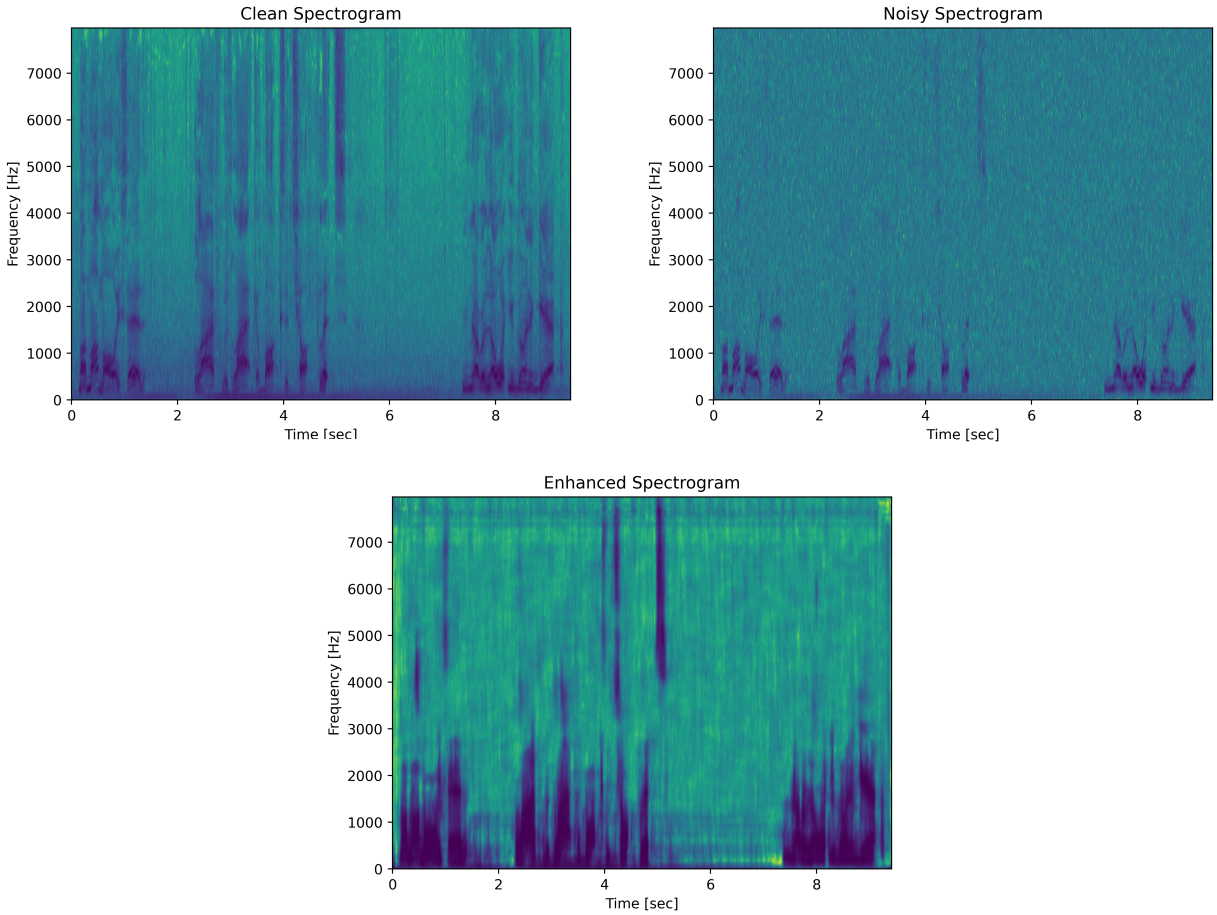
Bias-Free Convolutional Neural Networks for Speech Enhancement

- Master Thesis -



Project Report

Jakob Krarup Thomsen
James Peter Harris
Group 1072

Aalborg University
Department of Electronic Systems
Fredrik Bajers Vej 7B
DK-9220 Aalborg

This report was written in LaTeX and has been shared online in-between the group members using Overleaf licensed to all students at Aalborg University.

AALBORG UNIVERSITY

STUDENT REPORT

Department of Electronic Systems
Fredrik Bajers Vej 7
DK-9220 Aalborg Ø
http://es.aau.dk

Title:
Bias-Free Convolutional Neural Networks for Speech Enhancement

Theme:
Signal Processing and Acoustics

Project Period:
Spring Semester 2021

Project Group:
Group 1072

Participants:
Jakob Krarup Thomsen
jkth16@student.aau.dk

James Peter Harris
jharri16@student.aau.dk

Supervisor:
Zheng-Hua Tan
Jesper Jensen

Report Page Numbers: 39

Appendix Page Numbers: 13

Date of Completion:
June 3, 2021

Abstract:

Within the field of signal processing, a commonly occurring problem is that of denoising signals. This problem is especially relevant within the domain of speech processing. In recent years, deep learning models have shown state of the art performance in speech enhancement applications, surpassing previous methods in both objective and subjective performance. Deep learning models have previously suffered from reduced performance on unknown noise levels, however, a recent discovery within the field of image processing, indicates that bias-free models can generalise better across noise levels [1].
This report does not seek to create a new state of the art within speech enhancement, but instead investigates the implications of these bias-free models. For this, four different types of convolutional neural networks were selected and evaluated for their performance under both bias-free and conventional configurations.
Generally, bias-free networks are not found to have any significant improvement in generalisation over regular networks. However, UNet achieved significantly better performance, in a bias-free configuration, within known SNR ranges and marginally better outside known SNR ranges. A denoising CNN with a conventional configuration performed best overall.

Preface

We would like to thank the supervisors, Jesper Jensen and Zheng-Hua Tan, for their excellent supervision and motivation throughout this project. Additionally, we would also like to thank fellow student Thor Pilgaard Knudsen, who helped with us with some of the problems we had during implementation. Finally, none of the results presented in the project would have been possible without the use of the CLAAUDIA server, which is being maintained by Aalborg University IT support, who have also helped us check through code to ensure it ran properly on CLAAUDIA.

The project was undertaken with inspiration from the paper: *Robust and Interpretable Blind Image Denoising via Bias-free Convolutional Neural Networks* [1].

Contents

Chapter 1

Introduction

The need for speech enhancement arises from a speech signal being corrupted by some unknown noise, making it harder to hear what is being said. This corruption can take place in many environments, such as through a telephone when there is poor service, or when speaking in a factory or construction site where there is heavy machinery operating [2], or any situation where multiple people are speaking simultaneously, and someone is trying to focus on the words of one speaker, and in doing so ignoring the rest. This is commonly referred to as the cocktail party problem.

All of these are sources of noise. For a person with normal hearing in daily life, the brain is sufficiently adaptable to render the original speech intelligible in many, but not all cases. For the hearing-impaired however, this becomes a significant challenge, with hearing aid and cochlear implant users experiencing extreme difficulty communicating in noisy environments [2]. Thus, the principal aim is to design a system to remove this noise, thereby reducing fatigue and allowing hearing-impaired people to communicate better in daily life.

Note that there is no prior knowledge of the signal to noise ratio in any instance, and it is easy to see how this would vary in time, such as a person moving from a relatively quiet back room to the factory floor. This means that for any speech enhancement system to be relevant to real-world applications, it must work with a wide range of different signal to noise ratios.

Many methods for audio enhancement exists and it is still an active area of research, particularly within the field of deep learning. The oldest methods rely on statistical estimations of the noise, necessitating knowledge of the noise ahead of time, which does not always hold for real-world applications. Newer methods employ adaptive filter theory to better enhance the audio signals, with varying degrees of success on non-stationary noise types. Deep learning methods are the newest area of research and have already surpassed most traditional methods [3], especially in subjective comparisons [4].

Deep-learning based denoising is reliant on being trained on the exact noise types and levels they are to be used to remove. While this is not necessarily an issue as training is usually performed on powerful hardware, it still requires knowledge of the situation the model is to be used in. To potentially alleviate the requirement to train across all applicable noise levels, this project follows the work in [1], wherein the additive bias terms of neural networks are removed, resulting in better noise level generalisation for image denoising. Specifically, the bias-free methods applied in [1] will be applied to speech enhancement networks in this project.

Hypothesis: *Removing the additive bias terms from neural networks improves network generalisation across unknown noise levels for speech enhancement purposes.*

This report first provides an overview of the necessary theory and principals for deep learning methods in Chapter 2, followed by a state of the art analysis in Chapter 3. An overview of speech enhancement systems, including time-frequency domain conversions is found in Chapter 4. These conversions are ideal, as they allow the networks to interpret the audio information as spectrograms, which are more akin to images, hopefully resulting in better performance and comparison with the image denoising presented in [1]. Chapter 5 explains the methodology of the development of a speech enhancement system, intended to show the viability of bias-free speech enhancement. Results are presented in Chapter 6, investigating the generalisation benefits of bias-free neural networks. Finally, a discussion of the projects findings and any further work/shortcomings are discussed in Chapter 7 before being concluded in Chapter 8.

Chapter 2

Deep Learning Theory

In this chapter, a brief introduction to neural networks and back-propagation is provided, and more elaborate concepts such as convolutional layers, training algorithms and the idea behind bias-free denoising is explored.

2.1 Neural Networks

To properly understand the operations of a convolutional layer it is first necessary to understand densely connected layers and how they interact. A neural network is a collection of interconnected processing units referred to as *neurons*, organised into layers. Each neuron performs a calculation using the output of the previous layer as it's input. It then passes the result on to the next layer, until eventually an output layer is reached.

This result can take the form of either a classification such as whether the input is speech or noise, a male or female speaker etc., or a regression such as finding a speech signal from a corrupted input. There is little difference in network design between classification and regression problems. This means that for the purpose of understanding the principals behind neural networks, a regression problem can be assumed without loss of generality.

Within this framework, a neural network with n layers can be generally described as having a structure similar to Figure 2.1, with $\mathbf{x}$ and $\hat{\mathbf{x}}$ representing the input and output signals respectively, $\mathbf{l}$ being used to represent the layer of neurons as a vector, and $\mathbf{a}$ the vector output of each layer. The use of the term *feed-forward* implies that there are no stages where the output of one layer arcs back to influence previous layers. This is a simplification made for the purposes of initial explanation.

2.1.1 Neurons

Each layer consists of at least a collection of neurons, although other functionality may be included as well. For the purposes of this explanation, all layers are assumed to be densely connected and consist only of neurons.

Using the notation in Figure 2.1, it is possible to describe the functionality of an individual neuron both diagramatically as in Figure 2.2, and mathematically as;

$$a_l = f(\mathbf{w}_l \cdot \mathbf{a}_{l-1} + b_l) \tag{2.1}$$

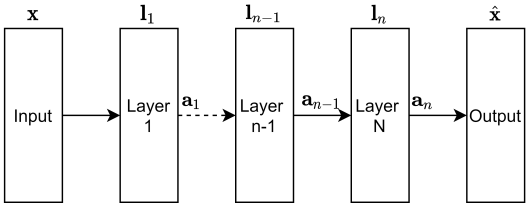


Figure 2.1: Block diagram of a feed-forward neural network

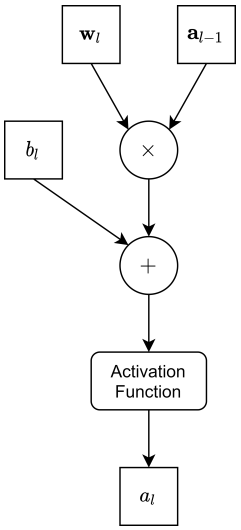


Figure 2.2: Block diagram of a single neuron

Where $\mathbf{w}_l$ is a row vector of weights associated with this particular neuron, which is multiplied with $\mathbf{a}_{l-1}$, the column vector containing the output of all the neurons in the previous layer. a bias term b_l associated with that perceptron is then also added, before evaluation with respect to the activation function $f(\cdot)$ to produce the output of a single neuron in the l-th layer, a_l.

2.1.2 Back-propagation

It is straightforward to see from both Figure 2.2 and Equation (2.1) how the output from one layer can be interpreted as the input for the next. This allows for both the diagram and equation to be cascaded outwards until a direct representation of the output for any given input is reached, with the equation taking the form of;

$$\hat{x} = f(\mathbf{w}_n \cdot (f(\mathbf{w}_{n-1} \cdot f(\mathbf{w}_{n-2} \cdot (\dots) + b_{n-2}) + b_{n-1})) + b_n) \tag{2.2}$$

The recursive nature of this equation makes it unwieldy to work with in all but the smallest networks. Also, recalling that this is only the equation for one value at the output, and that the whole output vector must be trained to produce the desired result. As such, there would be multiple, interdependant instances of this recursion taking place.

As this is extremely inefficient to work with, a back-propagation method is used instead. For this, a model is derived for how the error passes through the layer, and what adjustments to the weights and biases are required on each layer, given a specific error.

First, the *cost* of the output, relative to each input is defined, with a ground truth value $\bar{\mathbf{x}}$ being present for each input in the training cycle. This is a function that returns a score, relative to the veracity of the network's output. In this case, the Mean Squared Error (MSE) cost function is defined for simplicity, but others more appropriate for each application may be selected and applied without loss of generality. MSE is defined as;

$$c(\hat{\mathbf{x}}, \bar{\mathbf{x}}) = \sum_n (\hat{x}_{[n]} - \bar{x}_{[n]})^2 \tag{2.3}$$

where $[n]$ is used to index along elements in the output and ground truth vectors. Note also, that this

cost function implicitly encodes information about each weight and bias term in the network, from the output term $\bar{\mathbf{x}}$.

Now, it is possible to derive for each layer, the effect of changing each of the weight and bias terms. This is accomplished through the use of partial derivatives and the chain rule.

With the output of a single neuron already shown in Equation (2.1), a new variable z is introduced for future simplicity of notation. This allows us to re-write Equation (2.1) as;

$$a_l = f(\mathbf{w}_l \cdot \mathbf{a}_{l-1} + b_l) \tag{2.4}$$

$$\begin{aligned} a_l &= f(z_l) \\ z_l &= \mathbf{w}_l \cdot \mathbf{a}_{l-1} + b_l \end{aligned} \tag{2.5}$$

To find the effect of changing the weights of a neuron in a layer on the cost of the network, it is possible to use Figure 2.2, together with Equation (2.4) to define the relationship using the chain rule;

$$\frac{\partial c}{\partial \mathbf{w}} = \frac{\partial z}{\partial \mathbf{w}} \frac{\partial f}{\partial z} \frac{\partial c}{\partial f} \tag{2.6}$$

Similar is then in turn possible for both the bias b, and the input from the previous layer $\mathbf{a}_{l-1}$;

$$\frac{\partial c}{\partial b} = \frac{\partial z}{\partial b} \frac{\partial f}{\partial z} \frac{\partial c}{\partial f} \tag{2.7}$$

$$\frac{\partial c}{\partial \mathbf{a}_{l-1}} = \sum_{i=0}^{\dim(\mathbf{a}_{l-1})} \frac{\partial z}{\partial \mathbf{a}_{l-1}} \frac{\partial f}{\partial z} \frac{\partial c}{\partial f} \tag{2.8}$$

Note the summation in Equation (2.8) appears as a direct result of multiple neurons outputs from the previous layer having an effect on this derivative. The derivative of cost with respect to weight on the other hand does not need this summation, as all weights are self-contained within that neuron, and can be treated individually as a vector of gradients.

As such, back-propagation requires both the cost function, and activation function to be at least piecewise differentiable. This makes available a large range of possible cost and activation functions for implementation, while retaining the same updating scheme.

2.2 Convolutional Layers

The convolutional layer itself consists of a number i of filters $\mathbf{f}$ (sometimes referred to as kernels) of given dimensions $d_{\mathbf{f}}$, which convolve the input with their respective weights, both as matrices. Each filter, when convolved with the input $\mathbf{x}$, produces an output commonly referred to as a *feature map*, notated $\mathbf{y}_i$ [5].

Given the number of feature maps in a layer is equal to the number of filters, it is possible to detect many fundamental features of an image within a single layer, in a trainable manner. As a mental model, it is useful to think of these initial features as, for example, lines or edges. That is, the first

filter layers represent the most fundamental characteristics found within the input image. The key here is that the kernels that detect these features remain the same across the entire image.

The power of convolutional layers becomes apparent when multiple convolutional layers are stacked on top of one another. Recalling that in the first layer, one input image was mapped to many different feature maps. in subsequent convolutional layers, these feature maps are stacked on top of each other to produce a single many-channeled input to the second layer.

With all features from the previous layer now encapsulated, convolution can be performed again. Note however that when the number of channels increases, so too does the dimensionality (and hence number of parameters) of the kernel. As a worked example, consider a single channel image as an input, to be convolved with 16 3×3 kernels in the first 2D convolutional layer. This then produces 16 feature maps, the same size of the original image. The second 2D convolutional layer then sees these 16 feature maps stacked together, and so requires a $3 \times 3 \times 16$ kernel to produce each feature map, which still produces a single feature map of the same dimensionality as the first layer.

In stacking all feature maps before passing to the next layer, the next convolutional layer will consider combinations of features in creating the feature maps. This slowly builds complexity within the feature maps and is the primary motivation for their use. The translational invariance introduced by the convolution itself also aids in situations where features may not be static within an image, such as a series of aerial photographs, or a spectrogram of a sound signal through time [5].

This uses substantially less parameters within a layer than a densely connected layer, especially in the case of images [5]. The neurons within a densely connected layer take all pixels of the image as input simultaneously. This means that while any spatial characteristics of which parts of an image contribute to a neuron's activation may remain, they are obfuscated within the weights. Additionally, many more parameters are needed to accomplish the same task, as a larger number of neurons is needed to represent the image. This property of parameter sharing has also contributed to the widespread adoption of Convolutional Neural Networks (CNNs).

As in densely connected layers, a bias term b may also be added to the feature map of the convolutional layer. This is not pixel-wise, but determined separately for each feature map by back-propagation using the entire output of the convolutional layer.

Following, the back-propagation process is explored within the context of a 2-dimensional convolution, such as that used in CNNs for image enhancement. This method is based on the tutorial [6].

2.2.1 Forward Pass

For an image $\mathbf{x}$ of arbitrary size, indexed by (i, j), and a filter with weights given by $\mathbf{f}$ indexed by (k, l) the (biased) convolution operation may be expressed as;

$$\mathbf{y} = \mathbf{x} * \mathbf{f} + b \tag{2.9}$$

$$\mathbf{y}_{(i,j)} = \sum_k \sum_l \mathbf{f}_{(k,l)} \mathbf{x}_{(i+k,j+l)} + b \tag{2.10}$$

Where $\mathbf{y}_{(i)}$ corresponds to the i-th element of the feature map $\mathbf{y}$. It is this operation that the partial derivatives with respect to bias, kernel weights and previous layers output are found.

Each convolutional layer may have many kernels, each producing their own feature map. These are

then stacked, before further processing in subsequent layers. Note that this does not result in higher dimensional convolution, however the kernels do require proportionally more weights to account for the higher dimensionality.

After a signal passes through all layers of the network, it is evaluated with its ground truth to generate a loss, which indicates how well the network has performed on that sample in its current state. This loss is fundamental to the back-propagation process

2.2.2 Backward Pass

The backward pass through the network defines how the cost of the network is used to iteratively change the network's parameters, slowly converging on a (locally) optimal cost.

To accomplish this, the same back-propagation procedure as described in Section 2.1.2 is used. However, there are now convolutional layers in the place of densely connected neurons, for which a corresponding set of partial derivatives is required in order to maintain the same back-propagation framework. These will be derived in this subsection, following which the relationship between densely connected and convolutional layers will be explored.

The cost of the previous layer is first determined by a cost function $c(\mathbf{x}, \hat{\mathbf{x}})$. In the case of the final layer in the network, this will be the overall cost of the network, however this is treated in the same way. In the case of a convolutional layer, this loss is expressed element-wise on the feature map as;

$$d\mathbf{y}_{(i,j)} = \frac{\partial c}{\partial \mathbf{y}_{(i,j)}} \tag{2.11}$$

Gradient with Respect to Bias

The bias term, being constant on all elements within the feature map, is the easiest to calculate a gradient for. First, recalling the chain rule;

$$\frac{\partial c}{\partial b} db = \frac{\partial c}{\partial \mathbf{y}_{(i,j)}} \frac{\partial \mathbf{y}_{(i,j)}}{\partial b} \tag{2.12}$$

Then, a summation is taken to account for the fact that the bias is constant across all elements $\mathbf{y}_{(i,j)}$;

$$db = \sum_i \sum_j d\mathbf{y}_{(i,j)} \cdot \frac{\partial \mathbf{y}_{(i,j)}}{\partial b} \tag{2.13}$$

Then, it is possible to set all values in $\frac{\partial \mathbf{y}_{(i,j)}}{\partial b}$ to one. This is because the bias term both affects all elements in the feature map equally, and has a unitary gain to them. This then leads to the gradient with respect to the bias being defined in simplified form as;

$$\frac{\partial c}{\partial b} = db = \sum_i \sum_j d\mathbf{y}_{(i,j)} \tag{2.14}$$

Gradients with Respect to Filter Weights

It is necessary to account for the parameter sharing introduced by the convolution in the forward pass when calculating the gradients of the filter weights. Just as a change in a single filter weight

has a change on the entire feature map, it also has a change on the cost function. This necessitates all weights being changed simultaneously, with regards to the same cost function. In contrast to calculating the gradient of the bias, this is a significantly more complicated operation. First as in the case for the bias, the chain rule is recalled as;

$$\frac{\partial c}{\partial \mathbf{f}} = d\mathbf{f} = \frac{\partial c}{\partial \mathbf{y}_{(i,j)}} \frac{\partial \mathbf{y}_{(i,j)}}{\partial \mathbf{f}} = d\mathbf{y} \frac{\partial \mathbf{y}}{\partial \mathbf{f}} \tag{2.15}$$

Now the calculation of the weights in a convolutional layer can be shown to itself involve a convolution. To do this, the gradients of the kernel are also indexed by (m, n). This then allows each gradient of the kernel to be tied to the values it corresponds with in the feature map, as expressed as;

$$d\mathbf{f}_{(m,n)} = d\mathbf{y}_{(i,j)} \frac{\partial \mathbf{y}_{(i,j)}}{\partial \mathbf{f}_{(m,n)}} \tag{2.16}$$

Then, substituting the derivatives into the forward pass in Equation (2.9) means that;

$$\frac{\partial \mathbf{y}_{(i,j)}}{\partial \mathbf{f}_{(m,n)}} = \sum_k \sum_l \frac{\partial \mathbf{f}_{(k,l)}}{\partial \mathbf{f}_{(m,n)}} \mathbf{x}_{(i+k,j+l)} \tag{2.17}$$

Observing the properties of the partial derivative of the filter weights with respect to itself, it is noted that this is an indicator function such that;

$$\frac{\partial \mathbf{f}_{(k,l)}}{\partial \mathbf{f}_{(m,n)}} = \begin{cases} 1 & \text{if } k, l = m, n \\ 0 & \text{otherwise} \end{cases} \tag{2.18}$$

This means it is now possible to remove the summation, as only one case yields a non-zero gradient, as given by;

$$\frac{\partial \mathbf{y}_{(i,j)}}{\partial \mathbf{f}_{(m,n)}} = \mathbf{x}_{(i+k,j+l)} \tag{2.19}$$

Substituting back into Equation (2.16) then reveals the final result;

$$\frac{\partial c}{\partial \mathbf{f}} = d\mathbf{f} = d\mathbf{y}_{(i,j)} \mathbf{x}_{(i+k,j+l)} \tag{2.20}$$

Which is in fact a convolution of the cost of the output as defined in Equation (2.11) onto the input, which can be written as;

$$\frac{\partial c}{\partial \mathbf{f}} = d\mathbf{f} = \sum_i \sum_j d\mathbf{y}_{(i,j)} \mathbf{x}_{(i+k,j+l)} \tag{2.21}$$

This gives a vector containing all the gradients of the filter weights, allowing them to be updated accordingly.

Gradient with Respect to Input

The final result needed to be able to apply back-propagation through the convolutional layer and onto the next layer, is the gradient of the cost function with respect to the input. As the forward pass is of course a convolution, the principal behind this step is to "de-convolve" the gradient from the feature map, and as such this step also makes use of convolution. In a similar way to both the filter weights and bias, a derivative in chain rule form is obtained;

$$\frac{\partial c}{\partial \mathbf{x}_{(m,n)}} = d\mathbf{x}_{(m,n)} = d\mathbf{y}_{(i,j)} \frac{\partial \mathbf{y}_{(i,j)}}{\partial \mathbf{x}_{(m,n)}} \tag{2.22}$$

Just as in Equation (2.17), the forward pass is used to first define;

$$\frac{\partial \mathbf{y}_{(i,j)}}{\partial \mathbf{x}_{(m,n)}} = \sum_k \sum_l \mathbf{f}_{(k,l)} \frac{\mathbf{x}_{(i+k,j+l)}}{\mathbf{x}_{(m,n)}} \tag{2.23}$$

which, after simplification forms a similar indicator function;

$$\frac{\mathbf{x}_{(i+k,j+l)}}{\mathbf{x}_{(m,n)}} = \begin{cases} 1 & \text{if } k = m - i, l = n - j \\ 0 & \text{otherwise} \end{cases} \tag{2.24}$$

However, with a kernel significantly smaller than the input image, setting k and l such that $\frac{\mathbf{x}_{(i+k,j+l)}}{\mathbf{x}_{(m,n)}} = 1$ means that $\mathbf{f}$ is larger than its defined size. This is because just as the feature map from the forward pass is already of lower dimension than the input, convolving in the other direction, as in this step, will also reduce dimensionality. As such, the filter $\mathbf{f}$ is zero-padded so as to assure Equation (2.24) is non-zero. This then gives;

$$\frac{\partial c}{\partial \mathbf{x}_{(m,n)}} = d\mathbf{x}_{(m,n)} = \sum_i \sum_j d\mathbf{y}_{(i,j)} \mathbf{f}_{(m-i,n-j)} \tag{2.25}$$

The forward pass as defined in Equation (2.9) can expanded to show the multiplications of filter $\mathbf{f}$ and image $\mathbf{x}$ used to produce each element in the feature map $\mathbf{y}$.

This then proves that the convolution operation is linear, and can be represented as a matrix multiplication, with the filter coefficients arranged in a doubly block circulant matrix [5]. This then means that the forward pass can be optimised, as the convolution operation can be removed with no impact on mathematical accuracy.

This linearity then means that nonliniarities must be introduced within the network for it to have the desired "decision-making" characteristics, else the whole network could be simplified to a series of linear transforms. This is often accomplished by passing each feature map through some non-linear activation function, such as a ReLU. This can be seen in practice in [7].

2.2.3 Matrix-based Methods

Given that the 2-dimensional convolution can be framed as a multiplication of an (flattened) image $\mathbf{x}$ and a doubly block circulant matrix of filter weights f [5], the computations in the convolutional layer itself can be optimized, such that no actual convolution is performed, with the result remaining the same. This involves removing the summations, and in doing so also removes the need for loops in the computation.

This approach has also been used to minimise the computations in CNNs by extending the Strassen algorithm for matrix multiplication [8]. This new framework of understanding may have further implications to the interpretability of CNNs, however this remains outside the scope of this project.

2.3 Dimensionality of Feature Maps

The dimensionality of the input and output of the convolutional layer may not be the same, depending on whether zeros have been padded around the periphery of the image. This is because the number of unique ways a kernel of a given dimension can be convolved onto an image can be given by [6] [9]:

$$d_{\mathbf{y}} = \frac{d_{\mathbf{x}} - d_{\mathbf{f}} + 2p}{s} + 1 \tag{2.26}$$

where d_y, d_x, d_f is the dimension of the output, input and filter respectively, p is the amount of zero padding and s the stride of the filter. While this represents a single dimension, it can be applied to each dimension of the image and kernel similarly to give the total size of the image map.

Zero padding has been shown to at the very least not introduce boundary artefacts where the image dimensionality between input and output must remain constant [10]. While it is not clear from the literature if removing padding would result in these artefacts appearing, they remain undesirable in all aspects of this project, so zero padding shall be used to maintain dimensionality. Additionally, the inclusion of this "excess" information in the feature map that the zero padding causes is not expected to be of any detriment to performance.

Just as zero padding acts to preserve dimensionality between an input and it's feature maps, there also exist cases where it is desirable to have a change in dimensionality, in a controlled and well defined manner. In the following subsections, several methods of reducing and increasing the dimensionality of feature maps are discussed.

2.3.1 Reduction

Historically, pooling has seen most use for feature map reduction within convolutional neural networks, due to its ability to reduce the dimensionality of the feature map while broadly retaining the information contained within it [5].

Pooling is a form of dimensionality reduction applied after a convolutional layer, that seeks to reduce the feature map (and hence the input space of the next layer) by removing potentially irrelevant information. This is done by applying some averaging operation over the image in a manner similar to convolution, but whereby each element of the original feature map only appears in the pooling operation once.

There are several pooling strategies, such as average pooling (passing the average of a window of values) or max pooling (passing the maximum of a window of values). Pooling is commonly applied on the 2-dimensional image without overlap, such that each value in the original image features only once in any window.

So far, no applications of pooling within the realm of speech enhancement have been found. The current state of the art indicates away from any pooling operations, as they remove information wherever they are included. A suggested alternative is including convolutional layers with different strides, such as found in [11]. A noted caveat of this approach is that the network must be large *enough*

for the dataset it is being trained on, such that the properties of the data can be captured within the convolutional layers. Additionally, pooling-free networks are said to have better performance for generative models, such as variational autoencoders [12].

Note that the purpose of this project is to investigate the generalisation capabilities of bias-free networks. In this context, the pooling layer can be shown to have no impact on the transmission of the bias term. The (max) pooling operation is first defined on an input feature map in 2 dimensions $\mathbf{y}$ as;

$$\mathbf{y}_{(x,x+l),(y,y+l)} = \lceil \mathbf{y}_{(x,x+l),(y,y+l)} \rceil \tag{2.27}$$

where l determines the size of the pooling operation, and x, y are used to index the image in the x and y-axes respectively. Then, from the convolutional operation used to generate the feature map;

$$\mathbf{y}_{(x,y)} = (\mathbf{x}_{(x,y)} * \mathbf{f}) + b = (\sum_i \sum_j \mathbf{f}_{(i,j)} \mathbf{x}_{(x+i,y+j)}) + b \tag{2.28}$$

The bias is then shown to be an additive and constant across all x, y. This implies that the same information is contained within the feature map, and that the bias provides only some offset to that information, that when paired with some non-linear activation, would allow the network to learn to select parts of the feature map that are relevant, based on this bias.

Further in relation to the pooling operation itself, the value of b has no impact on which element within a pool is the maximum. This means that pooling can safely be removed from any desired network topology with no impact on the study of bias-free architectures. Based on this, and the current shift towards pooling-free architectures, pooling-free networks will be preferred over networks that include pooling.

2.3.2 Enlargement

As *the curse of dimensionality* generally means the reduction in dimensionality of data, both at the input and in the feature maps, is favoured, some noteworthy cases remain where it is necessary to increase the size of the feature maps in a controlled manner. One such example is the fully convolutional autoencoder [13], in which the sizes of the feature maps are first reduced to form a latent space representation, from which the original image may be reconstructed.

A simple form of upsampling found in convolutional neural networks involves simply repeating the value within the feature map. This is seen in some convolutional autoencoder implementations, such as [13]. This means that unlike in pooling, the information stored within the feature map has been preserved.

Another form of upsampling is transpose convolution. In this process, instead of looking at the image through a filter to produce an element in the feature map, each element in the lower dimensional input is fed through a filter to produce a cluster of values on the feature map. Then, convolving the kernel across the feature map in a similar way allows for a learned upsampling of the input. Similarly to convolution, transpose convolution can also use different strides in the filter in the same way. This enables both down- and upsampling using only the stride parameter.

Although the original UNet proposed in [14] includes upsampling by repeated values, the UNet used in [1] includes a transpose convolutional layer instead. Transpose convolutions will be used within the networks in this project due to their learned upsampling characteristics, although this design decision is somewhat arbitrary as contrary to downsampling, there is no potential for information loss.

2.4 Training algorithms

Whilst back-propagation details the method by which a cost is passed backwards through the network to update each parameter, it does not detail the exact updating method. One of the key implications of back-propagation is that it enables the implementation of a gradient descent optimisation method on each layer, regardless of layer type. Most intuitive of these is stochastic Gradient Descent (SGD), however adaptive moment estimation (ADAM) is the most contemporary and widely used in literature [15].

Stochastic Gradient Descent

In SGD, a fixed step size (often referred to as *learning rate*) α is introduced, that will be used to update all trainable parameters.

Initially, all parameters in the network are randomized. This is done to assure that the network gives some form of output that can be iterated over. However because of this randomization, and the highly non-convex nature of the objective, seperate initializations will not necessarily produce the same output. It is however sufficient to say that after the training process, the objective has been minimised to a local minima.

Following this, the training dataset is grouped at random into equally sized batches, and a validation dataset is partitioned to be kept for potential parameter adjustment within the training loop. This is the key difference between gradient descent, and stochastic gradient descent.

The average cost of each batch is then calculated, and back-propagated through the network using the relevant partial derivatives with respect to weights, biases and previous activations, on a layer-wise basis. The parameters of the network are therefore updated between batches.

Adaptive Moment Estimation

Adaptive moment estimation (ADAM) uses many of the same techniques found in SGD. The main difference is the handling of the learning rate. in SGD, this is fixed across all parameters, which leads to a slow initial convergence but a better defined minima with a lower learning rate, and the opposite when the learning rate is set too high. This introduces a trade-off within the selection of the learning rate. ADAM eliminates this trade-off by allowing the learning rates to adapt, according to the first and second statistical moments (mean and variance) of the gradients in each batch [15]. ADAM is the optimization algorithm used in this project, as it allows for faster training with variable learning rates ensuring decent convergence to a local minima.

While ADAM does still include a step size parameter, α, this should not be thought of in the same way as the step size in SGD. It also introduces two decay rates for the mean and variance estimates, β_1, β_2 respectively, with the published results given for $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$, with an epsilon $\epsilon = 10^{-8}$ used to prevent division by zero. These parameters will be used for this project, as the focus is not perfecting network performance, but investigating the impact of bias-free versions of known networks.

2.5 Bias-Free Denoising

This project draws inspiration from the work in [1], wherein Bias-Free Convolutional Neural Networks (BF-CNNs) are explored for image denoising. The goal of the paper is to obtain a greater understanding of the mechanisms that allow CNNs to achieve state of the art image denoising performance. One downside of such denoising networks is their performance on noise levels outside those present in the training data.

To better understand the cause of this lack of generalisation outside of known noise levels, the first order residual of the denoising process is investigated over different ranges of training noise levels. This is found by calculating the magnitude of the bias term by applying a first-order decomposition of the error based on Equation (2.2), expressing it in the form;

$$\mathbf{x} - \hat{\mathbf{x}} = (\mathbf{I} - \mathbf{A}_x)\mathbf{x} - b_x \tag{2.29}$$

where $\mathbf{A}_x$ is a matrix containing the Jacobian of the network, and b_x the net bias, both of which are variant on $\mathbf{x}$, as indicated by the subscript.

As can clearly be seen in Figure 2.3, the bias grows explosively outside of the trained noise levels, while the residual (removed noise) drops as well. This drop in the residual is also seen in the performance of the network.

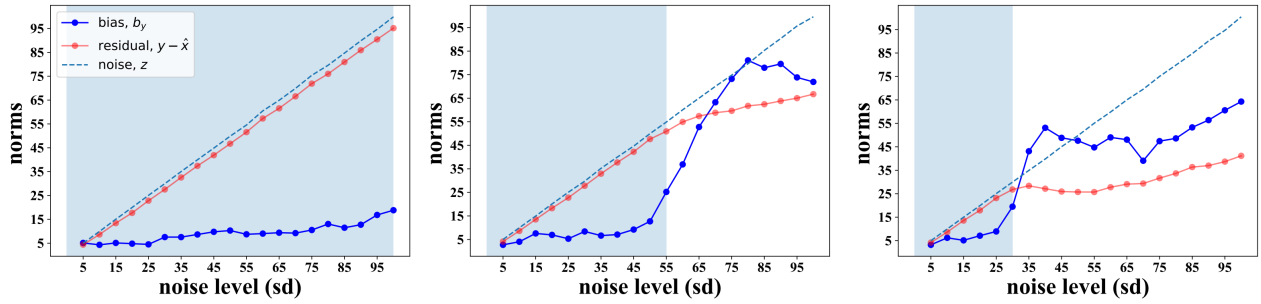


Figure 2.3: Graphs showing the average values for bias, residual and applied noise over noise levels (in standard deviations). The light blue background indicates the noise levels presented in the training data. [1].

Denoising generalisation across noise levels was found to improve dramatically by removing all additive terms from the neural networks. This method holds for a variety of network architectures, which further encourages the hypothesis that this will work for speech enhancement applications.

Chapter 3

State of the Art

In this chapter, a cross-section of various speech enhancement techniques are examined to determine their relevance and potential applicability in this project. Finally, using existing results from both speech and image enhancement, a new method is proposed to be more applicable under situations where the SNR is unknown.

It is also possible to classify speech enhancement systems into those that consider multiple microphones (channels), and those that use a single microphone. This project is primarily concerning single channel speech enhancement, and so that is the primary focus of this chapter. However, multi-channel methods are mentioned when relevant.

3.1 Non-Deep Learning Based Methods

Non-deep learning based speech enhancement algorithms can generally be classified into one of four categories; spectral subtractive, subspace, statistical model and Wiener-type [16][17].

The spectral subtraction method is considered to be the earliest method within the field, and relies on making a statistical estimate of the noise spectrum, before subtracting the estimated noise spectra from the speech and noise spectra. While spectral subtraction has been researched in the context of both the noise and signal's STFT coefficients following a known distribution [17], the assumption of known distributions does not hold in real-world applications.

Instead, a method based on adaptive filter theory is proposed for cochlear implant users in [18], which shows an improvement in hearing across all test subjects with speech shaped noise, but only a slight improvement with six speaker babble. The authors do note however that this could be attributed to cochlear implant users being more susceptible to babble noise however. The spectral subtraction method has also been seen to produce speech distortion [19], which will also be detrimental to the end user's experience.

Subspace methods seek to reduce the distortion seen in spectral subtraction methods by partitioning the noisy speech signal into two subspaces; one containing both speech and noise, and the other containing only noise. This decomposition into subspaces is commonly done by means of either an eigen- or singular value decomposition of the signal, however approaches based around the Karhunen-Loéve transform have also been seen [19]. This approach has the advantage of making no assumptions about the spectral characteristics of the noise, however subspace methods display comparatively poor performance in listening tests using noise sources experienced in daily life [16].

Various statistical speech enhancement methods exist, mainly based around the MMSE estimator. Of these, estimators that account for psychoacoustic characteristics in their design appear to perform best. This is seen in [20], where estimators that emphasise parts of the clean speech spectra with lower amplitude outperform those that emphasise high amplitude areas. One particular estimator presented appeared to perform well preserving weaker portions of speech (though inconsistent overall), however this was only seen through spectral analysis, and so there is no guarantee that this translates to better speech in practice. In listening tests, such as in [16], statistical estimators seemed to perform well, however they did not use the optimal parameters for the sample rate they selected. While speech sampled at 16 KHz was used in [16], the parameters used had been optimized for speech sampled at

8 KHz. This makes any comparison between non-deep learning based methods difficult, as it is now impossible to quantify the benefit in selecting a statistical estimator.

Wiener-based methods are noted as being the best-in-test when considering both good performance and computational complexity, although they did not perform as well as statistical or one spectral subtractive algorithm in terms of distortion . It is also mentioned that if the STFT coefficients of speech and noise are gaussian, then the wiener filter is optimal among all estimators, however non-linear MMSE estimators perform better in practice [17], as an assumption of gaussian coefficients does not hold in reality.

Generally, the field has progressed away from these methods, due to their assumptions of a segment-wise stationary signal and noise spectra, which is not seen in real-world applications.

Also worthy of mention are methods based around adaptive filtering, such as Least Mean Squares (LMS) and Recursive Least Squares (RLS). These have been explored with the same general objective as speech enhancement [4]. However, while speech enhancement is concerned specifically with making the spoken word easier to understand in a noisy environment, adaptive filtering is principally concerned with the noise, with relatively little regard to intelligibility.

3.2 Deep Learning Based Methods

Recently, deep learning techniques have been successfully applied to the field of speech enhancement, With studies showing a strong subjective preference towards deep learning based methods when compared to both spectral subtraction and statistical methods [4]. These are currently the general category of methods where state-of-the-art performance is observed.

in [21], a regression-based approach for speech enhancement is proposed, that decomposes a noisy speech signal into the frequency domain using a short-time Fourier transform in an overlapping manner, before enhancing it and reconstructing the enhanced signal using an inverse Fourier transform. This was trained on the TIMIT corpus [22] with 104 different noise types, with 100 being environmental in nature. This was found to improve the generalisation of the network, and also helped when encountering non-stationary noises. The Deep Neural Network (DNN) based method significantly outperformed a log-MMSE method in listening tests with both English and other languages.

These results are corroborated by [23], with the statistical method used for comparison being instead a short-time spectral amplitude MMSE. Additionally, knowledge of the type of noise is said to be more important to good performance in a DNN system.

Autoencoder-based methods can also be seen, such as [24] which uses an autoencoder as part of a noise estimation strategy before subtracting that noise with a wiener filter, and in [25] where they are deployed successfully in a multi-channel speech enhancement context, obtaining best results when single-channel and pre-enhanced speech features were concatenated and used as the input. More recently, [3] uses a deep denoising autoencoder alongside hard-coded features, outperforming spectral subtraction, statistical and wiener-based methods, although the framework is further towards noise reduction, distinguished here as being focused solely on the reduction of noise, than speech enhancement, distinguished as the active enhancement of the speech, including noise reduction.

Solutions involving recurrent neural networks have also been explored. Long short term memory (LSTM) based techniques have seen success in some speech enhancement applications such as in text-to-speech systems [26], though they appear to be constrained to more hardware-rich environments. [27] attempts to rectify this, proposing an equilibriated recurrent neural network (ERNN) that achieves

identical performance with fewer parameters, when compared to an LSTM-based design under similar circumstances.

Convolutional neural networks (CNNs), and in particular autoencoder-based CNNs are currently regarded as being state of the art in performance [7]. Several possible approaches exist, such as [28] where a frequency-domain loss function is defined and minimised to produce enhanced time-domain speech, similarly in [29] a convolutional autoencoder-based method that attempts to minimise the number of parameters (and by extension operations, make it more favorable for real-time applications).

No method seen to date has seen good performance on signal-to-noise ratios outside those which it has been trained on. This is a significant weakness when taking a network from a test environment to production, as it is impractical to train at all SNRs one would expect to see.

To mitigate against this, a bias-free convolutional neural network is proposed, based on the results of [1]. Here, removing the bias terms from the network was found to significantly increase generality over different signal to noise ratios. Although this is a method founded in image processing, the noisy speech signal may also be converted into an image, in the form of a spectrogram. This makes the results of [1] still potentially relevant in the context of speech enhancement.

Chapter 4

Speech Enhancement System

4.1 Overview

Figure 4.1 shows the structure of the proposed speech enhancement system, considering the findings of the State of the Art in Chapter 3. The following subsections will describe how each functionality is implemented, as well as possible alternatives and their potential implications on the speech enhancement system.

In this approach, the raw input $\mathbf{x}(t)$ is first converted into the frequency domain $\mathbf{X}(f)$ while still retaining temporal information, or the magnitude of each frequency observed at each timestep. The speech enhancement itself then takes place in the frequency domain by means of a neural network yielding $\hat{\mathbf{X}}(t)$, before conversion back to the time domain to output an enhanced version of the input signal $\hat{\mathbf{x}}(t)$

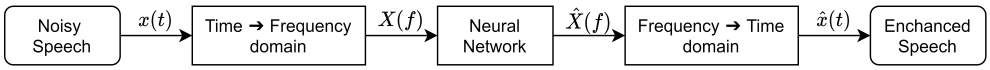


Figure 4.1: Structure of proposed speech enhancement system

4.2 Time-Frequency Domain Conversions

As can be seen from Figure 4.1, converting between the time and frequency domain is of critical importance to the speech enhancement system as a whole. Two possible transforms are considered for this conversion; the Fourier and Wavelet transforms.

4.2.1 Fourier Transform

The discrete Fourier transform takes the form

$$\mathbf{x}(f) = \mathbf{W}\mathbf{x}(t) \tag{4.1}$$

where x is our data in time (t) and frequency (f) domains, and $\mathbf{W}$ is a square symmetric matrix of weights defined as;

$$\mathbf{W} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \\ 1 & \omega^2 & \omega^4 & \omega^6 & \\ 1 & \omega^3 & \omega^6 & \omega^9 & \\ \vdots & & & & \ddots \\ 1 & & & & \omega^{(N-1)(N-1)} \end{bmatrix} \tag{4.2}$$

with ω in turn defined by

$$\omega(f) = e^{\frac{-2\pi i t}{N}} \tag{4.3}$$

where N is the number of samples in the time domain, which is equal to the number of bins in the frequency domain. This matrix vector product is naively quadratic as $O(N^2)$, however the fast Fourier

transform algorithm takes advantage of symmetries to reduce the complexity to $O(N\log(N))$. For the purposes of this report, these can be treated as identical, as they give the same end result [30].

Expressed in matrix form like this, it is easier to see how the STFT might be implemented. This would take the form of a FIFO buffer over the input signal, and applying the Fourier transform accordingly. This is effectively the same as taking the Fourier transform of a sliding window of data in time, and storing the results of each Fourier transform according to which timestep they correspond to. This may be expressed as;

$$\mathbf{X}(f,t) = \sum_{t=0}^{\infty} \mathbf{W}\mathbf{x}[t, t-N] \tag{4.4}$$

Where $\mathbf{x}[t, t-N]$ is a vector of length N containing the windowed input signal and $\mathbf{X}(f,t)$ is an $N \times l$ matrix containing the Fourier coefficients of $\mathbf{x}(t)$ for each unique timestep l over which the transform may be applied .

The Fourier transform also has a unique inverse, going from a frequency- to time-domain representation. This property is essential to reproducing the enhanced signal. This inverse is defined as;

$$\mathbf{x}(t) = \mathbf{W}'\mathbf{x}(f) \tag{4.5}$$

Where $\mathbf{W}'$ is defined similarly to Equation (4.2), however with $\omega(t)$ now defined as;

$$\omega(t) = e^{\frac{2\pi i t}{N}} \tag{4.6}$$

This inverse is readily extendable to the short-time Fourier transform, taking the form;

$$\mathbf{x}[t, t-N] = \mathbf{W}'\mathbf{X}(f,t) \tag{4.7}$$

This leads to ready conversions between time and frequency domain.

The STFT is also widely applied in contemporary deep learning-based speech enhancement systems, such as in [21], where a DNN-based framework for monaural speech enhancement using several hand-coded speech features is presented and tested on a variety of noise types, and more recently in [7], in which several objective functions are analysed with respect to their performance with a convolutional autoencoder.

One of the main issues with this technique is in relation to the lengths of the windows themselves. There is a lower limit on the frequency we can detect defined by the length of the window in time, but also a upper limit determined by the sample frequency. The fact that the length of the window is not adaptive, means that the STFT struggles to resolve sustained tones with a window that is too small. Alternatively, short duration and high frequency bursts are difficult to detect [31] with an excessively large window. This makes the selection of window size a trade-off between burst and sustained tone resolution. Eliminating this trade-off is the main motivation (and some of the intuition) behind the wavelet transform.

4.2.2 Wavelet Transform

A wavelet is best thought of as a single oscillation of a periodic function, i.e. one period of a sinusoid/square wave/etc. We define one possible wavelet as the *mother wavelet*, ψ, to form the basis of the wavelet transform. Note that the type of wavelet selected as the mother wavelet has a direct impact on the result.

It is possible to scale this mother wavelet in terms of both amplitude a and frequency b. This is the fundamental property behind the wavelet transform.

By sweeping over a variety of (perhaps arbitrarily spaced) amplitudes and frequencies, and convolving the result with the input signal, the wavelet transform is produced. This is defined mathematically as;

$$\mathbf{T}(a,b) = \frac{1}{\sqrt{a}} \int_{i=-\infty}^{\infty} \mathbf{x}(t)\psi(t) * \frac{t-b}{a} dt \tag{4.8}$$

This is the continuous wavelet transform, and takes every possible wavelet and scale combination. For reasons of practicality, a subset of wavelets is often selected (and indexed by m and n), thus discretizing the frequency and amplitude space. This then leads to the discrete wavelet transform [32];

$$\mathbf{T}_{m,n} = \int_{i=-\infty}^{\infty} \mathbf{x}(t)\psi_{m,n}(t) dt \tag{4.9}$$

The wavelet transform has been shown to produce a spectrogram, however there remain problems in time and frequency resolution, as resolution is not constant within the spectrogram [33].

The primary motivation behind the wavelet transform also means that it is especially beneficial and most used in cases involving high frequency, short time period oscillations within a broadband signal. An example of such a use case is the analysis of EKG and other biological signals [33].

There is however an existing body of literature surrounding the application of the wavelet transform to sound signals. in [34], a wavelet-based method is used to classify various sound sources, primarily distinguishing speech from musical origin, and a beat detection algorithm is presented.

In [35], a wavelet-based method for speech enhancement is presented, that takes the wavelet transform of the input signal, before applying a semisoft thresholding function to the resulting wavelet representation and applying the inverse wavelet transform. [36] appears to build on the results in [35], including pause detection and new thresholding methods, however it relies on the same fundamental method.

Historically, wavelet thresholding is the predominant wavelet-based speech enhancement method, with development and research into these methods being observed to at least 2017 (to authors knowledge) [37]. However, more contemporary methods exist, including [38], which uses a wavelet transform to split the signal into several subbands, then matrix factorization to extract and enhance the speech components from each sub-band.

4.2.3 Evaluation and Selection

With the growing adaptation of learned transforms with end-to-end speech enhancement systems (not covered here) replacing both Fourier and Wavelet transforms, neither approach will be able to be considered state of the art. The Fourier Transform is most readily understood and has a unique inverse, in comparison with the wavelet transform with additional complexity involved in choosing a

"good" set of wavelets, as well as a non-unique inverse if a continuous transform is selected [39], it is recommended that a Fourier-based approach be used. It is not expected that either approach will have a direct impact on the results, as they are only to be used to create a spectrogram.

4.3 Neural Network Structures

The neural networks trained in this project are based on the work in [1] and [40]. The first two, Denoising Convolutional Neural Network (DnCNN) and UNet, are two neural network structures based almost entirely on convolutional layers. These neural networks represent the state of the art within image denoising, which is closely related to audio enhancement with spectrogram preprocessing.

The third neural network, a convolutional autoencoder, represents the state of the art within audio enhancement. This network employs end-to-end audio processing, meaning no preprocessing is performed. In doing so, the neural network learns not only to enhance the speech, but a latent representation of the signal itself, given by some arbitrary transform.

The convolutional autoencoder has also been shown to be capable of speech enhancement going from a spectra to a sequence of samples, so the learned transform is only in one direction. However in this approach presented in [41], densely connected layers were also used during the reconstruction of the raw speech signal based on the output of the convolutional layers.

Additionally, convolutional autoencoders have shown to have good image denoising performance within the medical field, across a narrow range of noises and SNRs [13]. This indicates that convolutional autoencoders are a similarly valid approach in spectra-to-spectra speech enhancement. This comes from similar reasoning to that on the use of bias-free networks, with methods performing well on image enhancement being generally transferable to speech enhancement.

However, as the intention of this report is not to develop a new state-of-the-art, the network design will be somewhat arbitrary. An architecture similar to that presented in [13] is proposed, consisting of 3 convolutional layers, with 2 max pooling layers in between in the encoder to reach a latent representation consisting of 64 $\frac{d_x}{4} \times \frac{d_y}{4}$ feature maps. The decoder is then this in reverse to reach the same dimensionality as the original image, however with the use of up-sampling layers instead of max pooling.

As the kernel size is not described in [13] this will be set arbitrarily to be a 3×3 kernel. This is chosen to be consistent with both the DnCNN and UNet neural networks, however it is highly unlikely that this is optimal. In a similar manner, zero padding will also be used, while the stride of the kernel will be changed in order to produce the desired changes in feature map sizes.

- DnCNN: A feed-forward CNN with 20 layers. Each layer has 64 kernels of size 3×3, ReLU activation functions and batch normalization. Two different variants are included in this project, one without a skip connection as seen in Figure 4.2, and with a skip connection included as in [1] and seen in Figure 4.3.
- Convolutional Autoencoder: An autoencoder comprised of a convolutional layer with 128 filters with a stride of 1 at the input, with 2 further convolutional layers of 128 filters with a stride of 2 forming the encoder, giving a latent feature map of $\frac{d_{x,y}}{4}$. The decoder then consists of 2 transpose convolutional layers also with a stride of 2, before a convolutional layer of 128 filters of size 1 forms the final layer of the model. This can be seen in Figure 4.4

- UNet: A UNet-like network topology with a broadly similar structure to that of the convolutional autoencoder, with feed-forward connections across latent stages. This can be seen in Figure 4.5

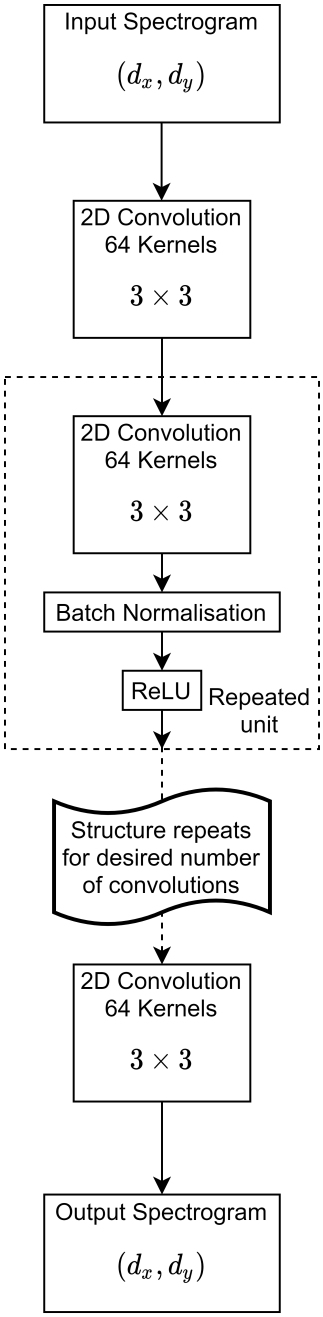


Figure 4.2: The DnCNN without skip connections.

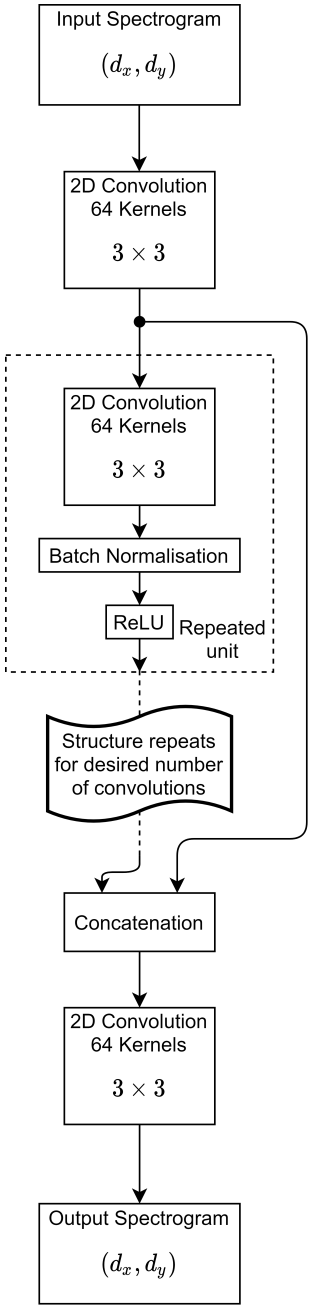


Figure 4.3: The DnCNN with skip connections.

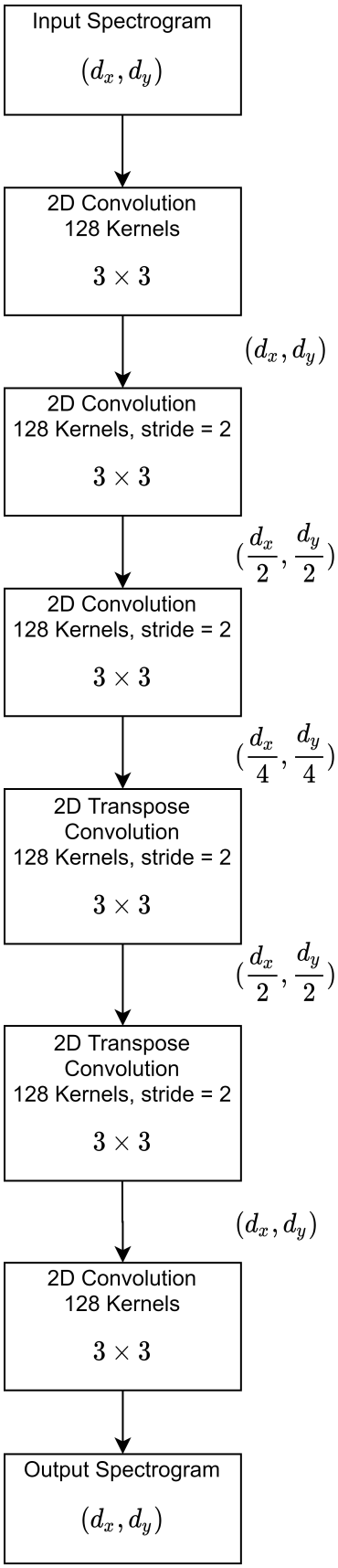


Figure 4.4: The convolutional autoencoder.

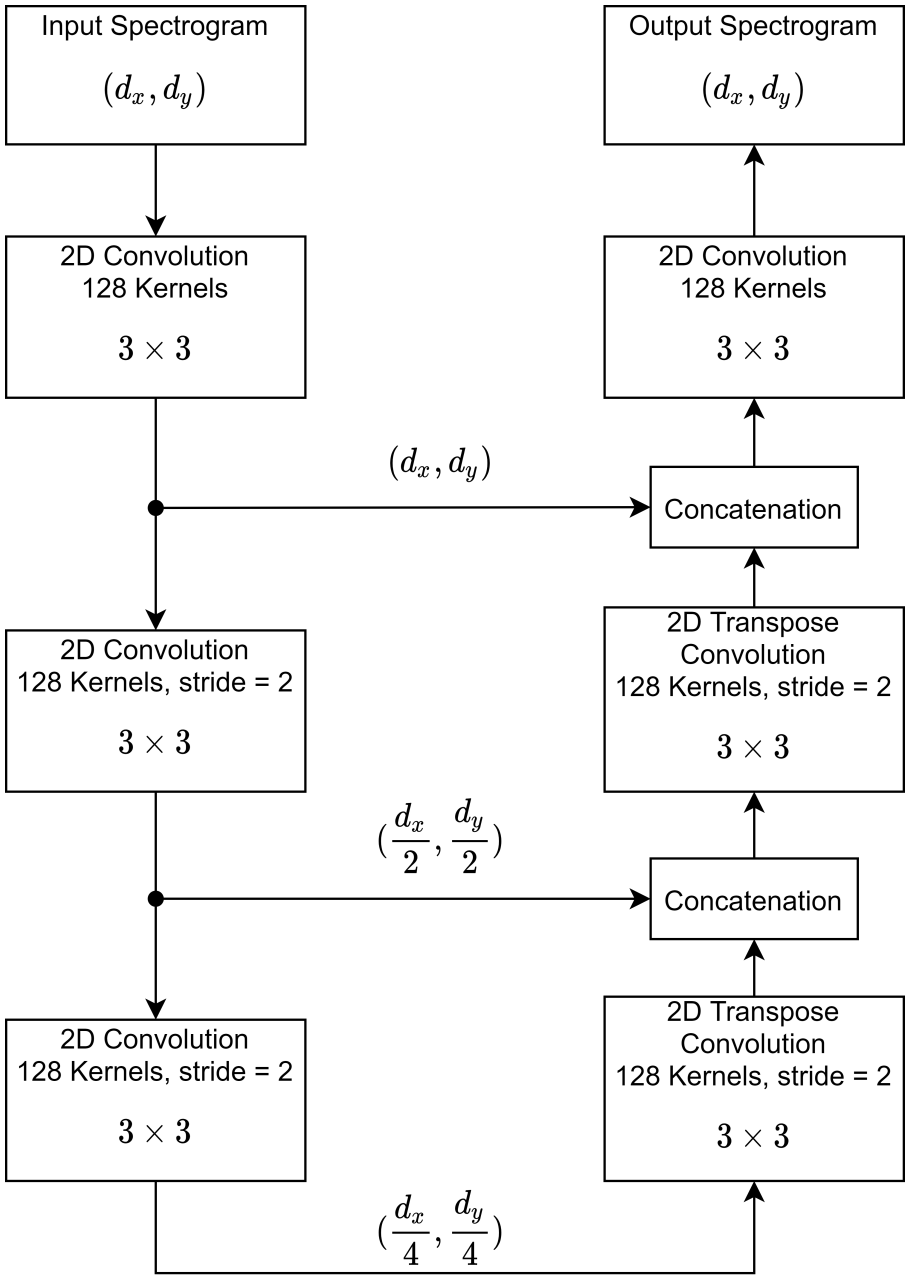


Figure 4.5: The UNet.

Chapter 5

Methodology

In this chapter, the method for noise generation with an arbitrary SNR is explained, as well as the training given to each network. Finally, several metrics are introduced to aid in the evaluation of the hypothesis.

5.1 Noise Generation

Training a network for speech enhancement necessitates a noisy dataset with a clean counterpart for training and evaluation. As the datasets used in this project contain only clean audio, the noise must be added before training. The two types of noise, as well as how they are scaled to different noise levels and applied to the datasets are presented in this section.

5.1.1 Additive White Gaussian Noise

Additive White Gaussian Noise (AWGN) is a simple form of additive noise. It is spectrally white, meaning equal intensity at different frequencies. As AWGN is additive, the noise is also independent of the signal it is applied to. While AWGN is not the most complex type of noise, it should yield a proof of concept for the generalisation of bias-free speech enhancement.

The AWGN is applied by drawing values from a zero mean normal distribution. The variance does not matter, as the noise is scaled relative to the signal power (see section 5.1.3).

5.1.2 Babble Noise

Babble noise aims to emulate the cocktail party effect, whereby a conversation is being held in a room with multiple speakers around the subject. This is a particularly difficult environment for people with hearing impairments [2]. Babble noise is more complex that AWGN, especially due to the non-stationarity of it. The increased complexity in conjunction with the emulation of a well-known speech enhancement problem makes it ideal for further proving the robustness of the trained models. Once it has been proven that the models can denoise speech with AWGN applied, the same will be done with babble noise, to further prove the effectiveness of the solution.

The babble noise is generated by creating roughly equally sized vectors of shuffled sentences from either the TIMIT or WSJ datasets. The amount of vectors is determined by the desired amount of *concurrent* speakers. Note that there is no guarantee that all speakers are audible at a given time instance. Once the vectors are generated, they are summed and scaled according to a desired SNR, as described in Section 5.1.3. Applying the babble noise is a matter of randomly indexing a section of the babble noise vector and adding this to the audio segment.

5.1.3 Noise Scaling

Since the purpose of this project is to investigate the generalisation benefits of bias-free networks across multiple noise levels, it is desired to scale the noise in a controlled manner. Scaling the noise according to a specific Signal to Noise Ratio (SNR), is a common method for this.

This is done by first computing the power of the signal onto which noise is to be added. The power p of a signal, s, is found by [42]:

$$p_s = \sum_{n=0}^{N-1} \frac{s_(n)^2}{N} \tag{5.1}$$

where N is the length of the vector s. The same can be done to obtain the power of the noise vector, which is then scaled according to the desired ratio, e.g. 40 dB. This is done by finding a value, α, to scale the noise with, according to:

$$\alpha = \sqrt{\frac{p_s}{10^{\frac{\text{SNR}}{10}} \cdot p_n}} \tag{5.2}$$

where p_s and p_n is the computed powers of the signal/noise vectors and SNR is the desired SNR in dB. The noise is then multiplied by α and added to the signal vector.

5.2 Dataset Overview

5.2.1 TIMIT

The Texas Instruments/Massachusetts Institute of Technology (TIMIT) Acoustic-Phonetic Continuous Speech Corpus is a speech Corpus developed for the purpose of acoustic-phonetic studies and speech recognition development [43]. The Corpus contains audio from 630 speakers from across the United States, each reading ten sentences out loud. This results in 6300 samples of audio of varying lengths.

The TIMIT dataset is the primary dataset used during the project. The dataset contains a decently large number of audio samples to ensure that the preprocessing and models work as intended. The TIMIT dataset was chosen due to being easier to work with, as well as to reduce memory use during initial development on local desktop machines. In addition to being smaller, a library for loading and converting the dataset to audio in an easy manner exists, which greatly reduced development time in this area.

5.2.2 WSJ

The Continuous Speech Recognition (CSR) Corpora, herein referred to as Wall Street Journal (WSJ) dataset, is the second dataset available for training and testing [44]. The dataset consists of a large number of audio samples recorded using different microphones. The WSJ is comprised of audio read from excerpts of the Wall Street Journal news texts, hence the name. The dataset is developed primarily for speech recognition, so it shares a similar structure to the TIMIT dataset. No loader

could be found for the dataset, so one had to be written from scratch. The dataset is compressed, meaning it has to be decompressed before use.

The WSJ dataset is much larger than TIMIT, however, this means that the whole dataset will not fit into local desktop memory. This means that only a part of the dataset can be loaded, which results in a similar dataset size as TIMIT. In addition to this, the dataset is compressed, meaning it has to be converted to .wav files before being loaded. In general, the benefit of the WSJ dataset is the size of it, however, as this will not fit into memory, it will require additional overhead during training.

Once the entire preprocessing and training pipeline has been verified, as well as a proof of concept for bias-free speech enhancement, a network can be trained using both datasets. Since the goal of this project is not to develop a state of the art speech enhancement model, this is a low priority.

5.3 Preprocessing

Since the purpose of this project is to investigate bias-free image-based deep learning for speech enhancement, the audio datasets are converted to spectrograms, which are essentially images. Converting the TIMIT and WSJ datasets to spectrograms requires some preprocessing as well as some restructuring.

5.3.1 Audio Preprocessing

As the TIMIT and WSJ datasets are developed and structured for speech recognition systems, the structure is defined by speaker and phonetic content [43]. This is invaluable for speech recognition systems, however, for the purposes of speech enhancement this is not required.

The audio is preprocessed by first normalising each utterance in the dataset to zero mean and unit variance concatenating all utterances into one audio vector. This vector is then split into equally sized audio samples, discarding any leftover audio. Splitting the audio this way guarantees the generated spectrograms to be of equal size, which greatly improves memory usage and eases the training process, even for networks which allow different sized inputs.

5.3.2 Spectrogram Generation

As stated in Section 4.2 the Fourier transform is used to convert from time to frequency domain. This conversion results in a matrix of complex weights, describing the amplitude and phase of each frequency component at a given time. These matrices are also referred to as spectrograms. Magnitude spectrograms are often used in audio processing, particularly log-magnitude spectrograms. The relationship between the magnitude of a spectrogram and the output of the STFT is:

$$S_{f,t} = |S_{f,t}| \cdot e^{(j\theta_{f,t})} \tag{5.3}$$

where $|S_{f,t}|$ is the magnitude of the spectrogram weights and $\theta_{f,t}$ is the phase angle, both indexed by time t and frequency bin f. The log-magnitude spectrograms are computed as;

$$A_{f,t} = \ln |S_{f,t}| \tag{5.4}$$

and also normalised to zero mean, unit variance for use in training and testing. Converting the log-magnitude spectrogram back to a regular spectrogram is done by;

$$S_{f,t} = \exp(A_{f,t}) \cdot e^{(j\theta_{f,t})} \tag{5.5}$$

where θ is the phase angle of the original signal at time t and frequency bin f.

For this project, the spectrograms are treated as images, meaning the time and frequency axes are treated as spatial axes instead. This approach has been used extensively in deep learning applications and despite the state of the art shifting towards end-to-end models, this method is still a viable solution. As the point of the project is to investigate the generalisation benefits of bias-free deep learning, the efficacy of the model is not as relevant. Additionally, the origin of this project [1] is within the field of image denoising. As such, it is theorised that the same methods will apply to spectrogram denoising/enhancement.

5.3.3 Postprocessing

To reconstruct the audio from the enhanced spectrograms requires taking the inverse STFT. The inverse STFT requires both magnitude and phase information to properly reconstruct the audio without audible artifacts. The phase information of the original spectrogram is saved before enhancement and is reapplied before taking the inverse STFT. Once the audio has been reconstructed, it is scaled to an appropriate level for listening to avoid clipping using peak normalisation.

5.4 Training

During training, the network are fed batches of noisy spectrograms with the target being the clean counterpart to the input spectrogram. This trains the network to reconstruct the clean spectrogram from a noisy input, thus enhancing the spectrogram and therefore the audio.

When the loss of the network begins stagnating, the learning rate will be reduced, which will allow the training algorithm to more easily descend potential minima. This also allows for coarser training in the early stages, which can help speed up training process.

For each network, two copies are trained on identical data, where one has the bias terms removed entirely. This should reveal the impact of removing the bias.

5.4.1 Reducing Over-fitting

When training a neural network, over-fitting is a common problem wherein the network will learn the training data "too well" and will not generalise well to input not present in the training dataset [45, 46]. This is due to the network not learning the characteristics of the problem it is designed to solve. Over-fitting leads to poor performance outside of training, but it can be alleviated by implementing features such as early stopping or using a separate validation dataset.

One way to reduce over-fitting when training a neural network is to stop training when the error or cost stops improving over successive epochs. This is not guaranteed to reduce over-fitting, as the validation data could be too similar to the training data. An additional benefit of early stopping is

that the training stops, meaning faster model prototyping, since less time is spent doing useless or potentially harmful training.

Utilising a validation dataset helps ensure that the network learns the broader aspects of the data, as opposed to just learning the data "by heart". The validation data is often a part of the training data, which is not used in updating the weights of the network, but is instead used to validate the cost of the current epoch. If the network has a low training cost, but a high validation cost, the network is not generalising well to the data.

The training done in this project is always utilising a validation dataset, which is derived from the training data unless otherwise specified. Methods such as learning-rate reduction and early stopping are based on the validation loss, meaning the training algorithm will only employ these methods if the validation loss is stagnating.

5.4.2 Multi-SNR Training

To be able to compare to the results from [1], the neural networks for speech enhancement are trained in a similar fashion to those for image denoising. In [1], the neural networks are trained across a range of standard deviations (SNRs) and are subsequently tested on noise levels in- and outside the training noise range.

To do this, the training data is essentially generated multiple times, each with a different SNR. The training data is then shuffled and fed in batches to the neural network for training. It is important that the network sees all generated SNRs in ONE epoch, so as to not train asymmetrically towards one specific SNR.

5.5 Metrics

While subjective listening tests are the preferred method of evaluation for speech enhancement systems in general, they are both expensive and time consuming to undertake. Due to the time limitations involved in this project, only synthetic metrics are selected for use.

Speech enhancement problems require metrics that also account for how the enhanced speech is perceived. This is due to the fact that the human ear does not perceive all frequencies equally, indeed higher frequencies are often indistinguishable from another even when the difference is 1 KHz or greater. This necessitates the need for specifically developed metrics that takes into account these more human characteristics.

This is in contrast from, mean squared error and other non-perceptually oriented metrics, where the mean distance between an enhanced speech signal and a clean reference is taken in either time or frequency domain.

Here, the Perceptual Evaluation of Speech Quality (PESQ) [47] and Short-Time Objective Intelligibility (STOI) [48] shall be used. The main focus of PESQ is the quality of each utterance, and how perturbed it is by noise, while STOI is primarily concerned with whether the utterances will be understood.

The PESQ method attempts to substitute a listening test and as such returns a score on the interval $[1, 5]$, with a 5 corresponding to no degradation, and a 1 corresponding to degradation significant enough to make the meaning of the speech intangible. Alternative mappings exist acknowledging that

participants in listening tests are generally more hesitant to give a "5" [49], however the principal that a higher score correlates to better quality still holds. The scores generated by PESQ have been shown in [47] to have a strong correlation with scores from listening tests under laboratory conditions.

PESQ has both a narrowband and wideband [50] configuration. The narrowband metric was originally designed for a bandwidth of 300-3100 Hz, while the wideband metric takes a bandwidth of 50-7000 Hz. With the fundamental vocal range for males at 60-150 Hz, and for females 200-400 Hz [2, p. 49], the PESQ metric in a narrowband configuration will likely provide unfavorable results due to the bandwidth limitations imposed cutting off the fundamental speech components for all male speech, and a significant proportion of female speech. To aid in comparison, both wideband and narrowband PESQ metrics have been calculated, however the wideband is to take precedence in evaluation.

In contrast, STOI considers the linear correlation coefficient between the speech in the frequency domain, with a short-time Fourier transform being used to segment the signal into time-frequency regions with a length of 400 ms [48]. As this is a correlation coefficient, the STOI score is inherently bounded on $[0, 1]$, with 0 representing no correlation and total degredation, and 1 representing perfect correlation between reference and enhanced speech.

An extended version of STOI (ESTOI) is also in common use, which uses many of the same steps as STOI. Key differences include normalisation of the time-frequency regions before any comparison, and the use of subspace methods instead of correlation coefficients [51]. Similarly to STOI however, scores are also bounded on $[0, 1]$ with the same meaning attached.

Chapter 6

Results

This chapter describes the tests performed to investigate the impact of removing the bias from state of the art neural networks. The results of these tests are presented and analysed to show the efficacy of the networks within the scores described in Section 5.5. Key results are presented in this chapter while a full overview of all results and plots can be found in Appendix A.

6.1 Test Overview

The following results were obtained by training the networks described in Section 4.3 on sets of training data with varying levels of noise. The full range training set covers the SNR values in the interval of -10 dB to 20 dB in steps of 5 dB, where the reduced range set covers SNR values in the interval of 0 dB to 15 dB in steps of 5 dB. All networks are then evaluated on the test set, which has SNR values identical to the full range training set. All results are displayed as bar plots showing the mean score values for each SNR value, with one standard deviation as the error bars.

Normally, such training would be performed multiple times to ensure statistical significance. However, due to having issues with getting the training running on the provided server (CLAAUDIA), this was not possible within the project time-frame. The results produced represent one or at most two realisations of the networks.

In addition to evaluating the networks according to the metrics in Section 5.5, the most promising networks are further investigated by listening to the enhanced audio produced by these networks. This is not a structured listening test, as it will be performed by the authors, however, it will ensure that the score is not just mathematically sound but also produces reasonable audio.

To supplement this listening test, spectrograms of the enhanced speech have been produced, and can be found in Appendix B. To limit the size of this report, only noise at one SNR has been considered when producing these spectrograms, however where relevant additional spectrograms are included within the body of the results.

6.2 DnCNN Results

Two realisations of the DnCNN network have been tested, with the difference between the two being the inclusion of a skip connection between the first and final layer. In the normal DnCNN network, this skip connection is included, however, due to not including this skip connection in the first network realisation, both are available for testing. While not exactly identical to the regular DnCNN, the variant without the skip connection will still be evaluated.

6.2.1 DnCNN without Skip Connections

The DnCNN without skip connection performed equally or worse than the DnCNN with skip connections, making most of the results from this test redundant. The network did not generalise better across noise levels outside the training range with a bias-free configuration. This, in conjunction

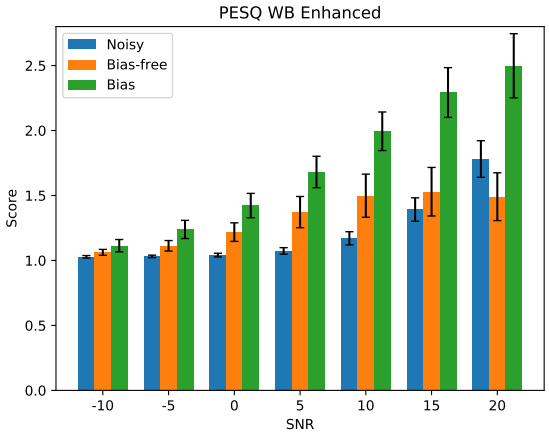


Figure 6.1: Bar plot showing the mean PESQ Wideband score for DnCNN with skip trained on the full SNR range.

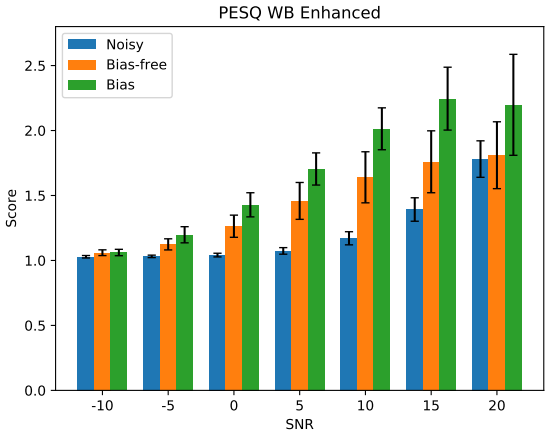


Figure 6.2: Bar plot showing the mean PESQ Wideband score for DnCNN with skip trained on the reduced SNR range.

with the overall poorer performance, both according to PESQ and STOI metrics, means that the network will not be analysed in more detail. For an overview of the performance of this network, see Appendix A.1.

6.2.2 DnCNN with Skip Connections

The DnCNN with skip connection performed the best of all tested networks in terms of overall denoising performance. The performance was highest for the regular configuration, meaning the bias-free configuration performed significantly worse across the all test metrics. This means that the bias-free configuration did not improve on the generalisation on noise levels outside the training range.

Figure 6.1 and 6.2 show the narrowband PESQ score for the DnCNN with skip connections. While this is the highest score achieved in this metric, it is clear that the bias-free configurations perform worse across noise levels, both for the full and reduced training set. Similar behaviour is seen in the STOI metrics. The DnCNN with skip connections performance on metrics is mirrored in audio enhancement. The network provides the clearest sounding denoising and introduces the least amount of distortion at lower noise levels. All results from the test of this network can be found in Appendix A.2.

6.3 Convolutional Autoencoder Results

Similarly to the DnCNN, bias-free convolutional autoencoders also underperformed relative to their biased counterparts. This holds for networks trained both on a reduced and a full range of SNRs. The complete results across all four metrics and the two SNR ranges used for training can be seen in Appendix A.4.

At SNRs of 15 dB and above, the distortion introduced by the network appeared to drive the metrics lower than the noisy signal in all cases. This distortion is also perceivable when listening to the samples, however it is clear that the noise has been attenuated. When listening to both the biased and bias-free network's output, the biased network is also perceivable sharper in tonality than the bias-free network, which seems to add a muddiness to the speech. This muffles the output somewhat, which is an undesirable characteristic. This deviation in quality is barely observable with an input

SNR of 15 dB, however it becomes more apparent at an SNR of 20 dB.

This change in clarity, as well as the effects of the latent space representation are also clearly observable in the spectrograms produced by both the biased (Figure 6.4) and bias-free (Figure 6.3) networks, here shown with the clean (Figure 6.5) and noisy (Figure 6.6) spectrograms besides for reference.

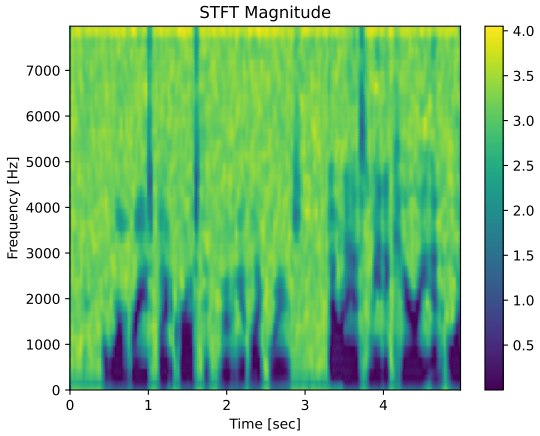


Figure 6.3: Log-magnitude spectrogram of the bias-free convolutional autoencoder's output with an SNR of 20 dB

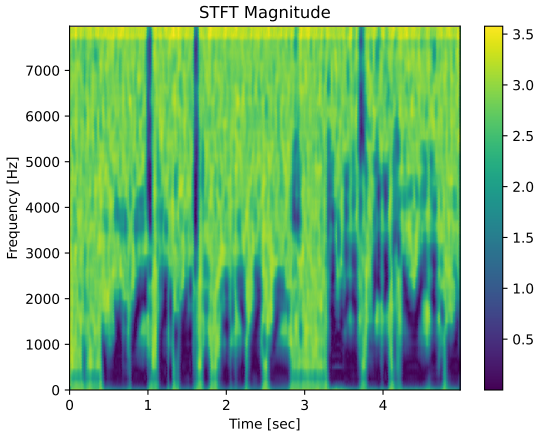


Figure 6.4: Log-magnitude spectrogram of the biased convolutional autoencoder's output with an SNR of 20 dB

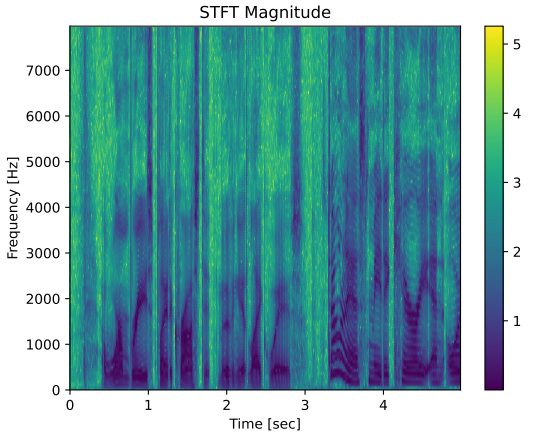


Figure 6.5: Log-magnitude spectrogram of the clean frame of speech

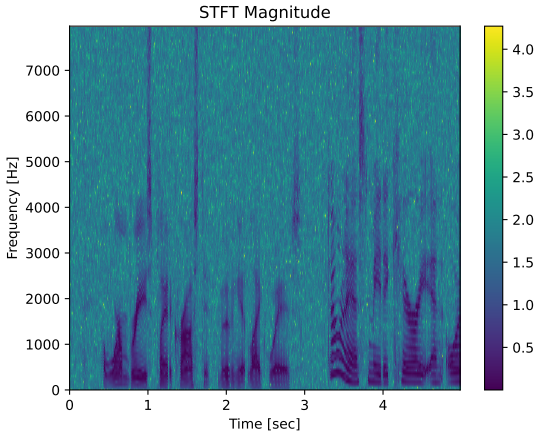


Figure 6.6: Log-magnitude spectrogram of the frame of speech with AWGN added at an SNR of 20 dB

6.4 UNet Results

The results from the UNet tests show some interesting distinctions between the regular and the bias-free network configurations. While all other networks tested showed a similar or worse performance on bias-free configurations, the bias-free UNet shows a clear improvement over the regular configuration. This improvement is however only statistically significant inside the training set noise values. For any noise values outside the training set, the benefit from the bias-free configuration is diminished to within the margin of error, especially given the limited network realisations. The STOI and ESTOI metrics (Figure 6.9 and 6.10) however only show minor benefits to the bias-free configuration, particularly in the narrow case. This behaviour can be explained by the network potentially not being able

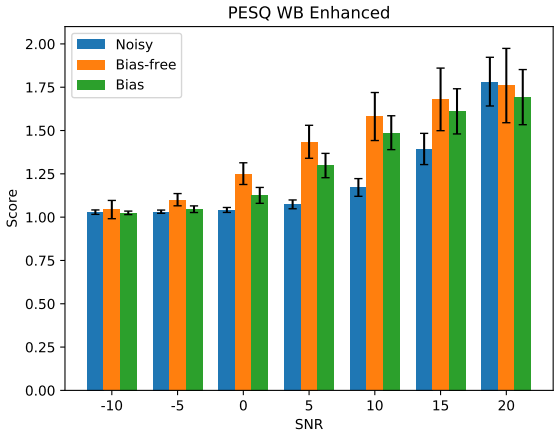


Figure 6.7: Bar plot showing the mean PESQ Narrow-band score for UNet trained on the full SNR range.

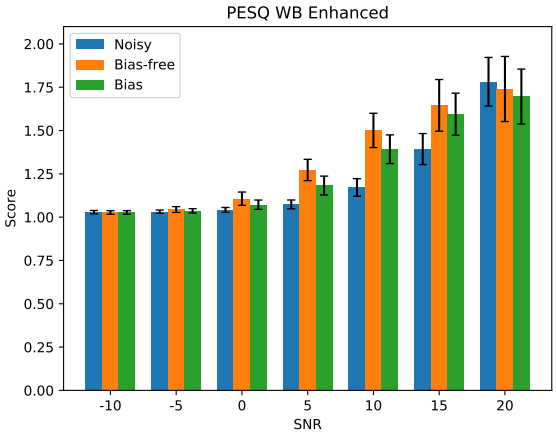


Figure 6.8: Bar plot showing the mean PESQ Narrow-band score for UNet trained on the reduced SNR range.

to capture the larger training set with the current amount of parameters. This remains untested and is therefore purely speculation. The UNet introduces less distortion at lower SNRs and does not introduce any appreciable muddiness to the bias-free enhanced audio. This can also be observed within the spectrograms of the enhanced speech both with and without bias in Figure B.5 and Figure B.4.

An example of these results can be seen in Figure 6.7 and 6.8. The full results can be found in Appendix A.3, which show similar relationships between bias-free and regular configurations.

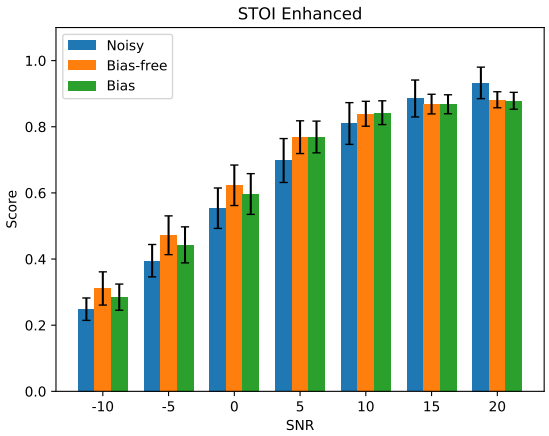


Figure 6.9: Bar plot showing the mean STOI score for UNet trained on the full SNR range.

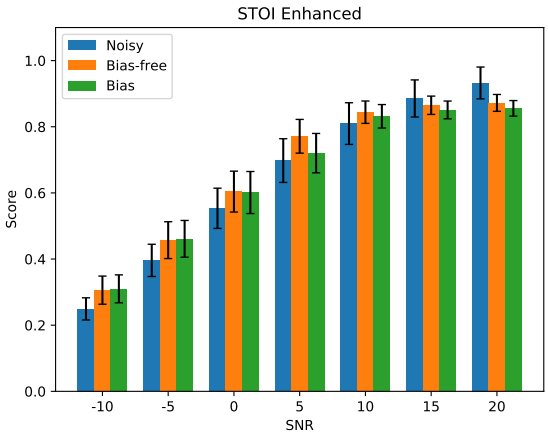


Figure 6.10: Bar plot showing the mean STOI score for UNet trained on the reduced SNR range.

Additionally, inspection of the spectrograms shows that there is significantly less "blockiness" to them when compared to the convolutional autoencoder. This indicates that the inclusion of skip connections between lower dimension feature maps aids performance significantly, irrespecttive of whether bias terms are included. This can be seen in Figure 6.11 and Figure 6.12, where some harmonics can be faintly observed in the spectrograms.

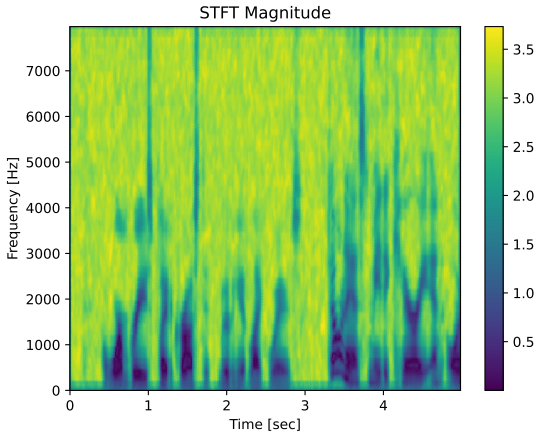


Figure 6.11: Spectrogram of speech enhanced using the UNet, trained on the limited SNR range. SNR = 20 dB

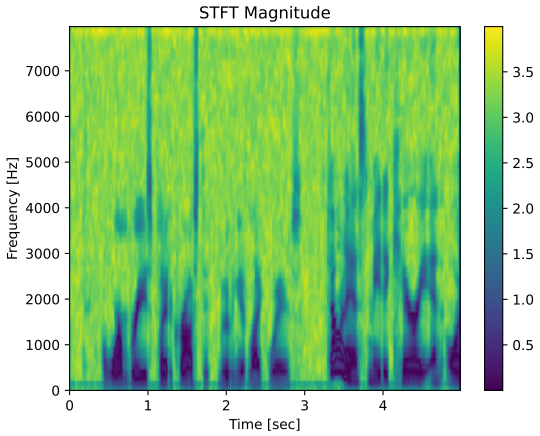


Figure 6.12: Spectrogram of speech enhanced using the bias-free UNet, trained on the limited SNR range. SNR = 20 dB

From these results, it can be concluded that the bias-free configuration does not improve the generalisation of the network across noise levels outside the training range. The improved performance of UNet with a bias-free configuration is however an interesting result, which can be further explored to potentially improve the model for full noise range training.

Chapter 7

Discussion

The results of this project show that only UNET has any appreciable gain from the selection of a bias-free variant. There are however several further details that could be investigated, as well as methods to make the experiments performed so far more powerful in terms of statistical significance. These will be detailed in this chapter.

Increasing the power of the experiments in this instance requires training multiple networks of the same topology, then considering the mean metrics and the variance between the networks, and not the variance on the test set as is currently performed. This would mitigate against random initializations of the network parameters having a direct influence on the results. As things stand, only 1 network of each type has been trained, meaning that there is the potential for a "poorly" initialized network to converge to a loss that would be considered sub-optimal, given any other parameter initialization. However, as things stand with most bias-free networks performing worse than their biased counterparts, this is sufficient to give a provisional conclusion.

The experiments thusfar have also all taken part on the TIMIT corpus. The Wall Street Journal corpus has also been available, and would provide another datapoint for any stronger conclusion. Further, visual inspection of the sentences within the TIMIT corpus shows that at least some of the sentences within the testing set also appear in the training set. As such, it is possible (though unlikely) for the network to learn to reconstruct these sentences, rather than generalise across all speech. A cross-validation using a similarly pre-processed alternative corpus such as Wall Street Journal would provide some insights in that regard.

Also of note is that training and testing has only used additive white gaussian noise so far. In real-world situations, speech enhancement methods must also be able to work on many sources of noise, with most being non-stationary. Although it is possible to make the assumption that the noise is stationary within each utterance (in this case, assumed to be a 5 second frame), this still leaves many unknowns in the spectral distribution of the noise. The noise may also take the form of other speech, such as in the cocktail party problem, further complicating the enhancement process. This training on multiple noise types was found in [21] to further improve generalisation capabilites, and so should be considered for implementation in any state-of-the-art solution.

AWGN was used for it's similarity to the noise used in [1], however while they use variance when determining the noise magnitude, this project uses the SNR. This makes direct comparison between the noise levels impossible without access to the same image database. Despite this, training on the range of SNRs used in this project is an accepted standard within speech enhancement [2].

A first-order decomposition of the network was also proposed in Section 2.5 to further understand the impact of the bias. This however involved taking the Jacobian of the network evaluated for each spectrogram $\mathbf{x}$, which was found to be very computationally intensive at the network sizes used in this report. This was therefore out of reach at the time of writing.

Chapter 8

Conclusion

In this project, four different types of convolutional neural networks were trained and evaluated with the objective of testing the hypothesis: *Removing the additive bias terms from neural networks improves network generalisation across unknown noise levels for speech enhancement purposes.*

Three of the network types, namely the convolutional autoencoder and Denoising Convolutional Neural Network (DnCNN), both with and without a skip connections, showed that bias-free configurations were inferior to configurations where bias terms were included. In these biased configurations, the DnCNN with a skip connection achieved the best overall performance.

The fourth network type, UNet, showed that a bias-free configuration performed significantly better inside known Signal to Noise Ratio (SNR) ranges and marginally better outside known SNR ranges. However, distortion dynamics introduced by the network impairs performance at higher SNRs. Additionally, no enhanced generalisation capabilities were observed in any bias-free networks.

With only one realisation of each network being trained and tested, it is not possible to conclude on the hypothesis in any statistically significant manner. The results at this time however indicate that the hypothesis is false.

Bibliography

[1] S. Mohan, Z. Kadkhodaie, E. P. Simoncelli, and C. Fernandez-Granda, "Robust and interpretable blind image denoising via bias-free convolutional neural networks," *arXiv preprint arXiv:1906.05478*, 2019.

[2] P. C. Loizou, *Speech enhancement: theory and practice*. CRC press, 2013.

[3] Y.-H. Lai, F. Chen, S.-S. Wang, X. Lu, Y. Tsao, and C.-H. Lee, "A deep denoising autoencoder approach to improving the intelligibility of vocoded speech in cochlear implant simulation," *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 7, pp. 1568–1578, 2016.

[4] T. M. Taha, A. Adeel, and A. Hussain, "A survey on techniques for enhancing speech," *International Journal of Computer Applications*, vol. 179, no. 17, pp. 1–14, 2018.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[6] P. JAUMIER, "Backpropagation in a convolutional layer," Jul 2019. [Online]. Available: https://towardsdatascience.com/backpropagation-in-a-convolutional-layer-24c8d64d8509

[7] M. Kolbæk, Z.-H. Tan, S. H. Jensen, and J. Jensen, "On loss functions for supervised monaural time-domain speech enhancement," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 825–838, 2020.

[8] J. Cong and B. Xiao, "Minimizing computation in convolutional neural networks," in *International conference on artificial neural networks*. Springer, 2014, pp. 281–290.

[9] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*. Ieee, 2017, pp. 1–6.

[10] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE transactions on image processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[11] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[12] F.-F. Li, R. Krishna, and D. xu, Apr 2020. [Online]. Available: https://cs231n.github.io/convolutional-networks/#pool

[13] L. Gondara, "Medical image denoising using convolutional denoising autoencoders," in *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2016, pp. 241–246.

[14] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[16] Y. Hu and P. C. Loizou, "Subjective comparison and evaluation of speech enhancement algorithms," *Speech communication*, vol. 49, no. 7-8, pp. 588–601, 2007.

[17] R. C. Hendriks, T. Gerkmann, and J. Jensen, "Dft-domain based single-microphone noise reduction for speech enhancement: A survey of the state of the art," *Synthesis Lectures on Speech and Audio Processing*, vol. 9, no. 1, pp. 1–80, 2013.

[18] L.-P. Yang and Q.-J. Fu, "Spectral subtraction-based speech enhancement for cochlear implant patients in background noise," *The Journal of the Acoustical Society of America*, vol. 117, no. 3, pp. 1001–1004, 2005.

[19] Y. Hu and P. C. Loizou, "A generalized subspace approach for enhancing speech corrupted by colored noise," *IEEE transactions on speech and audio processing*, vol. 11, no. 4, pp. 334–341, 2003.

[20] P. C. Loizou, "Speech enhancement based on perceptually motivated bayesian estimators of the magnitude spectrum," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 5, pp. 857–869, 2005.

[21] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, "A regression approach to speech enhancement based on deep neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 1, pp. 7–19, 2014.

[22] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1," *NASA STI/Recon technical report n*, vol. 93, p. 27403, 1993.

[23] M. Kolbæk, Z.-H. Tan, and J. Jensen, "Speech intelligibility potential of general and specialized deep neural network based speech enhancement systems," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 1, pp. 153–167, 2016.

[24] B. Xia and C. Bao, "Speech enhancement with weighted denoising auto-encoder." in *Interspeech*, 2013, pp. 3444–3448.

[25] S. Araki, T. Hayashi, M. Delcroix, M. Fujimoto, K. Takeda, and T. Nakatani, "Exploring multi-channel features for denoising-autoencoder-based speech enhancement," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 116–120.

[26] C. Valentini-Botinhao, X. Wang, S. Takaki, and J. Yamagishi, "Investigating rnn-based speech enhancement methods for noise-robust text-to-speech." in *SSW*, 2016, pp. 146–152.

[27] D. Takeuchi, K. Yatabe, Y. Koizumi, Y. Oikawa, and N. Harada, "Real-time speech enhancement using equilibriated rnn," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 851–855.

[28] A. Pandey and D. Wang, "A new framework for supervised speech enhancement in the time domain." in *Interspeech*, 2018, pp. 1136–1140.

[29] S. R. Park and J. Lee, "A fully convolutional neural network for speech enhancement," *arXiv preprint arXiv:1609.07132*, 2016.

[30] P. Duhamel and M. Vetterli, "Fast fourier transforms: a tutorial review and a state of the art," *Signal processing*, vol. 19, no. 4, pp. 259–299, 1990.

[31] A. Boggess, *A first course in wavelets with fourier analysis.*, 2nd ed. Hoboken, N.J: John Wiley Sons, 2009.

[32] S. Talebi, "The wavelet transform," Jan 2021. [Online]. Available: https://towardsdatascience.com/the-wavelet-transform-e9cfa85d7b34

[33] E. N. Kamavukao, "Wavelet analysis summary and applicatons," Feb (accessed) 2011 (accessed 2021). [Online]. Available: https%3A%2F%2Fperson.hst.aau.dk%2Fenk%2FST8%2FLecture7_Applications_of_Wavelet.pdf&usg=AOvVaw3yACUUJdz_uStltnPYbsld

[34] G. Tzanetakis, G. Essl, and P. Cook, "Audio analysis using the discrete wavelet transform," in *Proc. Conf. in Acoustics and Music Theory Applications*, vol. 66. Citeseer, 2001.

[35] J. W. Seok and K. S. Bae, "Speech enhancement with reduction of noise components in the wavelet domain," in *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2. IEEE, 1997, pp. 1323–1326.

[36] H. Sheikhzadeh and H. R. Abutalebi, "An improved wavelet-based speech enhancement system," in *Seventh European conference on speech communication and technology*, 2001.

[37] A. Bhowmick and M. Chandra, "Speech enhancement using voiced speech probability based wavelet decomposition," *Computers & Electrical Engineering*, vol. 62, pp. 706–718, 2017.

[38] S. Wang, A. Chern, Y. Tsao, J. Hung, X. Lu, Y. Lai, and B. Su, "Wavelet speech enhancement based on nonnegative matrix factorization," *IEEE Signal Processing Letters*, vol. 23, no. 8, pp. 1101–1105, 2016.

[39] i. The Mathworks, "Inverse continuous wavelet transform." [Online]. Available: https://www.mathworks.com/help/wavelet/gs/inverse-continuous-wavelet-transform.html

[40] M. Kolbæk, Z.-H. Tan, S. H. Jensen, and J. Jensen, "On loss functions for supervised monaural time-domain speech enhancement," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 825–838, 2020.

[41] T. Kounovsky and J. Malek, "Single channel speech enhancement using convolutional neural network," in *2017 IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM)*. IEEE, 2017, pp. 1–5.

[42] M. D. Sacchi, "Adding noise with a desired signal-to-noise ratio."

[43] "Timit acoustic-phonetic continuous speech corpus." [Online]. Available: https://catalog.ldc.upenn.edu/LDC93S1

[44] "Csr-i (wsj0) complete." [Online]. Available: https://catalog.ldc.upenn.edu/LDC93S6A

[45] I. V. Tetko, D. J. Livingstone, and A. I. Luik, "Neural network studies. 1. comparison of overfitting and overtraining," *Journal of chemical information and computer sciences*, vol. 35, no. 5, pp. 826–833, 1995.

[46] S. Lawrence and C. L. Giles, "Overfitting and neural networks: conjugate gradient and backpropagation," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, vol. 1. IEEE, 2000, pp. 114–119.

[47] A. W. Rix, J. G. Beerends, M. P. Hollier, and A. P. Hekstra, "Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs," in *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221)*, vol. 2. IEEE, 2001, pp. 749–752.

[48] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, "A short-time objective intelligibility measure for time-frequency weighted noisy speech," in *2010 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2010, pp. 4214–4217.

[49] O. GmbH. [Online]. Available: https://www.opticom.de/technology/pesq.php

[50] "862.2: Wideband extension to recommendation p. 862 for the assessment of wideband telephone networks and speech codecs. itu-telecommunication standardization sector."

[51] J. Jensen and C. H. Taal, "An algorithm for predicting the intelligibility of speech masked by modulated noise maskers," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 11, pp. 2009–2022, 2016.

Appendix A

Results

A.1 DnCNN without Skip Connections

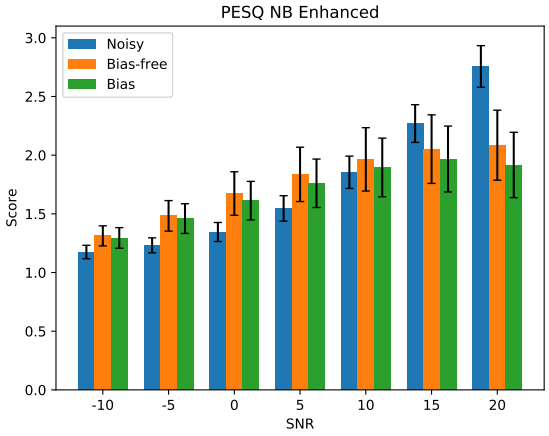


Figure A.1: Bar plot showing the mean PESQ Narrowband score for DnCNN without skip trained on the full SNR range.

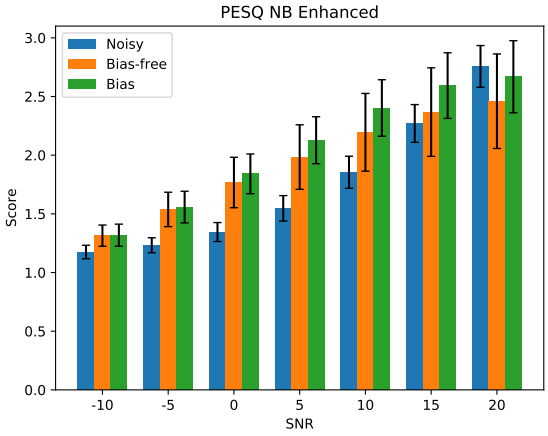


Figure A.2: Bar plot showing the mean PESQ Narrowband score for DnCNN without skip trained on the reduced SNR range.

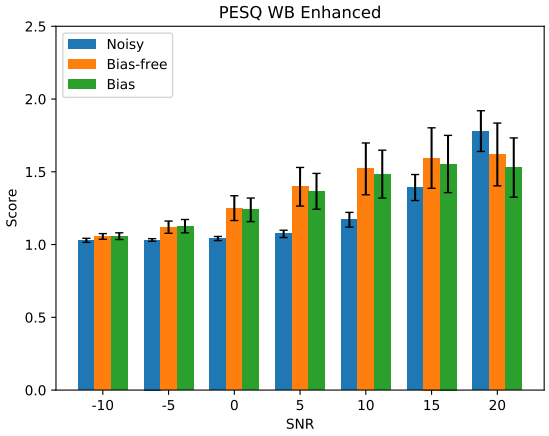


Figure A.3: Bar plot showing the mean PESQ Wideband score for DnCNN without skip trained on the full SNR range.

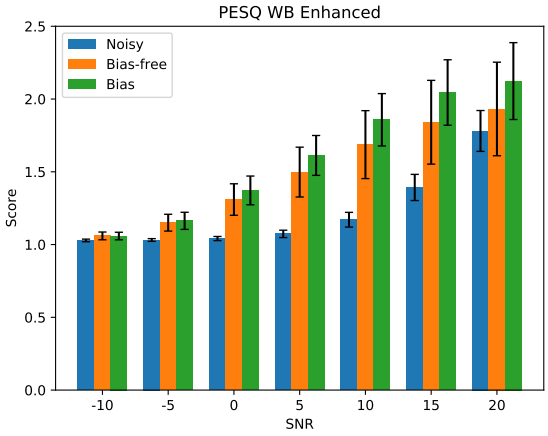


Figure A.4: Bar plot showing the mean PESQ Wideband score for DnCNN without skip trained on the reduced SNR range.

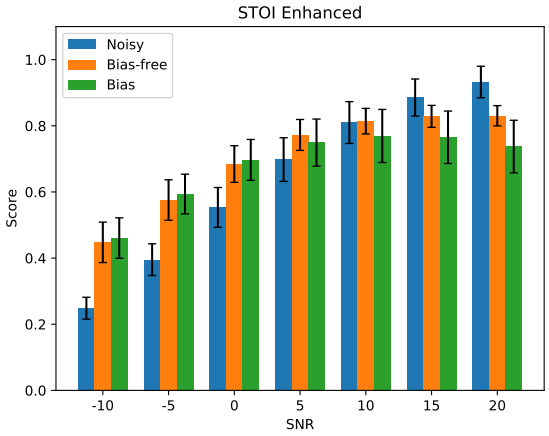


Figure A.5: Bar plot showing the mean STOI score for DnCNN without skip trained on the full SNR range.

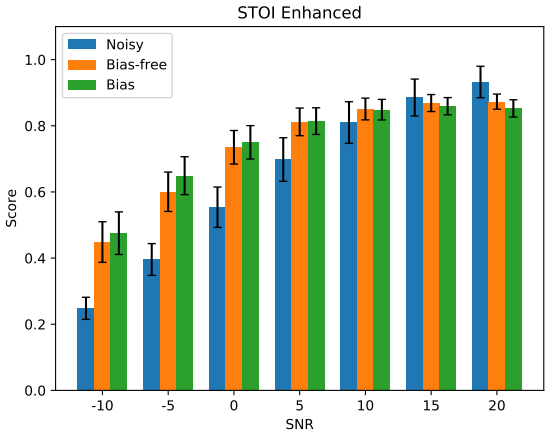


Figure A.6: Bar plot showing the mean STOI score for DnCNN without skip trained on the reduced SNR range.

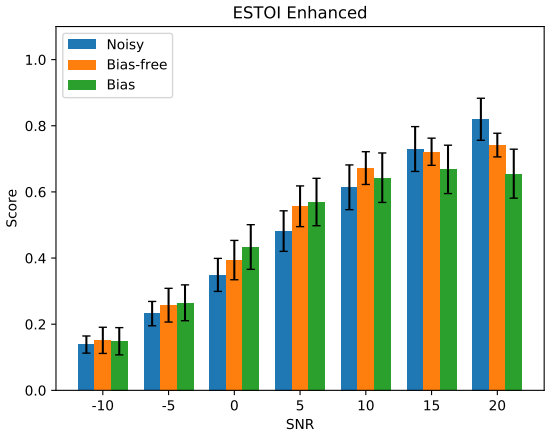


Figure A.7: Bar plot showing the mean ESTOI score for DnCNN without skip trained on the full SNR range.

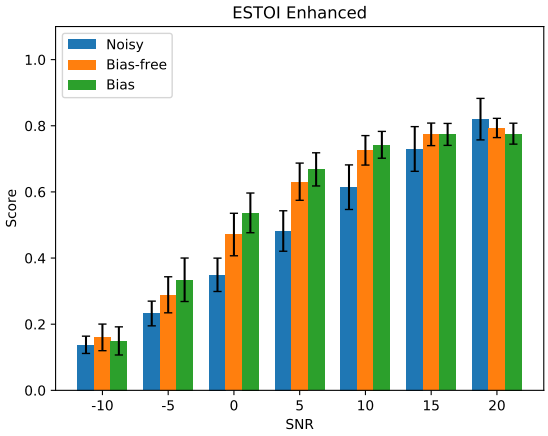


Figure A.8: Bar plot showing the mean ESTOI score for DnCNN without skip trained on the reduced SNR range.

A.2 DnCNN with Skip Connections

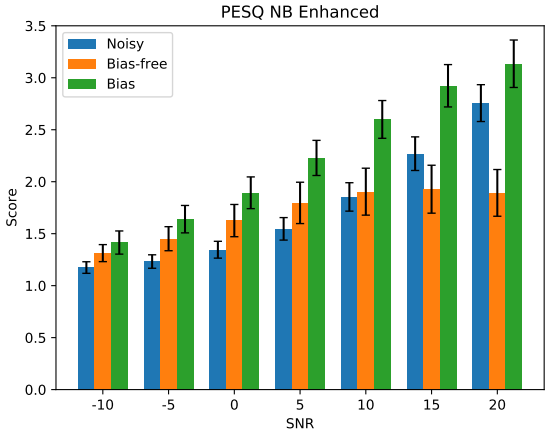


Figure A.9: Bar plot showing the mean PESQ Narrow-band score for DnCNN with skip trained on the full SNR range.

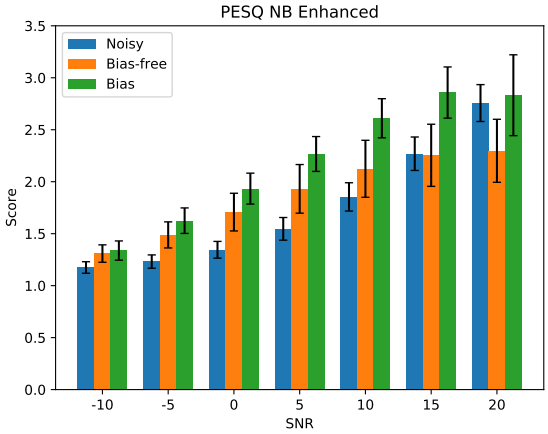


Figure A.10: Bar plot showing the mean PESQ Narrow-band score for DnCNN with skip trained on the reduced SNR range.

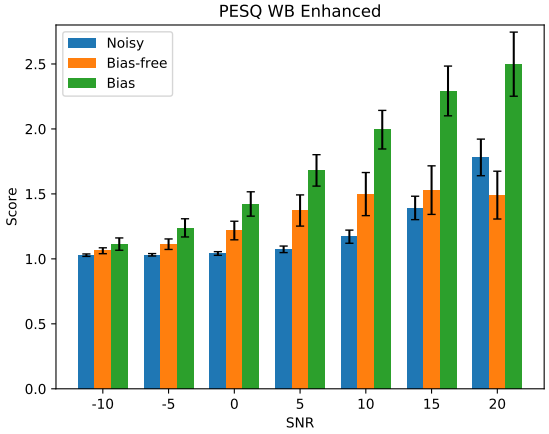


Figure A.11: Bar plot showing the mean PESQ Wide-band score for DnCNN with skip trained on the full SNR range.

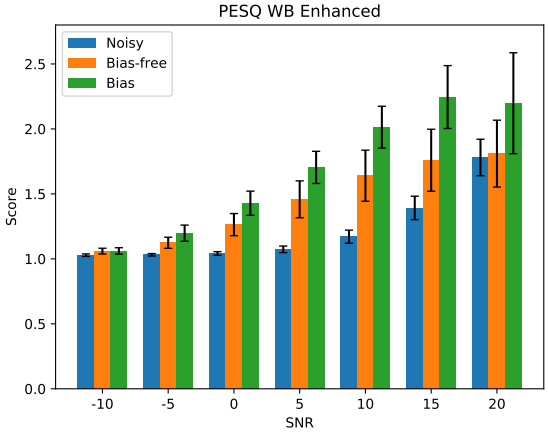


Figure A.12: Bar plot showing the mean PESQ Wide-band score for DnCNN with skip trained on the reduced SNR range.

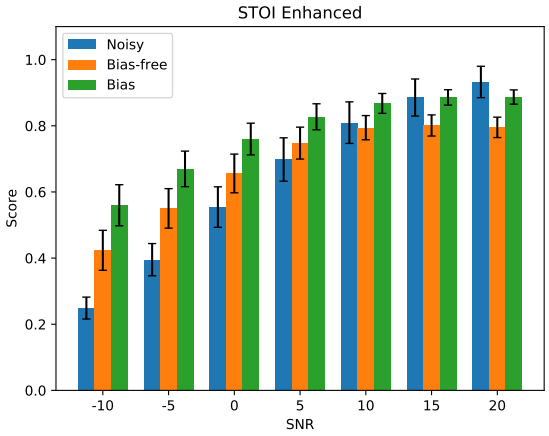


Figure A.13: Bar plot showing the mean STOI score for DnCNN with skip trained on the full SNR range.

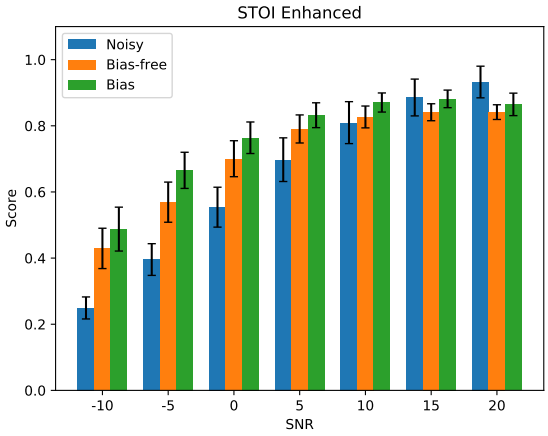


Figure A.14: Bar plot showing the mean STOI score for DnCNN with skip trained on the reduced SNR range.

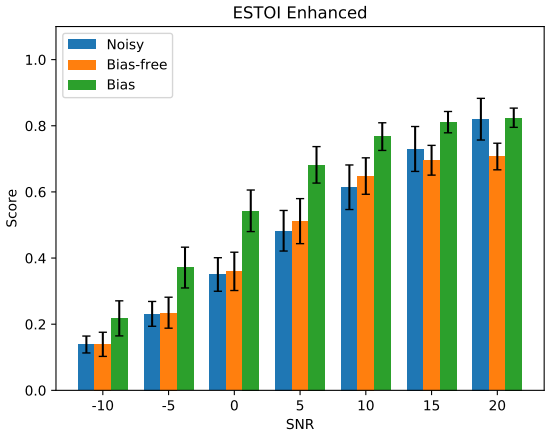


Figure A.15: Bar plot showing the mean ESTOI score for DnCNN with skip trained on the full SNR range.

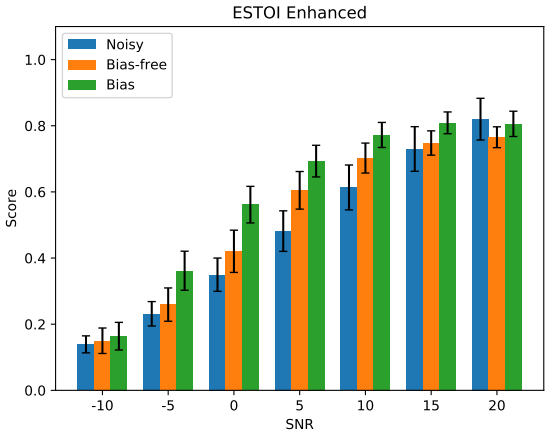


Figure A.16: Bar plot showing the mean ESTOI score for DnCNN with skip trained on the reduced SNR range.

A.3 UNet Results

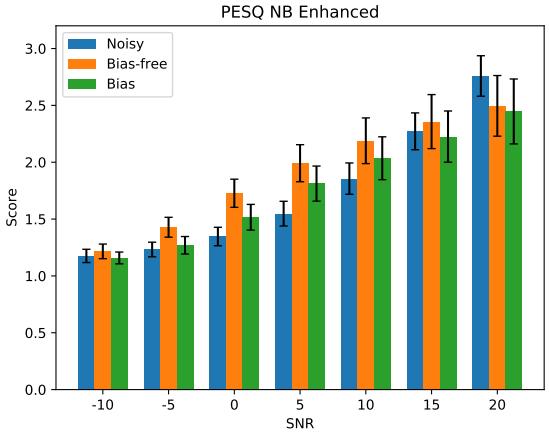


Figure A.17: Bar plot showing the mean PESQ Narrow-band score for UNet trained on the full SNR range.

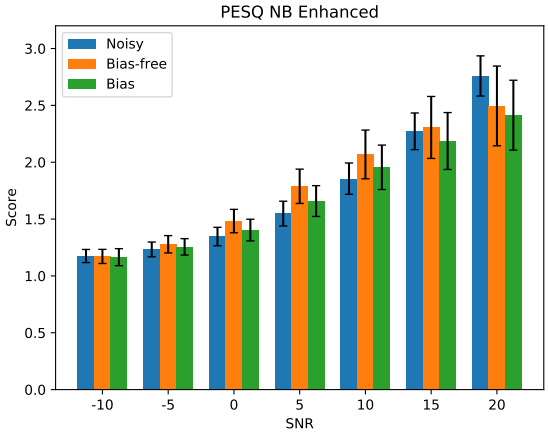


Figure A.18: Bar plot showing the mean PESQ Narrow-band score for UNet trained on the reduced SNR range.

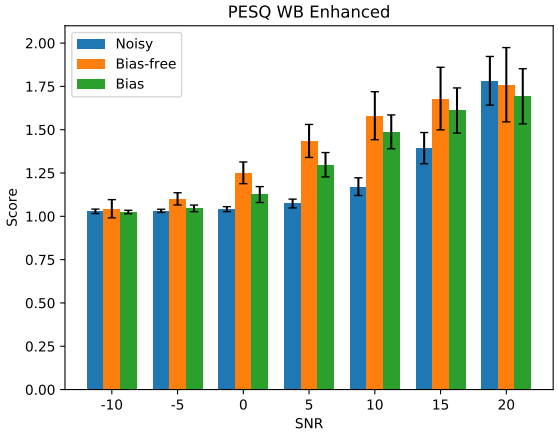


Figure A.19: Bar plot showing the mean PESQ Wide-band score for UNet trained on the full SNR range.

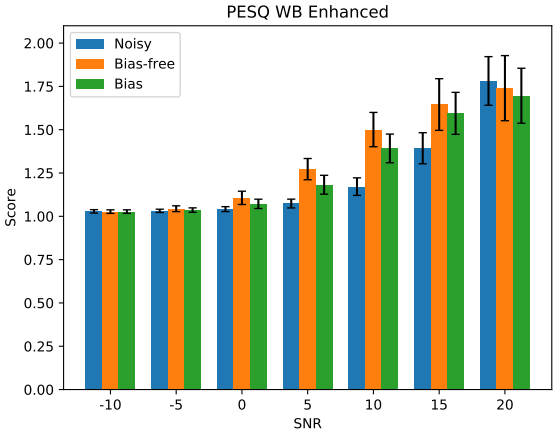


Figure A.20: Bar plot showing the mean PESQ Wide-band score for UNet trained on the reduced SNR range.

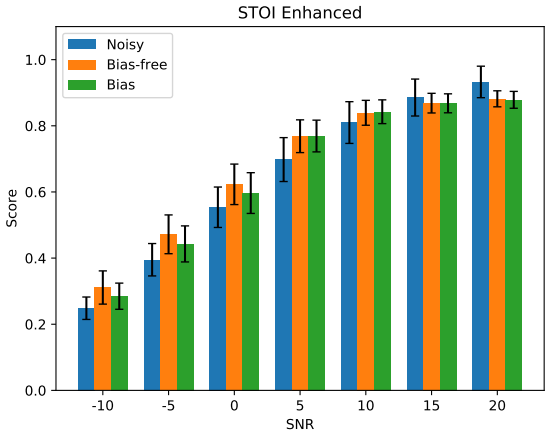


Figure A.21: Bar plot showing the mean STOI score for UNet trained on the full SNR range.

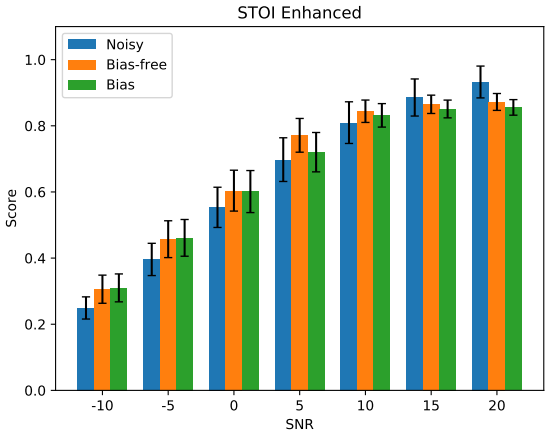


Figure A.22: Bar plot showing the mean STOI score for UNet trained on the reduced SNR range.

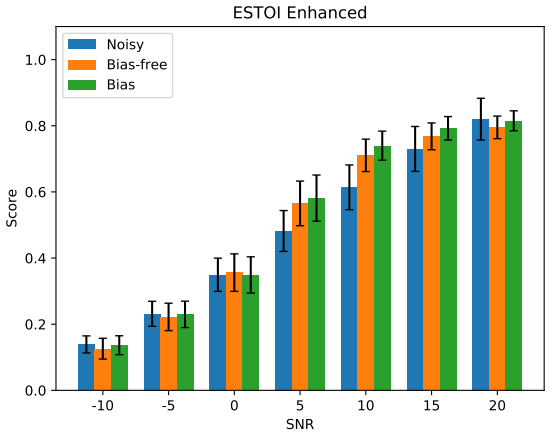


Figure A.23: Bar plot showing the mean ESTOI score for UNet trained on the full SNR range.

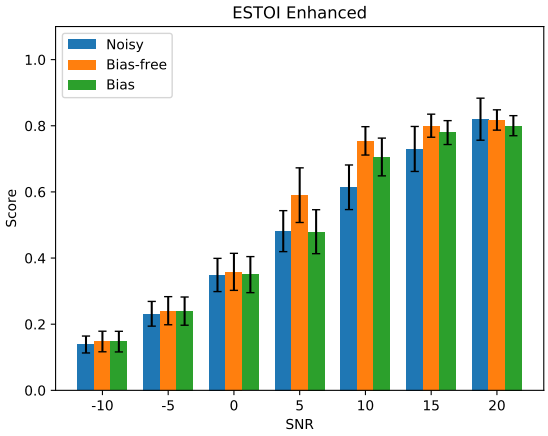


Figure A.24: Bar plot showing the mean ESTOI score for UNet trained on the reduced SNR range.

A.4 Convolutional Autoencoder Results

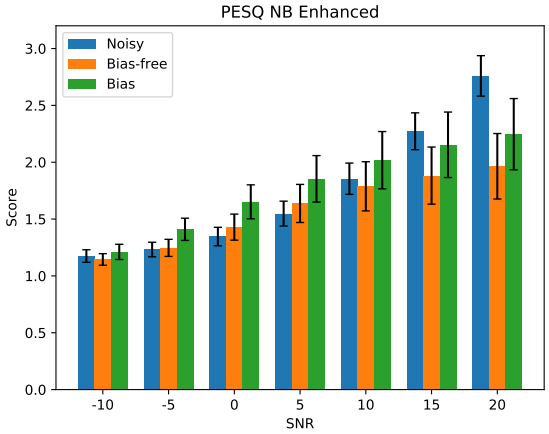


Figure A.25: Bar plot showing the mean PESQ Narrowband score for Convolutional Autoencoder trained on the full SNR range.

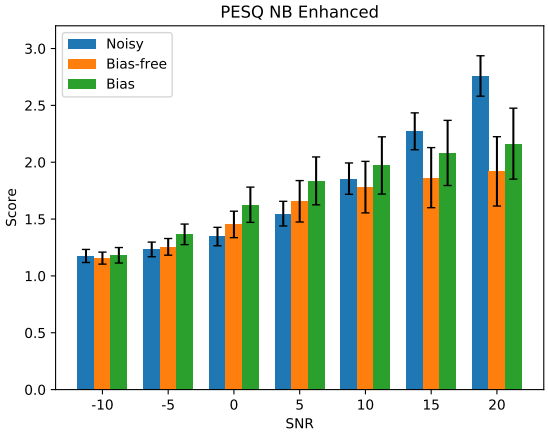


Figure A.26: Bar plot showing the mean PESQ Narrowband score for Convolutional Autoencoder trained on the reduced SNR range.

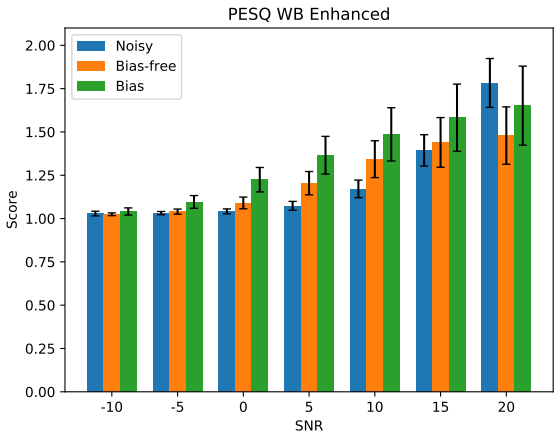


Figure A.27: Bar plot showing the mean PESQ Wideband score for Convolutional Autoencoder trained on the full SNR range.

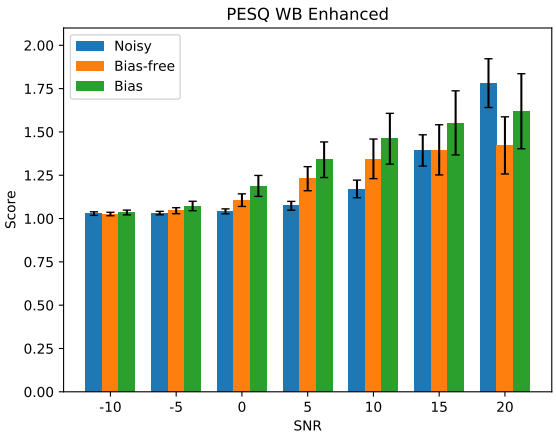


Figure A.28: Bar plot showing the mean PESQ Wideband score for Convolutional Autoencoder trained on the reduced SNR range.

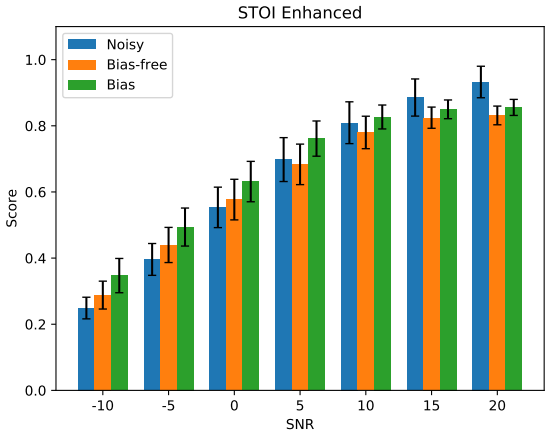


Figure A.29: Bar plot showing the mean STOI score for Convolutional Autoencoder trained on the full SNR range.

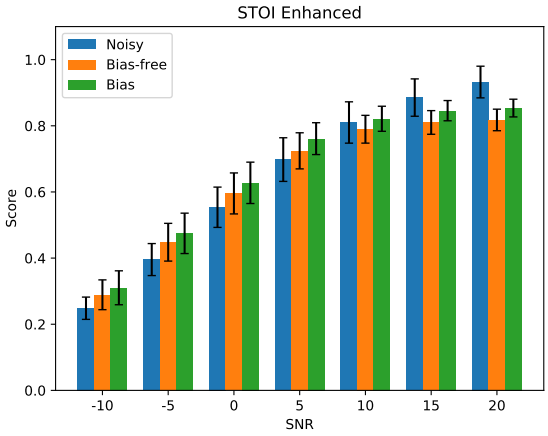


Figure A.30: Bar plot showing the mean STOI score for Convolutional Autoencoder trained on the reduced SNR range.

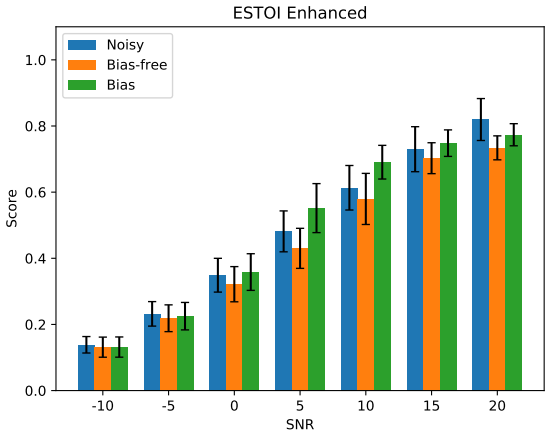


Figure A.31: Bar plot showing the mean ESTOI score for Convolutional Autoencoder trained on the full SNR range.

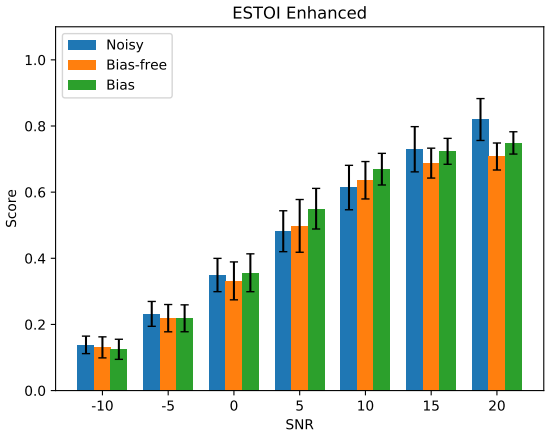


Figure A.32: Bar plot showing the mean ESTOI score for Convolutional Autoencoder trained on the reduced SNR range.

Appendix B

Sample Spectrograms

In this appendix, sample spectrograms of each network type, trained with both the limited and full SNR range, and with and without bias will be presented. Only the predictions on one SNR (10 dB) is shown however, due to the excessive number of plots that would otherwise be necessary. However, this is deemed by the authors to be sufficient to compare the outputs of each network visually. A reference spectrogram of the clean speech signal is also provided in Figure B.1 for ease of comparison.

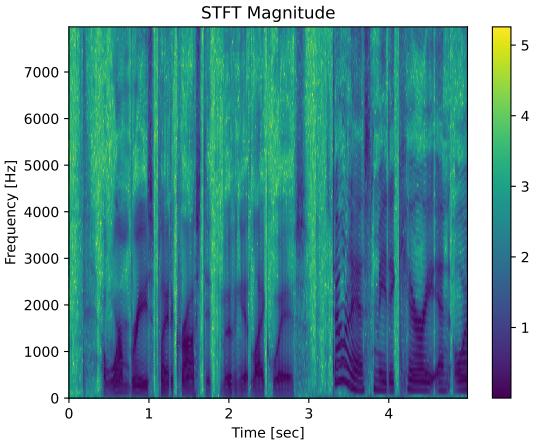


Figure B.1: Clean reference spectrogram for comparison to enhanced spectra provided below.

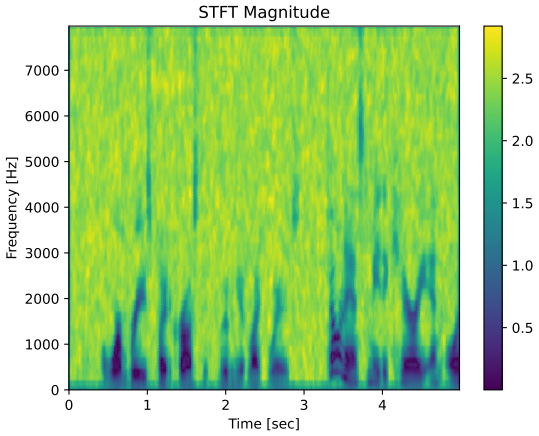


Figure B.2: Log-magnitude spectrogram of noisy speech enhanced using UNET, trained on the limited SNR range with bias

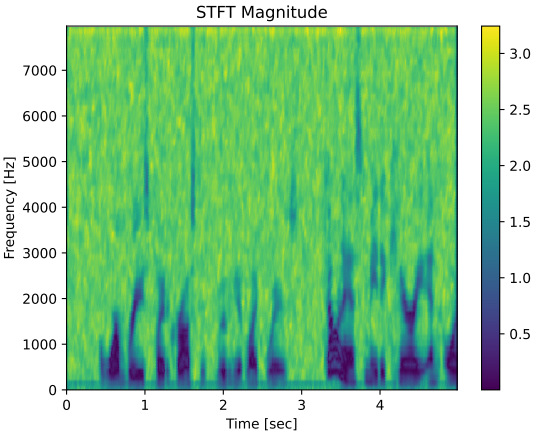


Figure B.3: Log-magnitude spectrogram of noisy speech enhanced using UNET, trained on the limited SNR range without bias

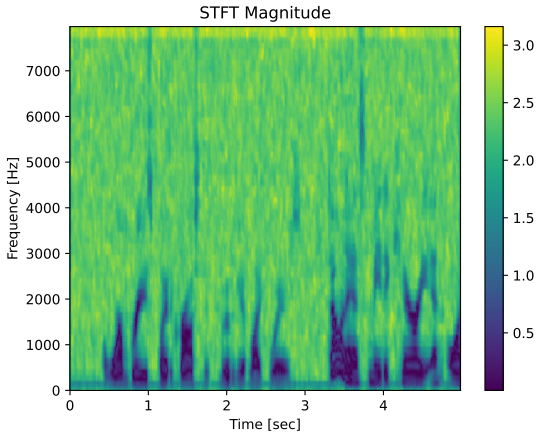


Figure B.4: Log-magnitude spectrogram of noisy speech enhanced using UNET, trained on the full SNR range with bias

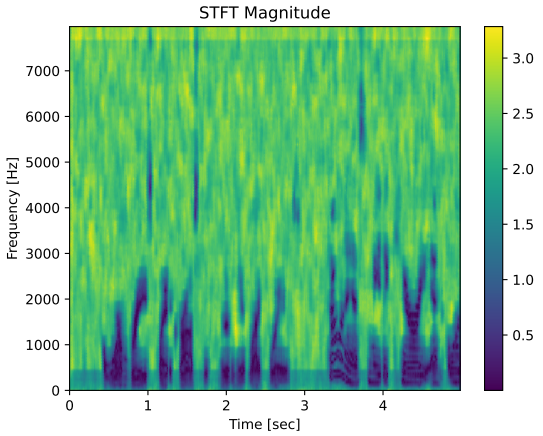


Figure B.5: Log-magnitude spectrogram of noisy speech enhanced using UNET, trained on the full SNR range without bias

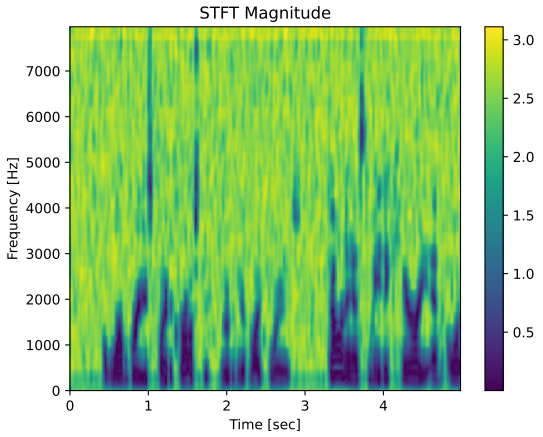


Figure B.6: Log-magnitude spectrogram of noisy speech enhanced using a convolutional autoencoder, trained on the limited SNR range with bias

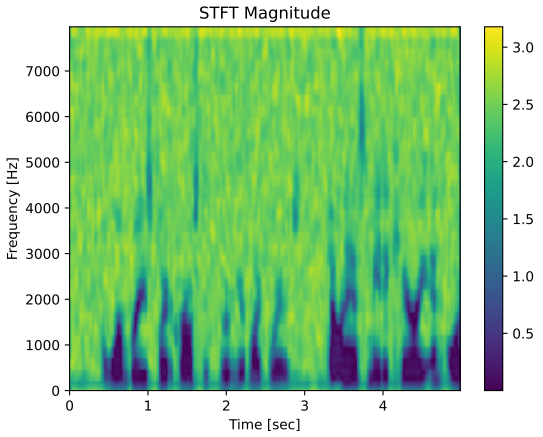


Figure B.7: Log-magnitude spectrogram of noisy speech enhanced using a convolutional autoencoder, trained on the limited SNR range without bias

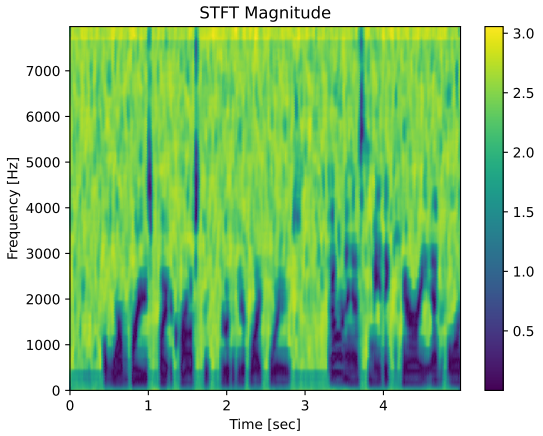


Figure B.8: Log-magnitude spectrogram of noisy speech enhanced using a convolutional autoencoder, trained on the full SNR range with bias

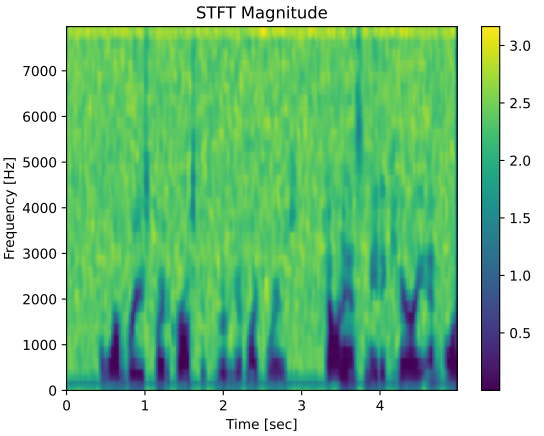


Figure B.9: Log-magnitude spectrogram of noisy speech enhanced using a convolutional autoencoder, trained on the full SNR range without bias

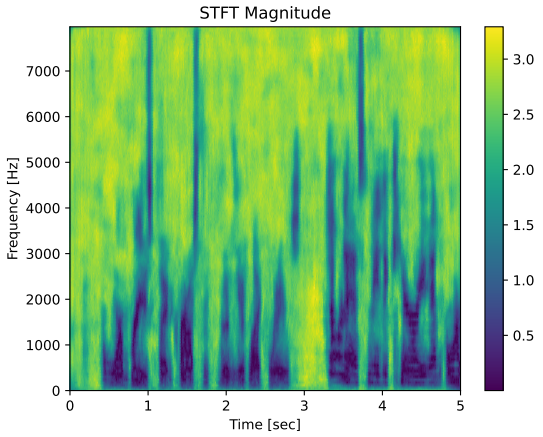


Figure B.10: Log-magnitude spectrogram of noisy speech enhanced using a DnCNN without a skip connection, trained on the limited SNR range with bias

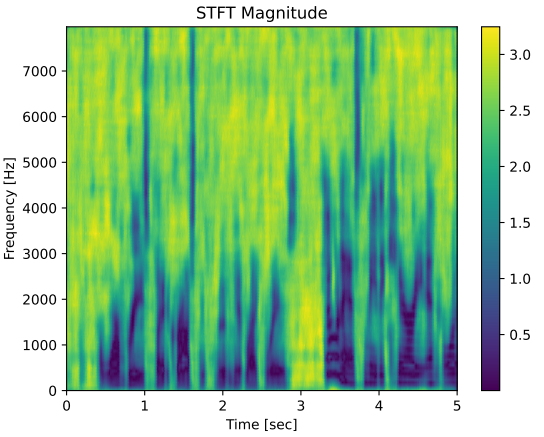


Figure B.11: Log-magnitude spectrogram of noisy speech enhanced using a DnCNN without a skip connection, trained on the limited SNR range without bias

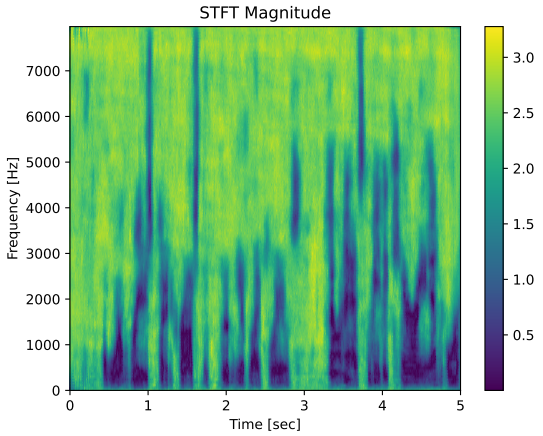


Figure B.12: Log-magnitude spectrogram of noisy speech enhanced using a DnCNN without a skip connection, trained on the full SNR range with bias

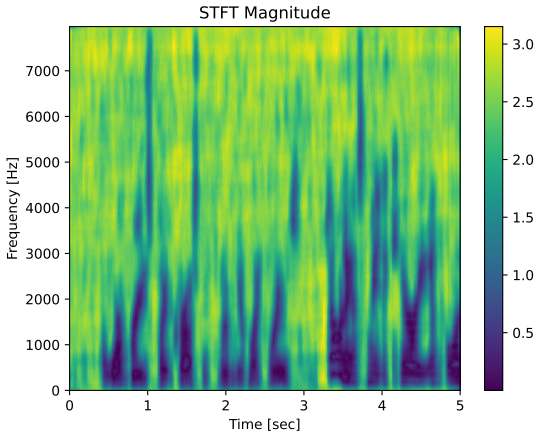


Figure B.13: Log-magnitude spectrogram of noisy speech enhanced using a DnCNN without a skip connection, trained on the full SNR range without bias

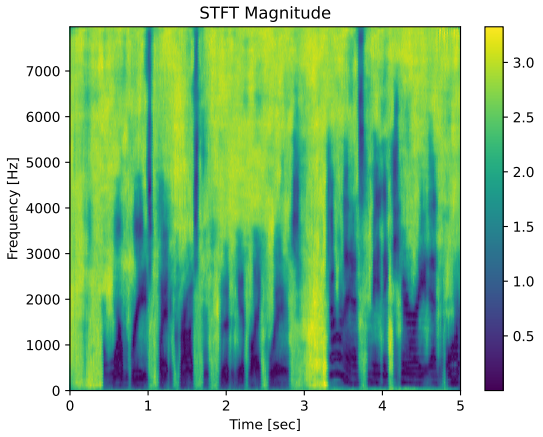


Figure B.14: Log-magnitude spectrogram of noisy speech enhanced using a DnCNN with a skip connection, trained on the limited SNR range with bias

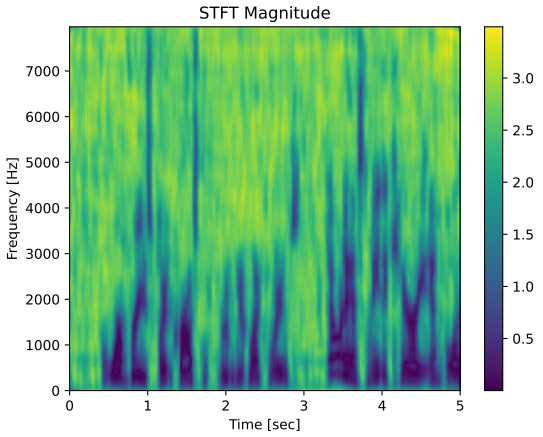


Figure B.15: Log-magnitude spectrogram of noisy speech enhanced using a DnCNN with a skip connection, trained on the limited SNR range without bias

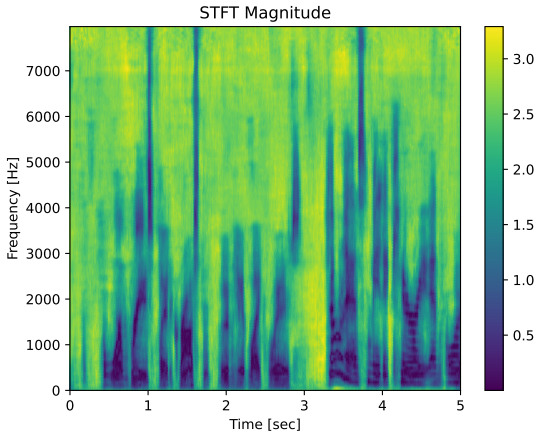


Figure B.16: Log-magnitude spectrogram of noisy speech enhanced using a DnCNN with a skip connection, trained on the full SNR range with bias

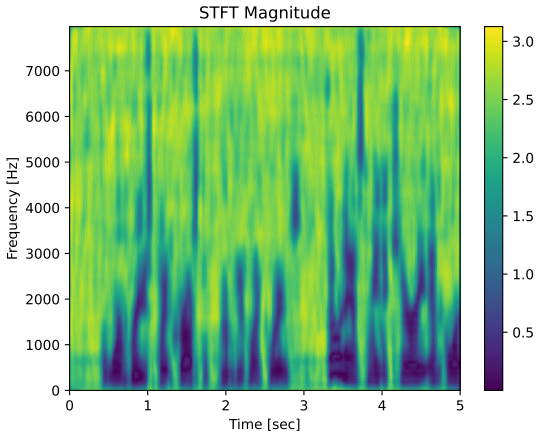


Figure B.17: Log-magnitude spectrogram of noisy speech enhanced using a DnCNN with skip connection, trained on the full SNR range without bias