech domain due to reverberation effects and background noise (Feng et al., 2014), with degradations in word error rate (WER) above 30% absolute possible in some cases (Swietojanski et al., 2013). For many systems using speech interfaces, it is often only possible (or at least more common) for one to interact with the system from a distance. Additionally, because a significant portion of speech recognition research has historically only focused on performance in the close-talking domain, there is generally more speech data available for the close-talking domain than there is for the distant domain.

One technique to address this performance degradation on distant speech data

is *speech enhancement*. Speech enhancement systems have the goal of "enhancing" incoming speech by transforming distant or otherwise noisy speech to appear as if it is clean, close-talking speech via filtering, masking or other techniques (Lim and Oppenheim, 1979). This allows one to use a speech recognizer for distant speech that has only been trained on a proportionally larger amount of close-talking speech data. This also replaces the time-intensive task of manually labeling hundreds or thousands of hours of new data and has the additional benefit of being a useful preprocessing step in an online system.

A common issue with most speech enhancement systems is that although they can improve recognition results for systems evaluated on distant or noisy speech, they tend to have side effects on the spectra for clean, close-talking speech that can degrade recognition results when the system is evaluated on such speech (Xu et al., 2014). A common technique for addressing this issue is to train a close-talking speech detector that uses zero-crossing rate and energy information to determine whether the speech needs to be enhanced before passing it to a speech recognizer (Xu et al., 2015). While this method has helped to improve overall results, it does not address the lack of robustness to clean speech present in current speech enhancement models themselves.

Therefore, I decided to research and develop a variety of *adaptive* speech enhancement models and explore their abilities to improve distant speech recognition performance without degrading clean speech recognition performance or requiring separate detectors to be trained. Some models focus on simply performing adaptive speech enhancement before passing it to an existing acoustic model trained on only close-talking speech data, whereas others contain their own acoustic model trained on close-talking speech data in conjunction with the enhancement model.

The rest of the thesis is organized as follows. I discuss related work and background information in Chapter 2. I then describe some initial experiments with data augmentation via unsupervised domain adaptation in Chapter 3, whose challenges motivated the interest in the adaptive speech enhancement models described in Chapter 4. I discuss experimental setup and outcomes in Chapter 5 before concluding the thesis

with a discussion of results and possible future work in Chapter 6.

Chapter 2

Related Work

Researchers have approached the problem of speech enhancement at various stages of the feature extraction pipeline for ASR systems. *Linear filtering* approaches seek to dereverberate the signal before features are extracted by attempting to find an optimal filter that cancels out the room impulse response (RIR) by blind deconvolution (Gillespie and Atlas, 2003; Hopgood and Rayner, 2003). *Spectral enhancement* approaches seek to enhance the speech in the frequency domain after a short-time Fourier transform (STFT) is performed on the signal by directly modifying the STFT coefficients with moving-average or predictive estimators (Yoshioka et al., 2012). Finally, *feature enhancement* approaches have been developed to denoise logarithmically-compressed feature vectors (for example, Mel-frequency cepstral coefficients (MFCCs) or log Mel-frequency filter bank features) directly before being passed into the acoustic model for the speech recognizer (Feng et al., 2014; Xu et al., 2014, 2015).

Regardless of where a given speech enhancement technique is being applied, one can consider the goal of the technique to be to retain the linguistic components of the speech signal necessary for speech recognition while filtering out noise and reverberation components of the signal that do not affect the spoken content of the signal (referred to by Hsu et al. (2017b) collectively as the "nuisance attribute"). If close-talking speech is considered to be the "source" domain and distant speech is considered to be the "target" domain, then speech enhancement can therefore be viewed as a *domain adaptation* problem, where the goal is to take a model that

performs well on the source domain and adapt it to also generalize well to data drawn from the target domain by making use of linguistic components while ignoring the nuisance attribute.

It is therefore of interest to investigate common machine learning models used to learn data representations encoding these linguistic and non-linguistic components and consequentially models that can perform domain adaptation based on these components. Below I review three generic model families for learning data representations useful for domain adaptation: autoencoders, domain adversarial neural networks and generative adversarial networks.

2.1 Autoencoder Models

A common type of model used in machine learning to learn data representations in an unsupervised fashion (i.e., without explicit labeling of what the representation should be) is the *autoencoder* (Bourlard and Kamp, 1988). Autoencoders are machine learning models that consist of two neural networks: an *encoder* Φ and a *decoder* Ψ. The autoencoder is trained end-to-end by using backpropagation and stochastic gradient descent to minimize *reconstruction loss*, a measure of how well the autoencoder recovered the original input after encoding and decoding. Formally, for input data $\mathcal{X} = \mathbb{R}^d$ and latent space $\mathcal{F} = \mathbb{R}^J$, define an encoder $\Phi : \mathcal{X} \to \mathcal{F}$ and decoder $\Psi : \mathcal{F} \to \mathcal{X}$, where $\Phi, \Psi = \arg\min_{\Phi,\Psi} \|x - (\Psi \circ \Phi)(x)\|^2$ for $x \in \mathcal{X}$.

A problem that commonly arises when there is insufficient regularization applied to the learning algorithm being used is when the autoencoder simply learns an identity function of its input without the encoder learning any salient features of the underlying data distribution. Another way of stating this is that the reconstruction loss for any $x \in \mathcal{X}$ will be nearly 0, even if x is not drawn from the generative process for the given dataset $S = \{x_1, \ldots, x_n\}$ where $x_i \in \mathcal{X}$.

2.1.1 Variational Autoencoders (VAEs)

One way to regularize autoencoders being trained on large datasets with intractable posterior distributions is to use the Auto-encoding Variational Bayes (AEVB) algorithm to optimize recognition models using a Stochastic Gradient Variational Bayes (SGVB) estimator. When neural networks are used for the recognition model, the resulting model is known as a *variational autoencoder* (VAE) (Kingma and Welling, 2014).

Formally, let the input data be n samples of some discrete or continuous variable $x \in \mathcal{X}$. Let ϕ be the learned model parameters and θ be the model parameters for the process generating $z \in \mathcal{F}$. Let the prior over the latent variables z be a centered isotropic multivariate Gaussian $p_\theta(z) = \mathcal{N}(z; 0, I)$. Additionally, let $p_\theta(x|z)$ be a multivariate Gaussian whose distribution parameters are computed from z with a multilayer perceptron. Because the true latent posterior $p_\theta(z|x)$ is intractable in this situation, one can instead approximate the latent posterior with a diagonal multivariate Gaussian $\log q_\phi(z|x^{(i)}) = \log \mathcal{N}(z; \mu^{(i)}, \sigma^{2(i)} I)$, where the mean $\mu^{(i)}$ and standard deviation $\sigma^{(i)}$ are outputs of the encoder E given input datapoint $x^{(i)}$.

The loss function used for VAEs (see Equation 2.1) then regularizes the model parameters ϕ by adding the negative Kullback-Leiber divergence of the approximated latent posterior $q_\phi(z|x)$ to the latent prior $p_\theta(z) = \mathcal{N}(z; 0, I)$. This allows VAEs to efficiently perform approximate inference of the true posterior $\log p_\theta(x|z)$ given L samples of z (using the reparametrization trick described in Kingma and Welling (2014)) and model parameters θ.

$$
\begin{aligned}
\mathcal{L}(\theta, \phi; x^{(i)}) &\simeq -D_{KL}(q_\phi(z|x)||p_\theta(z)) + \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(x^{(i)}|z^{(i,l)}) \\
&\simeq \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2\right) + \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(x^{(i)}|z^{(i,l)}) \\
&\text{where } z^{(i,l)} = \mu^{(i)} + \sigma^{(i)} \odot \epsilon^{(l)} \text{ and } \epsilon^{(l)} \sim \mathcal{N}(0, I)
\end{aligned}
\tag{2.1}
$$

A variation of the variational autoencoder, the *factorized hierarchical variational autoencoder* (FHVAE), was developed by Hsu *et al.* and has been shown to successfully disentangle sequence-level attributes (for example, speaker volume, noise environment, etc.) from segment-level attributes (linguistic content) in the latent space (Hsu et al., 2017a). These segment-level attributes were shown to improve recognition results for noisy speech recognition tasks on the Aurora-4 database when used as features.

2.1.2 Denoising Autoencoders (DAEs)

Another way to prevent autoencoders from learning identity functions is to corrupt the input to the model. Vincent et al. (2008) proposed the Denoising Autoencoder (DAE), where the learning criterion is modified so as to not just learn an accurate reconstruction of the input with the model, but to also be robust to partial destruction of the input by learning to map partially destroyed inputs to almost the same latent representation as the corresponding clean inputs. Formally stated, for clean input dataset $S_c = \{x_1, \ldots, x_n\}$ where $x_i \in \mathcal{X}$ and its corresponding dirty version $S_d = \{f_d(x_1), \ldots, f_d(x_n)\} = \{\widetilde{x}_1, \ldots, \widetilde{x}_n\}$ for $x_i \in S_c$ and some noising function $f_d : \mathcal{X} \to \mathcal{X}$, redefine the encoder Φ and decoder Ψ originally defined at the beginning of Section 2.1 to be $\Phi, \Psi = \arg\min_{\Phi,\Psi} \|x_i - (\Psi \circ \Phi)(\widetilde{x}_i)\|^2$ for corresponding datapoints $x_i \in S_c, \widetilde{x}_i \in S_d$. DAEs are then trained with similar optimization methods as standard autoencoders. Results indicate that DAEs can in fact learn better representations of input data suitable for learning tasks such as speech enhancement (Feng et al., 2014; Lu et al., 2013; Ishii et al., 2013) and image denoising (Xie et al., 2012). Additionally, multiple layers of DAEs can be stacked together to make deep learning models that learn useful higher-level representations for various problems (Vincent et al., 2010).

2.2 Domain Adversarial Neural Networks (DANNs)

Consider a setup similar to the one described for DAEs in Section 2.1.2, where both an annotated speech corpus and unannotated speech corpus are available, but no

correspondence between their datapoints is known. Formally, data $S_1 = \{x_1, \ldots, x_n\}$ where $x_i \in \mathcal{X}$ from one domain (for example, close-talking speech) are available in addition to data $S_2 = \{y_1, \ldots, y_m\}$ where $y_i \in \mathcal{X}$ from a different domain (for example, noisy speech), but no correspondence between the datapoints $x_i \in S_1$ and $y_i \in S_2$ is known. Additionally, define a set of L possible labels $\mathcal{Y} = \{0, 1, \ldots, L-1\}$. Assume that corresponding labels $T_1 = \{\ell_1, \ldots, \ell_n\}$ where $\ell_i \in \mathcal{Y}$ are available for data S_1 (i.e., labeled pairs (x_i, ℓ_i) are available), but no such labels are known for S_2.

In such a situation, it can be difficult to build models that predict labels with high accuracy for both data in S_1 and S_2. One way to build models that perform well on one domain and generalize to the other is to try to ensure that the learned representation of input data contains little to no information about the domain from which it originated while preserving enough information to predict labels. *Domain Adversarial Neural Networks (DANNs)*, originally proposed by Ganin et al. (2016), implement this idea by adding a *domain regressor* component to an existing neural network predictive model.

Formally, let this predictive neural network be the composition of a feature extractor $G_f(\cdot; \theta_f)$ and a label predictor $G_\ell(\cdot; \theta_\ell)$. The domain regressor $G_d(\cdot; \theta_d)$ therefore takes the feature outputs of G_f and predicts a value in $[0, 1]$ where a prediction of 0 indicates a belief that the features originated from data in S_1 and a prediction of 1 indicates a belief that the features originated from data in S_2. The predictive neural network and domain regressor can therefore be viewed as playing a two-player minimax game with the following value function $V(G_f, G_\ell, G_d)$, where λ is a tunable regularization parameter:

$$\begin{aligned}
\min_{G_f, G_\ell} \max_{G_d} V(G_f, G_\ell, G_d) &= \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_\ell(x_i, \ell_i) - \lambda \left(\frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_d(x_i, 0) + \frac{1}{m} \sum_{j=1}^{m} \mathcal{L}_d(y_j, 1) \right) \\
\text{where } \mathcal{L}_\ell(z, \ell) &= \log \frac{1}{G_\ell(G_f(z))_\ell} \\
\text{and } \mathcal{L}_d(z, c) &= c \log \frac{1}{G_d(G_f(z))} + (1 - c) \log \frac{1}{1 - G_d(G_f(z))}, c = \{0, 1\}
\end{aligned} \tag{2.2}$$

2.3 Generative Adversarial Networks (GANs)

In addition to learning representations from existing data, it is also possible to train models to generate new data that accurately resemble real data drawn from the unknown underlying probability distribution generating the real data. One such type of model that has enjoyed recent success is the Generative Adversarial Network (GAN) (Goodfellow et al., 2014). GANs consist of two models, a generator G that seeks to capture the underlying probability distribution of the real data and a discriminator D that seeks to determine whether a given sample originates from the real dataset or from G. D therefore acts as an "adversary" to G by challenging it to generate convincingly real-looking data.

As with other adversarial problems, these models are trained in a minimax fashion. First, define data distribution p_{data} on dataset $S = \{x_1, \ldots, x_n\}$ where $x_i \in \mathcal{X}$ and noise prior $p_z(z)$. Then, define a mapping $G(z; \theta_g)$ from noise z to data space $\mathcal{X}$, where generator G is a differentiable function represented by a multilayer perceptron with parameters θ_g. p_g is therefore the distribution over data x learned by G. Define another multilayer perceptron $D(x; \theta_d)$ that outputs a scalar representing the probability x came from p_{data} rather than G. The optimizer then alternates between training D by maximizing the probability of D assigning correct labels to both data from S and samples drawn from G, and training G by minimizing the probability that D correctly classifies samples produced by G as coming from G. This can be written as a two-player minimax game with value function $V(G, D)$ (see Equation 2.3).

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.3}$$

2.3.1 CycleGAN

It is possible to train systems that learn data representations that allow for transforming datapoints from one dataset to appear to have been generated from the generating distribution of a different dataset, even when no correspondence between datapoints

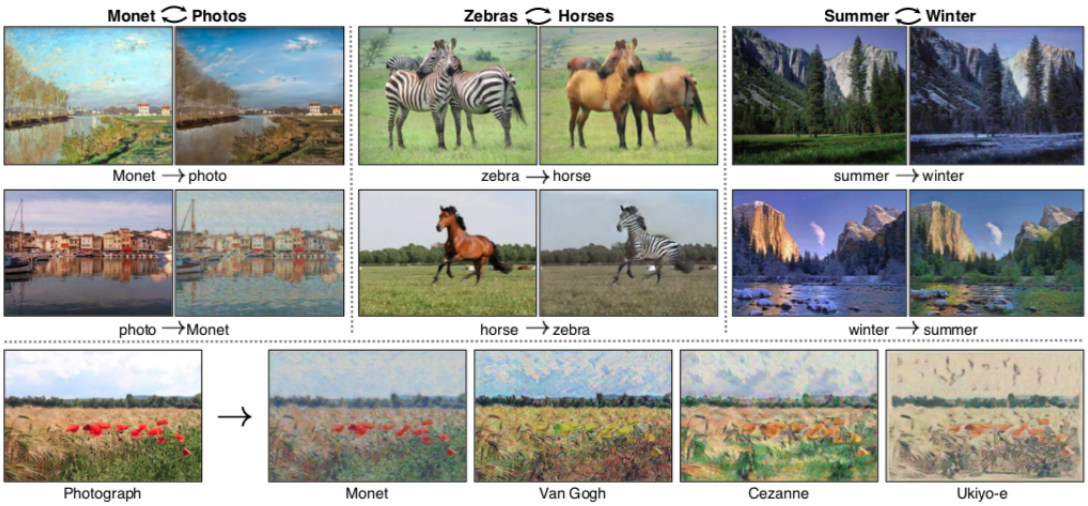

Figure 2-1: Sample image outputs from CycleGAN image-to-image translation model (Zhu et al., 2017)

in the two datasets is known (in contrast to Section 2.1.2). Formally, define datasets $S_1 = \{x_1, \ldots, x_n\}$ where $x_i \in \mathcal{X}$ and $S_2 = \{y_1, \ldots, y_m\}$ where $y_i \in \mathcal{X}$, where no correspondence between the data points in S_1 and S_2 is known. Additionally, define two models $A : \mathcal{X} \to \mathcal{X}$ and $B : \mathcal{X} \to \mathcal{X}$. The *forward cycle consistency* condition is then $B(A(x)) \approx x$ for $x \in S_1$ and the *backward cycle consistency* condition is $A(B(y)) \approx y$ for $y \in S_2$. These conditions were incentivized by Zhu et al. (2017) by using a cycle consistency loss based on the L1 norm (see Equation 2.4).

$$\mathcal{L}_{\text{cyc}}(A, B) = \|B(A(x)) - x\|_1 + \|A(B(y)) - y\|_1 \text{ for } x \in S_1, y \in S_2 \tag{2.4}$$

Results by Zhu et al. (2017) indicate that cycle consistency loss, when combined with GAN-style adversarial losses for discriminators D_1 (where S_1 is the real data and $B(y)$ for $y \in S_2$ is the simulated data) and D_2 (where S_2 is the real data and $A(x)$ for $x \in S_1$ is the simulated data), can be used to train image-to-image translation models that produce compelling results using unpaired data alone (see Figure 2-1).

Chapter 3

Initial Attempts at Data Augmentation via Unsupervised Domain Adaptation with Multidecoders

As a preliminary step in my thesis work, I investigated the setup described in Section 2.2, where speech data from both a close-talking (source domain) and distant recording environment (target domain) are available, but the data are not parallel (i.e., there is no known correspondence between close-talking and distant utterances) and word transcriptions are only available for the close-talking speech data. In this setup, one can only train acoustic models with the close-talking speech data, unless there is a way to label the distant speech data or use the labeled close-talking data to produce simulated, labeled distant speech data. As speech datasets can often exceed hundreds or even thousands of hours of audio, manually labeling additional data can quickly become unscalable. Therefore, I researched models that achieve better performance for the second option: data augmentation via unsupervised domain adaptation.

Attributes of speech data that are not correlated with linguistic changes can be referred to collectively as the *nuisance attribute* (Hsu et al., 2017b). Because the

speech data being considered for this setup have primarily non-linguistic changes (i.e., distance between microphone and speaker) between the two domains being examined, one might expect to see changes in the nuisance attribute across different domains, but relatively constant values across utterances from the same domain. One can therefore augment the data for the target domain by encoding labeled source domain data, transforming its nuisance attribute to that of the target domain, and decoding it to get simulated, labeled data that appears to have been generated by the target data distribution.

There are several such nuisance attribute transformations described by Hsu et al. (2017b) for performing data augmentation in an unsupervised domain adaptation setting with variational autoencoders (VAEs) that have been performed with encouraging results. However, they all require some form of estimation of the nuisance attribute (or the nuisance attribute *subspace*, assuming latent variables learned by a VAE are mapped to orthogonal subspaces). Therefore, I decided to investigate the possibility of a model that can implicitly estimate the nuisance attribute in the model itself without needing external estimation methods like these.

3.1 Multidecoder

I developed a new autoencoder-based domain adaptation architecture, the *Multidecoder*, which uses multiple decoders jointly trained end-to-end with a single encoder to allow one to decode a single encoded latent vector $z \in \mathcal{F} = \mathbb{R}^J$ into various domains. At a high level, a multidecoder is an autoencoder model where there is a single encoder function Φ followed by multiple decoder functions (Ψ_{src} for the source domain and $\Psi_{tar}^{(1)}, \cdots, \Psi_{tar}^{(T)}$ for T target domains). In most situations, there is only one target domain to which one is seeking to adapt the source domain data. Therefore, I consider T to be equal to 1 for the following description and analysis, although the model can be extended to values of T greater than 1.

A multidecoder that seeks to solve the domain adaptation problem can therefore be formally defined as one that achieves low reconstruction loss for both datapoints

drawn i.i.d. from the source distribution and the target distribution. In this setup, the source distribution p_{src} is defined on the source dataset $S_{src} = \{x_1, \ldots, x_n\}$ where $x_i \in \mathcal{X}$ and the target distribution p_{tar} is defined on the distant dataset $S_{tar} = \{y_1, \ldots, y_m\}$ where $y_i \in \mathcal{X}$. A multidecoder that successfully solves this problem would therefore have learned an encoder Φ that maps data from both domains into a common representation space (also referred to as an embedding or latent space), in addition to decoders Ψ_{src} and Ψ_{tar} that can faithfully reconstruct data from both domains using this common representation space.

Therefore, if the source domain data $S_{src} = \{x_1, \ldots, x_n\}$ where $x_i \in \mathcal{X}$ are close-talking speech and the target domain data $S_{tar} = \{y_1, \ldots, y_m\}$ where $y_i \in \mathcal{X}$ are distant speech, optimize the sum of the following two losses:

- **Close-talking reconstruction**: $\mathcal{L}_c^{recon} = \|x_i - (\Psi_{src} \circ \Phi)(x_i)\|^2$ for $x_i \in S_{src}$
- **Distant reconstruction**: $\mathcal{L}_d^{recon} = \|y_i - (\Psi_{tar} \circ \Phi)(y_i)\|^2$ for $y_i \in S_{tar}$

3.1.1 Architecture

Rather than processing speech data as raw waveforms, most ASR systems extract "frames" of features by sliding a window over the short-time Fourier transform (STFT) of the waveform (Mogran et al., 2004). As speech data is inherently sequential in time, it is useful for a multidecoder to take temporal context frames into consideration for reconstruction, rather than just using a single frame. In lieu of using recurrent connections (as with Hsu et al. (2016) and Graves et al. (2013), for example), I splice in f_L frames of features from the frames directly before the current frame and f_R frames from the frames directly after the current frame in time to get a feature vector that is a concatenation of $(f_L + f_R + 1)$ frames of features. For a frame with d-dimensional features, this results in a 2-dimensional feature matrix of size $((f_L + f_R + 1) \times d)$.

An architecture especially well-suited for processing 2-dimensional data is the convolutional neural network (CNN) (LeCun et al., 1995), known primarily for its successes in image processing (Simonyan and Zisserman, 2014). In a multidecoder, the encoder is a typical multilayer convolutional neural network, where each layer

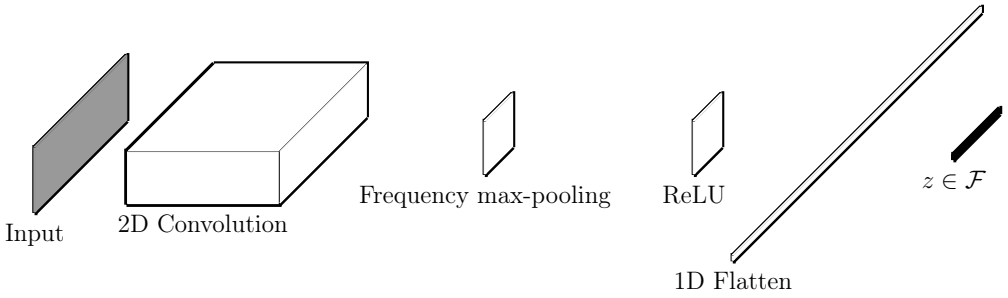

Figure 3-1: An example single layer CNN-based encoder for AMI multidecoder experiments

consists of several convolution filters that are convolved with its input features to yield feature maps as output. These feature maps are then passed through a rectified linear unit (ReLU) activation function (Glorot et al., 2011) before being downsampled by a max-pooling layer in the feature dimension only (Sainath et al., 2015). The encoder then flattens the last output to one dimension before passing it through a fully-connected layer to yield the latent vector z. This is depicted in Figure 3-1.

Because the decoders need to reconstruct z back to the original dimensionality of the input data, one approach is to make the decoder symmetric to the encoder. To do so with the multidecoder, each decoder first has a fully-connected layer that the latent vector z is passed through to yield the flattened input to the convolutional stages. Zero-padding is applied to the inputs of convolutional layers in the decoders so that the output is resized to the original input size to the corresponding convolutional layer in the encoder without requiring deconvolutional layers (Zeiler et al., 2010) or similar techniques. Additionally, instead of max-pooling layers, decoders use max-*unpooling* layers that make use of saved indices of the maximal values in the corresponding max-pooling layer in the encoder to upsample the input in the feature dimension and set all non-maximal values to zero (Zeiler and Fergus, 2014).

To prevent learning identity functions, experiments were conducted with using the following regularization techniques:

- **Variational autoencoder (VAE)**: instead of using a single latent vector, use two latent vectors $z_\mu \in \mathcal{F}$ and $z_\sigma \in \mathcal{F}$ for mean and log-variance, respectively,

and add Kullback-Leibler divergence into the loss function as described in Section 2.1.1.

- **Denoising autoencoder (DAE)**: corrupt the input feature matrices by zeroing out input elements randomly with probability p as described by Vincent et al. (2008).

- **Generative Adversarial Network (GAN)-style loss**: train a discriminator D to detect real versus fake output features in a minimax fashion with the multidecoder representing generator G as described in Section 2.3.

- **Domain adversarial loss**: as described in Section 2.2, train the multidecoder in a minimax fashion simulataneously with a discriminator D to detect whether latent vector z belongs to the close-talking or distant speech domain with the encoder representing generator G.

- **Cycle consistency loss**: without loss of generality over domains, for data in domain i, train with not only reconstruction loss, but also with reconstruction loss between the original data and the data passed through the decoder Ψ_j for the other domain j followed by Ψ_i. This resembles training with parallel data by using simulated data for domain j as described in Section 2.3.1, but I used L2 norm rather than L1 norm to be consistent with the reconstruction losses used elsewhere during training.

3.1.2 Nuisance attribute transformations for multidecoders

Fortunately, multidecoders are models that do not need to use a nuisance attribute transformation as described at the beginning of the chapter. While the methods described by Hsu et al. (2017b) involve changing the latent encoding itself before passing it through the decoder, multidecoders seek to learn a data representation that represents enough variance in the data for each of the decoders to reconstruct input data from its domain because the training regimen forces the encoder to minimize reconstruction losses incurred for both decoders. This allows each of the decoders

to not only preserve linguistic information, but also learn a function on the nuisance attribute to reconstruct speech attributes for its domain that are not related to linguistic information. Therefore, data augmentation for target domain T can be performed with a multidecoder simply by encoding labeled data from source domain S and decoding with the decoder corresponding to domain T. The quality of this learned transformation, however, is not guaranteed without using parallel data, so I conducted experiments to determine how effectively it creates simulated data for improving acoustic models.

3.2 Experiments

3.2.1 AMI Meeting Corpus

The dataset used for these initial multidecoder experiments was the AMI meeting corpus, a dataset that contains 100 hours of recordings of meetings between professionals in research and design workplace scenarios (Carletta et al., 2005). Multiple recording setups were used to get parallel recordings of the same dialogue, including individual headset microphones (the "IHM" setting), a single distance microphone (the "SDM1" setting) and eight distance microphones placed in an array (the "MDM8" setting). For these experiments, I only made use of the IHM and SDM1 (henceforth referred to as "SDM" for simplicity) data, with the IHM data modeling a close-talking speech scenario and the SDM data modeling a distant speech scenario. Word transcriptions are available for both of these setups, as they follow the same dialogue recorded in parallel, but in order to be consistent with the setup described at the beginning of this chapter, I only made use of the labels for the IHM data when training multidecoders and did not assume any utterances to be parallel. Additionally, to allow for rapid prototyping, I made use of a balanced subset of AMI that consisted of 10% of the original data (about 10 hours of speech).

3.2.2 Training

For feature extraction, I used the Kaldi speech recognition toolkit (Povey et al., 2011). 80 dimensional Mel-scale filter bank features were extracted with a 25 ms window and 10 ms frame shift, a common auditory approximation used in speech recognition (Mogran et al., 2004). Five frames of left context and five frames of right context are spliced into the feature vector for a resulting feature matrix size of 11×80. Batch size was fixed to 256 samples to balance memory and performance considerations with diversity of data, although multiple experiments were conducted to determine other hyperparameters.

Model development was performed using the PyTorch deep learning framework (Paszke et al., 2017). Weights in the network were initialized using Xavier initialization (Glorot and Bengio, 2010) and the models were optimized using the Adam optimizer with the parameters set to the defaults suggested by Kingma and Ba (2015) ($\beta_1 = 0.9$, $\beta_2 = 0.99$ and $\epsilon = 10^{-8}$) with the exception of the initial learning rate, which was determined to be optimal at 0.0001 rather than 0.001 in these experiments.

To train a given decoder Ψ_i, the reconstruction loss $\mathcal{L}_i^{recon}$ for a given minibatch passed through the shared encoder Φ and the given decoder Ψ_i was computed and then backpropagated to update their weights without modifying the weights of the other decoder. Minibatches for the two domains were passed through in an alternating fashion so that the model saw a balanced amount of data for the two decoders over the course of each epoch of training. The models were trained for 25 epochs or until 5 consecutive epochs of no improvement in the reconstruction loss on the development set were observed, whichever occurred first.

Kaldi was then used to train frame-level `tri3` hidden Markov models (HMMs) (see Guglani and Mishra (2018) for a thorough description of the `tri3` model), whose states were used as targets to train time-delay neural network (TDNN)-based HMM state recognizers (Povey et al., 2011; Cheng et al., 2017). The TDNN used contained 7 hidden layers with 450 units each and the same layer-wise context frames as described by Peddinti et al. (2018) that fed into a final layer consisting of a softmax over

each of the 3,984 HMM states. The model was then trained on the data described below in Section 3.2.3 for 20 epochs with a step size of 0.05. These frame-level state predictions can then be decoded into word-level predictions using trained transition and language models; however, this step was not taken for these experiments due to the disappointing frame-level results discussed below.

3.2.3 Results

I evaluated the frame-level classifiers based on Frame Error Rate (FER), a measure of how often the classifier predicts the correct frame-level state labels. The DAE variation of the multidecoder architecture was the only regularization technique used that actually led to any improvement in FER beyond the non-regularized multidecoder, so for brevity, I only report results for the best network setup (see Figure 3-2), which used 25% input noising. I used the following training setups for the frame-level classifiers:

- Baseline IHM: trained on only original IHM data S_{src}
- Baseline SDM: trained on only original SDM data S_{tar}
- IHM→IHM: trained on $\Psi_{src}(\Phi(x_i))$ for $x_i \in S_{src}$
- IHM→SDM: trained on $\Psi_{tar}(\Phi(x_i))$ for $x_i \in S_{src}$
- SDM→IHM: trained on $\Psi_{src}(\Phi(y_i))$ for $y_i \in S_{tar}$
- SDM→SDM: trained on $\Psi_{tar}(\Phi(y_i))$ for $y_i \in S_{tar}$
- IHM + IHM→SDM: trained on original IHM data S_{src} in addition to $\Psi_{tar}(\Psi(x_i))$ for $x_i \in S_{src}$

These frame-level classifiers are then evaluated on the IHM development set and SDM development set; results are shown in Table 3.1. Although some of the classifiers were able to nearly match the FER of the IHM baseline classifier on the IHM development set, none were able to make any substantial improvement towards the

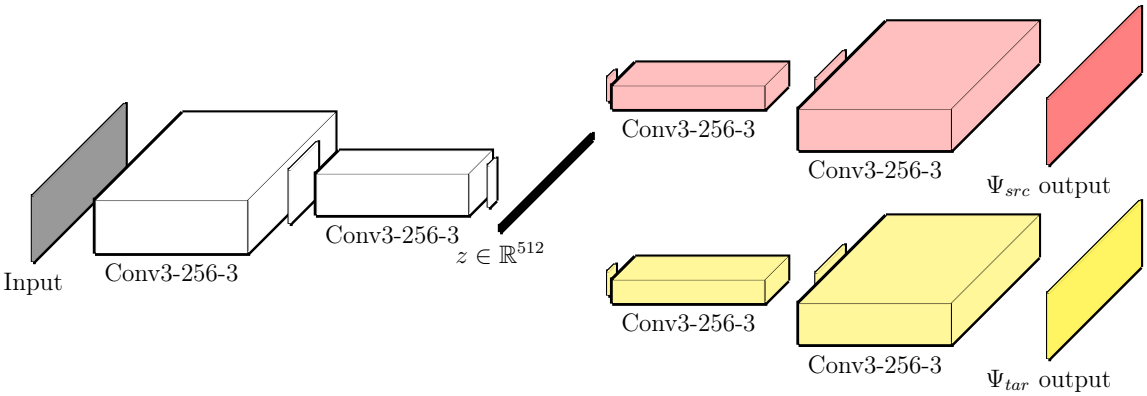

Figure 3-2: Optimal multidecoder architecture for AMI experiments. Convolutional layer parameters shown as "Conv⟨Receptive field size⟩-⟨Number of channels⟩-⟨Pool size⟩". ReLU not shown for simplicity.

Table 3.1: AMI 10% Subset: Unsupervised Domain Adaptation Results

Model	IHM Dev FER	SDM Dev FER
Baseline IHM	57.9%	86.7%
Baseline SDM	80.9%	77.3%
IHM→IHM	59.3%	87.0%
IHM→SDM	59.9%	85.9%
SDM→IHM	78.0%	82.1%
SDM→SDM	79.6%	**80.9**%
IHM + IHM→SDM	**58.1**%	85.8%

performance of the baseline SDM classifier on the SDM development set without also significantly degrading performance on the IHM development set. This indicates that despite the best efforts of the regularization techniques, the multidecoder in this specific training setup was unable to learn an encoder function and decoder functions that allowed the model to perform domain adaptation to a degree that improved results on a distant speech recognition task. It may still be possible to make architecture and/or dataset changes that allow a multidecoder to perform unsupervised domain adaptation, but these were not found in these preliminary experiments.

These disappointments motivated a desire to see how much better the performance could be if the domain adaptation were conducted in a supervised manner, where I instead assume that I have parallel recordings of speech in both close-talking and distant environments. This new setup and associated model architectures are

described in the following chapter.

Chapter 4

Adaptive Enhancement Models

After the difficulties in training unsupervised domain adaptation models, I decided to instead run experiments to determine the efficacy of *supervised* domain adaptation models, where recordings of the same speech are available for both a close-talking and distant speech setup. Although parallel recordings were assumed to be available in this new setup, I still made the assumption that I only have word-level transcriptions for the close-talking data. The reasoning for this is twofold: first, to model the scenario where a large amount of parallel recordings that have not yet been transcribed is added to an existing labeled close-talking dataset, and second, to evaluate the performance of acoustic models not trained on multiple conditions (as with Xu et al. (2014), for example).

Because this setup now assumes a large amount of close-talking and distant speech data available but still only acoustic models trained on close-talking data, I developed models to perform speech enhancement rather than data augmentation. As described in Chapter 1, speech enhancement is a technique used to transform distant or otherwise noisy speech data to clean, close-talking data. Formally, for close-talking data $S_c = \{x_1, \ldots, x_n\}$ where $x_i \in \mathcal{X}$ and corresponding distant data $S_d = \{f_d(x_1), \ldots, f_d(x_n)\} = \{\widetilde{x}_1, \ldots, \widetilde{x}_n\}$ where $x_i \in S_c$ and $f_d : \mathcal{X} \to \mathcal{X}$ is some noising function, speech enhancement systems seek to reduce the distance between corresponding data points $x_i \in S_c, \widetilde{x}_i \in S_d$ after passing $\widetilde{x}_i$ through the model by some metric (for example, ℓ_2 norm).

An *adaptive* speech enhancement system is one that not only seeks to reduce this distance between distant speech and close-talking speech, but also to accurately reconstruct the close-talking speech (i.e., keep the distance near 0). This is a desirable property because it prevents one from needing to train a separate close-talking speech detector (see Chapter 1) or otherwise handle close-talking speech separately to avoid degradations in performance on close-talking speech.

4.1 Modular network components

I developed a variety of models to perform adaptive speech enhancement on both close-talking and distant speech before passing the enhanced speech to an acoustic model trained on only close-talking speech. To have a set of control experiments to show the effect of speech enhancement in various model setups, I first defined the following networks as building blocks to the models:

- **Encoder** E: define E to be a multilayer convolutional neural network. As with the encoder portion of the multidecoders described in Section 3.1.1, each layer consists of several convolution filters that are convolved with its input features to yield feature maps as output. These feature maps are passed through the rectified linear unit (ReLU) activation function, with some layers (not necessarily all, as was the case in Chapter 3) also being downsampled by a max-pooling layer in the feature dimension only. The output of the convolutional stage is then flattened to one dimension before being passing through a fully-connected layer to yield the latent vector $z \in \mathcal{F} = \mathbb{R}^J$. A depiction of this component with sample layer parameters is shown in Figure 4-1.

- **Decoder** D: define D to be a multilayer convolutional neural network whose layers are symmetric to E (using zero-padding and max-unpooling in the same way as described in Section 3.1.1) and in reverse order. The input to D is a latent vector z produced by E and the output is a reconstructed feature matrix with the same dimensions as the input data to E. A depiction of this component

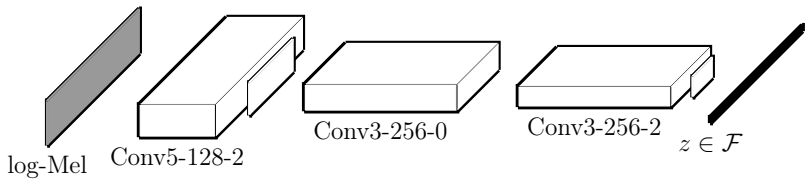

Figure 4-1: Sample encoder architecture for adaptive speech enhancement experiments. Convolutional layer parameters shown as "Conv⟨Receptive field size⟩-⟨Number of channels⟩-⟨Pool size⟩". ReLU and BatchNorm layers not shown for simplicity.

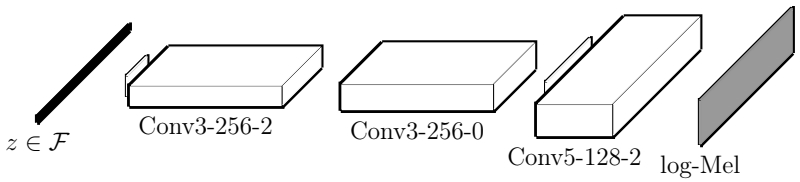

Figure 4-2: Sample decoder architecture for adaptive speech enhancement experiments (symmetric to the encoder in Figure 4-1). Convolutional layer parameters shown as "Conv⟨Receptive field size⟩-⟨Number of channels⟩-⟨Pool size⟩". ReLU and BatchNorm layers not shown for simplicity.

with sample layer parameters is shown in Figure 4-2.

- **Frame Classifier** C: define C to be a multilayer fully-connected neural network. C takes a latent vector $z \in \mathcal{F}$ as input and then passes it through multiple fully-connected layers, each using the aforementioned ReLU activation function with the exception of the final layer. The final layer uses a log-softmax function over all of the HMM states (see Section 3.2.2) to yield log probabilities for predicting the states at each frame. A depiction of this component with sample layer parameters is shown in Figure 4-3.

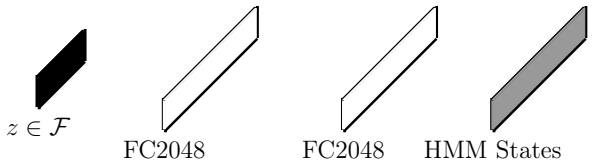

Figure 4-3: Sample frame classifier architecture for adaptive speech enhancement experiments. Fully-connected layer parameters shown as "FC⟨Hidden unit count⟩". ReLU and BatchNorm layers not shown for simplicity.

4.2 Model descriptions

Using these modular components, I then developed the following network configurations:

4.2.1 Baseline acoustic model

To first get baseline error rates without any form of speech enhancement, I trained baseline acoustic models that operate directly on the raw log-Mel filter bank features. Formally, a baseline acoustic model AM consists of the composition of an encoder and a frame classifier (i.e., $AM = C \circ E$) as depicted in Figure 4-4. The model is trained by minimizing the cross-entropy loss between the ground truth HMM states and the predicted states $\arg\max AM(\cdot)$. Two baseline acoustic models are trained for each set of experiments: a close-talking acoustic model AM_c trained on just the close-talking speech and a distant acoustic model AM_d trained on just the distant speech. Note that AM_d is only used as an oracle to determine what the error rate would be for an acoustic model if there were in fact word transcriptions available for the distant speech.

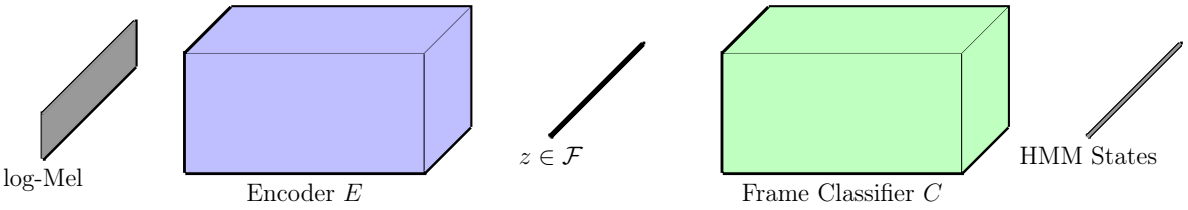

Figure 4-4: Sample baseline acoustic model architecture for adaptive speech enhancement experiments.

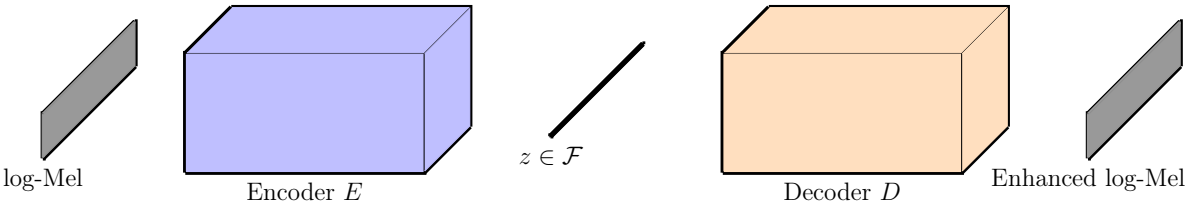

Figure 4-5: Sample enhancement network architecture for adaptive speech enhancement experiments.

4.2.2 Enhancement network

The first (and simplest) adaptive speech enhancement model is the *enhancement network*. An enhancement network N consists of the composition of an encoder and a decoder for close-talking speech (i.e., $N = D_c \circ E$) as depicted in Figure 4-5. Although N appears to be a simple autoencoder, it is distinguished by how it is trained. For close-talking data $x \in S_c$, the objective is to minimize reconstruction loss $\|x - N(x)\|^2$ as with normal autoencoders, but for distant data $y \in S_d$, minimize *transformation* loss $\|x - N(y)\|^2$ as well. Once speech has been processed by N, the resulting enhanced speech is passed to the baseline close-talking acoustic model AM_c for evaluation.

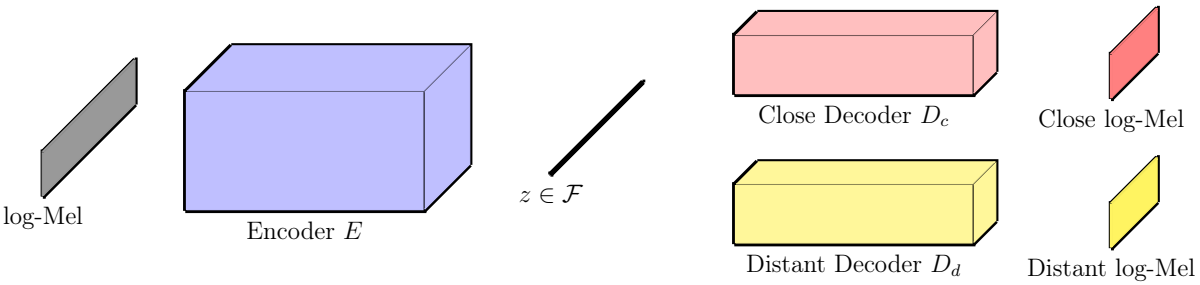

Figure 4-6: Sample enhancement multidecoder architecture for adaptive speech enhancement experiments. The encoder E is shared for reconstructing both close and distant log-Mel outputs.

4.2.3 Enhancement multidecoder

An enhancement multidecoder N_{MD} is the same model as an enhancement network N, but with an additional decoder D_d for distant speech data (see Figure 4-6). N_{MD} is trained in a similar fashion to N, where for close-talking data $x \in S_c$ and distant data $y \in S_d$, the objective is to minimize both reconstruction loss $\|x - D_c(E(x))\|^2$ and transformation loss $\|x - D_c(E(y))\|^2$. However, for this model, I additionally minimize reconstruction loss $\|y - D_d(E(y))\|^2$ and transformation loss $\|y - D_d(E(x))\|^2$. During evaluation, speech data is passed through the encoder E and the close-talking decoder D_c to enhance it before passing it to the baseline close-talking acoustic model AM_c. Training the distant decoder D_d can therefore be seen as an extra regularization step for the encoder E because it encourages E to encode enough information in the latent vector z to also reconstruct a distant version of the input speech.

4.2.4 Multitask network

Instead of just training the adaptive enhancement model and acoustic model separately, I decided to also investigate using a shared encoder for both models to see if the learned representation can be used for both speech recognition and speech enhancement. The first variant of such a model, the multitask network M, consists of an enhancement network N and an acoustic model AM that share a single encoder E as

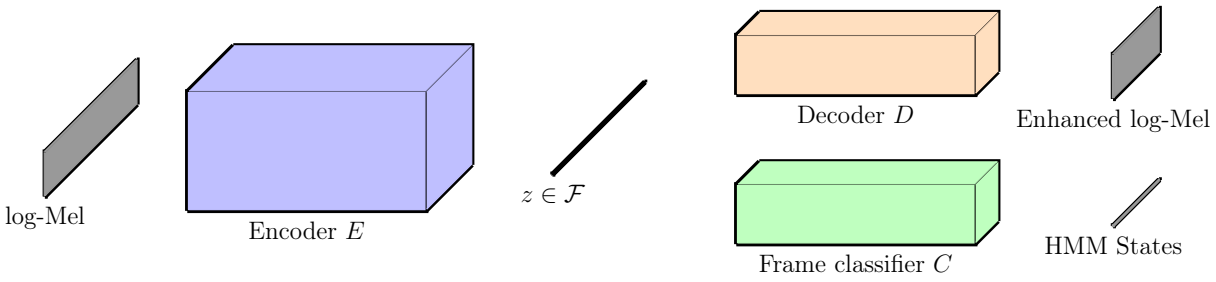

Figure 4-7: Sample multitask network architecture for adaptive speech enhancement experiments. The encoder E is shared for both speech enhancement and frame classification.

depicted in Figure 4-7. During training, the optimizer alternates between minimizing cross-entropy loss between the ground truth HMM states and the predicted states $\arg\max AM(\cdot)$ and the reconstruction and enhancement losses for N as described in Section 4.2.2. This encourages the encoder to preserve linguistic information well enough for speech recognition while simultaneously encoding enough information to reconstruct or transform input speech. During evaluation, speech is first enhanced by N before classifying HMM states with AM.

4.2.5 Multitask multidecoder

Similar to a multitask network, a multitask multidecoder M_{MD} consists of an enhancement multidecoder N_{MD} and an acoustic model AM that share a single encoder E as depicted in Figure 4-8. During training, the optimizer alternates between minimizing cross-entropy loss between the ground truth HMM states and the predicted states $\arg\max AM(\cdot)$ and the reconstruction and enhancement losses for N_{MD} as described in Section 4.2.3. This encourages the encoder to preserve linguistic information well enough for speech recognition while simultaneously encoding enough information to reconstruct or transform input speech in either the close-talking or distant speech domains. During evaluation, speech is first enhanced by the encoder E and close-talking decoder D_c before classifying HMM states with AM.

I then ran experiments by fixing the architectures for E, D and C, building these

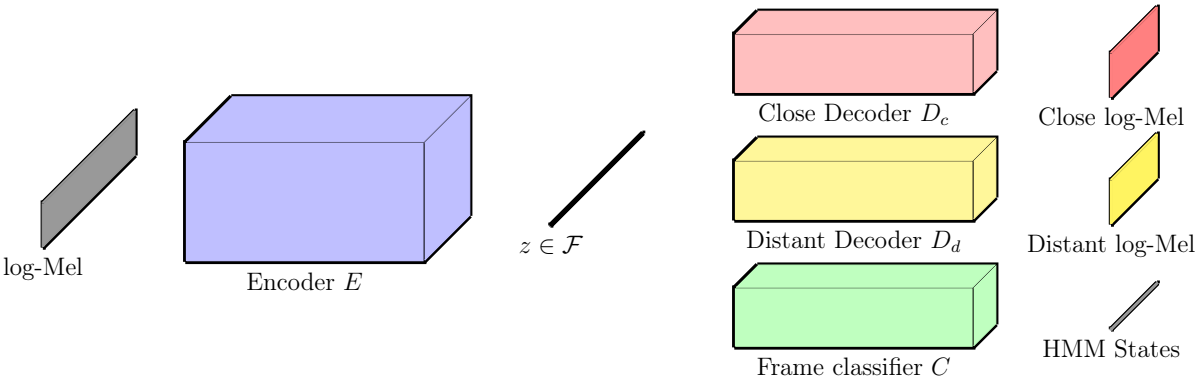

Figure 4-8: Sample multitask multidecoder architecture for adaptive speech enhancement experiments. The encoder E is shared for frame classification and reconstructing both close and distant log-Mel outputs.

models with these components and training them in parallel. The details of these experiments are described in detail in the following chapter.

Chapter 5

Experimental Setup and Outcomes

For feature extraction, I again used the Kaldi speech recognition toolkit (Povey et al., 2011). 80 log Mel-scale filter bank features were extracted with the same 25 ms window and 10 ms frame shift and context frames are still spliced into the feature vector as described in Section 3.2.2, but seven context frames (instead of five) were used, as this led to better recognition results without significantly changing training time or memory usage. Batch size was fixed to 256 samples for all datasets, a value that balanced memory and data diversity concerns regardless of dataset in these experiments.

Model development was again performed using the PyTorch deep learning framework. Weights in the network were initialized using Xavier initialization and the models were optimized using the Adam optimizer with the parameters set to the defaults suggested by Kingma and Ba (2015) ($\beta_1 = 0.9$, $\beta_2 = 0.99$ and $\epsilon = 10^{-8}$) with the exception of the initial learning rate. Experiments were conducted on initial learning rate, learning rate update schedule and number of epochs for each dataset and are described below.

Because the acoustic models being trained here already predict `tri3` HMM states, Kaldi was not needed to train TDNN models as in Chapter 3. However, these frame-level state predictions were in fact used to generate word-level predictions by decoding with trained transition and language models. I used NIST's `sclite` tool (NIS) to get recognition results in terms of word error rate (WER; see Equation 5.1).

$$WER = \frac{S + D + I}{N} \begin{cases} S,\ D \text{ and } I \text{ are substitution, deletion and insertion errors;} \\ N \text{ is the total number of ground truth word labels} \end{cases} \tag{5.1}$$

5.1 Simulating distant speech data

Because both of the datasets used for experiments only contain close-talking recordings, I needed to create parallel distant speech data. Rather than record large amounts of new audio in a variety of room and microphone setups, I simulated distant speech by convolving the original recording with simulated room impulse responses (RIRs) (Neely and Allen, 1979). RIRs were generated using the image method (Allen and Berkley, 1979) with a variety of speaker positions, rectangular room sizes and microphone positions as proposed by Ko et al. (2017). Three different sets of rooms $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$ were generated by sampling the following uniform distributions for width L_x, length L_y and height L_z (in meters), where $\mathcal{U}(a, b)$ stands for a uniform distribution between a and b:

$$\begin{aligned} \mathcal{S}_1 &: L_x \sim \mathcal{U}(1, 10), L_y \sim \mathcal{U}(1, 10), L_z \sim \mathcal{U}(2, 5) \\ \mathcal{S}_2 &: L_x \sim \mathcal{U}(10, 30), L_y \sim \mathcal{U}(10, 30), L_z \sim \mathcal{U}(2, 5) \\ \mathcal{S}_3 &: L_x \sim \mathcal{U}(30, 50), L_y \sim \mathcal{U}(30, 50), L_z \sim \mathcal{U}(2, 5) \end{aligned} \tag{5.2}$$

The speed of sound was fixed to 343 meters per second for each room, a common value used for when the propagation medium is dry air at a temperature of 20°C (Allen and Berkley, 1979). The wall, ceiling and floor reflection coefficents were sampled from $\mathcal{U}(0.2, 0.8)$ for each room. For each set, 200 rooms were sampled and then for each room, 100 RIRs were generated for a random source and microphone placement within the room, resulting in a total of $60,000$ simulated RIRs. A subset of 600 RIRs were used for experiments by selecting a random RIR for each room.

To create the distant speech data, each utterance in the close-talking dataset was

convolved with a random RIR from the subset. To keep the distant utterances in a reasonable volume range while staying aligned with the close-talking utterance and at the same length, I performed the following steps (depicted in Figure 5-1), where the speech signal s_c is of length N and the RIR is of length $M < N$:

1. Perform a "full" convolution on the signal by padding with $(M-1)$ zeroes on both sides of the speech signal. This yields a value at every point of signal overlap (including partial overlaps) to get a sequence of length $N + 2(M-1)$.

2. Remove the padding on the left side of the signal to recover a signal s_d of length $N + M - 1$.

3. Choose start index i in s_d to be the index of the maximum value in the RIR. The intuition behind this choice is that there is a gap between the first sound being produced by the speaker and it reaching the microphone placed in the room due to the speed of sound. Therefore, the time at which the original sound reaches the microphone before any dampening occurs will be the loudest peak in the RIR. Because the distance between the speaker and the microphone can differ depending on the RIR, the speech is aligned based on when it reaches the microphone using this shift. Trim distant speech signal to length N with the following slice: $s_d^{trim} = s_d[i : i + N]$.

4. Normalize volume of the distant speech to be similar to that of the original close-talking speech by multiplying distant speech by $\gamma * \frac{\max|s_c|}{\max|s_d^{trim}|}$, where γ is a value in $[0, 1]$ to reduce clipping. $\gamma = 0.95$ was used in my experiments.

5.2 TIMIT Acoustic-Phonetic Continuous Speech Corpus

The first dataset that I investigated in these experiments was the TIMIT Acoustic-Phonetic Continuous Corpus (Lamel et al., 1989). The corpus contains 6,300 phonetically-balanced sentences from 630 different speakers. I used a 462 speaker set with all SA

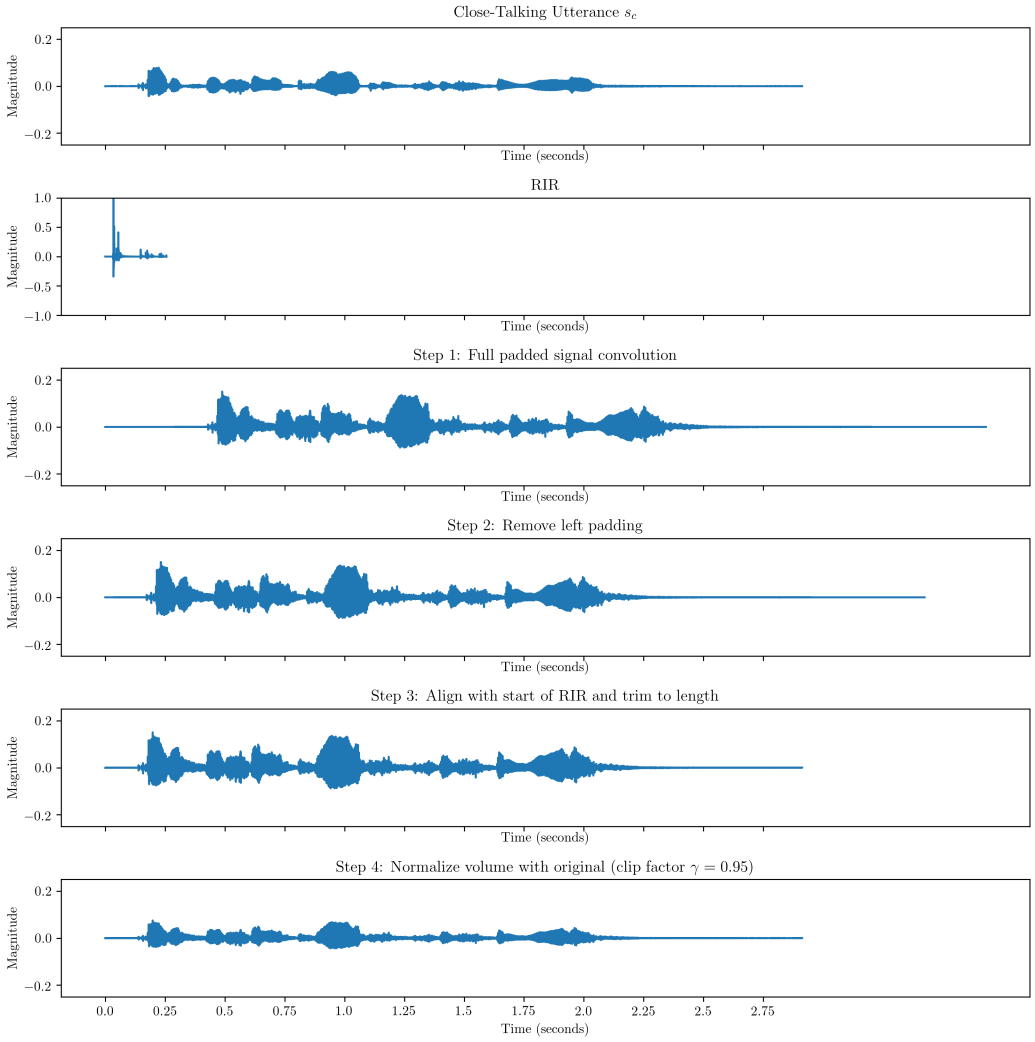

Figure 5-1: Convolution steps for creating distant speech data. Utterance shown is from TIMIT with ID: SX126, Speaker FELC0

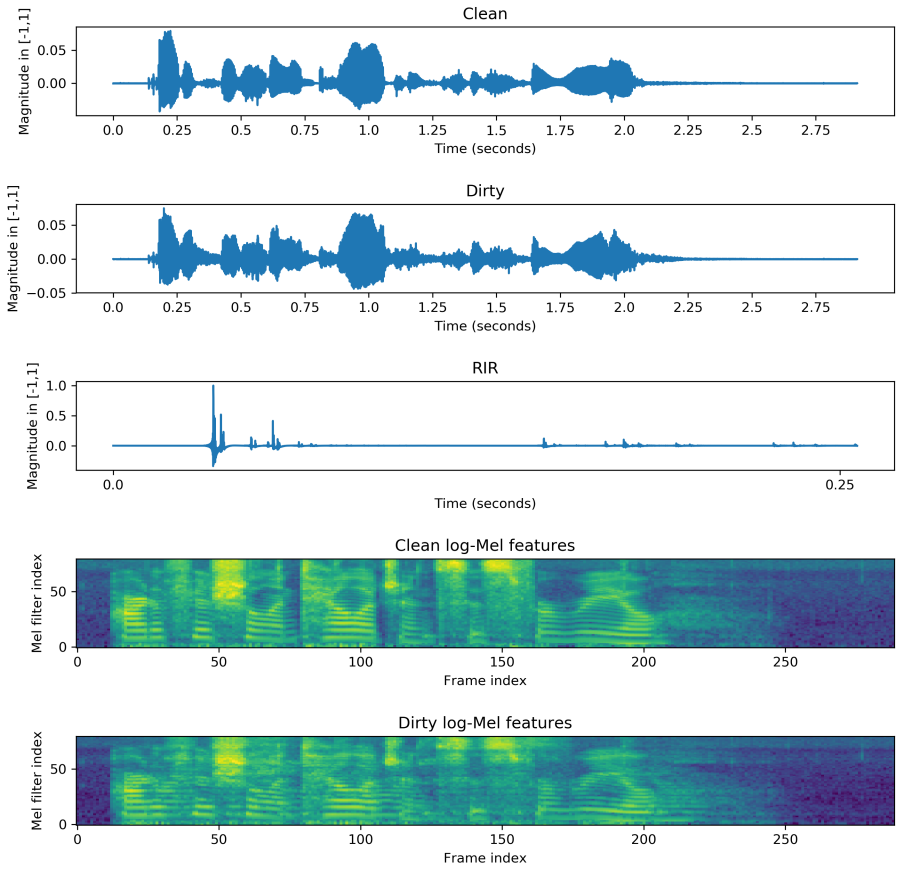

Figure 5-2: A sample TIMIT sentence pair and associated room impulse response. Utterance ID: SX126, Speaker FELC0

records (dialect-specific sentences) removed for training, a held-out 50 speaker set for early stopping and a 24 speaker test set, a standard setup used for comparing TIMIT results (Lopes and Perdigão, 2011). To conform to CMU/MIT standards, I used 48 phone labels during training, 39 of which were used during testing by not counting confusions between phones within five separate groups (Lee and Hon, 1989). Waveforms and associated filter bank features for an example pair of utterances, as well as the waveform for the associated RIR, are shown in Figure 5-2.

Because TIMIT is labeled at the phone level, researchers commonly train phone recognizers (rather than speech recognizers, which operate at the word level) and

report results in terms of Phone Error Rate (PER). PER is computed using the same formula as WER (Equation 5.1), but by measuring the number of phone substitution, deletion and insertion errors and total number of ground truth phone labels (Lopes and Perdigão, 2011).

5.2.1 Experimental setup

I conducted experiments to determine an effective initial learning rate and learning rate update schedule for TIMIT and determined that starting with a learning rate of 0.0001 and decaying it to 10% of the previous rate every time more than one epoch of training completed without an improvement in the loss on the development set worked reasonably well. With this schedule, it was determined that 35 epochs of training were sufficient to reach a good convergence point.

Experiments were also conducted to determine optimal architecture choices for E, D and P. Because TIMIT is a fairly small dataset, overfitting was observed on very deep architectures, as well as "wide" architectures (i.e., high number of hidden units per layer). Additionally, reconstruction error was observed to be higher when using convolutional units with larger receptive fields (7×7, for example), presumably due to a greater degree of upsampling in the padded convolutions performed in the decoder. Interestingly, this trend towards smaller receptive fields has also led to improvements in computer vision tasks as observed by Simonyan and Zisserman (2014).

The best performing architectures for the modular components on TIMIT were the following. Results for the close-talking and distant test sets by these architectures are shown in Table 5.1.

- **Encoder** E: 3 convolutional layers in the following order:
  - 64 channels with 5×5 receptive fields; followed by 1×2 max-pooling layer
  - 128 channels with 3×3 receptive fields; no max-pooling
  - 128 channels with 3×3 receptive fields; followed by 1×2 max-pooling layer

Table 5.1: TIMIT Speech Enhancement Results

Model	Close-talking Test PER	Distant Test PER
Baseline close-talking	20.3%	44.8%
Baseline distant	22.8%	30.5%
Enhancement Net	20.5%	**29.9**%
Enhancement Multidecoder	20.8%	30.5%
Multitask Net	20.4%	32.4%
Multitask Multidecoder	**20.3**%	32.3%

  - 512-dimension latent space

  Decoder D is symmetrical to this and in reverse order.

- **Frame Classifier** C: 3 fully-connected layers with 512 hidden units each

Several interesting results are to be seen here. First, the enhancement network and enhancement multidecoder models both improved the performance on distant speech to match or outperform both the close-talking *and* distant baseline acoustic models, while only suffering degradations on the order of tenths of a percentage point on PER for the close-talking test data when compared to the close-talking baseline. Despite these improvements, no major difference is seen between the results of the enhancement network and enhancement multidecoder in these experiments, suggesting that the added decoder for distant speech provided no useful regularization on the encoder and/or added enough parameters (and therefore complexity) to the training process such that one would need to train it with different hyperparameters or architecture choices to improve performance beyond that of the enhancement network.

Another interesting result is that the multitask network and multitask multidecoder have consistently worse performance on the distant data when compared to the enhancement network and enhancement multidecoder, despite performing slightly better on the close-talking data. One possible reason for this can be seen in Figure 5-3, which compares histograms of the ℓ_2 distances between the encoded latent vectors $E(x_i)$ and $E(\widetilde{x}_i)$ for all pairs of parallel data points $x_i \in S_c, \widetilde{x}_i \in S_d$ with those of the encoded latent vectors for all pairs of parallel data points that have already

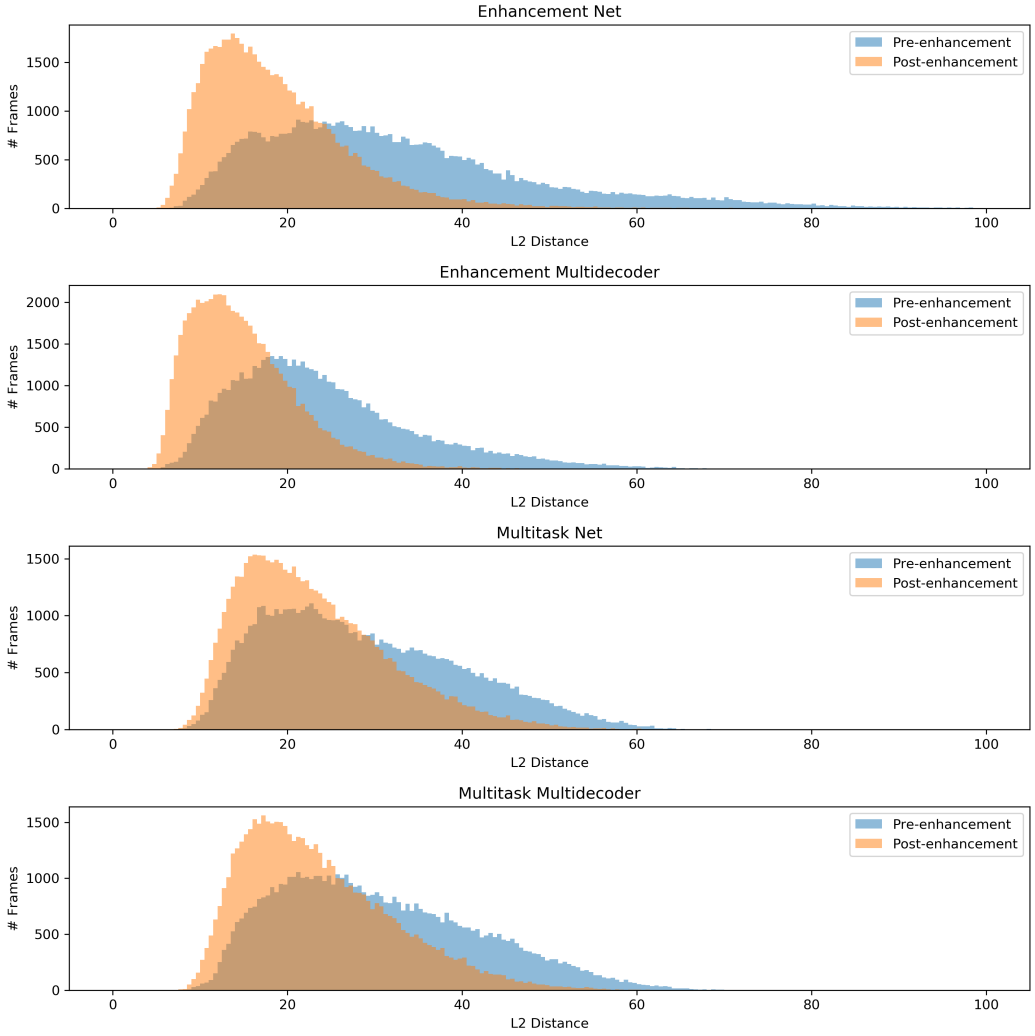

Figure 5-3: TIMIT ℓ_2 distances between encoded close-talking and distant speech latent vectors, pre- and post-enhancement. Plots are shown for the enhancement network, enhancement multidecoder, multitask network and multitask multidecoder models with the best performance in the TIMIT experiments.

been passed through the enhancement portion of the model once. The figure shows that the enhancement models more effectively reduce the mean and variance of the ℓ_2 distances after enhancement than the multitask models. The interpretation of this is that the enhancement models more effectively bring parallel data points close together in latent space after enhancement than the multitask models do, signifying that they learn a data representation that makes distant speech look similar to close-talking speech.

Code for the TIMIT experiments is available on GitHub at `https://github.com/atitus5/meng-timit`.

5.3 TED-LIUM Corpus Release 2

I also conducted experiments with a significantly larger speech corpus. The second release of the TED-LIUM corpus contains 207 hours of clean, close-talking recordings from TED (Technology, Entertainment, Design) talks with associated word transcriptions (Rousseau et al., 2014). The recordings come from 1,495 TED talks across a variety of topics from 1,242 unique speakers (68% male, 32% female). The dataset also contains 19 TED talks, 8 of which are used as a development set for validation and 11 of which are used as a held-out test set. Unlike TIMIT, TED-LIUM is labeled at the word level, so results are reported for the test set in terms of word error rates (WER; see Equation 5.1) achieved by speech recognizers rather than phone recognizers. The waveforms and associated filter bank features for an example pair of utterances for TED-LIUM, as well as the waveform for the associated RIR, are shown in Figure 5-4.

5.3.1 Experimental setup

Because TED-LIUM is a much larger dataset, I conducted experiments on a 10% subset of randomly selected utterances from the train and development sets (results are still reported on the full test set, however) before moving on to the full dataset.

Experiments to determine an effective initial learning rate and learning rate up-

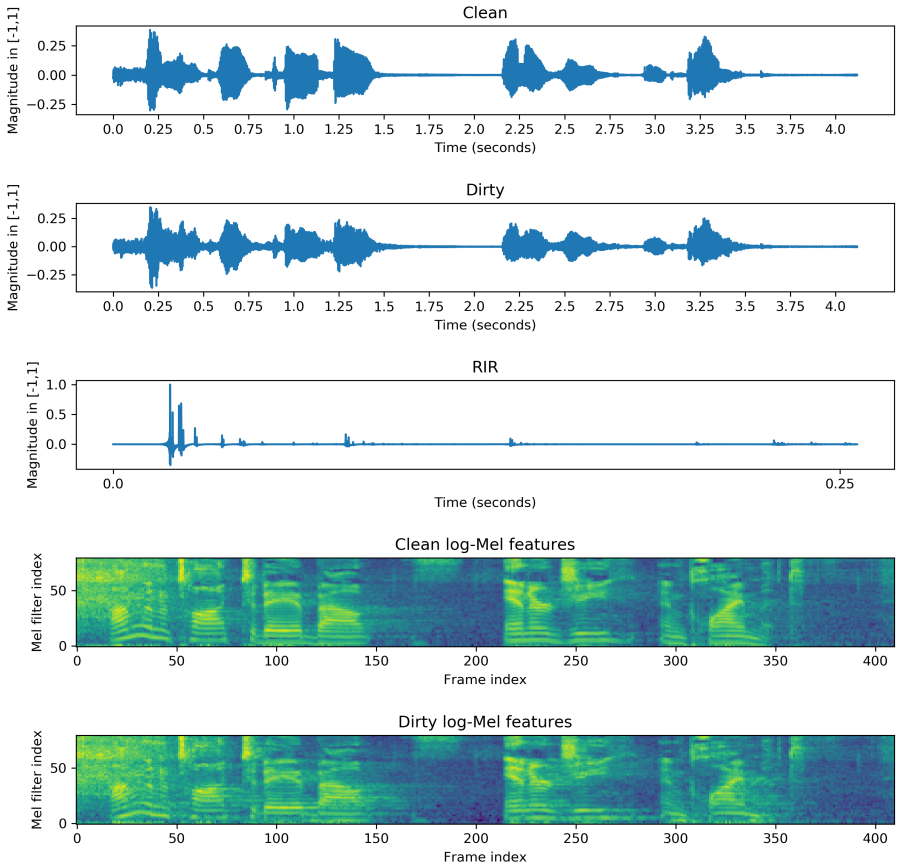

Figure 5-4: A sample TED-LIUM sentence pair and associated room impulse response. Utterance ID: 0001586-0001998, Speaker Bill Gates

date schedule for TED-LIUM determined that starting with a learning rate of 0.0001 and decaying it to 10% of the previous rate every time an epoch of training completed without an improvement in the loss on the development set. Additionally, to both speed up training and improve generalization on the development and test sets, I utilized batch normalization (Ioffe and Szegedy, 2015) with running mean and variance computations (momentum set to 0.1) and learnable affine parameters γ and β before each ReLU activation in the model, in addition to Dropout with $p = 0.5$ (Srivastava et al., 2014) after each fully-connected layer activation in P.

With this setup on the 10% subset, I determined that 10 epochs of training on all models except the baseline acoustic models (where 25 epochs were used) were sufficient to reach a reasonable convergence point without requiring several hours of training per epoch. On the full dataset, however, 5 epochs of training were used on all models except the baseline acoustic models (where 10 epochs were used) in order to reach a sufficient convergence without requiring many days of training or more advanced distributed training methods.

Experiments were conducted to determine optimal architecture choices for E, D and P for the 10% subset before running experiments with these choices on the full dataset. Because TED-LIUM is a much larger dataset than TIMIT (even with just a 10% subset), deeper and wider architectures were necessary to prevent overfitting. As with TIMIT, reconstruction error was observed to be higher when using convolutional units with larger receptive fields (7×7, for example), so smaller values were used instead.

The best performing architectures for the modular components on the 10% were the following. Results for the close-talking and distant test sets by these architectures are shown in Table 5.2. When trained on the full dataset, results improved to those shown in Table 5.3.

- **Encoder** E: 3 convolutional layers in the following order:
  - 64 channels with 5×5 receptive fields; followed by 1×2 max-pooling layer
  - 128 channels with 3×3 receptive fields; no max-pooling

Table 5.2: TED-LIUM (10% Subset) Speech Enhancement Results.

Model	Close-talking Test WER	Distant Test WER
Baseline close-talking	20.5%	36.9%
Baseline distant	22.7%	32.8%
Enhancement Net	**20.9**%	**33.4**%
Enhancement Multidecoder	21.3%	38.1%
Multitask Net	23.1%	38.3%
Multitask Multidecoder	23.3%	41.6%

Table 5.3: TED-LIUM Speech Enhancement Results.

Model	Close-talking Test WER	Distant Test WER
Baseline close-talking	17.7%	35.3%
Baseline distant	18.2%	26.9%
Enhancement Net	18.4%	**30.9**%
Enhancement Multidecoder	**18.0**%	35.6%
Multitask Net	19.9%	33.5%
Multitask Multidecoder	20.2%	40.6%

  - 128 channels with 3×3 receptive fields; followed by 1×2 max-pooling layer
  - 1024-dimension latent space

  Decoder D is symmetrical to this and in reverse order.

- **Frame Classifier** C: 2 fully-connected layers with 2048 hidden units each

In contrast to TIMIT, only the enhancement network model seemed to achieve any substantial improvement on distant speech for TED-LIUM. This may be explained by the histograms comparing the ℓ_2 distances between $E(x_i)$ and $E(\widetilde{x}_i)$ for all pairs of parallel data points $x_i \in S_c, \widetilde{x}_i \in S_d$ with those of the encoded latent vectors for all pairs of parallel data points that have already been passed through the enhancement portion of the model once (see Figures 5.3.1 and 5-6). The enhancement network is the only model that reduces the mean and variance of these distances in these experiments, whereas the other models keep these quantities roughly the same as

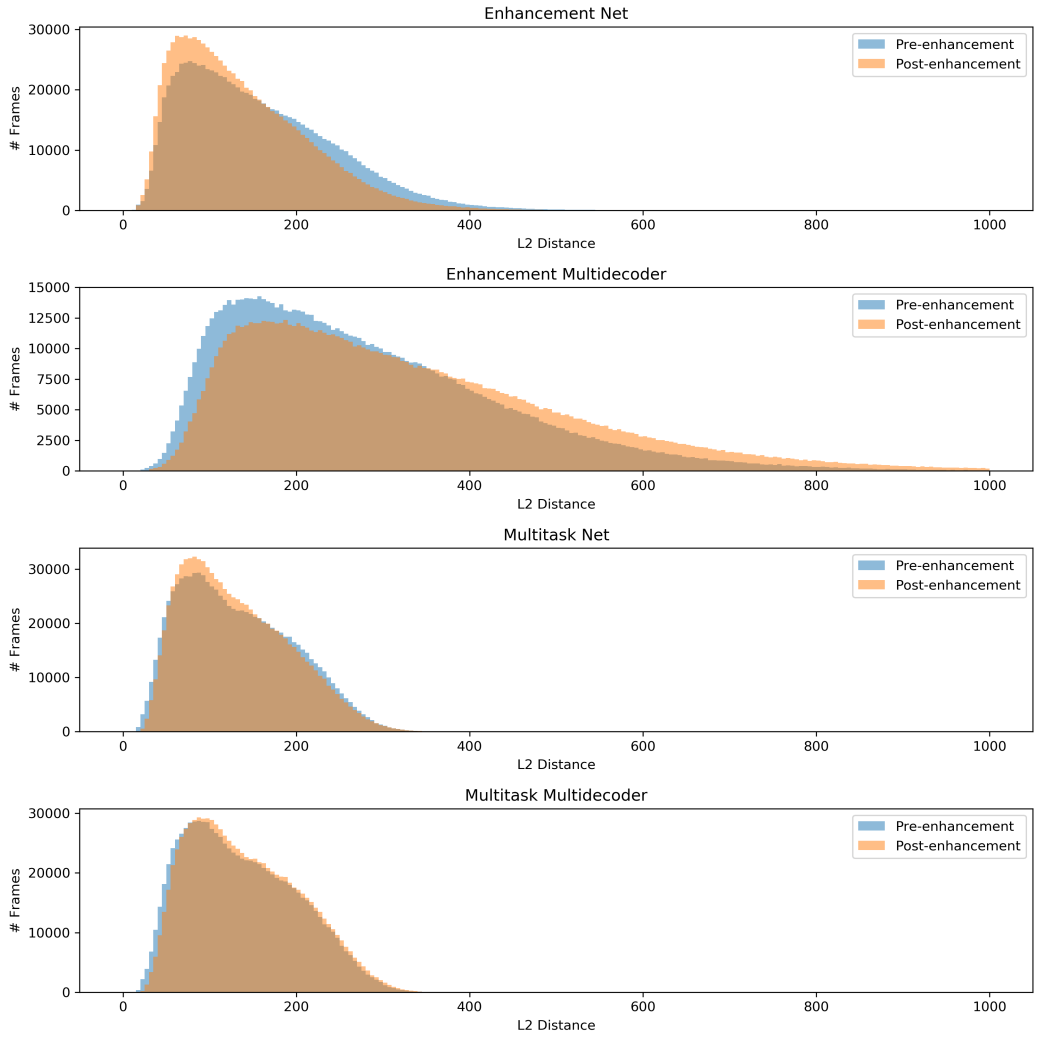

Figure 5-5: TED-LIUM (10% subset) ℓ_2 distances between encoded close-talking and distant speech latent vectors, pre- and post-enhancement. Plots are shown for the enhancement network, enhancement multidecoder, multitask network and multitask multidecoder models with the best performance in the TED-LIUM (10% subset) experiments.

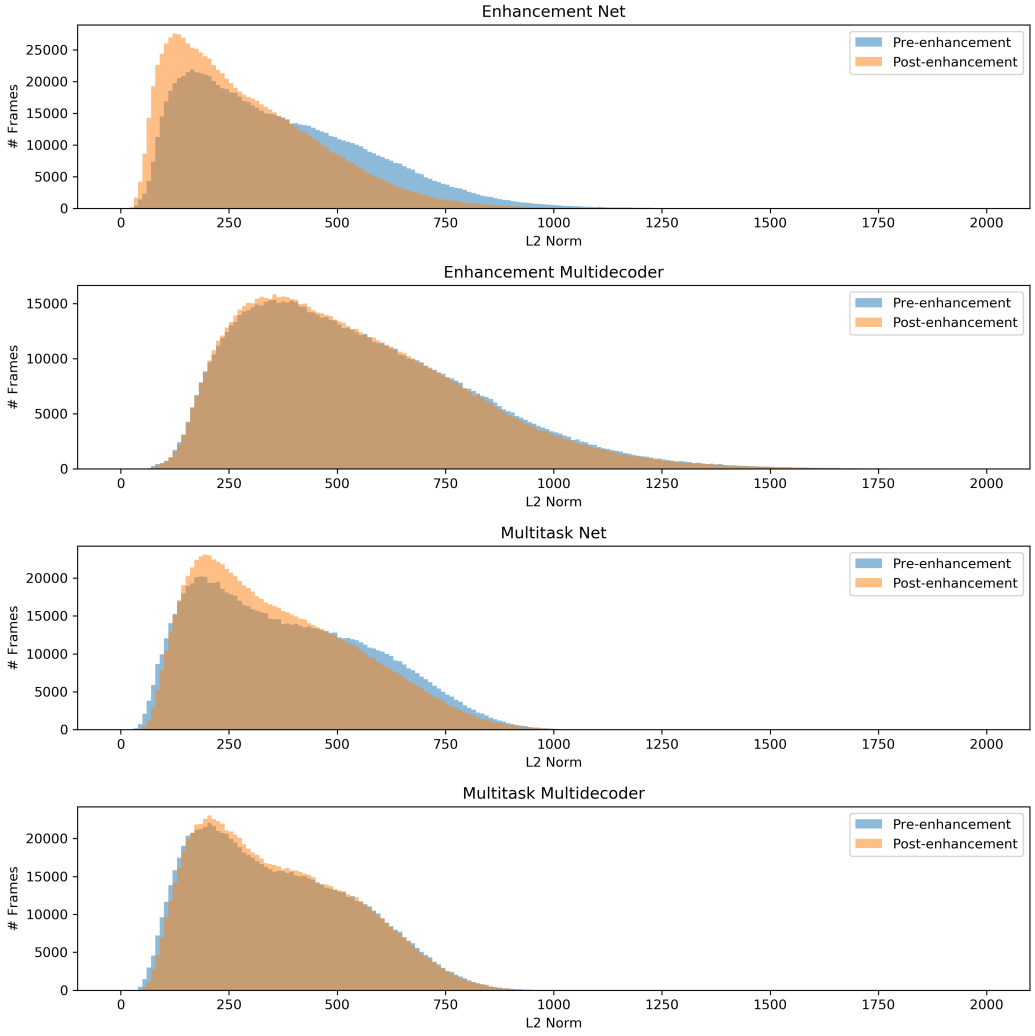

Figure 5-6: TED-LIUM ℓ_2 distances between encoded close-talking and distant speech latent vectors, pre- and post-enhancement. Plots are shown for the enhancement network, enhancement multidecoder, multitask network and multitask multidecoder models with the best performance as determined by the TED-LIUM (10% subset) experiments.

before. This suggests that the other models may be learning identity functions that fail to actually enhance distant speech, a problem discussed in Section 2.1 that may be solved with additional regularization. Another difference between TED-LIUM and TIMIT is that the gap in close-talking WER between the close-talking baseline acoustic model and the distant baseline model was only 0.5%, suggesting that it may be better to simply train the acoustic model on distant data, rather than training speech enhancement models, if labels were indeed available for the distant data in this setup.

Despite these differences with TIMIT, similar behavior was observed in a few instances. For example, WER did not improve for either close-talking or distant speech when a distant speech decoder D_d was added to the enhancement network or multitask network models to create the enhancement multidecoder and multitask multidecoder models, respectively. Not only did training take substantially longer to complete (roughly twice as long due to two times as many forward and backward passes through an encoder-decoder composition), the WER increased for both domains in most cases as well. Additionally, with the exception of a slight improvement in distant WER for the multitask network on the full TED-LIUM dataset (see Table 5.3), the WER for both domains is strictly worse for the multitask models than the enhancement models in these experiments despite having similar reconstruction and transformation losses and similar histograms in Figures 5.3.1 and 5-6. This suggests that it may be necessary to train the acoustic model portion of the model for proportionally longer than the enhancement portion and/or weight the HMM state classification cross-entropy more heavily than the reconstruction and transformation losses in the combined loss function in order to have similar acoustic model performance to the baselines.

Code for the TED-LIUM experiments is available on GitHub at `https://github.com/atitus5/meng-tedlium`.

Chapter 6

Conclusion

6.1 Summary of contributions

In Chapter 1, I described the problem of distant speech recognition and briefly discussed speech enhancement, a common technique used to improve distant speech recognition performance. I then discussed the lack of robustness of most speech enhancement systems to close-talking speech and motivated the desire to research adaptive speech enhancement models in this thesis that are robust to both distant and close-talking speech. In Chapter 2, I discussed existing speech enhancement systems in more detail and outlined several families of recent machine learning models that seek to learn data representations that can be useful for speech enhancement. In Chapter 3, I introduced the multidecoder model and summarized preliminary experiments with unsupervised domain adaptation on the AMI Meeting corpus whose challenges motivated the later, more successful experiments of this thesis.

The main contributions were described in Chapter 4 and 5. Chapter 4 introduced the problem of supervised domain adaptation and described the various adaptive speech enhancement models that I investigated over the course of the thesis. Chapter 5 then described the experiments conducted on both the TIMIT and TED-LIUM speech corpora, including a discussion of the process used to artificially simulate realistic distant speech data from existing close-talking speech data. I demonstrated that it is indeed possible to build adaptive speech enhancement systems that can

significantly improve recognition performance on distant data without substantially degrading recognition performance on close-talking data. Furthermore, I showed that the models not only bring close-talking and distant speech data closer in feature space, but also in the internal representation space learned by the models.

6.2 Future Work

Only a relatively small subset of possible model architectures for adaptive speech enhancement models were investigated in this thesis and only a relatively small space of possible model parameters and training hyperparameters was searched for each of these models. It is worth running a wider variety of experiments on this problem to see what other approaches can achieve better performance, both in terms of speech recognition and computational efficiency. One possible approach is to experiment with different weights for losses in the combined loss functions for the models investigated here. For example, in the multitask network, it may be beneficial to place more weight on the HMM state classification cross-entropy than the reconstruction and transformation losses because the enhancement models tended to converge more quickly than the baseline acoustic models.

Another area to improve is computational efficiency. The relatively high parameter count of these models (especially the multidecoder-based architectures) and multiple forward and backward passes through different encoder and decoder branches meant that it was impractical to train larger models without requiring many days of training. If these speech enhancement systems were to be deployed as a front-end processing step in online speech recognition systems, they would need to be reasonably efficient to keep latency as low as possible. The use of 10% subsets and smaller architectures for prototyping helped to alleviate these problems, but still required multiple days on larger datasets like the TED-LIUM corpus.

It is also worth looking at other datasets beyond the ones investigated here. In Section 5.1, I described a process for creating artificially reverberated data that could hypothetically be applied to other speech datasets with different characteristics than

AMI, TIMIT and TED-LIUM. Example characteristics include other languages (the datasets I investigated consist solely of English utterances), other spoken content, and other speaking styles. It would also be interesting to run more experiments on other noise environments, including non-stationary noise or channel noise in lossy environments (such as wireless communications).

In addition to running new experiments on different setups, it would also be insightful to evaluate the performance of these models in real applications with distant microphone setups. An example of such an application would be smart speakers that may be operated from distances anywhere from a few centimeters to tens of meters. Analyzing the features output by the speech enhancement model would be especially interesting to analyze in the situation where the speaker is moving in order to evaluate how well the system adjusts its output as the room impulse response changes over time.

6.3 Parting Thoughts

In order for speech recognition systems to become truly useful in today's society, they need to become more robust to a variety of noise environments. It is relatively rare that humans find themselves in nearly noise-free environments in their daily lives, but we still manage to understand each other in many noisy environments where speech recognition systems currently fail. In this thesis, I outlined a few techniques that can help to bridge this gap for one such noisy environment, distant speech recognition. I am glad to have contributed to one of the many efforts in speech recognition research to make machines understand us as well as humans, and look forward to a future where such interaction is commonplace.

Bibliography

NIST `sclite` scoring toolkit. `http://www.itl.nist.gov/iad/mig/tools/`.

Jont Allen and David Berkley. Image method for efficiently simulating small-room acoustics. *The Journal of the Acoustical Society of America*, 65(4):943–950, 1979.

Hervé Bourlard and Yves Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59:291–294, 1988.

Jean Carletta et al. The AMI meeting corpus: A pre-announcement. In *International Conference on Machine Learning for Multimodal Interaction*, pages 28–39, 2005.

Gaofeng Cheng et al. An exploration of dropout with LSTMs. In *INTERSPEECH-2017*, pages 1586–1590, 2017.

Xue Feng, Yaodong Zhang, and James Glass. Speech feature denoising and dereverberation via deep autoencoders for noisy reverberant speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1759 – 1763, 2014.

Yaroslav Ganin et al. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

Bradford Gillespie and Les Atlas. Strategies for improving audible quality and speech recognition accuracy of reverberant speech. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 676–679, 2003.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.

Ian Goodfellow et al. Generative adversarial nets. In *International Conference on Neural Information Processing Systems (NIPS)*, pages 2672–2680, 2014.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6645–6649, 2013.

Jyoti Guglani and AN Mishra. Continuous Punjabi speech recognition model based on Kaldi ASR toolkit. *International Journal of Speech Technology*, pages 1–6, 2018.

Geoffrey Hinton et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

James Hopgood and Peter Rayner. Blind single channel deconvolution using nonstationary signal processing. *IEEE Transactions on Speech and Audio Processing*, 11 (5):476–488, 2003.

Wei-Ning Hsu, Yu Zhang, and James Glass. A prioritized grid long short-term memory RNN for speech recognition. In *Spoken Language Technology Workshop (SLT)*, pages 467–473, 2016.

Wei-Ning Hsu, Yu Zhang, and James Glass. Unsupervised learning of disentangled and interpretable representations from sequential data. In *International Conference on Neural Information Processing Systems (NIPS)*, pages 1876–1887, 2017a.

Wei-Ning Hsu, Yu Zhang, and James R. Glass. Unsupervised domain adaptation for robust speech recognition via variational autoencoder-based data augmentation. *arXiv preprint arXiv:1707.06265*, 2017b.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456, 2015.

Takaaki Ishii et al. Reverberant speech recognition based on denoising autoencoder. In *INTERSPEECH-2013*, pages 3512 – 3516, 2013.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Diederik Kingma and Max Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

Tom Ko et al. A study on data augmentation of reverberant speech for robust speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5220–5224, 2017.

Lori Lamel, Robert Kassel, and Stephanie Seneff. Speech database development: Design and analysis of the acoustic-phonetic corpus. In *ESCA Tutorial and Research Workshop on Speech Input/Output Assessment and Speech Databases*, 1989.

Yann LeCun et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10), 1995.

Kai-Fu Lee and Hsiao-Wuen Hon. Speaker-independent phone recognition using hidden Markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(11):1641–1648, 1989.

Jae Lim and Alan Oppenheim. Enhancement and bandwidth compression of noisy speech. *Proceedings of the IEEE*, 67(12):1586–1604, 1979.

Carla Lopes and Fernando Perdigão. Phoneme recognition on the TIMIT database. In *Speech Technologies*. 2011.

Xugang Lu et al. Speech enhancement based on deep denoising autoencoder. In *INTERSPEECH-2013*, pages 436–440, 2013.

Nelson Mogran, Hervé Bourlard, and Hynek Hermansky. Automatic speech recognition: An auditory perspective. *Speech Processing in the Auditory System*, pages 309–338, 2004.

Stephen Neely and Jont Allen. Invertibility of a room impulse response. *The Journal of the Acoustical Society of America*, 66(1):165–169, 1979.

Adam Paszke et al. Automatic differentiation in PyTorch. In *International Conference on Neural Information Processing Systems (NIPS)*, 2017.

Vijayaditya Peddinti et al. Low latency acoustic modeling using temporal convolution and LSTMs. *IEEE Signal Processing Letters*, 25(3):373–377, 2018.

Daniel Povey et al. The Kaldi speech recognition toolkit. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2011.

Anthony Rousseau, Paul Deléglise, and Yannick Estève. Enhancing the TED-LIUM corpus with selected data for language modeling and more TED talks. In *International Conference of Language Resources and Evaluation (LREC)*, pages 3935–3939, 2014.

Tara Sainath et al. Deep convolutional neural networks for large-scale speech tasks. *Neural Networks*, 64:39–48, 2015.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Nitish Srivastava et al. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research (JMLR)*, 15(1):1929–1958, 2014.

Pawel Swietojanski, Arnab Ghoshal, and Steve Renals. Hybrid acoustic models for distant and multichannel large vocabulary speech recognition. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 285–290, 2013.

Pascal Vincent et al. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning (ICML)*, 2008.

Pascal Vincent et al. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research (JMLR)*, 11:3371–3408, 2010.

Junyuan Xie, Linli Xu, and Enhong Chen. Image denoising and inpainting with deep neural networks. In *International Conference on Neural Information Processing Systems (NIPS)*, pages 341–349, 2012.

Yong Xu et al. An experimental study on speech enhancement based on deep neural networks. *IEEE Signal Processing Letters*, 21(1):65–68, 2014.

Yong Xu et al. A regression approach to speech enhancement based on deep neural networks. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 23(1):7–19, 2015.

Takuya Yoshioka and Mark Gales. Environmentally robust ASR front-end for deep neural network acoustic models. *Computer Speech and Language*, 31(1):65–86, 2015.

Takuya Yoshioka et al. Making machines understand us in reverberant rooms: Robustness against reverberation for automatic speech recognition. *IEEE Signal Processing Magazine*, 29(6):114–126, 2012.

Matthew Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, pages 818–833, 2014.

Matthew Zeiler et al. Deconvolutional networks. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2528–2535, 2010.

Jun-Yan Zhu et al. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017.