Seq-U-Net: A One-Dimensional Causal U-Net for Efficient Sequence Modelling

Daniel Stoller[1*], **Mi Tian**[2], **Sebastian Ewert**[2] and **Simon Dixon**[1]
[1]Queen Mary University of London
[2]Spotify
d.stoller@qmul.ac.uk, {mtian,sewert}@spotify.com, s.e.dixon@qmul.ac.uk

Abstract

Convolutional neural networks (CNNs) with dilated filters such as the Wavenet or the Temporal Convolutional Network (TCN) have shown good results in a variety of sequence modelling tasks. While their receptive field grows exponentially with the number of layers, computing the convolutions over very long sequences of features in each layer is time and memory-intensive, and prohibits the use of longer receptive fields in practice. To increase efficiency, we make use of the "slow feature" hypothesis stating that many features of interest are slowly varying over time. For this, we use a U-Net architecture that computes features at multiple time-scales and adapt it to our auto-regressive scenario by making convolutions causal. We apply our model ("Seq-U-Net") to a variety of tasks including language and audio generation. In comparison to TCN and Wavenet, our network consistently saves memory and computation time, with speed-ups for training and inference of over 4x in the audio generation experiment in particular, while achieving a comparable performance on real-world tasks.

1 Introduction

Sequence modelling is an important problem central to many application domains, including language, audio, and video generation [Bai *et al.*, 2018; Yu *et al.*, 2017; Trinh *et al.*, 2018]. In some of these applications, the sequences can be millions of time-steps in length (e.g. in the case of audio generation due to the high sampling rate of audio signals), and it can be vital to model the long-term dependencies present in such sequences (for example to be able to repeat a melody in a music piece that occurred a minute earlier).

This problem is often framed as the task of predicting the next element in a sequence given all of the elements observed so far, giving rise to auto-regressive models. Recurrent neural networks (RNNs) are often used in this context since they can theoretically remember inputs for an arbitrary number of time-steps, and also offer quick inference at test time as the hidden state carries all the information about previous sequence elements and only needs to be updated using the next element. However, in practice, these models can be difficult [Bengio *et al.*, 1994] and slow [Trinh *et al.*, 2018] to train due to their strictly sequential nature. More recently, CNNs with dilated filters were shown to be competitive approaches for sequence modelling. Instead of relying on recurrence to retain information over a large number of steps, which might be difficult to achieve in practice, CNNs such as the temporal convolutional network (TCN) [Bai *et al.*, 2018] and Wavenet [van den Oord *et al.*, 2016] access far-away time-steps more directly through their dilated filters.

Despite their impressive performance, these architectures suffer from two issues. Firstly, each convolutional layer operates at the same time resolution as the input. This results in a high memory usage and training time especially with long sequences, rendering long-term modelling infeasible even with large scale, multi-GPU training [van den Oord *et al.*, 2016]. Secondly, inference is slow as elements have to be predicted sequentially and require a forward pass through the CNN's many layers. Although re-using layer outputs from previous steps helps, all layers still have to be traversed and updated to predict the next sequence element.

In this context, "slow feature analysis" [Wiskott and Sejnowski, 2002] poses that for a wide variety of tasks important features of an input signal vary only slowly over time – which leads to an interesting approach of increasing efficiency by computing some features at lower sampling rates compared to the input without compromising model performance. Notably, U-Nets [Ronneberger *et al.*, 2015] already incorporate the equivalent of this principle for image processing, by computing features at different time-scales with two-dimensional convolutions and combining them to make predictions at the same resolution as the input. A version with *one-dimensional* convolutions was presented for audio source separation [Stoller *et al.*, 2018]. We base our model on this U-Net variant, as it should be able to process many kinds of temporal sequences, not just audio signals. We show how to adapt it for our auto-regressive setting by making all convolutions causal, such that each prediction for the next time-step can only depend on past inputs.

As a result, we obtain the "Seq-U-Net"[1], a general-purpose

[*]Contact Author. Funded by EPSRC grant EP/L01632X/1. Present affiliation: Statistics and Machine Learning, DZNE, Bonn.

[1]Code available at https://github.com/f90/Seq-U-Net

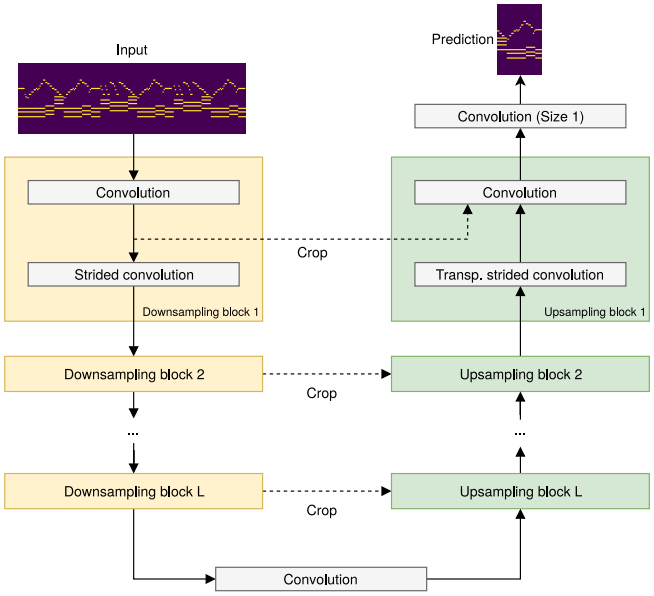

Figure 1: Architecture diagram of our proposed model. 1D Convolutions are applied across time with LeakyReLU activations followed by Dropout. Strided and transposed strided convolutions are used for down- and upsampling the features, respectively. Since the convolutions do not use padding, the output is smaller than the input and skip connections need to be cropped at the front.

network architecture that is not limited to audio tasks but can be applied to a wide range of sequence modelling problems – while providing considerable efficiency improvements over TCN and Wavenet. Inference is greatly accelerated by only computing new layer activations if they are not decimated in the downsampling process. This time-variant processing gives each layer its own "update rate", which is in contrast to fully-convolutional TCN and Wavenet approaches. In particular, we compare to TCN in the context of word- and character-level language modelling as well as symbolic music generation. Additionally, we tackle the task of generating piano music directly in the time-domain and compare performance with a Wavenet reimplementation using a log-likelihood metric as well as listening tests. Overall, we find that our architecture achieves competitive results while requiring less memory and training time.

2 Related Work

Recurrent neural networks (RNNs) are a commonly used approach in deep learning for sequence modelling, including LSTMs and GRUs [Graves *et al.*, 2013; Boulanger-Lewandowski *et al.*, 2012]. In practice, training these models to successfully model long-term dependencies can be difficult [Bengio *et al.*, 1994] and slow as computation is strictly sequential and can not be parallelised [Trinh *et al.*, 2018]. Hierarchical multi-scale RNNs [Chung *et al.*, 2016; El Hihi and Bengio, 1995] and the Clockwork RNN [Koutník *et al.*, 2014] model time series on multiple time-scales to enable longer-term dependency modelling, but the sequential processing in the high-resolution timescales is still computationally expensive. Further work in this direction also includes the DilatedRNN [Chang *et al.*, 2017]. The SampleRNN [Mehri *et al.*, 2016] is a three-layer RNN specifically developed for audio generation. While it also employs a multi-scale approach, it inherits the disadvantages of RNNs mentioned above, and the "slower" layers have to compute high-level features directly from raw audio and forward them to the "faster" layers, which is arguably more difficult than computing them bottom-up.

Alternative approaches involve CNNs with filters that have increasing dilation factors to cover longer distances between inputs [Kalchbrenner *et al.*, 2016; Campos *et al.*, 2017], of which we want to highlight TCN [Bai *et al.*, 2018] and Wavenet [van den Oord *et al.*, 2016] for sequence modelling. Due to their depth, these neural models require a large amount of memory and have slow inference.

The parallel Wavenet [van den Oord *et al.*, 2017] provides fast inference by using a flow-based student network to emulate the outputs of an already trained Wavenet. For long-term dependency modelling in audio, Dieleman et al. [2018] use a complex, multi-stage training with auto-encoding networks to compress the audio before using Wavenets to model the latent state evolution. However, since these approaches involve training a Wavenet, they inherit its computational complexity.

Other approaches were developed such as FFTNet [Jin *et al.*, 2018], WaveRNN [Kalchbrenner *et al.*, 2018] and MelNet [Vasquez and Lewis, 2019], which do provide large efficiency gains by means of optimisations specific to the audio domain, but at the cost of generality.

Finally, the Transformer network [Vaswani *et al.*, 2017] has shown great potential, but the complexity of its attention mechanism is quadratic in the length of the sequence, preventing the use for long sequences. Sparse Transformers [Child *et al.*, 2019] restrict the attention modules to a sparse subset of all previous inputs to remedy this, but could still benefit from introducing a multi-scale architecture.

We are unaware of another multi-scale approach evaluated across a variety of sequence modelling problems, but similar approaches were used for video segmentation [Shelhamer *et al.*, 2016] and audio separation [Stoller *et al.*, 2018].

3 Method

We present two variants of our multi-scale approach. The first is an adaptation of the Wave-U-Net to the auto-regressive setting and shown in Section 3.1. The second variant, presented in Section 3.2, further adds residual connections to stabilise training for tasks with very long-term dependencies such as raw audio generation.

3.1 Seq-U-Net

Our model is based on the Wave-U-Net [Stoller *et al.*, 2018] and shown in Figure 1. The network features L *levels* of downsampling (DS) and upsampling (US) blocks, and a convolutional bottleneck and output layer. Each downsampling block features a convolution, whose outputs are used as a shortcut connection for the respective upsampling block, followed by another convolution with stride k to downsample the features across time. Each upsampling block has a transposed convolution with stride k to upsample the previously

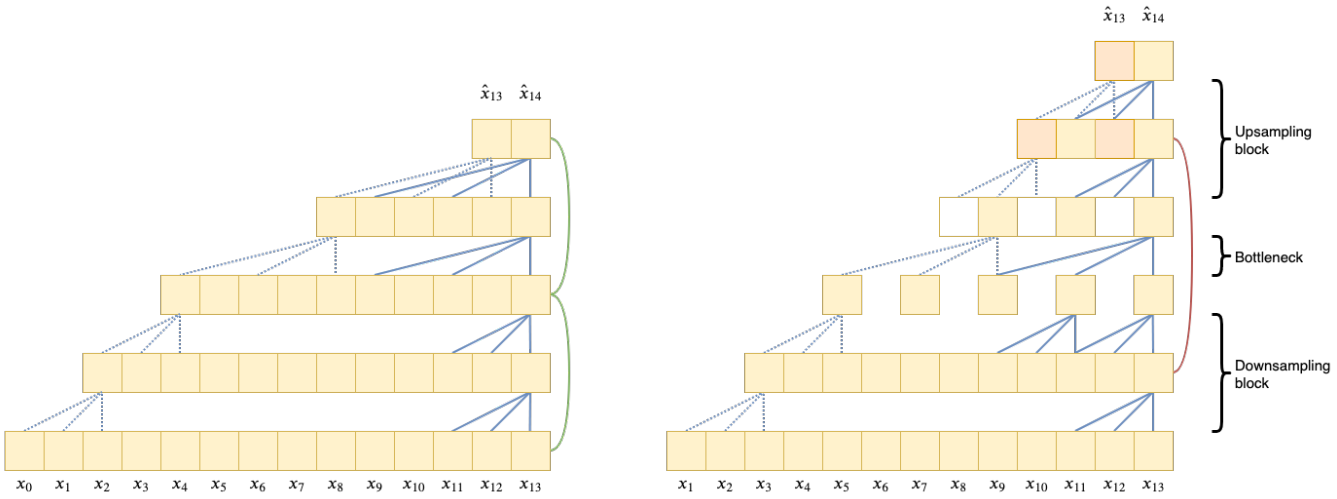

Figure 2: Compared to TCN (left, two residual blocks), Seq-U-Net (right, one down- and upsampling block) computes features only at certain intervals to save memory and training time. Zero-padding is used in the upsampling blocks (white squares), leading to different computational paths throughout the network (red squares). The red line indicates feature cropping and concatenation.

obtained coarse-grained features. The result is concatenated with the features from the shortcut connection, and input to another convolution to combine high- and low-level features. In this paper, we set the stride k to 2. All convolutions have the same filter width and a LeakyReLU activation followed by Dropout, except for the output convolution.

Like in the original Wave-U-Net, the convolutional layers do not use zero-padding so that all model predictions are made with the necessary input context. As a result, there are more feature frames in the shortcut output of a DS block than in the output of the transposed strided convolution in the corresponding US block. Zeros are prepended to the beginning of input sequences to allow predicting the first sequence elements. In the Wave-U-Net, the outputs at each level of the network are interpreted as features describing the center part of the input, so the shortcut features are center-cropped before concatenation. Consequently, source signals are predicted for the center part of the mixture excerpt.

Our key idea is to interpret the filters as causal instead: the output of a filter covering input timesteps $n-k$ to $n+k$ should now help predict input x_{n+k+1} *outside* of its receptive field instead of some feature of the input at timestep n, i.e. the current source audio signal. Therefore, we instead crop the first feature frames of each shortcut connection to make sure that features are aligned in time properly. As a result, we obtain an auto-regressive model for sequence modelling, similar to Wavenet and TCN, but significantly sparser in terms of activations due to the decreased resolution in most of the layers.

Fast Inference

From a signal processing perspective, TCN and Wavenet are time-invariant systems as they apply the same set of operations at each time step. In contrast, the multi-scale architecture of Seq-U-Net allows us to employ a time-variant processing scheme (inspired by [Koutník *et al.*, 2014]) that drastically accelerates inference, as many operations do not have to be computed at every step: If an output computed for the latest time-step in a DS block is decimated, only the US blocks on the same or higher resolution need to be updated, since the input to the other blocks does not change. This means that a block on level $i \in \{1, \dots, L\}$ only needs to be updated every k^{i-1} time-steps. To implement this procedure, all blocks are given an internal clock based on their level to determine when to compute a new output. To predict the very first sample from a given context, a normal forward-pass is conducted and caches for the resulting layer activations are set up before switching to the above step-wise procedure. For further details on the implementation, please refer to our source code.

3.2 Residual Variant

Since raw audio generation benefits from a large receptive field, we employ much deeper instances of our model for the experiment in Section 5.2. With this increase in layers however, we observed training instability. Residual networks can be trained stably even with hundreds of layers [He *et al.*, 2015], so we also propose a residual variant of our model.

Compared to the baseline model from Section 3.1, we employ an additional convolution on the input with F output channels, and also use F input and output channels for all up- and downsampling blocks to allow for residual connections. We replace each convolutional layer in the base model with a residual layer similar to the one in Wavenet [van den Oord *et al.*, 2016], whose outputs y are given by

$$y = I(x) + \tanh(C_1(x)) \cdot \sigma(C_2(x)), \tag{1}$$

where x are the layer inputs, σ is the sigmoid function, C_i applies convolutional layer i to its input and I processes the input x to provide an identity connection in case the convolutions change the feature dimensionality.

For the convolutions with stride used in the DS blocks, I first decimates the input x to provide the identity for the residual layer. For the transposed convolutions with stride in the US blocks, I takes the input and repeats the feature vector

at each time step $k-1$ times to perform upsampling[2]. For both down- and upsampling, I finally crops the resulting feature sequence at the front to ensure it matches the number of residual features, which is reduced due to not using padding for convolutions. To refine the high-resolution shortcut features using the low-resolution features from the upsampling path, we use the shortcut as input x and use the concatenation of the shortcut and the upsampled features as input to the residual convolutions C_i.

To easily scale the network in size for more complex tasks, we employ $D+1$ residual layers in each block (one layer for up- or downsampling), with D as hyper-parameter, allowing features to be processed more flexibly at each time resolution.

4 Complexity Analysis

We will analyse the memory consumption and computational complexity of our approach at both training and test time and compare with Wavenet[3].

4.1 Training

Due to the size N of receptive field increasing exponentially with the number of layers for the Seq-U-Net and Wavenet, roughly $L = \log_k(N)$ levels of processing are required. For the Wavenet, we define k as the factor with which dilation increases in each layer.

When presented with $I \geq N$ inputs during training, Wavenet needs to compute I feature activations in each of the L layers, since it operates on the same resolution as the input, reaching a total of $I \cdot \log_k(N)$. The Seq-U-Net on the other hand computes $3I + \frac{I}{k}$ feature activations in the first down- and upsampling block, $3\frac{I}{k} + \frac{I}{k^2}$ on the second level, and so on, in addition to a bottleneck convolution with $\frac{I}{k^L}$ outputs. For the Seq-U-Net, we thus obtain at most $\sum_{i=0}^{L} 4\frac{I}{k^i} \leq 8I$ feature activations regardless of the number of layers[4]. The above calculation not only demonstrates the time complexity, but also the required memory, since the computed feature activations need to be maintained for the backward-pass.

4.2 Inference

At test time, and auto-regressive models such as Wavenet and Seq-U-Net require a forward pass to generate the next element in the sequence, which can be prohibitively slow when sequences are long (e.g. in audio generation) or when a real-time application is desired. While caching previously computed outputs in the Wavenet reduces computation time, it still involves evaluating all L layers, which especially affects deep models (e.g. $L = 30$ in [Kalchbrenner *et al.*, 2018]).

In the Seq-U-Net, each level in the network only has to be updated at certain intervals as described in Section 3.1. In particular, the average number of levels we have to update for each time-step is $\sum_{i=1}^{L} \frac{1}{k^{i-1}} \leq 2$ and thus a constant number of layers independent of the number of levels L in the network. While this is an amortised analysis of the average time per step, in the worst case all layers need to be updated, although this is not relevant for offline sequence generation.

5 Experiments

We evaluate our method on a variety of sequence modelling tasks regarding its performance, training time and memory complexity. Due to the architectural similarity, we will firstly compare our method with TCN in Section 5.1 on language modelling as well as symbolic music modelling. To test whether our model can capture long-term dependencies, we also compare to TCN on a synthetic copy task and to a Wavenet baseline on the task of audio generation in the time-domain. Note that the Wave-U-Net can not be used as a baseline model for these experiments, since it has access to sequence element x_{t+1} when predicting the successor to x_t and can therefore easily achieve perfect prediction.

For time and memory measurements, we use a single NVIDIA GTX 1080 GPU with Pytorch 1.2, CUDA 9 and cuDNN 7.5[5]. We compare the average time required for each training step and the maximum memory allocated throughout a training epoch[6].

5.1 Comparison With TCN

We will compare our model against TCN across three sequence modelling tasks. To match model complexity, we use the same filter length, Dropout rate, and levels of resolution, which results in very similar receptive field size. Then, the number of features in each layer is adapted for Seq-U-Net so it matches TCN in the number of parameters.

We optimise each model for 100 epochs using a batch size of 16 and an Adam optimiser with initial learning rate α, which is reduced by half if validation performance did not improve after P epochs and more than 10 epochs have passed since the beginning of training. Finally, the model that performed best on the validation set is selected.

To prevent the training procedure from favouring one model over the other, we perform a hyper-parameter optimisation over the learning rate $\alpha \in [e^{-12}, e^{-2}]$ and optional gradient clipping with magnitudes between $[0.01, 1.0]$. This hyper-parameter optimisation is performed for each combination of model and task using a tree of Parzen estimators[7] to find the minimum validation loss. All results are shown in Table 1, using the hyper-parameters shown in Table 2.

Character-based Language Modelling

We perform character-based language modelling, where the task is to predict the next character given a history of previously observed ones, on the PTB dataset [Marcus *et al.*, 1993]. The average cross-entropy loss is used as training objective, and patience is set to $P = 5$.

For both models, we use 100-dimensional character embeddings with 0.1 Dropout as input, and their output is projected back to character probabilities using the transposed

[2]This operation does not violate the auto-regressive condition.

[3]Comparison with TCN is omitted as it is very similar to Wavenet but differs slightly in the number of layers per level of resolution

[4]This disregards the reduction in size due to not using padding for convolutions since it occurs in all models

[5]We use Pytorch's benchmark mode to find the best algorithm for training each network.

[6]Does not include memory used for purposes such as caching

[7]"Hyperopt" package: http://hyperopt.github.io/hyperopt/

Task	Model	Train	Test	Time (s)	Mem.
Char-LM	TCN	1.066	1.31	0.0694	445.9
Char-LM	Seq-U-Net	1.08	1.30	0.0286	304.9
Word-LM	TCN	47.21	108.47	0.0480	580.5
Word-LM	Seq-U-Net	40.43	107.95	0.0234	382.1
M-Muse	TCN	5.789	6.931	0.0059	108.5
M-Muse	Seq-U-Net	5.794	6.969	0.0065	75.3
M-Nott	TCN	1.409	2.783	0.0071	73.1
M-Nott	Seq-U-Net	1.850	2.97	0.0067	52.5
M-JSB	TCN	6.178	8.154	0.0034	13.1
M-JSB	Seq-U-Net	6.151	8.173	0.0037	8.2
Piano	Wavenet	1.76	1.88	1.4616*	5294*
Piano	Seq-U-Net**	1.83	1.93	0.3621*	1514*

* Measurements were taken with a batch size of 2 instead of 16 due to the high amount of memory required.
** Residual variant

Table 1: Performance (lower is better) for Seq-U-Net and comparison models across different tasks ("M-" denotes symbolic music modelling). Times denote the duration for a forward- and backward-pass. "Mem." is the required GPU memory in MB

version of the embedding matrix. We evaluate models using the bits-per-character (bpc) metric.

As shown in Table 1, our model performs as well as its TCN counterpart in this regard, while requiring 59% less time per training step, and 32% less GPU memory during training. These results suggest that many of the required features are on a higher level of abstraction and vary only slowly, e.g. per word or per sentence, and so do not need to be recomputed for each new character – a hypothesis also put forth in [Chung *et al.*, 2016].

Word-based Language Modelling

For our second experiment, we perform word-based language modelling, which involves predicting the next word following a given sequence of words. As in the previous experiment, we use the PTB dataset with a vocabulary of 10,000 words. Following TCN's experimental set-up [Bai *et al.*, 2018], we use 600-dimensional word embeddings with 0.25 Dropout as input, and use the transpose of the embedding matrix to project the 600-dimensional outputs from the models to probability vectors over all words. For training, we minimise the average cross-entropy with a patience of $P = 5$, and for evaluation we use the per-word perplexity.

Similarly to the results for character-based language modelling in Section 5.1, Table 1 shows that both models perform very similarly, but the Seq-U-Net architecture is substantially more efficient to train (reducing the training time by 51% and memory usage by 34%).

Symbolic Music Modelling

For our final comparison with TCN on real-world data, we model polyphonic music in the symbolic domain. Each music piece is represented as a piano roll – a binary matrix of size $88 \times T$ that indicates which of the 88 pitches are active at each of the T time frames. Our models predicts a whole time-frame at each step in an auto-regressive manner, and we use the sum of binary cross-entropies over each pitch, averaged over all time frames as training objective. We use a patience of $P = 10$ for early stopping.

Three different datasets of varying complexity and content are used: Muse[8], Nottingham (Nott)[9] and the JSB chorales [Allan and Williams, 2005]. For evaluation, we use the frame-wise perplexity introduced in [Boulanger-Lewandowski *et al.*, 2012].

Table 1 shows the perplexity on the training and test sets for both models on all datasets. We find that both models are very closely matched in terms of training and test perplexity on the Muse and JSB datasets. For the Nott dataset, TCN achieves a noticeably lower perplexity than the Seq-U-Net on the training partition. This performance gap also appears on the test set, although it is considerably smaller, indicating that incorporating the slow feature hypothesis induces a regularising effect on the model.

For these datasets, no improvement in training time is observed, unlike the previous language modelling experiments. This is due to the much smaller size of the models, where the higher number of convolutional layers in the Seq-U-Net has a larger impact than the reduction in computation time for each layer. Nevertheless, the memory footprint is substantially reduced by an average 32%.

Copy Task

Finally, we compared our model to TCN on the *copy task*, following the experimental setup outlined in [Bai *et al.*, 2018]. The input to the model is a one-dimensional sequence consisting of 10 integer numbers randomly chosen between 1 and 8, followed by M zeroes, and 11 entries filled with the digit 9, acting as a signal for the model to output the initial 10-number sequence at the end of the input sequence.

Using the same setting of $M = 1000$ used in [Bai *et al.*, 2018], we found that Seq-U-Net was not able to retain the number sequence and output it at the end (reaching an accuracy of 12.7%), in contrast to TCN. Theoretically, this can be explained by the resampling operations contained in the Seq-U-Net, through which the number sequence needs to be transported. Neighbouring elements (feature vectors) of the sequence need to be encoded into a single feature vector so that subsequent downsampling of this sequence of feature vectors does not result in information loss. Similarly, even if the information successfully passes through all downsampling layers, the original sequence has to be decoded in the upsampling path. Both of these operations would require very specific configurations of the convolutional filters to be successful. However, it seems that retaining such high-frequency information over large numbers of time-steps is rarely needed in many real-world applications, since Seq-U-Net performs well on all real-world benchmarks investigated in this paper.

5.2 Raw Audio Generation

To test whether our model can capture long-term dependencies found in complex real-world sequences, we apply it to the generation of audio waveforms, using the residual variant

[8] See http://www-etud.iro.umontreal.ca/~boulanni/icml2012
[9] See http://ifdo.ca/~seymour/nottingham/nottingham.html

Task	Model	W	L	H	Dropout	Context	Params	P	LR	Clip
Char-LM	TCN	3	4	600	0.1	80	5.9M	5	0.00014	0.213
Char-LM	Seq-U-Net	3	4	390	0.1	73	5.9M	5	0.00073	No
Word-LM	TCN	3	4	600	0.5	73	14.7M	5	0.00115	No
Word-LM	Seq-U-Net	3	4	390	0.5	73	14.9M	5	0.00037	0.722
Music-Muse	TCN	5	4	215	0.2	Full	1.7M	10	0.00023	No
Music-Muse	Seq-U-Net	5	4	150	0.2	Full	1.7M	10	0.00047	No
Music-Nott	TCN	5	4	215	0.2	Full	1.7M	10	0.000067	0.601
Music-Nott	Seq-U-Net	5	4	150	0.2	Full	1.7M	10	0.00108	No
Music-JSB	TCN	3	2	220	0.5	Full	534k	10	0.00134	No
Music-JSB	Seq-U-Net	3	2	170	0.5	Full	522k	10	0.00051	0.324

Table 2: Hyper-parameters for TCN and Seq-U-Net comparison models investigated in Section 5.1. W is the convolutional filter width, L the number of layers, H the number of convolutional filters per layer, P the early stopping patience, and LR and Clip are the best learning rate and clipping magnitude found by hyper-parameter optimisation.

Model	Layers	Features	Context	Filter width
Wavenet	13	128	32764	2
Seq-U-Net	11	180	32748	5

Table 3: Models used for audio generation. Context is given as a number of audio samples.

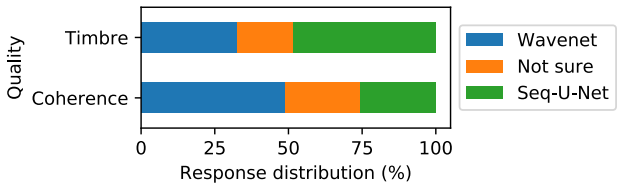

Figure 3: Overall distribution of listening test responses for both the timbre and the musical coherence questions

presented in Section 3.2. Since our architecture resembles the Wavenet with its use of stacks of residual convolutions, we use it as our comparison model in the following.

In particular, we use the classical piano recordings as used by Dieleman et al. [2018] amounting to about 607 hours in duration, and partition them into a training and test set, while avoiding pieces overlapping between the two partitions. Note that our version of the dataset is different as we were not able to obtain all the recordings listed in [Dieleman *et al.*, 2018].

We train two models in this experiment, listed in Table 3. The first one is a Wavenet baseline comprised of 4 Wavenet stacks with 13 dilated convolutional layers each and 512 features in the skip connection, and the second one is a Seq-U-Net model that matches the Wavenet in terms of receptive field size, and uses a residual depth of $D = 2$.

Besides downsampling the audio to 16 KHz mono signals, no further preprocessing is applied. During training, audio excerpts are loaded from random positions within the audio files, and each audio sample is transformed into a 256-dimensional one-hot vector using 8-bit mu-law encoding, following the Wavenet approach [van den Oord *et al.*, 2016]. A training batch consists of 16 examples and uses the last 5000 audio samples in each example as simultaneous training targets for the model. The average cross-entropy is minimised over 246000 iterations (equivalent to just over one epoch) with an Adam optimiser and a learning rate of 0.0005.

Experimental Setup

For evaluation, we report the likelihood of the models in bits per audio samples (bpa) on the test set. However, the bpa metric might not reflect perceptual audio quality very well, especially since the model uses its own predictions as input and not real samples at test time. This discrepancy is well known in the literature [Huszár, 2015], and we also found in practice that the two models vary in their stability at generation time. While the Wavenet starts to introduce progressively more noise into its outputs with longer generation, the Seq-U-Net appears stable throughout. Since this effect is very pronounced with durations of 10 seconds or longer, making Seq-U-Net clearly preferable, we conducted a listening test with samples of 5 seconds. We used a temperature of 0.95, meaning the unnormalised model outputs were divided by 0.95 before applying the softmax to obtain probabilities. In preliminary experiments, we found this stabilises the generation process, resulting in increased quality for both models.

Each of the 20 questions presented the participant with a 1.5 second excerpt of real piano randomly sampled from our test dataset. This was followed by two continuations produced by our two models that also include the real excerpt in the beginning. This conditional generation setting allows directly comparing between outputs of different models for the same input context: The participants were asked which excerpt has "better timbre (does it sound like a piano, is the audio free of distortions?)" and "more musical coherence (with respect to melody, harmony, rhythm)". An additional "Not sure" option was available when the participant thinks the quality is the same for both excerpts. The total number of participants is 22.

Results

As seen in Table 1, the Wavenet slightly outperforms the Seq-U-Net in terms of the bpa metric, albeit achieving a small

relative improvement of 2.6% on the test set, indicating the models are closely matched in terms of performance. The training set results indicate this might be due to the Wavenet fitting the training set more closely in the given number of training iterations. At the same time, the required training time and memory are drastically reduced for the Seq-U-Net by a factor of 4 and 3.5, respectively.

The results of the listening test are shown in Figure 3. While the Seq-U-Net exhibits better timbral characteristics, producing better continuations than the Wavenet in 15 out of the 20 provided examples, it falls behind in terms of musical coherence. We suspect this is due to the Seq-U-Net sometimes producing an unexpected transition from the real excerpt to the generated section, but then producing sounds more stably as time goes on. Overall, the two models appear to have different strengths and weaknesses – we encourage the reader to listen to the audio examples provided in our code repository. Additionally, the high amount of "Not sure" responses, especially for such a sensitive paired discrimination task, indicates that the models are quite evenly matched in this setting.

Finally, we measure the performance impact of our inference method introduced in Section 3.1 by comparing to the Wavenet's generation speed when caching previous activations. With a batch size of 1 on a single NVIDIA GTX 1080 GPU, we achieve 69 audio samples per second for the Wavenet, and 309 for the Seq-U-Net and thereby a speed-up with a factor greater than 4.

6 Discussion

The predictive performance of the Seq-U-Net as outlined in Table 1 is remarkably similar to that of Wavenet and TCN comparison models across all tasks we tested. While efficiency gains are not very noticeable for very small instances of our model with few levels of resolution, they rapidly increase when moving towards larger and deeper models as used in language and audio modelling, and we can expect these gains to become more pronounced for even deeper models with even longer receptive fields.

Since the metrics used in Table 1 are based on how much probability the models assign to the test data (log-likelihood) and not directly on how realistic their generated output is, we performed a listening test for the piano audio generation task. Surprisingly, despite better log-likelihood, our implementation of the Wavenet accumulates noise during generation, making it unsuitable to generate longer music pieces, whereas the Seq-U-Net is stable but less capable of smoothly continuing the real excerpts, for reasons that remain unclear. A more unified approach to training and evaluating generative models would be desirable.

7 Conclusion

In this paper, we demonstrated how a causal variant of a U-Net architecture with one-dimensional convolutions across the time domain can perform on par with existing state-of-the-art models in a variety of real-world sequence modelling tasks, while significantly reducing training time and memory requirements. Leveraging the idea that many relevant features in real-world sequences are only slowly varying over time allows the use of convolutional layers that compute features at progressively lower resolutions. These efficiency gains make it feasible to train generative models with much longer receptive fields in the future, which can be very useful in domains such as music and language generation. While results on the synthetic copy task show that high-frequency information can not be retained over large numbers of time steps, the competitive performance of our model on real-world benchmarks suggests only modelling long-term dependencies between "slow features" might be sufficient – although this should be investigated further in the future.

A limitation of our approach is that the levels of resolution along with the processing capacity at each resolution has to be manually pre-defined, which could limit performance. Future work could include potential solutions as used in the Phased LSTM [Neil *et al.*, 2016] so the model can adapt its levels of resolution more dynamically to the task.

Finally, attention mechanisms have shown great potential for sequence modelling and could be integrated into our approach by using attention operations in each down- and upsampling block alongside or instead of convolutions to further improve performance, as suggested in [Child *et al.*, 2019].

References

[Allan and Williams, 2005] Moray Allan and Christopher Williams. Harmonising Chorales by Probabilistic Inference. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 25–32. MIT Press, 2005.

[Bai *et al.*, 2018] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Convolutional Sequence Modeling Revisited. In *International Conference on Learning Representations (ICLR) Workshop Track*, 2018.

[Bengio *et al.*, 1994] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[Boulanger-Lewandowski *et al.*, 2012] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *Proc. of the International Conference on Machine Learning (ICML)*, pages 1159–1166, 2012.

[Campos *et al.*, 2017] Victor Campos, Brendan Jou, Xavier Giro-i-Nieto, Jordi Torres, and Shih-Fu Chang. Skip RNN: Learning to skip state updates in recurrent neural networks. *CoRR*, abs/1708.06834, 2017.

[Chang *et al.*, 2017] Shiyu Chang, Yang Zhang, Wei Han, Mo Yu, Xiaoxiao Guo, Wei Tan, Xiaodong Cui, Michael Witbrock, Mark A Hasegawa-Johnson, and Thomas S Huang. Dilated Recurrent Neural Networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 77–87. Curran Associates, Inc., 2017.

[Child *et al.*, 2019] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating Long Sequences with Sparse Transformers. *CoRR*, abs/1904.10509, 2019.

[Chung *et al.*, 2016] Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical Multiscale Recurrent Neural Networks. *CoRR*, abs/1609.01704, 2016.

[Dieleman *et al.*, 2018] Sander Dieleman, Aäron van den Oord, and Karen Simonyan. The challenge of realistic music generation: Modelling raw audio at scale. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 8000–8010, 2018.

[El Hihi and Bengio, 1995] Salah El Hihi and Yoshua Bengio. Hierarchical Recurrent Neural Networks for Long-term Dependencies. In *Proceedings of the 8th International Conference on Neural Information Processing Systems*, NIPS'95, pages 493–499, Denver, Colorado, 1995. MIT Press.

[Graves *et al.*, 2013] Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6645–6649, 2013.

[He *et al.*, 2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[Huszár, 2015] Ferenc Huszár. How (not) to Train your Generative Model: Scheduled Sampling, Likelihood, Adversary? *CoRR*, abs/1511.05101, 2015.

[Jin *et al.*, 2018] Zeyu Jin, Adam Finkelstein, Gautham J. Mysore, and Jingwan Lu. FFTNet: A Real-Time Speaker-Dependent Neural Vocoder. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[Kalchbrenner *et al.*, 2016] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural Machine Translation in Linear Time. *CoRR*, abs/1610.10099, 2016.

[Kalchbrenner *et al.*, 2018] Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron van den Oord, Sander Dieleman, and Koray Kavukcuoglu. Efficient Neural Audio Synthesis. In Jennifer Dy and Andreas Krause, editors, *Proc. of the International Conference on Machine Learning (ICML)*, volume 80, pages 2410–2419, Stockholmsmässan, Stockholm Sweden, 2018. PMLR.

[Koutník *et al.*, 2014] Jan Koutník, Klaus Greff, Faustino Gomez, and Jürgen Schmidhuber. A Clockwork RNN. *CoRR*, abs/1402.3511, February 2014.

[Marcus *et al.*, 1993] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a Large Annotated Corpus of English: The Penn Treebank. *Comput. Linguist.*, 19(2):313–330, June 1993.

[Mehri *et al.*, 2016] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio. SampleRNN: An Unconditional End-to-End Neural Audio Generation Model. *CoRR*, abs/1612.07837, 2016.

[Neil *et al.*, 2016] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Phased LSTM: Accelerating Recurrent Network Training for Long or Event-based Sequences. *CoRR*, abs/1610.09513, 2016.

[Ronneberger *et al.*, 2015] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proc. of the International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[Shelhamer *et al.*, 2016] Evan Shelhamer, Kate Rakelly, Judy Hoffman, and Trevor Darrell. Clockwork Convnets for Video Semantic Segmentation. *CoRR*, abs/1608.03609, 2016.

[Stoller *et al.*, 2018] Daniel Stoller, Sebastian Ewert, and Simon Dixon. Wave-U-Net: A Multi-Scale Neural Network for End-to-End Source Separation. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, volume 19, pages 334–340, 2018.

[Trinh *et al.*, 2018] Trieu H. Trinh, Andrew M. Dai, Minh-Thang Luong, and Quoc V. Le. Learning Longer-term Dependencies in RNNs with Auxiliary Losses. *CoRR*, abs/1803.00144, 2018.

[van den Oord *et al.*, 2016] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. *arXiv:1609.03499*, 2016.

[van den Oord *et al.*, 2017] Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis C. Cobo, Florian Stimberg, Norman Casagrande, Dominik Grewe, Seb Noury, Sander Dieleman, Erich Elsen, Nal Kalchbrenner, Heiga Zen, Alex Graves, Helen King, Tom Walters, Dan Belov, and Demis Hassabis. Parallel WaveNet: Fast High-Fidelity Speech Synthesis. *CoRR*, abs/1711.10433, 2017.

[Vasquez and Lewis, 2019] Sean Vasquez and Mike Lewis. MelNet: A Generative Model for Audio in the Frequency Domain. *CoRR*, abs/1906.01083, 2019.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *CoRR*, abs/1706.03762, 2017.

[Wiskott and Sejnowski, 2002] Laurenz Wiskott and Terrence J. Sejnowski. Slow Feature Analysis: Unsupervised Learning of Invariances. *Neural Computation*, 14(4):715–770, 2002.

[Yu *et al.*, 2017] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. In *AAAI Conference on Artificial Intelligence*, pages 2852–2858, 2017.