Transformer and LSTM Models for Automatic Counterpoint Generation using Raw Audio

Lars Ødegaard Bentsen*
Department of Technology Systems
University of Oslo
lars.bentsen@its.uio.no

Riccardo Simionato*
Department of Musicology
University of Oslo
riccardo.simionato@imv.uio.no

Benedikte Wallace
Department of Informatics
University of Oslo
benediwa@ifi.uio.no

Michael Krzyzaniak
Department of Informatics
University of Oslo
michakrz@ifi.uio.no

ABSTRACT

A study investigating Transformer and LSTM models applied to raw audio for automatic generation of counterpoint was conducted. In particular, the models learned to generate missing voices from an input melody, using a collection of raw audio waveforms of various pieces of Bach's work, played on different instruments. The research demonstrated the efficacy and behaviour of the two deep learning (DL) architectures when applied to raw audio data, which are typically characterised by much longer sequences than symbolic music representations, such as MIDI. Currently, the LSTM model has been the quintessential DL model for sequence-based tasks, such as generative audio models, but the research conducted in this study shows that the Transformer model can achieve competitive results on a fairly complex raw audio task. The research therefore aims to spark further research and investigation into how Transformer models can be used for applications typically dominated by recurrent neural networks (RNN). In general, both models yielded excellent results and generated sequences with temporal patterns similar to the input targets for songs that were not present in the training data, as well as for a sample taken from a completely different dataset.

1. INTRODUCTION

Automatic counterpoint generation tasks aim to generate harmonically interdependent melodies added above or below a given melody. A melody can be considered as a group of notes played in sequence, one after the other, while a harmony is a group of simultaneous notes, played in the background and around the melody. Counterpoint is multiple concurrent melodies that follow a set of melodic rules, with respect to the sequential notes in each melody and all of the simultaneous notes in all of the melodies at each moment. Therefore, a model should generate a stylistically plausible counterpoint in order to create a single harmonic texture supporting a given melody. Such a problem can be considered as a branch of algorithmic musical composition, which has most commonly been used to support human creativity and develop tools to explore musical ideas, such as melodic or harmonic motifs.

Algorithmic musical composition have primarily been explored using symbolic music representations. In this paper we instead aim to explore the domain of algorithmic music composition by focusing on learning harmonic relations and dependencies from raw audio data. Recently, DL-based models have successfully been able to learn directly from raw audio [1]. As for most sequence-based applications, Convolutional (CNN) and Recurrent Neural Networks (RNN) have been predominant. However, a drawback of these methods is that they struggle to learn temporal dependencies across many time-steps, due to the limitations of CNNs' receptive fields and RNNs' internal state. WaveNet- and Long short-term memory (LSTM)-based models have been proposed to combat this impediment of CNNs and RNNs, respectively. WaveNet is a fully probabilistic and autoregressive model, which can be regarded as an extension of PixelCNNs [2], where the conditional probability distribution is modelled by a stack of convolutional layers. Dilated causal convolution is used, where dilation increases the model's receptive field by skipping input values with a certain step, and the causality ensures that future context is not taken into account when making a prediction. Several alterations of the WaveNet model have been explored for different audio applications, such as speech denoising [3], instrument conversion [4], vocoder [5], frequency estimation [6] and virtual modelling [7–9].

Despite the success of WaveNet, RNN-based models still seem the preferred choice for many audio applications, such as for virtual analog tasks [10, 11], where these models have even managed to meet tough real-time constraints [12, 13]. Furthermore, the accuracy of RNN-based architectures have often proved better than WaveNet-based models for black-box modelling of nonlinear audio systems, while requiring significantly less processing power to run [12]. A hybrid model, which used a combination of

* Equal contribution. Listing order is random.

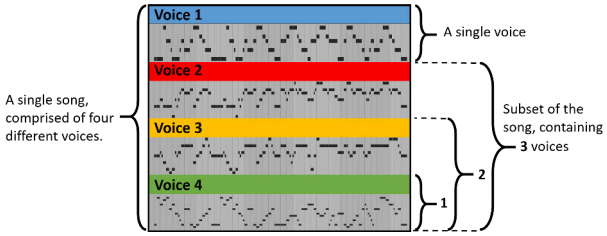

Figure 1: Example of MIDI representation of the four voices that comprise a single song.

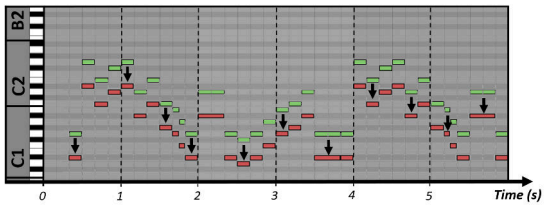

Figure 2: Example of pitch transposition for a single voice.

Input	Target
$[0,0,0,1]$	$[1,0,0,0]$
$[0,0,1,0]$	$[1,0,0,0]$
$[0,0,1,1]$	$[1,0,0,0]$
$[0,1,0,0]$	$[1,0,0,0]$
$[0,1,0,1]$	$[1,0,0,0]$
$[0,1,1,0]$	$[1,0,0,0]$
$[0,1,1,1]$	$[1,0,0,0]$
$[0,0,0,1]$	$[0,1,0,0]$
$[0,0,1,0]$	$[0,1,0,0]$
$[0,0,1,1]$	$[0,1,0,0]$
$[1,0,0,0]$	$[0,1,0,0]$
$[1,0,0,1]$	$[0,1,0,0]$
$[1,0,1,1]$	$[0,1,0,0]$
$[0,0,0,1]$	$[0,1,0,0]$
$(\cdots)$	$(\cdots)$
$[1,1,1,0]$	$[0,0,0,1]$

Table 1: Example showing the construction of the inputs and targets for a single song in binary encoding.

different DL architectures, was investigated in [14], where the model was divided into three parts; an adaptive front-end, latent-space and synthesis back-end. An important motivation behind the hybrid architecture, was to present a general-purpose DL framework for modelling audio effects.

After the Transformer was first introduced [15], it has outperformed many state-of-the-art architectures for a plethora of sequence-based applications, becoming increasingly popular after the release of the GPT-3 [16] and BERT [17] models. The success of these models for NLP tasks have later sparked research into the application of these models for various visual tasks, such as for image generation [18], image recognition [19–22], object detection [23], and segmentation [24, 25]. Transformer architectures have also started to make their way into the audio domain [26,27]. Different to RNNs, the Transformer relies on attention mechanisms to capture global context, instead of a recurrent unit with memory, making it potentially easier for the Transformer to capture temporal dependencies across much longer sequences than RNN-based models. This aspect is particularly important for music, as patterns can repeat or reappear after many time-steps, such as the chorus.

Considering the musical domain, Transformers have mainly focused on symbolic representations [26, 28], with only a few studies considering raw audio [29–31]. Li et al. [29] did not look at music, but introduced the Transformer TTS network for raw audio speech synthesis, which yielded unprecedented results, with mean opinion scores of 4.39, compared to 4.44 for real recordings. In the case of music, the analysis is typically more challenging than speech synthesis, as the latter usually considers a single speaker, while music signals are made up of multiple sources. Instruments like the piano are polyphonic, producing multiple pitches simultaneously, making the domain of music increasingly complex [32]. Child et al. [31] proposed the Sparse Transformer, which was able to model sequences of tens of thousands of time-steps, and showed impressive results generating raw audio classical music pieces of sequence lengths up to 65,536. Verma et al. [30] developed an auto-regressive and causal Transformer model for audio synthesis, using piano recordings from YouTube, containing both monophonic and polyphonic sounds. The inputs were waveform amplitudes, discretised in $0-255$, and the model aimed to predict the next value, one time-step ahead. They showed that the Transformer could be successfully applied to raw audio synthesis and outperform WaveNet-style models for certain tasks [30]. Aside from these studies, the literature is scarce with regards to the application of Transformers using raw audio data, and further research is therefore required to thoroughly explore the potential of such architectures, especially focusing on comparing against current state-of-the art.

The aim of this study was to generate a counterpoint for a given melody, using raw audio. With reference to Fig. 1, this meant generating a single missing voice, from a song containing any subset of the other three voices, shown here using symbolic (MIDI) representation for readability. The performance of the Transformer was investigated when applied to raw audio and compared against an LSTM model, which represents the current state-of-the-art. By studying a slightly more complex application than that in [30], this study aimed to further investigate the potency of the Transformer for use in raw-audio applications, which will be important to establish the architecture within the musical domain. Generally, the study investigated whether the Transformer could be successfully deployed and more extensively used for sequence modelling applications within music, which are different to the NLP domain.

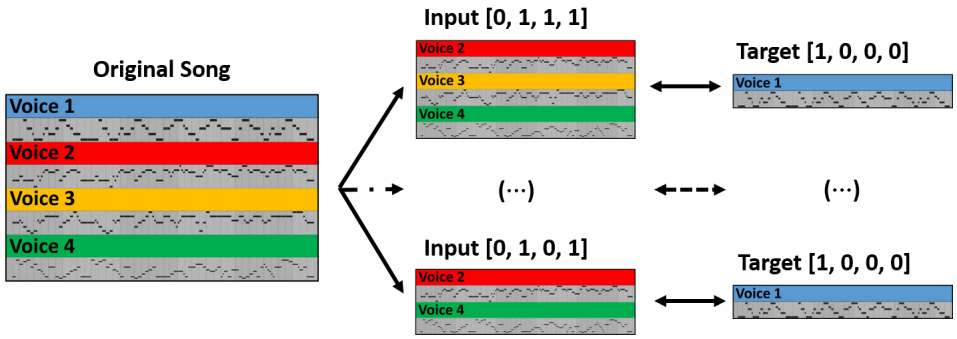

Figure 3: Illustration of the splitting of a song into inputs and targets, where the binary representation of inputs and targets are with reference to Table 1.

2. DATASET

The dataset used for this study was based on 5 different songs from some of Bach's work, each one composed of 4 voices in total. First, voices were transcribed in a MIDI format. To virtually increase the size of the dataset, the pitch was transposed up 5 semitones and down 6 semitones from the original, resulting in $(11+1)$ audio files for each song. The transposing of a single song, down 4 semitones, is illustrated in Fig. 2 and each of the 12 variations of a song therefore had different key roots. To produce the new representation of a song, all four voices (as depicted in Fig. 1) were transposed together, in the same direction and with the same magnitude. Instead of the 5 original songs, the new dataset now consisted of 60 different MIDI files.

The aim for this study was to propose a model that could predict a missing voice, given an input containing any subset of the remaining three voices in a song. Table 1 demonstrates the splitting of a single song into the corresponding inputs and targets using a binary representation, also illustrated in Fig. 3. Here, $[0, 0, 0, 1]$ indicates an audio file for a song, in which only voice 4 is present, $[0, 1, 1, 0]$, a file containing voices 2 and 3, and so on. Each possible combination was therefore extrapolated before rendering the raw audio files. This resulted in 15 combinations for each song, and 900 MIDI files in total. Files containing 4 voices were not considered, as the aim was to generate a missing voice from an input containing $1-3$ of the remaining voices, hence we obtain $16-1=15$ combinations for each song. Finally, the MIDI files were rendered to raw audio files using different instruments.

2.1 Data Preparation

The principal drawback of the Transformer, is that the complexity of its attention operations scales quadratically with sequence length. This impose significant computational and memory constraints, which typically limits the feasible sequence lengths that can be analysed. Because of the sequential architecture of an LSTM model, all relevant past information has to be stored in a single memory vector, resulting in these models struggling to capture temporal characteristics over a very large number of time-steps. LSTM networks could in theory be applied to arbitrarily long sequences, but because of the sequential architecture, there is typically an upper limit to the maximum feasible sequence length, as for the Transformer. Furthermore, the sequential architecture also means that recurrent models become slow for longer sequences. Since audio is sampled at tens of thousands of Hertz, the resulting raw audio sequences can be very long, even for short snippets of recording. For this particular study, each song was around 0.5 - 1.0 minutes long, which meant that, with a sampling rate of 44.1 kHz, each raw audio file would be represented by a sequence of around $1.3M-2.6M$ values. Considerable effort was therefore spent in reducing input lengths to be able to use the Transformer and for the LSTM to be able to learn long-term temporal dependencies.

First, raw audio files were downsampled to 2940 Hz. By downsampling, the spectrum range was narrowed to a Nyquist frequency of 1470 Hz. Since notes present in the dataset were not higher than *DO6* (*C6*), i.e. 1046 Hz, all fundamental frequencies were still present in the new spectrum, while a large part of the overtones were discarded. This was decided as a trade-off between computational complexity, sequence lengths and frequency resolution.

For the task of counterpoint generation, spectral information, and in particular pitch information, was deemed the most important for the networks to learn. As a result, it was decided to use short-time Fourier transform (STFT), with a window length of 200 ms, to produce spectrograms from the raw audio files. Phase information was neglected and only magnitudes from the frequency spectra were taken.

The data was scaled in $(0, 1)$, using a standard Min-Max Scaler and split into test, train and validation sets. All files corresponding to a single, randomly chosen, song were used for testing and another for validation, while the files for the remaining three songs comprised the training set. Since there was no overlap between the test, train and validation sets, the aim was to improve regularisation by not allowing the models to overfit to the particular songs present in the training data.

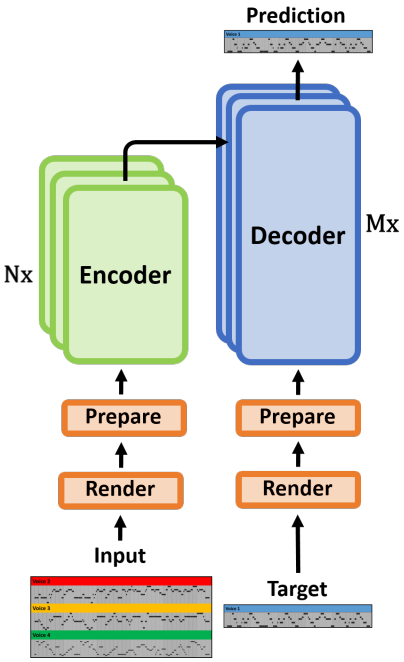

Figure 4: Proposed method for a generic encoder-decoder model, here taken as either an LSTM or Transformer model. *Render* refers to creating the raw audio files from MIDI representation. Even though the final output here are shown as MIDI, this was just for clarity, while the actual output was a raw audio file. Rendering, downsampling and STFT was done prior to training.

3. METHODS

3.1 Transformer

The Transformer model should take an input sequence, here a spectrogram, $x \in \mathbb{R}^{T \times d}$, and predict a missing melody, $y \in \mathbb{R}^{T \times d}$, where T is the number of time-steps and d the number of features (i.e. frequencies).

3.1.1 Encoder

The original Transformer architecture follows an encoder-decoder framework, as illustrated for a generic model in Fig. 4. The encoder is comprised of a multi-head attention mechanism and a traditional feed-forward neural network, as well as two residual connections and layer normalisations, as in the original formulation [15]. The input to the encoder is a sequence, which is added some positional encoding in order to capture temporal dependencies in the input. As in the original formulation of the Transformer, it was decided to use sine and cosine functions of different frequencies to represent the positional encoding.

The multi-head attention block in the encoder layer uses full self-attention, meaning that each attention operation would attend to the full input sequence. Multi-head attention, which computes multiple attention weights for updating each input, has also been found useful, as heads can learn to focus on different aspects of the input. Outputs from all heads are concatenated and passed through a learned linear transformation layer to produce the output.

The outputs from the multi-head attention block are added with a residual connection and applied layer normalisation, before being fed into a multilayer perceptron (MLP), typically with two hidden layers. The MLP is applied identically to all the inputs, individually. As for the multi-head attention, a residual connection and layer normalisation are applied to the outputs from the MLP, to produce the final outputs from the encoder layer. Finally, multiple encoding layers are stacked to introduce depth to the model, as shown in Fig. 4. The layers typically follow the same architecture, but each with different learnable weights, optimised through normal back propagation.

3.1.2 Decoder

The inputs to the decoder are the shifted outputs. First, the decoder employs an additional masked multi-head attention block, where the outputs are masked, so that the decoder only has access to $y_0, y_1, ..., y_{t-1}$ to make a prediction at time t, $\hat{y}_t$. The decoder then employs a multi-head attention block and MLP network in exactly the same manner as for the encoder. However, the inputs to compute the values and keys are now from the outputs of the final encoder layer, while the queries are from the outputs of the previous attention block in the decoding layer. This ensures that the model does not have access to outputs for future time-steps. As for the encoder, stacked decoding layers add depth to the model. For more details on the Transformer model and its implementation, the authors refer the reader to [15].

3.2 LSTM

Long short-term memory (LSTM) units were first introduced by Hochreiter et al. [33], in order to solve some of the shortcomings of vanilla RNNs when modelling long sequences. LSTMs partially solve the vanishing gradient problem [34] of RNNs, by employing a gated recurrent unit with skip-connections to allow gradients to flow across many time-steps. An LSTM unit consists of a cell, an input, output and forget gate. The cell stores values across multiple time-steps, acting as a memory, and the gates regulate the flow of information into and out of the cell. The forget gate controls what information in the cell state to forget, the input controls which new information will be encoded into the cell state, while the output controls what information encoded in the cell state is being outputted.

The LSTM model was also implemented in an encoder-decoder fashion, as shown in Fig. 4, where both the encoder and decoder could consist of multiple, stacked, LSTM layers. The encoder processes the input sequence and returns its final internal state and output. These values are then used as a sort of conditioning for the decoder, where the final state vector and output from the encoder set the first internal state of the decoder. In essence, the decoder therefore learns to predict the target at time-step t, given all past values of the target, $[y_0, y_1, ..., y_{t-1}]$, and conditioned on the input sequence. As for the Transformer, the decoder was trained to predict the target signal at the next time-step, given the previous outputs.

Model	NumParams	Architecture		LR	BS
Transformer	1,191,553	NumLayers:	3	0.001	16
		DimModel:	128		
		HiddenLayer:	256		
		NumHeads:	4		
		Dropout:	0.1		
LSTM	444,417	EncoderLayers:	1	0.001	16
		DecoderLayers:	1		
		EncoderUnits:	64		
		DecoderUnits:	64		
		OutputUnits:	256		
		Dropout:	0.1		

Table 2: Model configurations after hyperparamter tuning. LR and BS refers to learning rate and batch size.

Model	MSE	MAE
Transformer	$1.0404 \pm 0.003\text{e}{-5}$	$7.6733 \pm 0.2410\text{e}{-4}$
LSTM	$1.0388 \pm 0.004\text{e}{-5}$	$7.9989 \pm 0.5274\text{e}{-4}$

Table 3: Prediction Losses on the test dataset. Results are the mean taken from five different training iterations, along with the corresponding standard deviations.

3.3 Hyperparameter Tuning

To arrive at appropriate configurations for the models, a tuning process was conducted in Optuna, a framework for automated search of optimal hyperparameters [35]. The final architectures are summarised for both the LSTM and Transformer models in Table 2, where LR and BS refers to the learning rate and batch size, respectively. In addition to the parameters shown in the table, both the Stochastic Gradient Descent (SGD) and Adam optimisers were tested for, with the latter yielding the best results. The tuning was conducted in two stages. First, the models were trained to 500 epochs, before the search space was narrowed and models trained to 800 epochs. The final models were trained to 5000 epochs, when validation losses had properly converged for both models.

4. RESULTS & DISCUSSION

4.1 Prediction Losses

Both models were trained to minimise the Mean Squared Error (MSE), but also computing the Mean Absolute Error (MAE), given by the following equations:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_y)^2 \tag{1}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_y), \tag{2}$$

where n are the number of samples, y denotes the true labels (target) and $\hat{y}$ the predictions. Even though the two losses seem very similar, an important distinction is that the MSE metric will penalise large errors much more than the MAE metric. The authors also designed and tested various other loss functions, but this did not yield significant improvements for the final models and the MSE metric was therefore decided as the most appropriate loss function.

The final results for the LSTM and Transformer models when predicting on the test set are given in Table 3, as the mean and standard deviation from five different runs. Both models achieved excellent performance, with very small MSE and MAE values. The LSTM model yielded a slightly smaller MSE on the test set, while the Transformer performed better with regards to MAE. Nevertheless, due to the very small difference between the two models it was difficult to conclude on a better model out of the two. The fact that the Transformer achieved performance on-par with the LSTM model on a fairly complex raw-audio task, cements the hypothesis that Transformer-based architectures can be very competitive against the current state-of-the-art within sequence modelling. Having different models to choose from was thought desirable, as certain features of a particular architecture might be useful for the particular study or research problem.

4.2 Spectral Visualisation of the Results

Even though the MAE and MSE metrics were useful to evaluate the models, they are not very informative when trying to understand how good the models are at generating harmonies. The first two columns in Fig. 5 shows the generated spectrograms from the LSTM and Transformer models for two randomly selected test samples. The top row gives the true labels, i.e. what we wanted the models to predict, and the last two rows show the respective predictions. To listen to the different samples in Fig. 5, the raw audio files are also provided [1] . Looking at the first column, it was seen that both models generated samples that aligned closely with the target, as would be expected from the results in Table 3. The main melody was clearly captured by both models, as seen by the lower frequencies in the spectrograms. Interestingly, the models were also able to generate some of the, less pronounced, higher frequency harmonics. However, for the higher frequencies, the models tended to include more frequencies than were present in the target and it was more challenging to extract clear patterns for these. One reason for this drawback, might be that it was difficult for the models to learn the exact characteristics for frequencies with smaller amplitudes, as the models would not be penalised as heavily, with regards to MSE, for making wrong predictions for upper harmonics, compared to the main melody. Furthermore, the degree to which upper frequencies were present also varied significantly in the training samples, and it might be thought that the models slightly overfitted to the training data.

Moving to the second column of Fig. 5, the target seemed more challenging to predict than the previous sample, with much more frequencies present and generally lower amplitudes. Nevertheless, the models still performed well, capturing the main patterns present in the target, but seeming to include a few too many frequencies, some of which were not present in the target. The models also seemed slightly too decisive. In the target, it was seen that a number of frequencies close together were often present, while the Transformer and LSTM models generally seemed to only capture the main frequency in each group of frequencies.

[1] https://github.com/LarsBentsen/TransLSTM_RawAudio

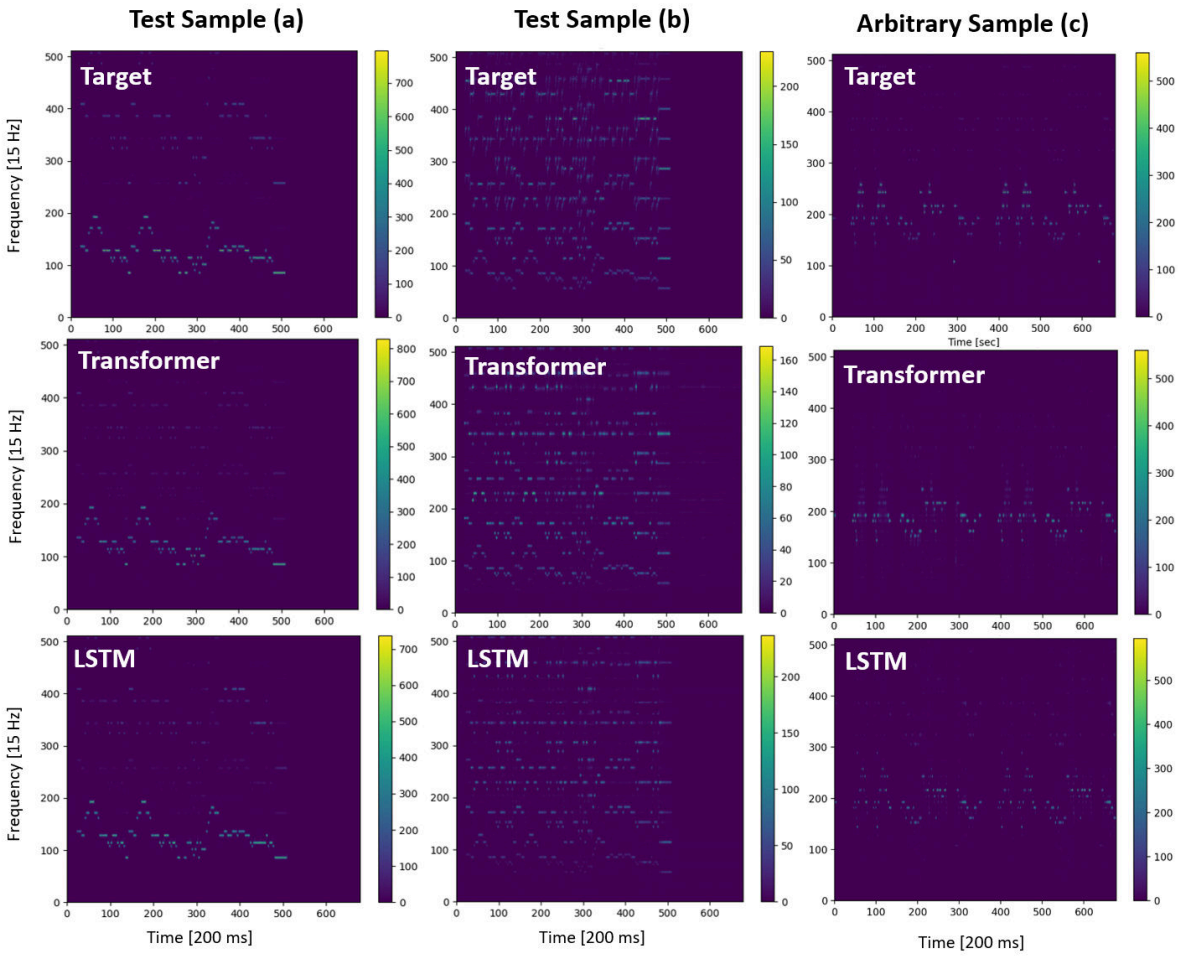

Figure 5: Spectral plots showing the predicted spectrograms and the corresponding targets (top row) for two randomly chosen samples from the test set and for one out-of-distribution sample (c), not taken from the original dataset. The plots are the raw output predictions from the Transformer and LSTM models, before performing inverse STFT and resampling, meaning that frequencies and times are given in 15 Hz and 200 ms resolutions, respectively.

Overall, it was seen through the spectrogram predictions and prediction losses that both models were well equipped to tackle the task at hand. Listening to the generated audio files, it was more challenging to distinguish between targets and predictions, than when considering spectrograms. Comparing the LSTM and Transformer models, the differences were minuscule, which proved the effectiveness of the Transformer for use on raw audio data.

The right hand column of Fig. 5, shows the Transformer and LSTM predictions for an arbitrary piece, which was not from the original dataset, but an extract taken from George Frideric Händel's Jubilate Deo (O be Jouyful in the Lord). It was seen that predictions from both models were slightly noisier than those for samples (a) and (b). A significant amount of higher frequency harmonics were predicted by the LSTM model, which were not present in the target. Nevertheless, both the LSTM and Transformer managed to predict the main melody reasonably well, indicating that the models were able to generalise to out-of-distribution samples, which further proved the prediction capabilities of both models.

For a given input, it is possible that a number of, completely different, melodies could follow the rules of counterpoint. Looking at the results in Fig. 5, it was seen that all predictions were very similar to the targets, even though ideal models should be able to generate a number of different melodies which fit the particular input. For this study, the models were not made autoregressive, but was conditioned on the previous, true, outputs to make the next predictions. Since the target spectrograms, in the top row of Fig. 5, were fed to the models with a look-ahead mask, it meant that the models were trained to output specific voices for a particular input. For future research, it will be particularly interesting to study ways in which the models could be made more creative, for which the authors of this paper think it would be important to make the models autoregressive and investigate ways in which the loss function could be altered. Since the MSE metric was used for

this study, the models would find a single melody for each input which minimised this loss. If instead, the loss function was founded on musical theory or more specific rules of counterpoint, it is thought that it might enable the models to generate more diverse sequences that can be different from the particular targets in the dataset.

Finally, the processing speed was estimated by running the models on a 2,3 GHz 8-Core Intel Core i9 processor. For a melody of 45.0 s, the Transformer computed all its predictions in 0.39 s, while the LSTM took 1.3 s. Furthermore, while the LSTM model took on average 3.5k epochs to converge, all Transformer models converged in < 50 epochs, significantly reducing training times.

For future work, the authors note a few additional areas of research thought particularly interesting. Since the complexity of the attention operations scale quadratically with sequence length, the model place some restrictions on the data that can be analysed. It would therefore be interesting to further investigate how the Transformer architecture could be adapted to reduce the complexity, such as the Longformer [36] and LogSparse Transformer [37], or other data preparation techniques which might be better equipped for extracting additional features in the raw audio samples. Further investigating architectures that facilitate the study of longer sequences is also thought to potentially improve the performance of Transformers over LSTMs. This is because the attention operation allows these models to attend directly to all previous time-steps for each computation, instead of storing all relevant information in a single memory array. The authors also believe that there is a significant potential with regards to the attention networks to improve the interpretability of the models. For instance, the Transformer relies on computing attention weights, which are analogous to how humans learn to focus on important aspects of a problem to arrive at a solution. By investigating a model's attention weights computed for a particular input, one could visualise which parts of the input the network learns to focus on. For music, this feature could be useful for a range of applications to better understand how the model learns and which parts of a sequence the model focuses on. Furthermore, the intuitive workings of the attention mechanisms are also thought interesting for allowing user inputs into a system to influence what to generate. However, this last point is not well studied and was merely thought as a potentially novel research direction for Transformer models in general.

5. CONCLUSION

In this study, the Transformer and LSTM models have been studied to generate counterpoint melodies from an input melody, using raw audio samples. Through analysis of the generated samples and prediction errors, it was shown that both models were well equipped for solving this problem. As a result, we argue that the Transformer model can be effectively deployed to musical applications, typically dominated by recurrent models. Due to the very long sequences when using raw audio samples, this study focused on downsampling and STFT to reduce the input lengths and create spectrograms. Furthermore, it was promising to see that both models could generalise well, producing appreciably accurate predictions for an out-of-distribution sample. For future work, it would be interesting to further investigate ways in which the Transformer and LSTM models could be altered to more effectively be used on raw audio data with very long sequences.

6. REFERENCES

[1] S. Dieleman and B. Schrauwen, "End-to-end learning for music audio," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 6964–6968.

[2] A. v. d. Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, "Conditional image generation with pixelcnn decoders," *arXiv preprint arXiv:1606.05328*, 2016.

[3] D. Rethage, J. Pons, and X. Serra, "A wavenet for speech denoising," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5069–5073.

[4] N. Mor, L. Wolf, A. Polyak, and Y. Taigman, "A universal music translation network," *arXiv preprint arXiv:1805.07848*, 2018.

[5] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan *et al.*, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4779–4783.

[6] P. Verma and R. W. Schafer, "Frequency estimation from waveforms using multi-layered neural networks." in *INTERSPEECH*, 2016, pp. 2165–2169.

[7] A. Wright, E.-P. Damskägg, L. Juvela, and V. Välimäki, "Real-time guitar amplifier emulation with deep learning," *Applied Sciences*, vol. 10, no. 3, p. 766, 2020.

[8] M. A. M. Ramírez and J. D. Reiss, "End-to-end equalization with convolutional neural networks," in *21st International Conference on Digital Audio Effects (DAFx-18)*, 2018.

[9] M. A. M. Ramirez and J. D. Reiss, "Modeling nonlinear audio effects with end-to-end deep neural networks," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 171–175.

[10] Z. Zhang, E. Olbrych, J. Bruchalski, T. J. McCormick, and D. L. Livingston, "A vacuum-tube guitar amplifier model using long/short-term memory networks," in *SoutheastCon 2018*. IEEE, 2018, pp. 1–5.

[11] T. Schmitz and J.-J. Embrechts, "Nonlinear real-time emulation of a tube amplifier with a long short time

memory neural-network," in *Audio Engineering Society Convention 144*. Audio Engineering Society, 2018.

[12] A. Wright, E.-P. Damskägg, V. Välimäki *et al.*, "Real-time black-box modelling with recurrent neural networks," in *22nd international conference on digital audio effects (DAFx-19)*, 2019.

[13] E.-P. Damskägg, L. Juvela, V. Välimäki *et al.*, "Real-time modeling of audio distortion circuits with deep learning," in *Proc. Int. Sound and Music Computing Conf.(SMC-19), Malaga, Spain*, 2019, pp. 332–339.

[14] M. A. M. Ramírez, E. Benetos, and J. D. Reiss, "A general-purpose deep learning approach to model time-varying audio effects," *arXiv preprint arXiv:1905.06148*, 2019.

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

[16] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

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

[18] N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, A. Ku, and D. Tran, "Image transformer," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4055–4064.

[19] K. Han, A. Xiao, E. Wu, J. Guo, C. Xu, and Y. Wang, "Transformer in transformer," *arXiv preprint arXiv:2103.00112*, 2021.

[20] X. Chu, Z. Tian, B. Zhang, X. Wang, X. Wei, H. Xia, and C. Shen, "Conditional positional encodings for vision transformers," *arXiv preprint arXiv:2102.10882*, 2021.

[21] Y. Tang, K. Han, C. Xu, A. Xiao, Y. Deng, C. Xu, and Y. Wang, "Augmented shortcuts for vision transformers," in *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[22] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z. Jiang, F. E. Tay, J. Feng, and S. Yan, "Tokens-to-token vit: Training vision transformers from scratch on imagenet," *arXiv preprint arXiv:2101.11986*, 2021.

[23] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," *arXiv preprint arXiv:2010.04159*, 2020.

[24] E. Xie, W. Wang, W. Wang, P. Sun, H. Xu, D. Liang, and P. Luo, "Segmenting transparent object in the wild with transformer," *arXiv preprint arXiv:2101.08461*, 2021.

[25] S. Zheng, J. Lu, H. Zhao, X. Zhu, Z. Luo, Y. Wang, Y. Fu, J. Feng, T. Xiang, P. H. Torr *et al.*, "Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6881–6890.

[26] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer," *arXiv preprint arXiv:1809.04281*, 2018.

[27] P. Verma and J. Berger, "Audio transformers: Transformer architectures for large scale audio understanding. adieu convolutions," *arXiv preprint arXiv:2105.00335*, 2021.

[28] Y.-S. Huang and Y.-H. Yang, "Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 1180–1188.

[29] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu, "Neural speech synthesis with transformer network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 6706–6713.

[30] P. Verma and C. Chafe, "A generative model for raw audio using transformer architectures," *arXiv preprint arXiv:2106.16036*, 2021.

[31] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," *arXiv preprint arXiv:1904.10509*, 2019.

[32] G. Peeters and G. Richard, "Deep learning for audio and music," 2021.

[33] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[34] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[35] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2623–2631.

[36] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *arXiv preprint arXiv:2004.05150*, 2020.

[37] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," *Advances in Neural Information Processing Systems*, vol. 32, 2019.