End-to-end Speech Enhancement Using Perceptual Losses

CS 490 : RnD

by

Neelesh Verma
Roll No. 160050062

under the guidance of

Prof. Balamurugan Palaniappan

Computer Science and Engineering
Indian Institute of Technology, Bombay
Mumbai 400 076

Contents

Abstract

In today's rapidly evolving digital landscape, the demand for real-time video and voice communication applications has surged. However, the quality of audio on the recipient's end is often compromised by background noise. This abstract addresses the critical issue of speech enhancement, which has become increasingly important as the usage of speech data continues to rise. Speech enhancement finds applications in various domains, including mobile phones, speech recognition systems, hearing aids, teleconferencing, and Voice over Internet Protocol (VoIP). The challenge lies in improving audio quality by effectively reducing noise.

Traditional speech enhancement systems rely on a range of signal-processing techniques and algorithms. However, the emergence of deep neural networks (DNNs) has revolutionized this field. Our report focuses on a novel approach that simplifies the speech enhancement process by eliminating the need for preprocessing. We propose an end-to-end deep learning architecture, featuring a fully-convolutional denoising network, where raw audio waveforms are directly input into the network. This streamlined approach is detailed in the report, along with the architecture's intricacies, training process, evaluation, and comparison with state-of-the-art.

We utilize human listeners to assess the quality of the denoised output subjectively. This approach provides insight into the real-world effectiveness of our method. In conclusion, our research represents a significant advancement in speech enhancement. Our end-to-end deep learning approach, free from preprocessing constraints, offers a compelling solution to the persistent problem of audio quality degradation in noisy environments. By outperforming existing methods in perceptual experiments, our work not only advances the state of the art but also sets a new benchmark for high-quality voice communication.

Chapter 1

Introduction

Speech enhancement, often referred to as denoising, is a pivotal process that revolves around the intricate task of eliminating unwanted noise—typically recognized as background noise—from speech signals. Crucially, this must be accomplished while preserving the clarity and authenticity of the underlying speech. In this study, our primary focus is on mono signals, which constitute recordings and playback utilizing a single audio channel, as opposed to stereo signals that utilize two audio channels. This particular emphasis guides our exploration into the realm of Single Channel Speech Enhancement, where the central challenge centers on the delicate equilibrium between minimizing noise interference and safeguarding the integrity of the speech signal.

The complexity of the Speech Enhancement endeavor is further heightened by the consistently high sampling rate of all audio files involved, which stands firmly at 16 kHz. This heightened density of data samples introduces an additional layer of intricacy to the problem. In typical human-to-human communication settings, our perceptual mechanisms are astonishingly sensitive to the smallest errors, thereby making precision and efficacy of paramount concern.

Within the pages of this report, we devised an innovative approach to speech enhancement—an end-to-end deep learning methodology. In particular, our pipeline uses a fully convolutional neural network to generate clean audio from a given noisy audio. For training such a network, we rely on a second network - Feature Loss network, that learns feature embeddings for audio signals. Using the differences between these feature embeddings for generated output and clean audio, we guide our Denoising network to generate smooth, perceptually similar (to clean au-

dio) denoised audio. These perceptual losses, gleaned from well-established sources [9, 11, 4], furnish a distinctive lens through which we evaluate the performance of our model. In brief, this feature loss network, initially designed to classify audio, calculates losses by comparing internal activations induced by the input signals within the network. Throughout this report, we consistently refer to this distinctive loss function as "Perceptual Feature Losses." Our findings robustly affirm its superiority over prevailing techniques, a fact substantiated through perceptual experiments involving human listeners. These outcomes point towards a promising avenue for enhancing speech quality and noise reduction.

The crux of our approach is anchored in the training of the Denoising network, a process wherein it is exposed to noisy audio inputs and is subsequently trained by evaluating the perceptual feature losses between its output and the corresponding clean audio. This critical loss mechanism ensures that the denoising network, over time, learns to generate representations that closely mirror the clean audio signals at multiple stages of processing. By infusing the training process with reconstruction losses at diverse layers, we strategically encourage the Denoising network to preserve both the minutiae of low-level details and the nuanced subtleties of high-level information from the clean audio. This layered approach yields a discernible enhancement in audio quality, heightening its intelligibility. Furthermore, by endowing deeper layers with higher weights in the summation of losses from various layers, we aptly harness the unique capabilities of each layer to effectively amplify speech clarity and comprehensibility.

Chapter 2

Literature Survey

There have been many algorithms for speech enhancement methods. Broadly, all these algorithms can be classified into the following 3 classes -

1. **Filtering Techniques**: Different types of filters are used that minimize some loss function between the original and enhanced audios. One of the commonly used filters is **Weiner Filter** [2]. It is a linear estimator and the loss function here is the mean squared error between the original and enhanced speech.
2. **Spectral Restoration**: The objective here is to find the estimate of the minimum mean square error of the speech spectrum from a noisy version. There are many ways to find the estimator using the noise-free speech [26], [14].
3. **Speech-Model-Based** : Our approach falls in this category. Some other popular architectures are **Wavenet** [24], **SEGAN** [19]. Such approaches aim to fully exploit the expressive power of deep networks and avoid going into spectrogram domains.

Initially, the denoising systems used to rely on spectrogram-domain-based methods. It was followed by spectrogram-factorization-based methods. Even with the introduction of deep neural networks, most of the pipelines operated on spectrograms [25]. Inverse short-time Fourier was used to obtain time-domain enhanced audio which then gave rise to signal artifact.

There have been recent changes in the design of the deep networks denoising pipelines that are much more optimized and operate on the raw waveforms directly.

This avoids expensive computations for time-frequency transformations. Some of these approaches used simple loss functions [22]. Some have even experimented with some advanced loss functions but with very little success [12].

In our case, we are using a different loss function that we call "Perceptual Feature Loss". This is inspired by the research in the computer vision field, especially in the **Style Transfer Algorithms** [9, 4], where activations in the pre-trained classification network were found to produce very effective loss functions for the synthesis of images. Basically, if we have to find the loss between two images, then these images are passed through a pre-trained network. Each of the images produces a set of internal activations in the network and the loss is computed on the basis of dissimilarities between the set of activations. Such training losses have been shown to be pretty successful in other tasks as well without the need for some expert knowledge and without adding further complexity to the network.

Now, in the next section, we will be looking at the architecture design and also we will try to reason about the choice of architecture, but that is somewhat subjective. Note also that we will also call the Denoising network as **Context Aggregation Network** sometimes and the other network (for computing perceptual feature losses) as **Feature Loss Network**.

Chapter 3

Network Architecture

There are two different architectures for two different but related tasks - 1) Denoising the audio and 2) Learning the audio features. The first one takes raw input signals as an input (which are supposed to be noisy) and gives a clean and denoised signal as an output. Since this network operates end-to-end on audio signals, it makes sense to use a fully convolutional network. We will be calling this the Denoising Network.

To train the denoising network, we can simply use L2 loss between the output of the Denoising Network and the original clean audio. However, it has been shown that deep neural networks can learn transferrable features [17]. VGG [23] based features have been extensively used in different vision tasks - Style transfer [15, 11, 9], Object Detection and Segmentation [6, 21, 8, 16]. Taking inspiration from that, we have leveraged VGG-style architecture to design our feature learning network, which we will be referring to as the Feature Loss Network.

3.1 Denoising Network

Let's introduce some notations to get started. Assume there is a noisy audio signal $\mathbf{x}$. The corresponding clean audio is $\mathbf{y}$. There is an additional background signal $\mathbf{n}$ such that $\mathbf{x} = \mathbf{y} + \mathbf{n}$. Our Denoising network is a *fully convolutional* network architecture based on **context aggregation networks** [27]. The reason for using context aggregation networks is that such networks have been used and proven to be efficient in these tasks already. **WaveNet** [24] is also a context aggregation network, previously used in speech synthesis. Though our denoising architecture is much simpler than WaveNet, we do not use skip connections and gated activations used in Wavenets. Another important feature of our network is the use of **Adaptive**

Normalization [20]. We describe a context aggregation architecture and adaptive normalization below.

3.1.1 Context Aggregation

The network consists of a total of 16 convolutional layers. The input to the first dimension and the output from the last dimension are the same which is $N \times 1$. Here the number of samples in the input signal N varies. We can see the architecture in the Figure 3.1.

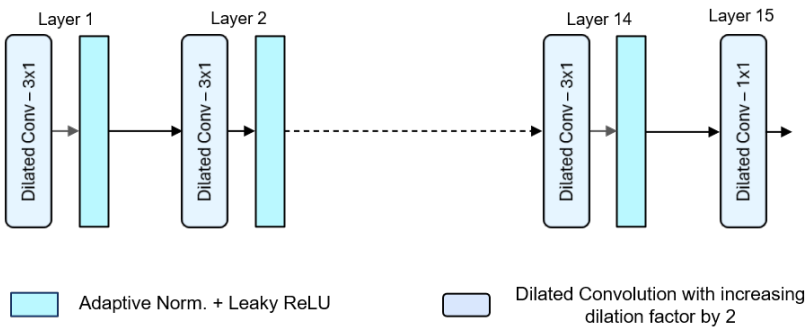


Figure 3.1: The context aggregation network consists of 15 dilated convolutional layers. The dilation factor increases exponentially from 2^0 to 2^13. There is no dilation for the 14th and the last layer.

Each of the intermediate layers is a 2D tensor of dimensionality $N \times W$ where W is the number of channels. We took constant $W = 64$ for all the layers. The convolutional kernels are of dimension 3×1. The following 3 operations are performed on activations from previous layers to get the activations of the current layer :

1. **Dilated Convolution**: First of all Dilated convolution with convolutional kernels is performed. The dilation operator basically aggregates long-range contextual information without changing the sampling frequency. We are increasing the dilation factor from 2^0 for the first intermediate layer to 2^{12} for the 13th layer. For the 14th layer and the last one, no dilation is being performed.

2. **Adaptive Normalization**: It is just a small variant of the standard batch normalization and is explained later.

3. **Leaky ReLU**: Finally to the normalized features, we perform point-wise leaky rectified linear unit $max(0.2x, x)$.

These are the operations up to 13 layers. For the 14th and 15th layers, the dilation part is removed. For the output layer (final layer), we are using 1×1 convolution with bias and no normalization and non-linearity. We describe adaptive normalization below.

3.1.2 Adaptive Normalization

Batch normalization generally improves performance and hastens the training process but degrades performance on some image processing operators. We thus employ an adaptive version of the batch norm that was first proposed in a fast image processing paper [20]. It is a combination of batch norm and identity mapping and is given as:

$$\Psi^s(x) = \lambda_s(x) + \mu_s BN(x) \tag{3.1}$$

where μ_s, λ_s are scalar weights and BN is standard batch normalization. The weights μ_s, λ_s are learned by back-propagation alongside other parameters of the network. These weights allow the model to adapt to the characteristics of the approximated operator, adjusting between the strength of the identity branch and batch normalization.

Now, if we look at our context aggregation network, the receptive field of the pipeline is $2^{14} + 1$ samples, which is about 1 sec of audio having frequency $f_s = 16$ kHz. Thus, overall we expect our network to capture the context on the time scales of spoken words. We move on to our next architecture which is the Feature Loss Network.

3.2 Feature Loss Network

The impressive performance of VGG [23] type networks (e.g.VGG-16, VGG-19) is well-known in computer vision tasks like object detection (e.g. Datasets like **ImageNet**). The design of our Feature Loss network is very much inspired by the VGG-16 architecture. The two important aspects of the network are design and the loss function. We describe them below.

3.2.1 Network Design

Since VGG is a very efficient (in terms of performance) architecture and there are no such standard classification networks in audio processing, the idea of a feature loss network and its design is somewhat similar to VGG. Here, we have a total of 15 convolutional layers. The convolutional kernels are of the same dimensions as in the Denoising network 3×1. Here we aren't doing any adaptive batch norm, a simple standard batch norm is sufficient. After this, we pass it through Leaky ReLU units to get the final activations from a layer. We are using a stride of 2×1, so after each layer we have a decimation by a factor of 2, resulting in halving the length of the subsequent layer as compared to the previous one. In the first intermediate layer, the number of channels equals 2, and after every 5 layers, the number of channels is doubled. In the last feature layer, we do the average pooling to get the output vector. The receptive field of the network is 2^{15} - 1 samples. The architecture is shown in Figure 3.2.

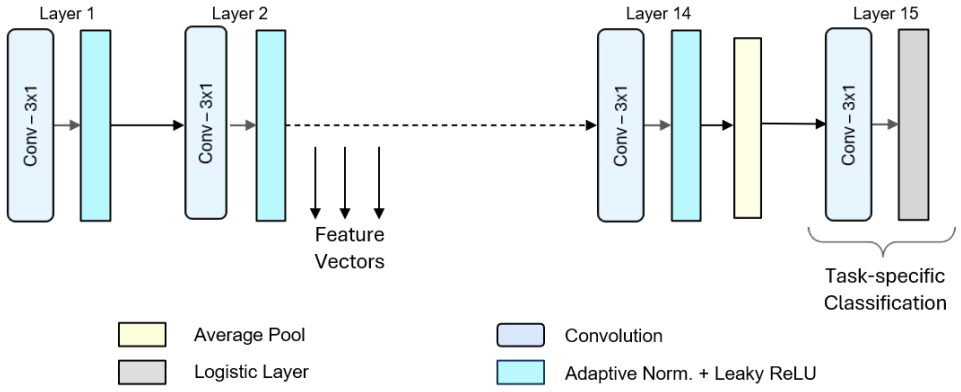


Figure 3.2: The loss network is VGG-inspired architecture. It consists of 15 convolutional layers, with batch normalization. From the output of the last layer, each channel is average-pooled to get the feature vector which is then used for training purposes of 2 different classification tasks

3.2.2 Denoising Loss Function

Suppose we denote the function computed by the m-th feature layer to be Φ^m. Let the input noisy audio be $\mathbf{x}$ and the generator learns the function $g(x;\theta)$ where θ represents the parameters of the generator. The actual clean audio is $\mathbf{y}$. The feature loss between the clean audio signal and the enhanced audio signal by the network

is defined as the weighted L1 loss between the feature activations induced by these signals in different layers of the network. It is computed as:

$$\mathcal{L}_{y,x}(\theta) = \sum_{m=1}^{M} \lambda_m \|\Phi^m(y) - \Phi^m(g(x;\theta))\|_1 \tag{3.2}$$

where the weights λ_m corresponding to the m-th layer are set to balance the contribution from each layer. We are setting these weights to the inverse of the relative values of $\|\Phi^m(y) - \Phi^m(g(x;\theta))\|_1$ after 10 training epochs. We started with weights = 1 for all layers for the first 10 epochs.

We have mentioned already our special loss function. We also experimented with simple loss functions - L1 loss and L2 loss. The output quality was very degraded even at low signal-to-noise ratios (SNRs). It seems like the network doesn't properly process low-energy speech information. So basically, by the nature of this layered network, activations at different layers (or depths) in the network correspond to different time scales in the signal. So we are actually comparing many features at different audio scales and penalizing them.

This kind of approach of using features from different layers and computing losses by using distances between these features (from the noisy and the clean audio) has been used in different contexts. Image stylization [9, 11] heavily relies on the pre-trained VGG (on ImageNet) and uses these feature losses for the stylization. VGG-based feature perceptual losses are good at preserving the content of the input image/audio. The perceptual losses also encourage perceptual consistency between the generated and target output. It makes sure that the generated output not only matches the target output but also captures similar high-level features and structures. Moreover, these losses can help reduce artifacts that might be introduced by the generator network (Denoising network in this case). This is because the loss is based on high-level features, which tend to be smoother and more coherent than pixel-level losses (or audio data losses).

Chapter 4

Network Training

We have two networks to train - the Denoising Network and the Feature Loss Network. We describe the details of the training phase of both the networks along with the dataset being used.

4.1 Feature Loss Network Training

Our feature loss network is not just specific to this speech-denoising task, it is more of a general-purpose feature loss network. That's why we train it on two different audio classification tasks. The reason for choosing these two tasks is because these are very standard and have been used for competitions as well (the same way VGG was used for ImageNet). Thus we can say that our Feature Loss Network is actually doing **Multi-Task Learning**. These two tasks are - 1) Acoustic Scene Classification and 2) Domestic Audio Tagging. We discuss both of these tasks and their datasets.

4.1.1 Acoustic Scene Classification Task

The aim of this task is to classify an audio signal into one of the predefined classes that characterize the environment in which it was recorded. There are 15 acoustic scenes for this task such as "park", "home", "train", "office", etc.

(I) Acoustic Scene Classification Dataset [1]

The training set consists of audio files sampled at 44.1 kHz. The duration of all the audio files is 30 seconds. We resampled the data at 16 kHz. All the files were stereo, so we converted each of them to 2-mono files. This way, we get a total of

2,340 audio files. Talking about the distribution of each class - each acoustic scene has 78 segments, which in total is 39 minutes of audio for each task.

(II) Loss function

Since this is a simple classification task, we use the standard cross-entropy loss to train the network. So if the output from the last layer of the network is say $\hat{y}$, we will first perform a softmax operation on it. The softmax will give the probabilities for each of the 15 classes. Then, we will calculate cross-entropy loss which is backpropagated for training :

$$\text{Loss} = -\sum_{i=1}^{15} y_i \log(\text{softmax}(\hat{y}_i)), \tag{4.1}$$

where all the y_i's are 0 except one that represents the actual class and that element is a 1.

4.1.2 Domestic Audio Tagging Task

This task is based on audio recordings made in a domestic environment. The aim of the task is to perform multi-label classification (assign zero or more labels to each sample). There are 7 classes here like "child speech", "adult male speech", "video game/TV", etc.

(I) Domestic Audio Tagging Dataset

The dataset is taken from **CHiME-Home-refine** [18]. The audio files are already sampled at 16 kHz, so a separate resampling step is avoided. The duration of the audio files is 4 seconds. The training set consists of a total of 1,946 files. The distribution of each class is shown in 4.1.

(II) Loss function

This is a multi-label classification task. We also use the same cross-entropy loss function described in equation 4.1. Whatever the output we get from the last layer, we perform a sigmoid operation on it. The reason is - Sigmoid, unlike softmax

Label	Description	No of occurrences
c	Child Speech	1214
m	Adult Male Speech	174
f	Adult Female Speech	409
v	Video game / TV	1181
p	Percussive sounds	765
b	Broadband noises	19
o	Other identifiable sounds	361

Table 4.1: Data distribution of Domestic Audio Tagging training set

doesn't give probability distribution around the no of classes as output, but independent probabilities. The softmax will give the probabilities for each of the 7 classes. Then, we will calculate the cross-entropy loss :

$$\text{Loss} = -\sum_{i=1}^{7} y_i \log(\text{sigmoid}(\hat{y}_i)), \tag{4.2}$$

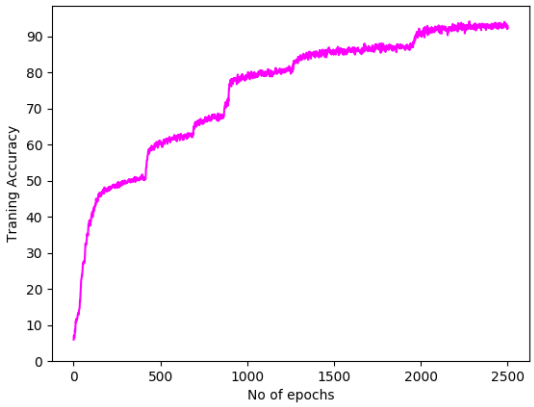
where all the y_i's are 0 except those that represent the actual classes and those are all 1.

Now, we illustrate the training of our Feature Loss network.

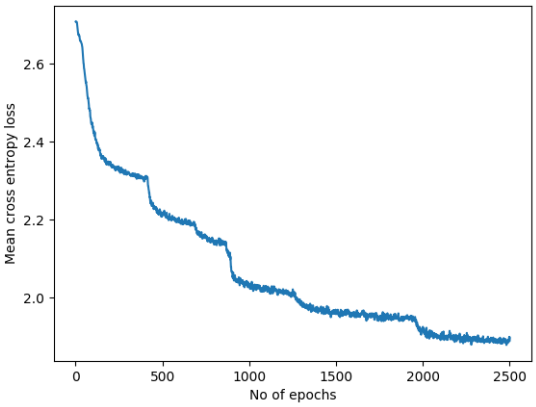
(a) Training Process

We have two different tasks to train our network. Please note, that network initialization is already taken care of by the PyTorch libraries. We are using **Adam Optimizer** [13] with a learning rate of 10^{-4}. The model is trained for a total of 2500 epochs. In each epoch, we loop over the training data from each task and also alternate between samples corresponding to different tasks, which means that we are not first training our network on the first task completely, and then the second task. Instead, in each epoch we iterate over the data for each task and alternate between training samples of each task.

The dataset for the first task is greater than that of the second task, so some of the files in the second dataset are presented a second time to maintain a strict alteration between both tasks. This training procedure is given in the original paper that we implemented [5]. We could have done a weighted resampling consider-



(a) Training Accuracy vs Epochs (Task 1)



(b) Mean cross Entropy Loss vs Epochs (Task 1)

Figure 4.1: Variation of (a) Training Accuracy and (b) Mean Cross Entropy Loss

ing multiple times those samples corresponding to the class with lower cardinality. Thus, in our single epoch, we go through 4680 training samples.

(b) Training Results

The training accuracy that we got for task 1 is **92.8 %**. We also took the mean of the cross entropy loss for each epoch and plotted it as well. It varied from 2.708 to 1.889. Both the graphs are shown in Figure 4.1. We observe fluctuations in the plots because of using batch size = 1.

For the second task, we didn't use the training accuracy measure due to the uneven distribution of the classes instead we used **Equal Error Rate**. It can be defined as a point where false acceptance rates and false rejection rates are equal. It is a variant of the extension of training accuracy for multi-label tasks.

4.2 Denoising Network Training

The denoising network is trained on a standard dataset used in denoising tasks. The training is pretty simple here as compared to the Feature Loss network. We now describe the dataset and loss function.

4.2.1 Dataset

We are using the dataset that is publicly available for speech enhancement methods - **Voice Bank Corpus** [3]. This is the largest available dataset that provides pre-mixed data. It means the noises have already been added to the data. The procedure for this mixing is also there. This dataset is also used in **WaveNet** [24] testing and **SEGAN** [19] testing. So we can compare our results as well. The training set consists of data generated by the speech of 28 speakers (14 males + 14 females). Background noise consists of 10 unique background noises. Each of these noise types is used to generate 4 files with 0, 5, 10, and 15 SNRs. These files are sampled at 48 kHz. We then resample the files to 16 kHz. There are a total of 11,572 files. There are around 400 sentences available from each speaker.

Background Noises

As we already mentioned there are a total of 10 background noises. Out of these two are artificially generated and eight are real noise recordings from a standard database - **Demand Database** [10]. The two artificially generated noises are as below:

1. **Speech-shaped noise**: It was created by filtering white noise with a filter whose frequency response matched that of the long-term speech level of a male speaker.

2. **Babble Noise**: It was generated by adding speech from six speakers from the Voice Bank corpus that were not used either for training or testing.

The other 8 noises were: domestic noise (inside a kitchen), office noise (in a meeting room), three public space noises (cafeteria, restaurant, subway station), two transportation noises (car and metro), and street noise (busy traffic intersection).

4.2.2 Loss Function

We already have described the loss function in Section 3.2.2. Whatever the output that we get from our Denoising network, we pass this enhanced audio and the corresponding clean audio through the Feature Loss network. There we take the L1 loss between the feature activations induced by these signals from the first 6 (the actual paper that we are implementing mentions the use of the 6 layers and the reason is described in the Observations Section) layers of the network.

$$\mathcal{L}_{y,x}(\theta) = \sum_{m=1}^{6} \lambda_m \|\Phi^m(y) - \Phi^m(g(x;\theta))\|_1 \tag{4.3}$$

where the weights λ_m corresponding to the m-th layer are set to balance the contribution from each layer. We are setting these weights to the inverse of the relative values of $\|\Phi^m(y) - \Phi^m(g(x;\theta))\|_1$ after 10 training epochs. We started with weights = 1 for all layers for the first 10 epochs. This loss is backpropagated to train the Denoising network.

(a) Training Process

The network weights are initialized using **Xavier Initialization** [7]. No biases have been used. The adaptive normalization parameters are initialized with $\lambda = 1$ and $\mu = 0$. The loss is computed using the first 6 layers. We are using the **Adam Optimizer** [13] with learning rate = 10^{-4}. The network is trained for a total of 320 epochs. In each epoch, the entire training set is presented in randomized order.

(b) Training Results

We saved the mean of deep feature loss for every epoch. The plot of deep feature loss vs number of epochs is plotted and shown in Figure 4.2. The deep feature loss is continuously decreasing (as we have expected).

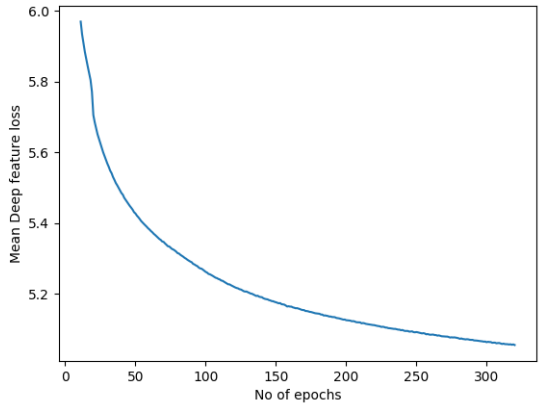
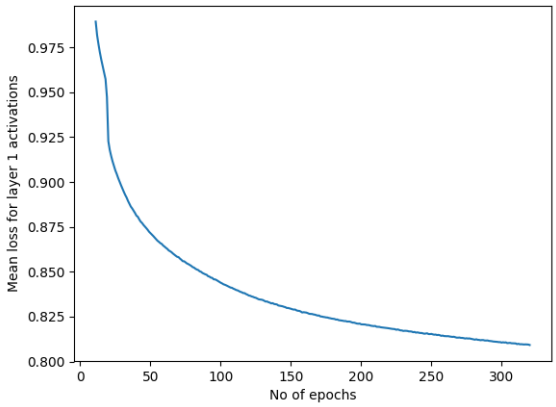


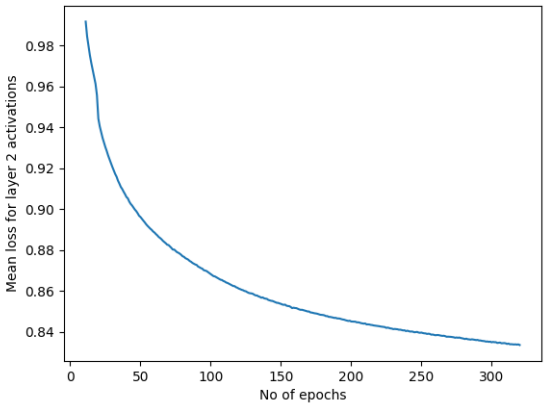
Figure 4.2: Variation of mean deep feature loss vs No of Epochs

To get more insights about the loss, we also analyzed the contribution from all

the layers to the loss. So we also plotted the feature losses between enhanced audio and the true audio for all these 6 layers. The plots for layers 1 and 6 can be seen in Figure 4.3.



(a) Mean of feature Loss vs Epochs (Layer 1)



(b) Mean of feature Loss vs Epochs (Layer 6)

Figure 4.3: Variation of Feature Loss for (a) Layers 1, and (b) Layers 6

We observe that the way in which the feature loss is decreasing is almost the same for layers 1 and 6. One difference to mention is that at any epoch, the feature loss is less for layer 1 as compared to layer 6.

Chapter 5

Experimental Setup and Test Results

For all the training, coding, and testing parts, we have used some environment, language, GPU, etc. It's important to discuss these things as well to get an idea about the overall performance of the model. We describe the experimental setup first, then we will discuss the results.

5.1 Experimental Setup

All the experiments (including training/testing) are done on a machine with the following specifications:

1. Operating System - Ubuntu 18.04.1 LTS
2. CPU - Intel(R) Xeon(R) CPU E5-2620 v4 2.10 GHz with 32 cores
3. RAM - 96 GB
4. GPU - Nvidia GeForce(R) GTX 1080 Ti(2 * 12 GB)
5. Programming Language - Python
6. Machine Learning Library - Pytorch

The model has been trained on a single GPU. The training of the Feature Loss network took approx. 72 hours for 2,500 epochs, and the Denoising network took around 68 hours to train for 320 epochs. The preprocessing part just includes resampling of the data (of both the Denoising and the Feature Loss networks) to 16 kHz. This was done using **SoX** command.

5.2 Testing Results

Now, we have done training both our networks. We will discuss the performance shortly. Let's first talk about the datasets used for testing (both for the Denoising and the Feature Loss network).

5.2.1 Feature Loss Network

We know that the Feature Loss network has been trained on two tasks. So we have these two tasks to test the network as well.

(I) Acoustic Scene Classification Test Dataset

Each acoustic scene has 26 segments totalling 13 minutes of audio for each of the 15 labels. There are a total of 390 audio files each having a 30-second duration. The files are sampled at 44.1 kHz, so we resampled it to 16 kHz.

(II) Domestic Audio Tagging Test Dataset

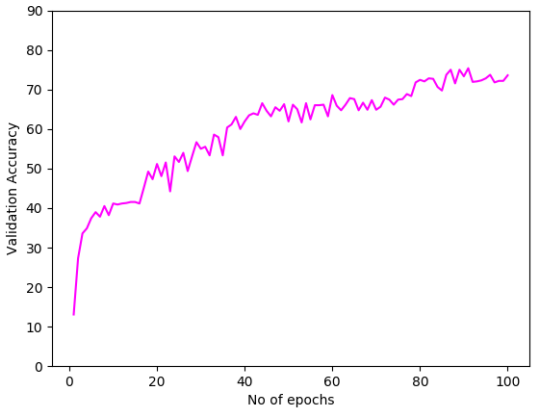
The audio is of duration 4 seconds. Total number of audio files are 816. The total number of labels is 7 (as already told in the training set). All the audio files are already sampled at 16 kHz.

(III) Plots and Results

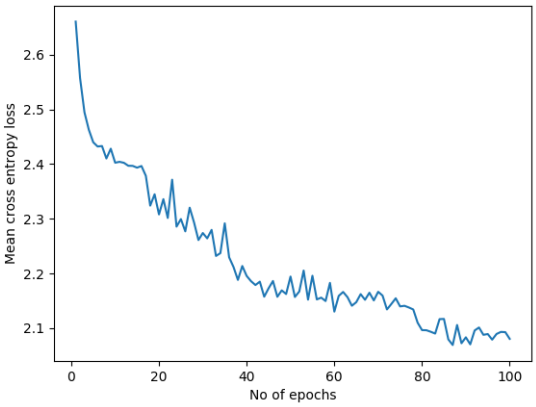
The Feature Loss network has been trained for 2500 epochs. Every 25 epochs of the network's training phase, we tested for this test dataset. Thus, we have a total of 100 epochs where we have observations for test accuracy measured on the test dataset.

For the first task, we plot a graph of accuracy vs epochs to see how testing accuracy is increasing with epochs. We also plot the mean of cross entropy loss vs these 100 epochs. Both the graphs are shown in Figure 5.1. The validation accuracy reaches 75.2 %. The mean cross entropy loss varies between 2.6614 and 2.0201.

For the second task, the same as in the training, here also, we didn't use the training accuracy measure. We again used the Equal Error Rate here. Also to view the variation of loss, we also saved the mean cross entropy loss (same as for Task



(a) Testing Accuracy vs Epochs (Task 1)



(b) Mean cross Entropy Loss vs Epochs (Task 1)

Figure 5.1: Variation of (a) Testing Accuracy and (b) Mean Cross Entropy Loss

1). The equal error rate varies between 0.581 to 0.226. The plot for the same is shown in Figure 5.2.

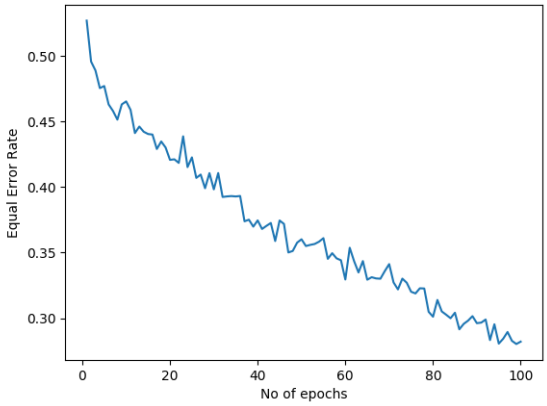


Figure 5.2: Variation of Equal Error Rate vs No of Epochs

5.2.2 Denoising Network

All the testing has been done in mismatched conditions. We first describe the dataset first, and then we will discuss the results.

(a) Dataset

The data source is the same as for training - Voice Bank Corpus. Here, the speech is obtained from 2 speakers (1 male + 1 female). The background data is obtained from 5 different background types. Then each of the background data is used to generate 4 files with 2.5, 7.5, 12.5, and 17.5 SNRs. The complete test set comprises 824 such files. All the data is sampled at 48 kHz, so we resampled the data at 16 kHz.

The chosen noises were domestic noise (living room), office noise (office space), one transport (bus), and two street noises (open area cafeteria and a public square).

(b) Plots

Just like the training, here also we saved and plotted the deep feature loss for the test set. We were running the training dataset for 320 epochs. We tested the test dataset after every 10 epochs, so we basically have 32 epochs of testing. We can see the variation of deep loss vs the number of epochs in Figure 5.3.

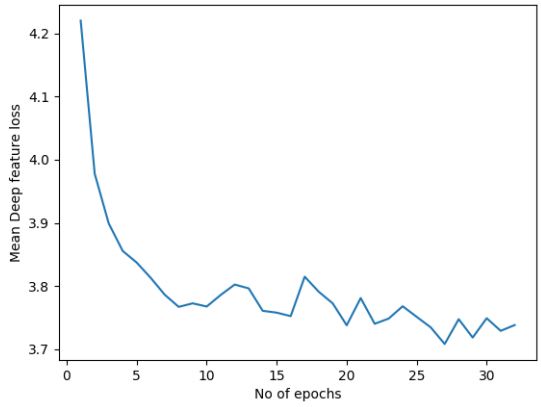
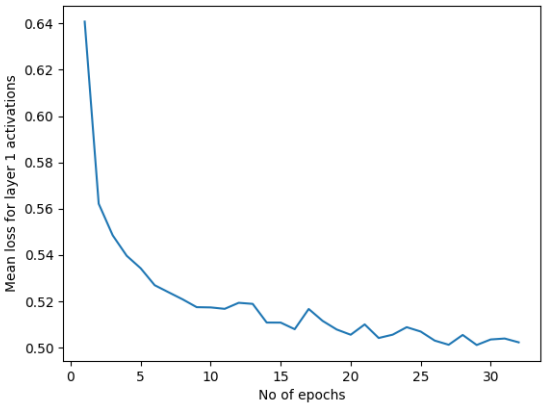


Figure 5.3: Variation of mean deep feature loss vs No of Epochs

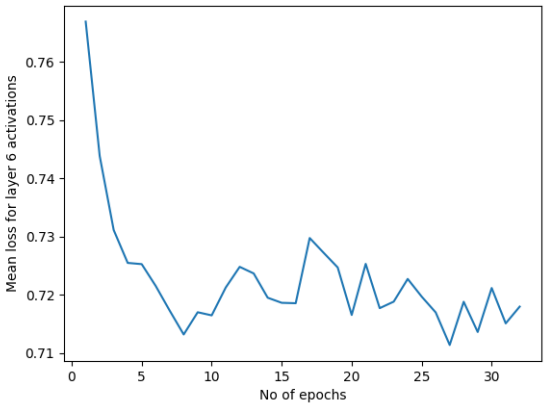
Similar to the training part, here also we plotted the variation of feature loss for layers 1 and 6. The plot can be seen in Figure 5.4.

(c) Evaluation Metric and Results

Objective metrics, which are quantitative measures used to assess audio quality, are recognized for their limited ability to fully align with human audio quality ratings.



(a) Mean of feature Loss vs Epochs (Layer 1)



(b) Mean of feature Loss vs Epochs (Layer 6)

Figure 5.4: Variation of Feature Loss for (a) Layer 1, and (b) Layer 6

To address this, we complement our evaluation process with meticulously crafted perceptual experiments involving human listeners. These experiments are designed to capture the subjective aspect of audio quality. The methodology for gathering human listener feedback is rooted in A/B testing, where two different versions of audio content (A and B) are presented to participants, and their preferences are recorded. These A/B tests are conducted substantially using the Amazon Mechanical Turk platform. This platform enables us to crowdsource human input efficiently. To streamline the execution and analysis of these perceptual experiments, we organize the A/B tests into sets known as **Human Intelligence Tasks** (HITs). This structured approach ensures a systematic and organized evaluation of audio quality as perceived by human listeners.

Each Human Intelligence Task (HIT) in our experiment comprises 100 pairwise comparisons vs. baseline. In each comparison, a worker is presented with two audio clips, and they have the flexibility to play these clips in any order and as many times as needed. One of the audio clips is generated by our approach, while the other is produced by one of the baseline methods, both from the same input within the test dataset. The presentation of these audio files is entirely randomized, both within each pair and among pairs. This means that the worker has no information about which clip comes from our approach and which is from the baseline. The worker's task is to identify, within each pair, the audio clip that contains cleaner speech quality. Following the approach of [5], we also added 0 "sentry" comparisons. These sentry pairs are designed to have a clear and obvious correct answer. They

are mixed into the HIT in a random order. The purpose of these sentry comparisons is to prevent careless or inattentive workers from affecting the results. If a worker provides incorrect responses for two or more of these sentry pairs, the entire HIT is disregarded.

In summary, each HIT comprises a total of 110 pairwise comparisons, consisting of 100 experimental comparisons and 10 sentry comparisons, to assess the cleanliness of speech in audio clips generated by our approach compared to baseline methods. The results are shown in table 5.1. For the Very Easy test set, SEGAN [19] and Weiner [2] perform on similar levels as ours. The superiority of our approach can be seen in the Hard test set. This illustrates the robustness of our algorithm within a specific context, one in which background signal degradation is considerably more pronounced. In such scenarios, the need for denoising is particularly evident. In contrast, for less challenging cases with lower levels of input degradation, both our method and the baseline methods tend to yield satisfactory results. In these cases, listeners may encounter greater difficulty in discerning differences between the various processed audio files.

Comparisons	**Hard**	**Medium**	**Easy**	**Very Easy**
Vs Wiener	94.5 %	85.1 %	78.3%	88.1%
Vs SEGAN	80.8 %	65.7 %	60.5%	59.6%
Vs Wiener	80.2 %	64.8 %	58.3%	54.7%

Table 5.1: Each cell in the table represents the proportion of blind randomized pairwise comparisons in which the listener judged our approach's output as cleaner than the output of a baseline.

Chapter 6

Summary

So far, we have looked at the training and testing part. Based on the results and graphs, we note down some observations and indicate what's remaining to be done.

6.1 Observations

First of all, let us consider the Feature Loss network. Since the network is trained on 2 different tasks, we can see that the test accuracy didn't reach beyond 80 %. Because somehow, the network weights have to change in a way it can learn about both the tasks and not just single task. So the training isn't independent. That's why the accuracy for task 1 also depends on how the network is trained for task 2. We can also see fluctuations in the graph in Figure 5.1. The reason is due to the use of batch size = 1. The same fluctuations happen with cross-entropy losses.

Now, considering the Denoising network results, we can see that the fluctuations in the losses are maximum for layer 6 followed by 5, then 4, and so on. We know that the actual deep feature loss function is a weighted sum of all these layers' losses. So in some way, the high fluctuations in the later layers help the initial layers to learn more. If there weren't many fluctuations in the later layers' losses, then the learning would have been very slow for the initial layers as the losses from later layers get backpropagated to train initial layer parameters. Also, the lowest losses have been recorded for layer 1, which also supports our theory that initial layers are getting faster trained than the later ones.

6.2 Issues and Future Work

6.2.1 Issues

One of the major issues was to decide how to train the Feature Loss network. The reason is because of multi-task learning. Should the optimizers be the same for both or two different optimizers should be used? Another concern is if the optimizers used for different tasks should be the same or not. We used a single optimizer for both tasks. Also, we have used the cross-entropy loss function for both tasks. We are not sure how the Feature Loss model would have converged if we had used two different loss functions. The reason is that if the loss function is the same, then the rate at which weights will change would have differed, and both the losses could have tried to take the weights to some other minimal.

There wasn't any major issue with the Denoising network. A small issue was with the input audio length, as it was a variable that forced us to use batch size = 1. We could have tried clipping all the audio to have some fixed duration but that could have resulted in the loss of information. Also, the training part for the Denoising network was **consuming too much of GPU** (around 6.5 GB).

6.2.2 Future Work

For future work, the most important thing to develop is the performance metric for our Denoising model. Human perception is a very subjective measure. Also, instead of just the first 6 layers, we can use something like Neural Style Transfer [9] loss, which uses losses from some layers at the start and some layers in between - layer 1, layer 3, layer 5, etc. Large fluctuations in the later layers can lead to fast learning for the initial layers.

As already mentioned, the GPU consumption for the Denoising network should be minimized and the code should try to use more than one cores wherever possible. Right now, it just uses a single core mostly in spite of the availability of many cores.

Bibliography

[1] T. Heittola A. Mesaros and T. Virtanen. Tut database for acoustic scene classification and sound event detection. *European Signal Processing Conference (EUSIPCO)*, 2016.

[2] Jacob Benesty, Jingdong Chen, and Yiteng Huang. Study of the widely linear wiener filter for noise reduction. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 205–208, 2010.

[3] S. Takaki C. Valentini-Botinhao, X. Wang and J. Yamagishi. Investigating rnn-based speech enhancement methods for noise-robust text-to-speech. *ISCA Speech Synthesis Workshop*, 2016.

[4] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1511–1520, 2017.

[5] Francois G Germain, Qifeng Chen, and Vladlen Koltun. Speech denoising with deep feature losses. *arXiv preprint arXiv:1806.10522*, 2018.

[6] Ross Girshick. Fast r-cnn. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[7] Xavier Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research - Proceedings Track*, 9:249–256, 01 2010.

[8] Md Foysal Haque, Hye-Youn Lim, and Dae-Seong Kang. Object detection based on vgg with resnet network. In *2019 International Conference on Electronics, Information, and Communication (ICEIC)*, pages 1–3, 2019.

[9] A. Alahi J. Johnson and L. Fei-Fei. Perceptual losses for realtime style transfer and super-resolution. *European Conference on Computer Vision (ECCV)*, 2016.

[10] N. Ito J. Thiemann and E. Vincent. The diverse environments multi-channel acoustic noise database: A database of multichannel environmental noise recordings. *J. Acoust. Soc. Am*, 2013.

[11] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Computer Vision – ECCV 2016*, pages 694–711, Cham, 2016.

[12] S. Chang X. Yang D. Florencio K. Qian, Y. Zhang. Speech enhancement using bayesian wavene. In *Proc. Interspeech*, 2017.

[13] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diega, CA, USA, 2015.

[14] B. Li, Yu Tsao, and K.C. Sim. An investigation of spectral restoration algorithms for deep neural networks based noise robust speech recognition. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 3002–3006, 01 2013.

[15] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. *Advances in neural information processing systems*, 30, 2017.

[16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016.

[17] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105. PMLR, 2015.

[18] S. Krstulovic J. Barker P. Foster, S. Sigtia and M. D. Plumbley. Chime-home: A dataset for sound source recognition in a domestic environment. *IEEE Work-*

shop on Applications of Signal Processing to Audio and Acoustics (WASPAA), 2015.

[19] Santiago Pascual, Antonio Bonafonte, and Joan Serra. Segan: Speech enhancement generative adversarial network. *arXiv preprint arXiv:1703.09452*, 2017.

[20] J. Xu Q. Chen and V. Koltun. Fast image processing with fullyconvolutional networks. *International Conference on Computer Vision (ICCV)*, 2017.

[21] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[22] X. Lu S.-W. Fu, Y. Tsao and H. Kawai. Raw waveform-based speech enhancement by fully convolutional networks. *arXiv:1703.02205*, 2017.

[23] K Simonyan and A Zisserman. Very deep convolutional networks for large-scale image recognition. pages 1–14. Computational and Biological Learning Society, 2015.

[24] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv:1609.03499*, 2016.

[25] Y. Wang and D. Wang. Cocktail party processing via structured prediction. *Neural Information Processing Systems (NIPS)*, 2012.

[26] Richard M. Warrren, Keri Hainsworth, B Brubaker, James Bashford, and E Healy. Spectral restoration of speech: Intelligibility is increased by inserting noise in spectral gaps. *Perception & Psychophysics*, 59:275–283, 1997.

[27] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations (ICLR)*, 2016.