Creating compelling music through deep convolutional networks

Louis Winder
law68@kent.ac.uk

The University of Kent

COMP6200 Research Project

Abstract

This paper explores the ability of convolutional neural networks (CNNs) to produce waveforms that sound musical, specifically by utilising the WaveNet model, introduced by Google DeepMind in [1]. By training on fragments of music, the model can learn to represent these waveforms and thus generate its own unique raw audio that is musical in nature. The model is trained on a broad range of data in the domain of solo piano. Constraining the domain to a single instrument makes the data significantly easier to model. When trained on a wide range of piano audio, the model is able to generate unique and often highly compelling samples of its own. Performing signal analysis on generated waveforms evidences this fact, whilst highlighting the model's ability to produce aesthetically pleasing output.

Contents

Chapter 1

Introduction

Using computers to create music that sounds pleasing to humans has long been a challenge for artificial intelligence research. In the past, endeavours such as Experiments in Musical Intelligence (EMI) [2] have produced some good results, but with the rise of novel machine learning techniques and resources in the past few years these works can be extensively built upon to provide many results quicker.

One of the main problems with neural networks producing music is that audio waveforms are often large in resolution. For example, high-quality audio is usually sampled at 44.1 kHz, which means that each second of audio has 44,100 samples. As we seldom have music that is only a few seconds in length, the amount of samples for say a musical piece will be in the millions. Furthermore, each sample is usually represented by 16 bits (its bit depth) and so to be able to model this large range is extremely challenging. This paper will utilise various methods to make this more tractable while still being able to produce very pleasant audio fragments.

Modelling in the raw audio domain provides a number of distinct advantages over other methods such as using symbolic representations (e.g. notes values from a score). Training on recordings of raw audio allows us to capture the subtleties and expression each musician conveys in their interpretations. Due to these embellishments being conditioned on, generation will contain the same features, which would simply not be considered in symbolic modelling methods.

Convolutional neural networks are useful for modelling waveforms as they act as good feature extractors [3]. WaveNet is an autoregressive model, meaning it uses past values in order to predict the future values and thus performing convolutions on these values yields meaningful representations of their structure. This paper will utilise the WaveNet convolutional neural network to learn from musical samples and be able to generate compelling, unique fragments of its own.

Chapter 2

Background to WaveNet

The primary underlying structure for the WaveNet model is a convolutional neural network (CNN). CNNs are based upon the mathematical operation of a convolution[1], which measures the amount of overlap between one function f as it is shifted over another g [4], with * commonly being used as the convolution operator. Convolutional neural networks have their roots in neuroscience, notably in studies on the visual cortices of cats by Hubel and Wiesel [5] that coined the concept of a receptive field which determined what neurons would fire in response to visual stimuli. This concept was then elaborated over time and would come to be integral to the way CNNs interpret their input data.

The WaveNet model at its core is a convolutional network, however in some respects it deviates from how a traditional CNN [6, 7] would be structured. For instance, as described in the paper [1], WaveNet does not utilise any pooling layers which are often present in traditional CNNs in order to reduce the spatial dimension of the input and act as a non-linear downsampler. Therefore, this chapter will omit discussing most of these typical features and instead focus on explaining the specific architecture behind WaveNet.

2.1 Convolutional layers

The most fundamental part of a CNN is the convolutional layer. The convolutional layer allows learning of features through kernels (or filters) which provide a small view of the input, called its receptive field. The kernel is an $n * n$ matrix which is often multiple times smaller than the input, but that convolves over it fully to produce a feature map. It does this by sequentially computing the dot product between the receptive field and kernel values (Figure 1). What makes convolution filters so powerful are the fact that they are learnable by the network, allowing for independent discovery of relevant features (in opposed to hard-coding kernels and testing them as was the standard before CNNs).

[1] In practice, implementations are typically based on cross-correlations which are identical to convolutions but use a mirrored kernel.

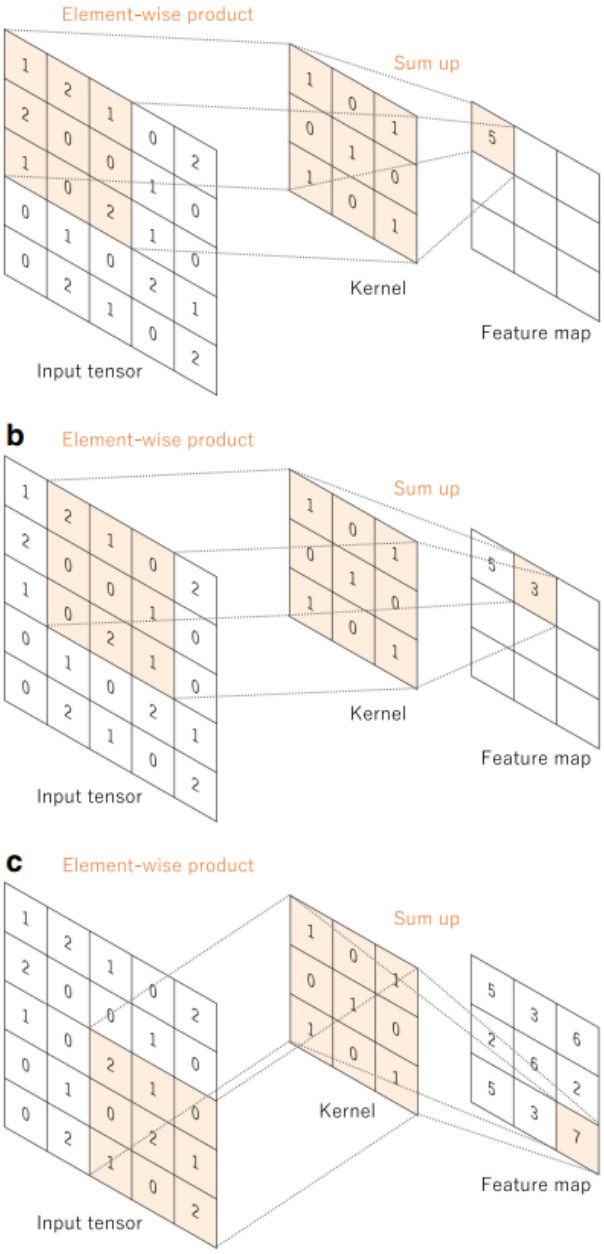


Figure 1: An example convolution operation on a 5x5 input tensor and 3x3 kernel. Convolving over the entire input tensor, a feature map of the input is constructed. Image taken from [8].

A CNN is made up of multiple convolutional layers which over time transforms the input into a set of distinctive features that capture important parts of the input. For example, feature maps of an image of a cat might highlight its ears or paws. Convolution operations seek to highlight the tight relationships between neighbouring values in the input by exploiting the fact that these are often closely linked. This is why convolutional networks perform well on image tasks [6, 7]; as feature pixels are inherently clustered.

2.2 Activation layers

An activation layer in a CNN is used to ensure non-linearity in the computation of outputs from inputs. There can be many such layers in an entire network. The activation layers compute the activation function of the feature maps that are output from prior convolutional layers. Thus, these activation layers come immediately after convolutional layers (Figure 2). There are many different activation functions that are used e.g. sigmoid, hyperbolic tangent, ReLU, or a combination of these. The rectified linear unit (ReLU) activation function is used frequently, being shown to perform better on multi-layered neural networks [9].

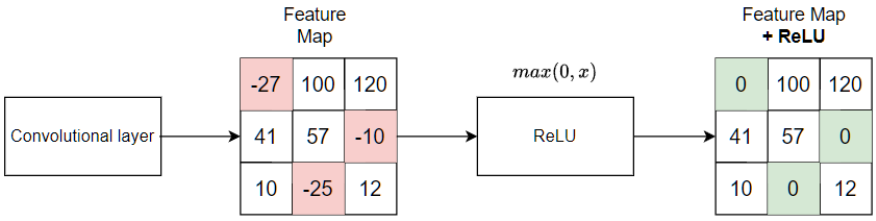


Figure 2: An example of how the ReLU activation function alters the output from a regular convolutional layer. Negative values become zero while all others remain unchanged.

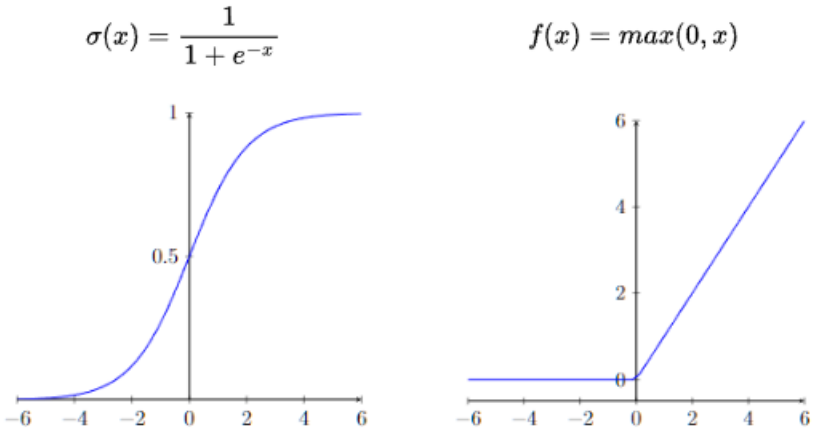


Figure 3: Graphs of two popular activation functions: Sigmoid and ReLU.

2.3 Fully-connected layer

The fully-connected layer is the penultimate layer in a CNN, after inputs have passed through all convolutional layers. It is where the final classification (prediction) is determined. The fully-connected layer connects a neuron to each output of the final convolutional layer, in the same way an ordinary artificial neural network does. For each neuron in the fully-connected layer, its activation is an affine transformation between input x and weight matrix w plus a learnable bias term:

$$y = wx + b \tag{1}$$

As outputs from convolutional layers are in the form of n-dimensional feature maps, they must be appropriately flattened for processing by the fully-connected layer. Typically, this is done by simply sequentially arranging the input values to fit a 1-dimensional space.

2.4 Loss layer

The final layer of a CNN is the loss layer (more commonly referred to as the loss function). Its purpose is to provide the resulting classification over a set of target values, comparing them to the output of the fully-connected layer to compute an error term that quantitatively evaluates the network's prediction. Different loss functions are suited for different classification problems. For the case of WaveNet, outputs are probability distributions, so cross-entropy loss is used which calculates the discrepancy between output distributions over K target classes. Equation 2 shows the formula for calculating general[2] cross-entropy loss between the predicted distribution q_i and true label p_i.

$$H(p, q) = -\sum_{i} p_i \, log \, q_i \tag{2}$$

[2]Implementations can vary

Chapter 3

WaveNet architecture

This chapter will detail the specific architecture of the WaveNet model. WaveNet is an autoregressive model, meaning its outputs depend linearly upon its inputs. This way, over time it builds up a global representation of its entire input and can predict future outputs using the information it has learnt. WaveNet was developed for audio waveforms because a similar model architecture, PixelCNN [10], had been shown to produce good results when using convolutional layers to model products of conditional probabilities between pixels in images. Hence, the authors applied similar logic to digital audio which also resulted in extremely convincing generation. PixelCNN was conceived out of a desire to improve the slow training time of its sister network PixelRNN [11]. Both are networks that model the conditional probabilities over pixels in images.

3.1 Audio as the product of conditional probabilities

WaveNet models the joint probability of a waveform $x = \{x_1, x_2, ..., x_T\}$ as a product of the conditional probabilities of samples:

$$p(x) = \prod_{t=1}^{T} p(x_t \mid x_1, ..., x_{t-1}) \tag{3}$$

This results in each sample x_t being conditioned on samples at all previous timesteps $x_1, ..., x_{t-1}$. The network learns the how to predict the next samples in a waveform given the previous ones, building up a global representation of its structure.

Conditional probabilities in the network are modelled by stacks of convolutional layers. As mentioned in the original paper, there are no pooling layers so there is little downsampling present. This is important as we want to retain as much information we can about local structure. The output of the model is a probability distribution over possible amplitudes (256 here, see 4.1.2) for *all* input samples. For example, for an input waveform of 16,000 samples the output would be a [16000 x 256] vector of distributions. We sample randomly from this distribution to obtain the prediction for x_t.

3.2 Dilated causal convolutions

Due to the temporal nature of audio data, we need to ensure samples are modelled chronologically. WaveNet does this through a special type of convolution called causal convolutions. Causal convolutions constrain the regular convolution operation such that when predicting next samples no "future" values are considered, only "past" ones. For modelling images in PixelCNN, causality is achieved by masking the convolution using a specially constructed binary tensor (Figure 4). For audio, we can more easily implement causality by doing a normal convolution and cutting off necessary samples at the end.

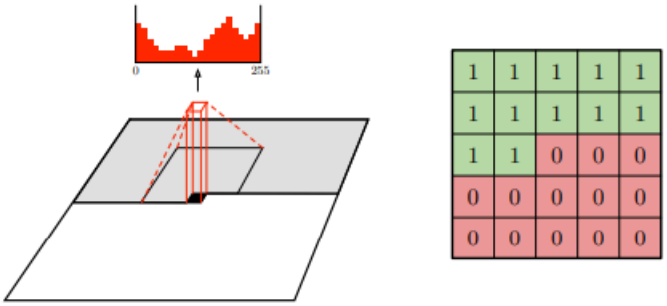


Figure 4: How causality is achieved in PixelCNN. Element-wise multiplication between the input and masked tensor limit the receptive field to a particular section of the image. Image taken from [11].

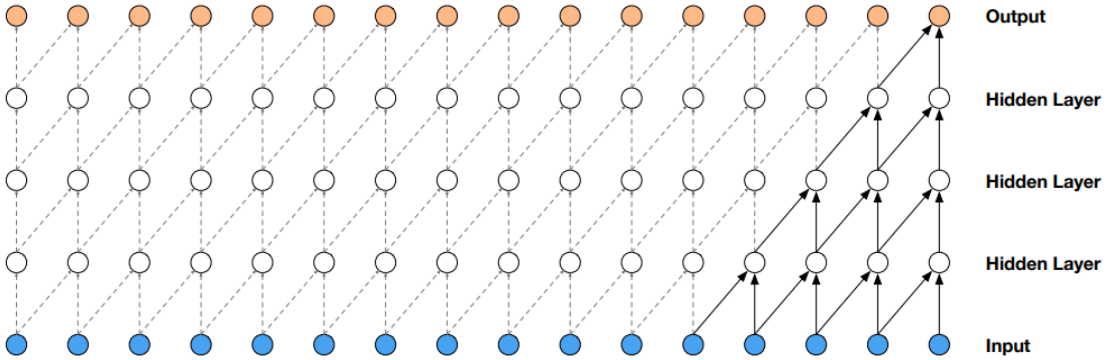


Figure 5: One stack of *causal* convolution layers (4). The convolution operations are highlighted in bold. Causality ensures the prediction for timestep x_{t+1} (top-right) cannot depend upon future samples $x_{t+1}, ..., x_T$ (not shown here). Image taken from [1].

WaveNet's convolutional layers are *dilated* causal convolutions. Dilated convolutions are convolutions where the kernel is artificially enlarged so that it covers a larger input space. They are useful for modelling audio as it allows us to extend the receptive field to many more samples. Due to music having correlations over a very large range (multiple seconds), this sort of coarse representation is imperative. Intuitively, dilated convolutions can be described as the further we look into the past, the better we can predict the future.

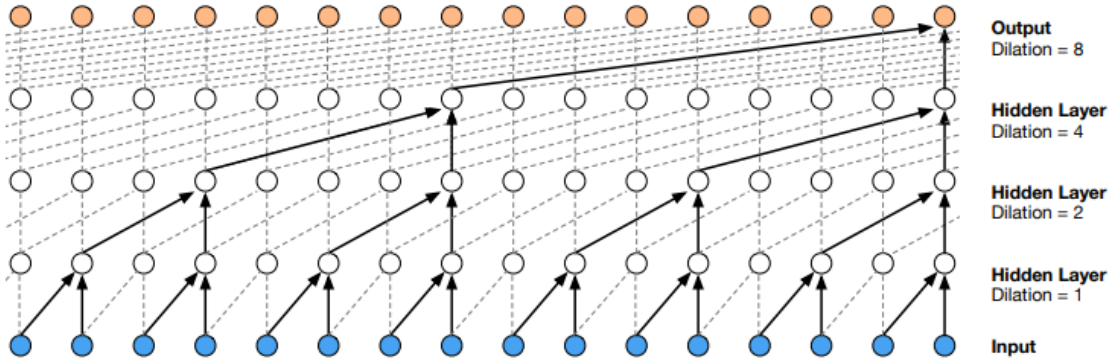


Figure 6: One stack of *dilated* causal convolution layers. Image taken from [1].

Dilations are doubled for each layer in a stack, increasing the receptive field exponentially per layer added. An entire WaveNet model can have multiple stacks of such layers which further increases the receptive field. The dilation factor is reset to 1 for every new stack to ensure outputs of all layers are being adequately considered.

3.3 Overall structure

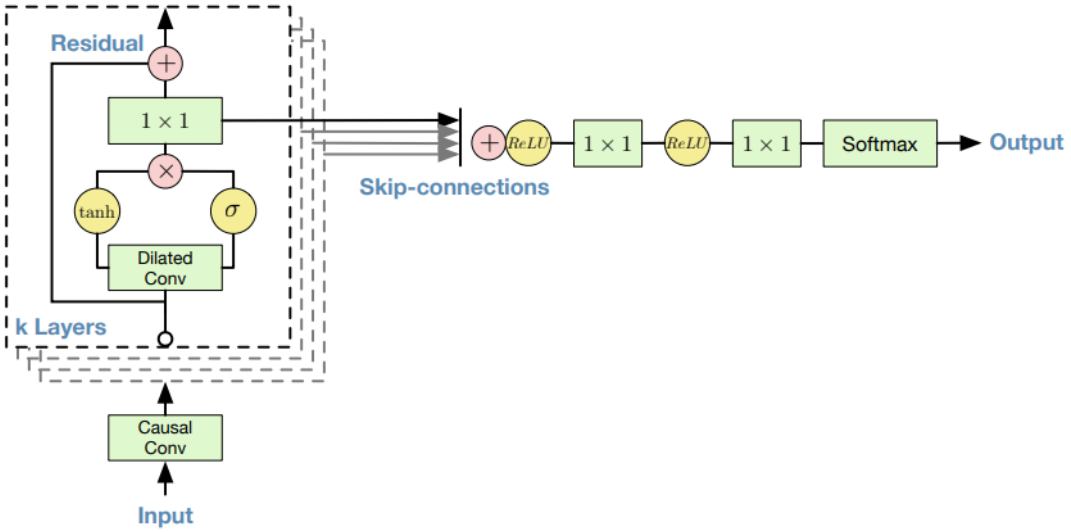


Figure 7: The overall architecture of WaveNet. Image taken from [1].

3.3.1 Residual layers and skip connections

The WaveNet architecture has residual layers that are stacked multiple times. The input first goes through a causal convolution then is split into two, with one side going through a dilated causal convolution and the other being passed directly to an addition operation at the end of the residual stack. Residual connections were introduced by He et al. [12] as a way to improve the performance of deeper networks. This is beneficial in the case of WaveNet as the more layers we can have the longer the receptive field will be.

Further to this, skip connections are also utilised. Outputs of the final 1x1 convolution layer (for each residual stack) are similarly divided in two and one half are summed before going through the final postprocessing layers. Skip connections here serve 2 purposes: first, by not passing information through subsequent layers we ensure valuable operations are not underrepresented, and second it enables more recent timesteps to have a higher weight which conveniently improves the network's predictions.

3.3.2 Gated activation unit

Rather than using a single activation function (e.g. ReLU), the output of the dilated convolutional layers use a specialised function called the gated activation unit. Although somewhat unclear from the diagram, the unit consists of two separate but identical dilated convolutions, "filter" and "gate", whose outputs are then collated as:

$$y = tanh(W_{f,k} * x) \odot \sigma(W_{g,k} * x) \tag{4}$$

where $*$ is a convolution operation, $\odot$ is an element-wise multiplication operation, σ is the sigmoid activation function, k is the current layer, f and g are the filter and gate convolutions and W is the learnable convolution filter. The authors of WaveNet indicated that they found this highly non-linear activation function to be more effective than others for modelling raw audio [1].

3.3.3 Softmax output distribution

To model the conditional probability distributions over each audio sample we normalise via a softmax layer. Softmax is a function that converts a list of real numbers into a probability distribution that conveys the likelihood of possible outcomes, in this case over the 256 different amplitudes each sample can take (4.1.2). Predictions are then made by sampling from these distributions. The standard softmax function σ is defined as:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{5}$$

where z is the unnormalised input vector, e is a standard exponential function and K is the number of classes to classify. The function ensures all outputs are within the interval $(0, 1)$ and that they sum to 1.

3.4 Training

WaveNet models the conditional predictions for all input timesteps T in order, thus, the ground truth labels used to calculate the loss and update the network weights are allocated as the input shifted forward a single timestep. Because for a single training data all ground truth values are known, we can parallelise the entire process to speed up computations significantly.

Training loss is measured as the cross-entropy (Equation 2) between the output predictions and the corresponding ground truth labels for all input samples T. We split the initial dataset into training and validation sets: 80 percent used to train for updating the network's weights and 20 percent for validating how the model can generalise to never-before-seen data.

Convolutional neural networks due to their structure are much faster to train than, for example, recurrent neural networks such have been used for time series data (including music) [13]. Their recurrent connections mean training is both computationally expensive and time-consuming [10, 14] which explains the motivation behind a CNN structure for the task of raw audio generation.

3.5 Audio generation

Once the model has been trained for an adequate number of epochs, we can use it to generate new and unique audios. In contrast to the training phase where we can predict in parallel, generating sample-by-sample is sequential. As each prediction x_{t+1} is dependent upon samples $x_1, x_2, ..., x_t$, in order to build up a waveform over time model predictions must be fed back in to the input for the next prediction (Figure 8).

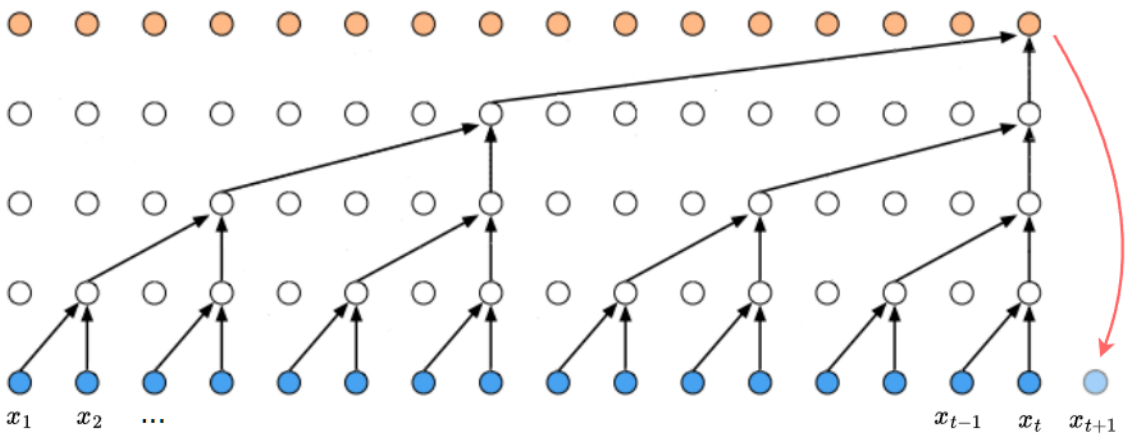


Figure 8: WaveNet's sequential generation process. Once sample x_{t+1} is predicted it is concatenated with the previous model input $(x_1, ..., x_t)$ and fed back in to predict the next sample. This operation is repeated for the amount of desired samples to generate.

In order to kick-start the generation we need at least one sample. Two different methods can be used to initiate the generation process: providing a single random sample (*unique* generation) or by supplying an existing seed (*assisted* generation).

3.5.1 Sampling from prediction distributions

In order to obtain an actual amplitude value for every prediction, we must sample from the network's output distribution. One naïve approach would be to simply choose the sample that has the highest probability each time. The problem with this method however is that it results in very limited generation variety. From tests using this method

there were a number of notable oddities such as the waveform frequently resolving to silence, suggesting this sampling method is somewhat untenable. Because of this, it's better to sample in a non-uniformly random manner. We implement a function that randomly chooses a sample based on the softmax distributions: samples are therefore weighted by chance but in contrast to highest sampling are given a larger possibility space. Empirically, this has shown to produce markedly better results.

3.5.2 Unique

Unique generation allows generation of fully novel waveforms. To begin generation an initial sample is chosen at random from an 8-bit space (256 possible values) corresponding to the $\mu-law$ encodings of amplitudes (see 4.1.2). Alternatively, we can choose an initial sample ourselves[3]. Once an initial sample is chosen, it is fed into the network to obtain the first prediction. Generation continues through the input-prediction-concatenation process outlined above until the desired length of samples is generated.

3.5.3 Assisted

Generation can also begin with an input seed of x number of samples which is a pre-existing waveform. Seeding the generation process can condition the model to continue generation along certain lines which is useful for example if a certain kind of sound is required. Seeding waveforms benefit from being similar to the data the model was trained on, although this should not be a strict requirement. Assisted generation works in the same way as unique generation but instead of providing a single sample as initial input we provide a list of samples and generation continues from there. For these experiments, the unique generation scheme was favoured for testing as it provides a clearer benchmark for generation quality.

[3]From testing choosing explicit samples in different ranges, it was observed that the subsequent generated audios can vary greatly. For example, choosing from around the median range results in quieter waveforms than choosing from either extreme. Given we are modelling conditional probabilities, intuitively this makes sense.

Chapter 4

Datasets

In order to test the abilities of WaveNet to their full potential, different datasets will be used to train the network and also to generate with it. This chapter will outline such datasets, describing their contents in detail and why they were chosen for evaluation. Also outlined will be the appropriate pre-processing of the audios in the dataset to prepare them for use by the network.

4.1 Data preparation

In order for the data in the datasets to be used by WaveNet, they must first go through a preparation phase involving three main stages: preprocessing, $\mu - law$ quantisation and input/target initialisation.

4.1.1 Dataset preprocessing

Audio data obtained from the raw datasets is sampled at high quality: typically 44.1 kHz or higher. This many samples presents a challenge for training of the model as even a single second of audio will have 44,100 samples for which each one has a probability distribution needed to be output over it. Modelling this many samples will soon lead to memory and time constraints hence we need a method of reducing the quality while retaining enough overall coherence. To remedy this, all datasets used went through an initial downsampling process, removing a large amount of samples from each waveform and thus lowing their qualities. Downsampling for all audio in the datasets was done using the Audacity digital audio editor[4].

In the original WaveNet paper [1], audio was downsampled to 16 kHz. In this paper, sampling rates of 16 kHz and also 8 kHz will be used. A further downsampling to 8 kHz allows for testing of longer receptive fields (number of samples used to predict the next) while retaining just enough quality for audio characteristics to be noticed. A sampling rate of lower than 8 kHz was not used as it resulted in incoherent audio representations.

[4] https://www.audacityteam.org/

After this downsampling process, where necessary all audios in the datasets were split into multiple shorter sub-audios (typically 5-6 seconds in length) using a custom-written program. This is to ensure each data example can fit into memory during training.

4.1.2 $\mu - law$ quantisation

The WaveNet architecture uses the softmax function to output a probability distribution over the possible values (amplitudes) any one sample of audio can take, randomly sampling from which yields likelihood values of samples at successive timesteps. Typically, each audio sample is stored as a 16-bit integer value meaning there are 2^{16} different possible amplitudes a sample can take. Performing a softmax on this would require 65,536 probability outputs and so to make it more tractable the audio is companded and quantised to an 8-bit space (256 possible values) using the equation for $\mu - law$ [15]:

$$f(x) = sign(x)\frac{ln(1+\mu|x|)}{ln(1+\mu)} \tag{6}$$

where $-1 \leq x \leq 1$ and $\mu = 255$. Doing this reduces the output dimensionality by a factor of 256 and therefore makes it much simpler to model.

In the generation phase, once an entire waveform is predicted its samples can be reconstructed back to their original form using the inverse of Equation 6:

$$f^{-1}(y) = sign(y)\frac{(1+\mu)^{|y|} - 1}{\mu} \tag{7}$$

where $-1 \leq y \leq 1$ and $\mu = 255$ as above. It should be noted however that this kind of non-linear discrete mapping will reduce the quality of audio upon its reconstruction[5].

4.1.3 Input/target initialisation

Given an input length of samples $x = \{x_1, x_2, ..., x_{t-1}\}$, its targets y is the list of samples x shifted forward by one timestep: $y = \{x_2, x_3, ..., x_t\}$. Hence, for each training example the model iteratively learns how to predict the next samples. The difference between the ground truth y and the model's predictions is calculated as the cross-entropy over $t-1$ samples (Equation 2) which is the metric used to adjust the weights during training.

The input to WaveNet is a one-hot encoding of a collection of x audio samples, and its output is a probability distribution of the same x number of samples over 256 possible amplitudes for each. This results in the output having the same shape as the input. Input tensors are in the form [B x C x T], where B is the batch size, C is the number of classes to predict (possible amplitudes in $\mu - law$, 256 in this case) and T is the list of samples. The model outputs a tensor of shape [B x T x 256] which is the predictions over all next samples.

[5]Authors of the WaveNet paper found that this reconstruction produced results that were very similar-sounding to the original, suggesting that this is an effective compression method.

4.2 MAESTRO

The first dataset that will be tested is the MAESTRO dataset [16]. MAESTRO is a dataset composed of virtuosic piano recordings. Audios for the dataset as-is are high-definition 44.1 kHz recordings but in order to make it tractable for modelling they have been downsampled to 16 kHz. The default (5.0) dataset of training audios contains 10,000 sub-audios each of 5 seconds in length, comprising around 14 hours of music.

4.3 MusicNet

The second dataset that will be tested is the MusicNet dataset [17]. MusicNet is a classical music dataset that, unlike MAESTRO, isn't just restricted to virtuosic performances. It contains a variety of pieces in the classical genre, including for instruments other than piano. Despite this, for our experiments the domain has been restricted to only pieces with the tag "Solo Piano". This was done to simplify the modelling process and to ensure fair benchmarks when evaluating against the MAESTRO dataset. The dataset of training audios contains 9,000 sub-audios each of 5 seconds in length, comprising 12.5 hours of music.

Chapter 5

Results

Documented in this chapter are the results from training and generating with different configurations of the WaveNet model and its training data. Experiments were conducted on various aspects of the model. The primary dataset used was MAESTRO, however a dedicated analysis has been performed on the MusicNet dataset to highlight potential differences. Due to specific computational resource limits, training of models was only able to be done in maximum 24-hour increments. Because of this, testing for a large number of epochs may appear staggered in loss graphs and other figures. Generated audios will be analysed using various signal analysis techniques, including constant-q transforms (CQTs) [18] and signal-to-noise ratio (SNR). All generated audios and their CQTs are available in the project's GitHub repository[6].

5.0 Default parameters

In order to ensure integrity of the testing process, further will be the default parameters that outline the configuration that will be used when testing individual parameters. An example would be when comparing how the model reacts to the different datasets: all other default settings should remain constant here as we are not testing them — the default for the variable we are currently testing is the only thing that is changed.

Default parameters:

- Model size: Medium[1]
- Dataset: MAESTRO
- Dataset audios sample rate: 16 kHz
- Dataset audios length: 5 seconds
- Number of dataset items: 10,000

In a select few cases some other parameters must also be adjusted when testing other things. These changes will be indicated when they occur.

[6] https://github.com/LouisWinder/WaveNet

5.1 Size

A WaveNet model has two adjustable parameters that are used to increase the receptive field of the model: number of stacks and number of layers (per stack). We refer to this as the size of the model. Larger models are able to look back further at past samples but take longer to train whereas small models may not be able to capture enough long-term structure to be able to make good predictions. For these experiments, we utilise three different model sizes and analyse their performance on generation.

	Model		
	Small	Medium	Large
No. Layers	10	13	14
No. Stacks	5	5	4
Rec. Field (samples)	5,115	40,955	65,532

Table 1: Model sizes and their respective receptive field lengths.

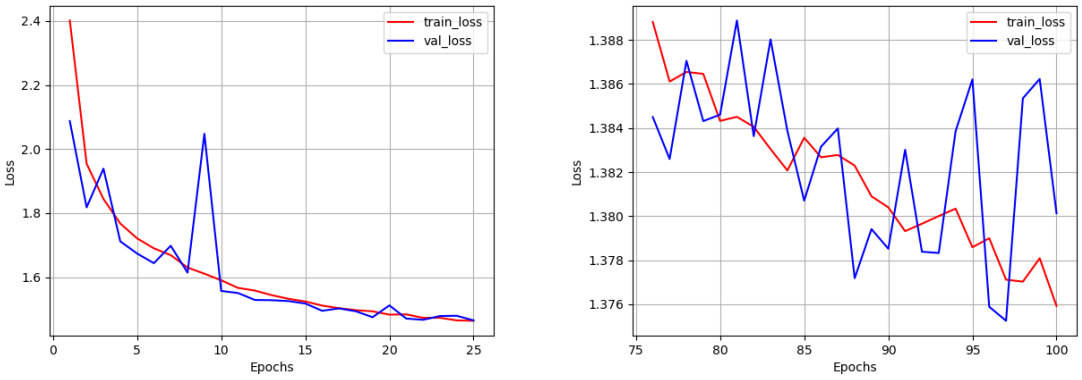
A model's receptive field can be calculated as the sum of dilations for all stacks using the equation:

$$rf = s \cdot \sum_{l=0}^{L-1} 2^l \tag{8}$$

where s is the number of stacks and L is the number of layers.

5.1.1 Small

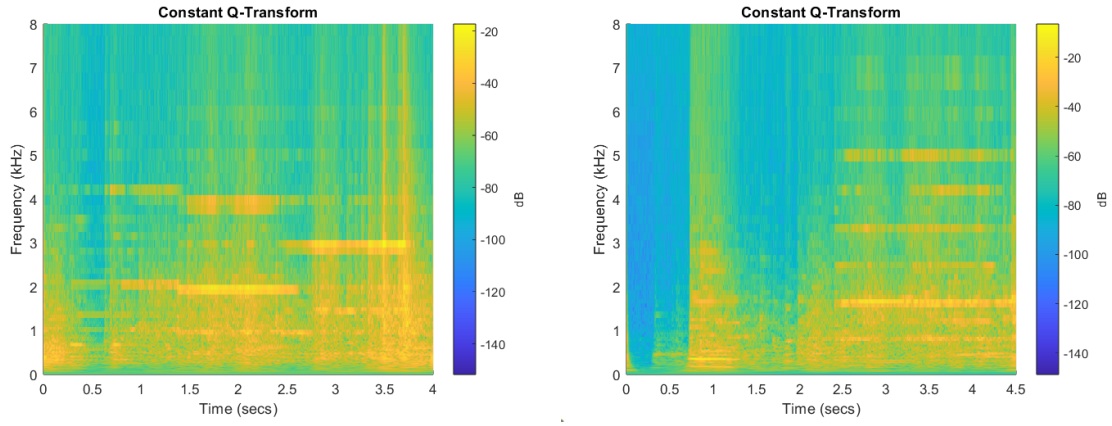
The small model is the fastest to train however the receptive field is restricted to 5,115 samples which means it struggles to capture long-term dependencies in its training waveforms. Small results in generation results that are less convincing than larger models however in practice the difference is relatively subtle. The small model was trained for 100 epochs using 10,000 training examples from the MAESTRO dataset.



(a) 0-25 epochs (b) 75-100 epochs

Figure 9: Loss plots of the small model from 0-25 and 75-100 epochs.

After 100 epochs training loss was able to get down to 1.376 with validation loss slightly lagging behind at 1.38. Compared to the medium and large models, for the same number of epochs the loss is noticeably higher. However, for the small model the downward trend appears to be steeper, suggesting that it can get down to the same loss values but it just needs more time to train.



(a) Generation after 25 epochs (b) Generation after 100 epochs

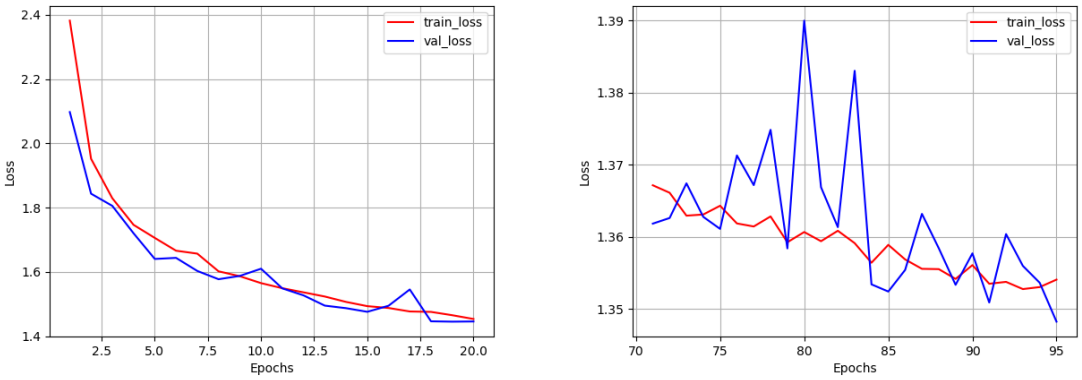
Figure 10: Constant-Q transforms of audios generated with the small model.

As we can see the generated audio after 25 epochs is relatively irregular but nonetheless it has coherent notes which are also audibly evident. After 100 epochs the results have improved slightly, with the model able to produce very interesting generations[7]. As evidenced by the 100 epoch generation the model can sometimes get stuck in a cycle which leads to repetitive generation which can be likened to trills, tremolos etc.

[7] https://github.com/LouisWinder/WaveNet/tree/main/Generated%20audios/Size/Small/Paper%20generations

5.1.2 Medium

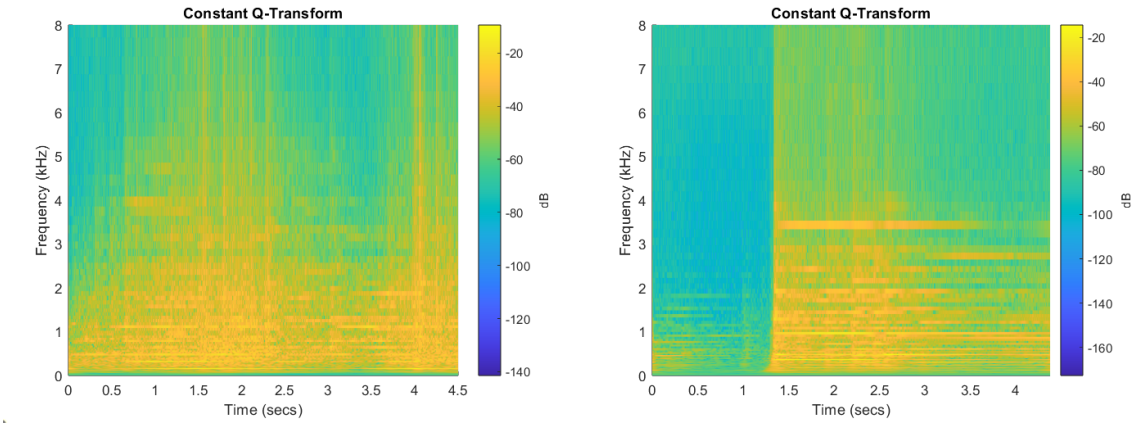
Medium is the model that was used most for testing. It gives a good trade-off between receptive field length and complexity so that the model can be trained relatively quickly but can also make good predictions. The medium model was trained for 95 epochs using 10,000 training examples from the MAESTRO dataset.



(a) 0-20 epochs (b) 70-95 epochs

Figure 11: Loss plots of the medium model from 0-20 and 70-95 epochs.

After 95 epochs the loss was able to get down to around 1.35 for both training and validation. As expected it slows down gradually per more epochs completed. Validation loss fluctuates a fair amount.



(a) Generation after 20 epochs (b) Generation after 95 epochs

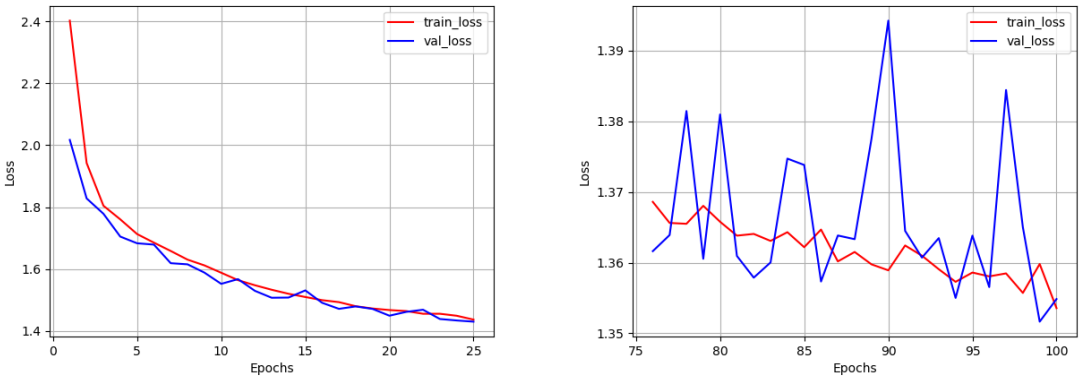
Figure 12: Constant-Q transforms of audios generated with the medium model.

The generated audios after 20 epochs were often not very convincing and generally noisy, as evidenced by the constant-q transforms (CQTs). Again, as the model was trained for longer it produced better quality waveforms[8].

[8] https://github.com/LouisWinder/WaveNet/tree/main/Generated%20audios/Size/Medium/Paper%20generations

5.1.3 Large

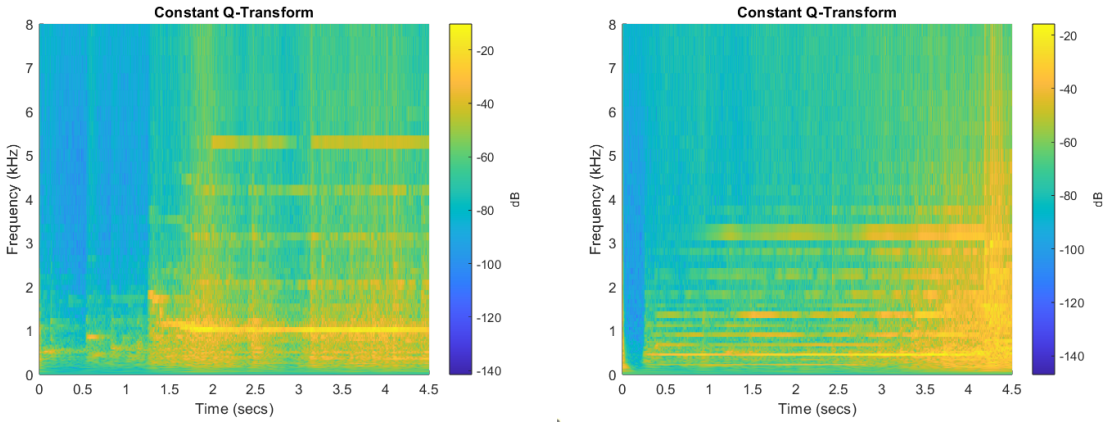
The authors of the WaveNet paper claimed that in order to produce waveforms that sounded coherently musical the receptive field must be multiple seconds [1]. The large model is used to explore whether this claim has any ground for these experiments. The large model was trained for 100 epochs using 10,000 training examples from the MAESTRO dataset.



(a) 0-25 epochs (b) 75-100 epochs

Figure 13: Loss plots of the large model from 0-25 and 75-100 epochs.

The large model is able to get down to around the same loss as with the medium model: training and validation losses are both around 1.355. This could suggest that medium and large models in general make similar predictions and so medium might have an advantage due to its lower training/inference time.



(a) Generation after 25 epochs (b) Generation after 100 epochs

Figure 14: Constant-Q transforms of audios generated with the large model.

Somewhat surprisingly, it seemed that the large model generally performed worse than the medium (and in some cases even small) models. Generation after both 25 and 100

epochs are relatively noisy as evidenced by the CQTs (Figure 14). Often, especially permitting longer generation times, outputs would resolve to predominantly noise with little musicality present. This wasn't always the case however, as some generations using the large model were aesthetically pleasing[9]. This outcome could suggest the model performs best at modelling local structure but struggles when predicting on a coarser scale, giving weight to an argument for limiting the receptive field.

5.2 Different sampling rates

Digital audio can be converted into a number of different sample rates (number of samples per second, see Appendix A). Higher sample rates provide higher quality recordings because a larger number of discrete numbers are being used to model a continuous analogue waveform. While desirable, a high sample rate is infeasible for the task of raw audio generation as it requires predictions from potentially hundreds of thousands of timesteps in the past, meaning the network will struggle to produce good quality output. We also will quickly run into memory constraints when modelling this much data.

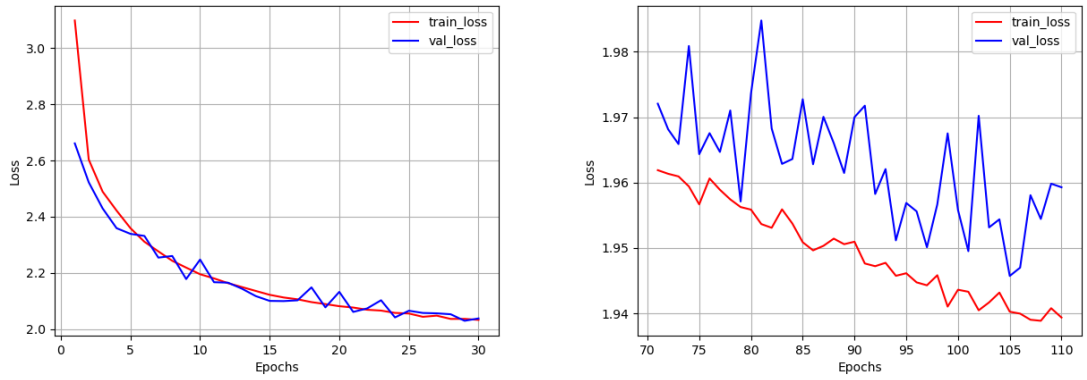
The default sampling rate used for experiments was 16 kHz. This was the rate used by the authors of the original WaveNet paper [1] as it provides a sufficiently high-quality audio while being simple enough to model.

5.2.1 8 kHz

A sampling rate of 8 kHz was tested as a minimum threshold as any further downsampling would cause excessive reduction to audio quality. 8 kHz sampling rate is used in telecommunications but is seldom used to broadcast music as it results in many omissions, however for these experiments it was considered as it allows generation of 2x longer waveforms.

The default model (5.0) was tested up to 110 epochs using 10,000 training examples from the MAESTRO dataset. Contrary to 16 kHz training examples, audios in the 8 kHz set are 6 seconds in length instead of 5. In practice this difference is subtle.

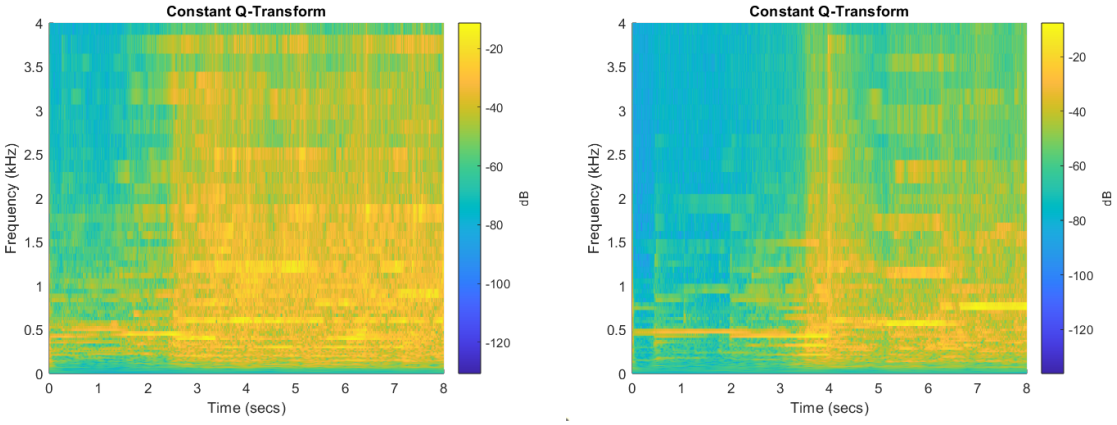
[9] https://github.com/LouisWinder/WaveNet/tree/main/Generated%20audios/Size/Large/Paper%20generations



(a) 0-30 epochs (b) 70-110 epochs

Figure 15: Loss plots on the 8 kHz MAESTRO dataset from 0-30 and 70-110 epochs.

For the 8 kHz audios the loss is notably higher than the regular 16 kHz dataset. After 30 epochs both training and validation losses got down to around 2.05 — an increase of more than 25% lower compared to the ∼1.5 obtained on the 16 kHz training data. This significant disparity between results is likely due to how 8 kHz audio is composed: less samples used to represent waveforms makes predicting with certainty difficult.



(a) Generation after 30 epochs (b) Generation after 110 epochs

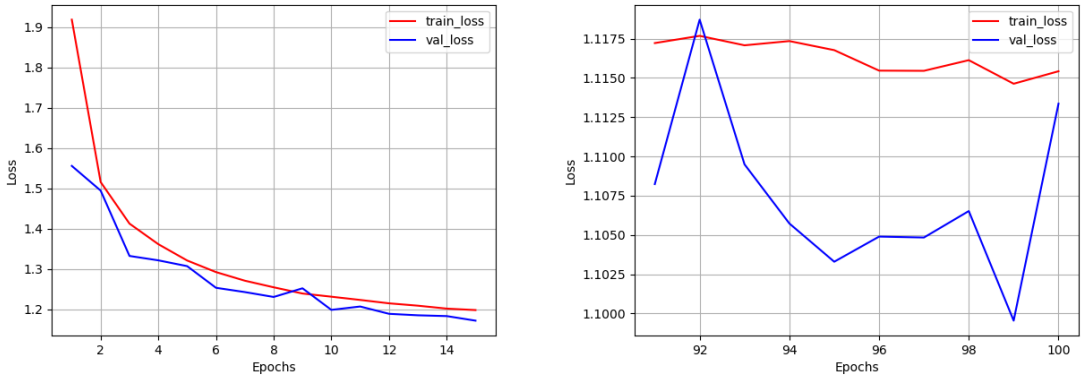
Figure 16: Constant-Q transforms of audios generated using a medium model trained on the 8 kHz MAESTRO dataset.

Generation using the model trained on the 8 kHz MAESTRO set produced fairly unsatisfactory results. While there did exist some tonal quality, the outputs severely lacked any sort of coherent structure or purpose. In fact, after multiple separate generations it can be observed that most of them sound very similar. This could be due to the fact the training loss remained at a high level and thus the model cannot predict well so ends up generating similar sounds. Generation after 110 epochs followed a similar pattern, albeit with a few less omissions[10].

The overall results from experiments with the 8 kHz set suggests heavily that it is not adequate for the task as, even with significant training, it fails to predict the next samples well enough which leads to noisy generation. This is somewhat of a shame as the lower the sample rate the longer we can generate audio without running into memory constraints so we would benefit from this kind of generation being feasible.

5.2.2 22.05 kHz

An upper bound sampling rate of 22.05 kHz was tested as any higher resulted in excessive memory use. 22.05 kHz is half the sample rate of the base rate audios in the training datasets take before they are downsampled. The default model (5.0) was tested up to 100 epochs using 10,000 training examples from the MAESTRO dataset.



(a) 0-15 epochs

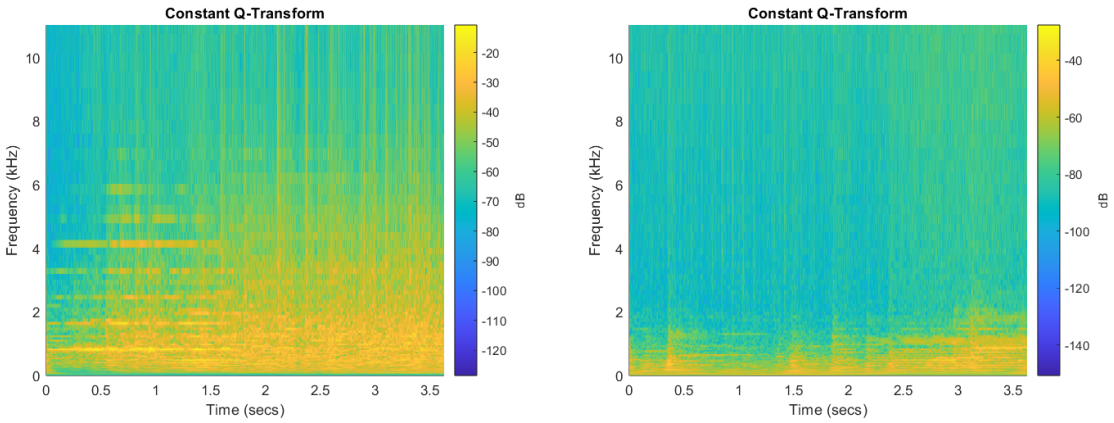
(b) 90-100 epochs

Figure 17: Loss plots on the 22.05 kHz MAESTRO dataset from 0-15 and 90-100 epochs.

As is visible from the graphs, training on the 22.05 kHz dataset is a fair amount more efficient. Loss gets down to around 1.2 in only 15 epochs which, compared to the best-performing model thus far, is over 0.2 lower. The fact that more samples are being used to represent each training waveform evidently results in a greater prediction ability by the model. It holds that this intuition is logical — a higher sample rate equals more closely correlated local structure which is being exploited by the model to produce better predictions. From 90-100 epochs the training has slowed down significantly to the point where practically no learning is being done. Perhaps this is the lowest point one can

[10] https://github.com/LouisWinder/WaveNet/tree/main/Generated%20audios/Different%20sampling%20rates/8kHz/Paper%20generations

obtain in regards to the prediction accuracy of the model. In reality this loss should be adequate for the purposes of producing reasonably coherent audio waveforms.



(a) Generation after 15 epochs (b) Generation after 100 epochs

Figure 18: Constant-Q transforms of audios generated using a medium model trained on the 22.05 kHz MAESTRO dataset.

Even though the loss after 15 epochs is as low as 1.2, generation at 15 epochs is not very convincing and relatively noisy. Generation after 100 epochs is a lot more coherent and less noisy but is observed to be lacking in dynamic range. The samples tended to be biased towards low-frequency representations, with some struggling to produce any meaningful tones[11].

5.3 Amount vs length of training audios

For training up to now, models have been trained on datasets that each have around 10,000 training examples to learn from. This gives them a wide variety of waveform structures which facilitates well the learning of the conditional probabilities that make them up. What if, however, we were to have a more restricted dataset? Would the model still be able to generate well, and what effect does the amount of data we use to train the model affect its generation? In addition, our training audios thus far have been no more than 6 seconds in length, primarily done to allow sustainable memory usage. Here we will additionally test on datasets containing audios of 10 seconds in length to see if providing a longer input sequence to the model allows it to better predict and generate.

5.3.1 1000 training examples

In order to test the first question, a model will be trained using only 1000 examples and, to make the test fair, for up to 100 epochs as done with the other tests in this section. The default model (5.0) was tested up to 100 epochs using 1000 training examples from the MAESTRO dataset.

[11] https://github.com/LouisWinder/WaveNet/tree/main/Generated%20audios/Different%20sampling%20rates/22.05kHz/Paper%20generations

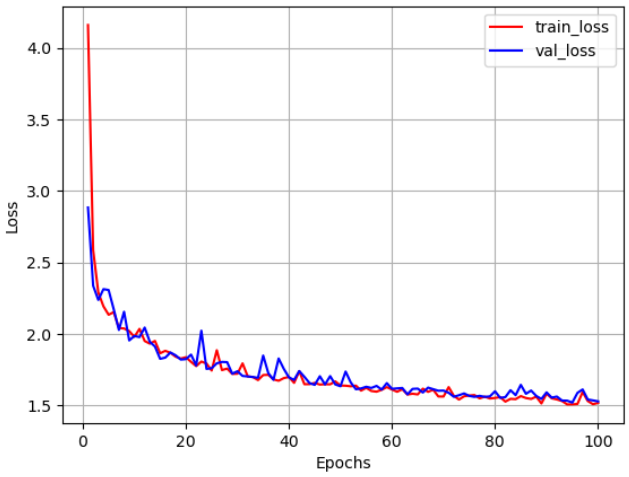
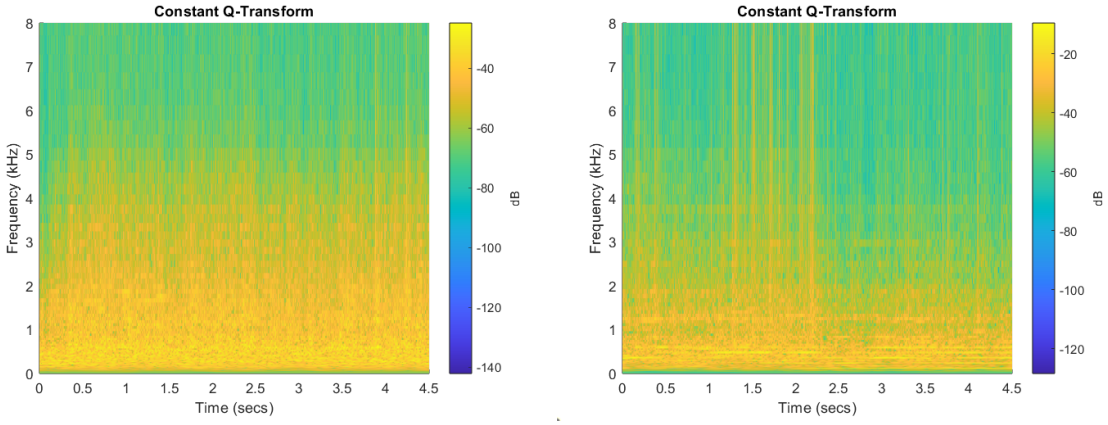


Figure 19: Loss plot on the 1000-example MAESTRO dataset from 0-100 epochs.

Training and validation losses follow a similar path, both flattening out towards 100 epochs at a loss of 1.5. In theory we should expect to see a lower loss compared to the tests with 10,000 training examples as the model has less to try and remember, but this does not seem to be the case. It appears the model struggles to get past certain thresholds to the point it cannot learn any longer. One may think that with loss values this low the generation will be satisfactory, however the loss values here are deceiving — it is only when generating with the model that the drawbacks of few training examples is truly evidenced.



(a) Generation after 25 epochs

(b) Generation after 100 epochs

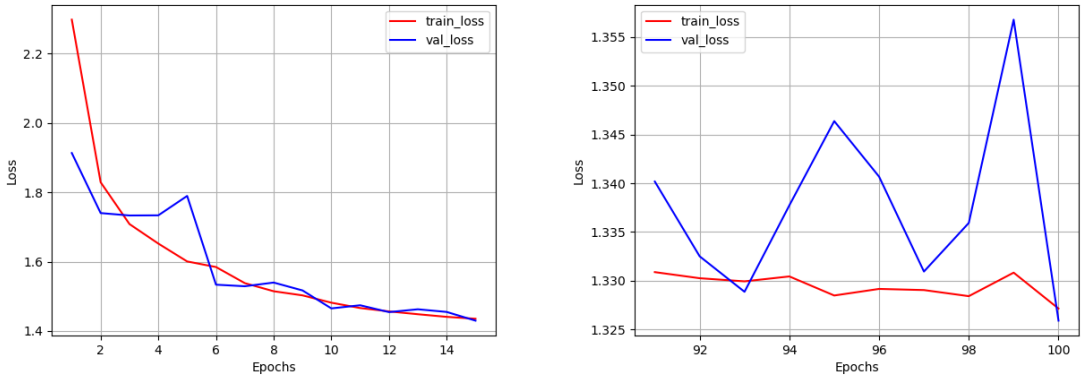
Figure 20: Constant-Q transforms of audios generated using a medium model trained on the 1000-example MAESTRO dataset.

As expected, the generated audios for the 1000-example dataset were extremely noisy. Even after 100 epochs of training the results were not pleasing, which suggests that in

order to train a WaveNet model efficiently we need a lot of training examples[12]. As audio waveforms have an extremely high degree of diversity, it makes sense for the model to struggle to predict the next samples with high accuracy as there are many different possible forms each one can take. Hence, a wide variety of training examples are required for the model to be able to generate convincingly.

5.3.2 10 second training examples

To test the second question of whether or not training on longer audio samples gives better generation, a model will be trained on a set where all samples are of 10 seconds in length. The default model (5.0) was tested up to 100 epochs using 8000 training examples from the MAESTRO dataset. Due to the high memory usage that comes from using 10-second training samples, the model was only trained on 8000 examples instead of the desired 10,000.



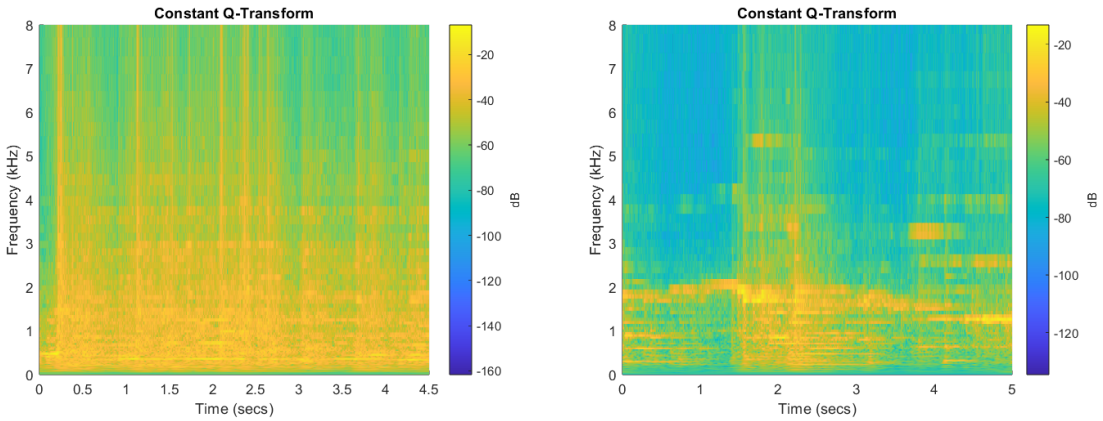
(a) 0-15 epochs

(b) 90-100 epochs

Figure 21: Loss plots on the 10-second example MAESTRO dataset from 0-15 and 90-100 epochs.

Again, here there is little difference in how the network is learning compared to other model configurations; loss follows a similar downward trend. However, after a short while the loss reduces at a slower rate compared to other models. At around 90 epochs the model appears to stop learning completely which suggests there is no point in training for longer using this specific model.

[12] https://github.com/LouisWinder/WaveNet/tree/main/Generated%20audios/Amount%20vs%20length/Amount/Paper%20generations



(a) Generation after 15 epochs

(b) Generation after 100 epochs

Figure 22: Constant-Q transforms of audios generated using a medium model trained on the 10-second example MAESTRO dataset.

Generation using a model trained on 10-second audio samples produced results on par with the best generations thus far. After only 15 epochs the results were relatively noisy but nonetheless had apparent tonal quality, even if many sounded similar. After 100 epochs of training the generated audios were of high quality with little audible noise present. Occasionally, the generations would contain divergent undertones that made the audio less natural sounding (for example, individual piano notes would stay at the same volume for long periods of time which wouldn't occur on a real instrument). Overall however, the resulting generations were pleasant and unique[13].

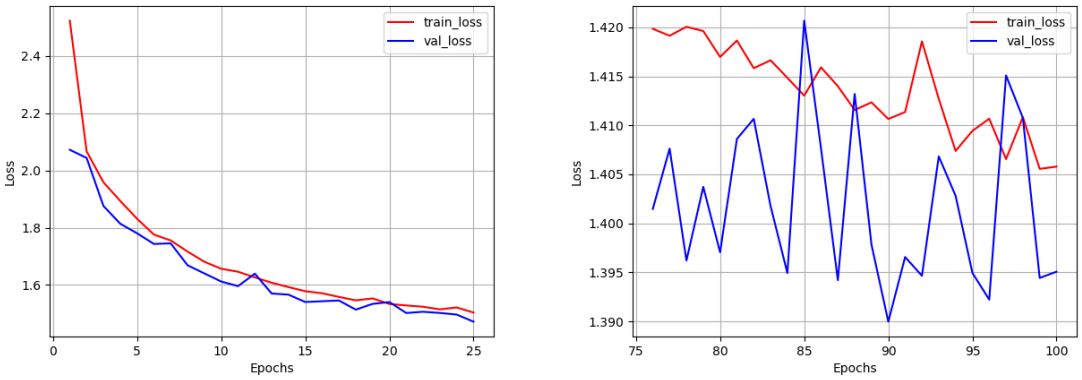
5.4 MusicNet dataset

For this section, the model will be tested solely on a different dataset: the MusicNet dataset [17]. Up to now, testing with the model has been performed using only the MAESTRO dataset. The primary reason for this is because there is an observable lack of datasets deemed suitable for the task. Compared to other mediums (for example images) there are few high-quality classical recording datasets that allow for scalable modelling with a system like WaveNet which requires a lot of training examples to generate well.

MAESTRO is a very large dataset with audios that are typically virtuosic in nature, meaning they contain a lot of different note frequencies in short spaces of time. A drawback of this is that the sporadic changes in local structure can make the waveforms unpredictable. Furthermore, virtuosic passages often have sections that are noisy and contain lots of omissions. It is for these reasons that the model could struggle to predict with high accuracy, warranting the testing of a similar but more restrained dataset. The MusicNet dataset has, like MAESTRO, been restricted to only pieces that are solo piano, making it considerably simpler to model.

[13] https://github.com/LouisWinder/WaveNet/tree/main/Generated%20audios/Amount%20vs%20length/Length/Paper%20generations

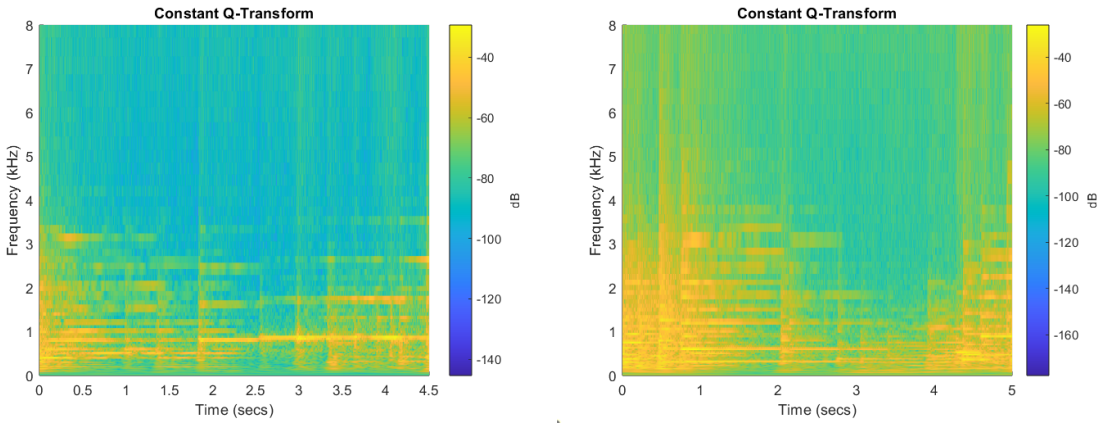
The default model (5.0) was tested for up to 100 epochs using 9,000 training examples solely from the MusicNet dataset.



(a) 0-25 epochs (b) 75-100 epochs

Figure 23: Loss plots on the MusicNet dataset from 0-25 and 75-100 epochs.

As expected the loss graphs follow a similar pattern as on the MAESTRO dataset, however even after 25 epochs both training and validation losses are approximately 0.1 higher than for the same model tested on MAESTRO. Nevertheless, the training process appears to be operating at a similar rate.



(a) Generation after 25 epochs (b) Generation after 100 epochs

Figure 24: Constant-Q transforms of audios generated using a medium model trained on the MusicNet dataset.

Generation using a model trained on the MusicNet dataset generally performed better than the MAESTRO dataset. They often audibly appear much less noisy than the samples generated by models operating on the MAESTRO set[14]. A reason for this is

[14] https://github.com/LouisWinder/WaveNet/tree/main/Generated%20audios/MusicNet/Paper%20generations

likely due to the differing nature of the audios in each dataset. MusicNet compared to MAESTRO's pieces include a lot less notes overall and have a much lower dynamic range which results in generation that is comparatively subdued.

5.5 Analysis/evaluation

As music is hard to objectively analyse, for all generations performed, an objective/subjective scoring system will be devised to rank each one based on how well the specific model that was used to generate it performed. This scoring system will be composed of 2 parameters. The first (objective) benchmark is a mathematical representation of the amount of "noise" in the generated signal, named its signal-to-noise ratio (SNR). This gives a quantitative measure of the undesirable sound in the signals and is one important instrument in determining the quality of generated waveforms. The second benchmark that will be used is a subjective score of how "musical" each generation sounds. This score will be on a scale of 1-5, 1 being not at all musical and 5 being extremely compelling. The score for each will be chosen based on a variety of factors such as:

- How realistic-sounding is the audio?
- How pleasing is the audio to listen to?
- How well does the generation model the training data?

The entire subjective scale will be as follows:

- **1:** Not at all musical
- **2:** Some tonal elements audible
- **3:** Coherent, but not musical
- **4:** Coherently musical
- **5:** Extremely musically compelling

To maintain integrity, generation will only be analysed on the ∼100 epoch cases that are outlined in the CQTs from above tests. Signal-to-noise ratio will be calculated using the MATLAB programming language[15]. MATLAB provides a range of tools for signal analysis, one being calculation of signal-to-noise ratio using the function *snr(x)*[16] where x is the signal to be analysed. The lower the value, the better the generation. Table 2 outlines the signal-to-noise ratio scores for each model and Table 3 shows the subjective scores for each generation.

[15] https://uk.mathworks.com/products/matlab.html
[16] https://uk.mathworks.com/help/signal/ref/snr.html

Model	Signal-to-noise ratio
Small	-8.4598
Medium	0.6387
Large	8.4122
8 kHz	-2.1622
22.05 kHz	-9.2349
1000 examples	-2.5453
10-second examples	-9.8975
MusicNet	-1.9210

Table 2: Signal-to-noise ratios for generation using different models.

Model	Subjective Score
Small	3
Medium	4
Large	2
8 kHz	2
22.05 kHz	3
1000 examples	1
10-second examples	4
MusicNet	4

Table 3: Subjective scoring for generation using different models.

Interestingly, it was identified that although generation with a model sounded pleasant and musical, it did not necessarily exhibit a low signal-to-noise ratio. For example, generations using the medium and MusicNet models generally produced audios that sounded pleasant however they had relatively high SNR values compared to other models. This suggests models that are good at capturing the underlying patterns in the training data do not necessarily provide the highest quality generation. In practice we would prefer the former over the latter as it would be easier to employ quality-enhancing methods post-generation to already-adequate musical fragments.

Overall however, subjectivity should still remain the primary evaluation method when reviewing creative forms such as music. Signal analysis methods should not be ignored as they do provide objective value to us by indicating the overarching generational ability of models, however we as humans evaluating subjective art forms have a responsibility to draw on our unique appreciation of the arts to give more weight to the music that sounds good to us personally.

Chapter 6

Conclusion

This paper has shown the potential of the WaveNet model to produce compelling, unique musical samples by modelling in the raw audio domain. We have demonstrated the ability of the model to represent high-quality audio up to 22,050 samples per second by operating on conditional probabilities of samples, while ensuring the time for training is low through the ability to parallelise the process. We have shown how different parameters and configuration of the network affects the resulting generation, highlighting settings that cause output to be unsatisfactory. Both signal analysis techniques and subjective analysis have been utilised to evaluate generation which are indicative of the performance of different models and their data.

Although not utilised in this paper, since WaveNet's introduction in [1] multiple enhancements have been made that improve the network's performance greatly. Arguably the most notable of these is Parallel WaveNet [19] which can generate over 3 orders of magnitude faster than the traditional architecture by utilising a pre-trained "teacher" model from which a separate "student" network samples. This provides a considerable boost in speed which has facilitated its deployment for real-time systems at Google such as text-to-speech for Google Assistant[17].

A big drawback of the default WaveNet model is the fact that, due to memory and time constraints, it can only generate samples of a few seconds in length. Future improvements should focus on finding solutions to this problem so that generation can be more critically analysed.

The foundations of this work provide a clear benchmark upon which further research can expand. WaveNet has been shown to perform much better when it is trained on a larger pool of data, therefore future experiments have much scope to build upon the somewhat limited tests conducted here. The model's ability to learn from time data in the raw audio domain means its ability does not have to just be restricted to classical piano music as done in this paper. It would be interesting to see what generation on other genres, instruments or voices looks like.

[17] https://deepmind.google/technologies/wavenet/

Bibliography

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

[2] David Cope. *Experiments in musical intelligence*. A-R Ed., 1996.

[3] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 33(12):6999–7019, 2021.

[4] Eric W Weisstein. Convolution. *https://mathworld. wolfram. com/*, 2003.

[5] David H Hubel and Torsten N Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148(3):574, 1959.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[7] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[8] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, 9:611–629, 2018.

[9] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

[10] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems*, 29, 2016.

[11] Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pages 1747–1756. PMLR, 2016.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] Douglas Eck and Juergen Schmidhuber. A first look at music composition using lstm recurrent neural networks. *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale*, 103(4):48–56, 2002.

[14] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[15] CCITT Recommendation. Pulse code modulation (pcm) of voice frequencies. In *ITU*, 1988.

[16] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *International Conference on Learning Representations*, 2019.

[17] John Thickstun, Zaid Harchaoui, and Sham Kakade. Learning features of music from scratch. *arXiv preprint arXiv:1611.09827*, 2016.

[18] Judith C Brown. Calculation of a constant q spectral transform. *The Journal of the Acoustical Society of America*, 89(1):425–434, 1991.

[19] Aaron Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George Driessche, Edward Lockhart, Luis Cobo, Florian Stimberg, et al. Parallel wavenet: Fast high-fidelity speech synthesis. In *International conference on machine learning*, pages 3918–3926. PMLR, 2018.

[20] Eva Tuba, Nebojša Bačanin, Ivana Strumberger, and Milan Tuba. Convolutional neural networks hyperparameters tuning. In *Artificial intelligence: theory and applications*, pages 65–84. Springer, 2021.

[21] Şaban Öztürk, Umut Özkaya, Bayram Akdemir, and Levent Seyfi. Convolution kernel size effect on convolutional neural network in histopathological image processing applications. In *2018 International Symposium on Fundamentals of Electrical Engineering (ISFEE)*, pages 1–5, 2018.

[22] Anh-Duc Nguyen, Seonghwa Choi, Woojae Kim, Sewoong Ahn, Jinwoo Kim, and Sanghoon Lee. Distribution padding in convolutional neural networks. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 4275–4279, 2019.

[23] Tom Le Paine, Pooya Khorrami, Shiyu Chang, Yang Zhang, Prajit Ramachandran, Mark A Hasegawa-Johnson, and Thomas S Huang. Fast wavenet generation algorithm. *arXiv preprint arXiv:1611.09482*, 2016.

[24] Jonathan Boilard, Philippe Gournay, and R. Lefebvre. A literature review of wavenet: Theory, application and optimization. 03 2019.

Appendix A

Audio representation background

Digital audio waveforms are represented by a discrete set of samples, each of which having an amplitude which describes its relative intensity (or volume). The digital signal's x-axis represents time whilst its y-axis is commonly a logarithmic scale of decibels (dB).

The sampling rate of a waveform refers to the number of samples per second of audio, usually displayed in Hz. For example, CDs use a sampling rate of 44,100 Hz, or 44.1 kHz. The higher the sample rate, the higher the quality of the sound, as more samples are being used to represent the signal.

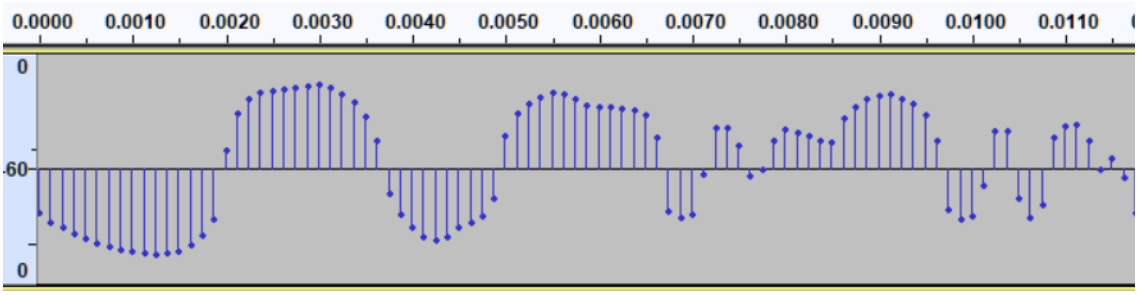


Figure 25: An example waveform using sample rate of 8 kHz, meaning there are 8000 samples per second of audio.