

Time Series and Sequence Learning

Temporal Convolutional Networks, cont'd

Fredrik Lindsten, Linköping University

Temporal Convolutional Network

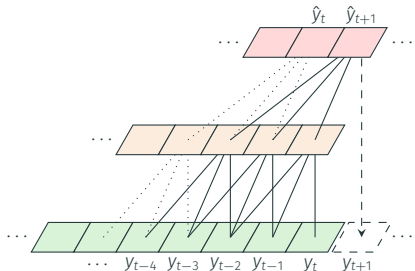
2-layer TCN:

$$h_t^{(0)} = y_t,$$

$$h_t^{(1)} = \sigma(W_t^{(1)} H_t^{(0)} + b^{(1)}),$$

$$y_{t+1} = W_t^{(2)} H_t^{(1)} + b^{(2)} + \varepsilon_{t+1},$$

with $\varepsilon_{t+1} \sim N(0, \sigma_\varepsilon^2)$.

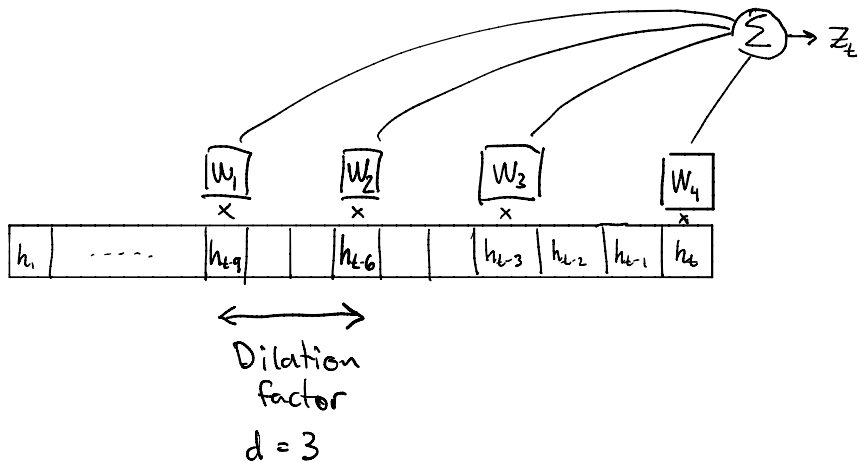


Can extend to multiple layers of hidden signals, $h_t^{(2)}, h_t^{(3)}, \dots$

- ▲ Multiple layers \Rightarrow very flexible models
- ▲ Receptive field increases with depth...
- ▼ ...but only linearly

Dilated convolutions

$$W = [w_1 \ w_2 \ w_3 \ w_4]$$



Dilated convolutions

Formally

$$H_t = \begin{bmatrix} h_{t-(p-1)d} \\ \vdots \\ h_{t-2d} \\ h_{t-d} \\ h_t \end{bmatrix},$$

$$Z_t = W$$

TCN with dilated convolutions

TCN with dilated convolutions:

$$h_t^{(0)} = y_t,$$

$$h_t^{(\ell)} = \sigma(W^{(\ell)} H_t^{(\ell-1)} + b^{(\ell)}), \quad \ell = 1, \dots, L-1,$$

$$y_{t+1} = W^{(L)} H_t^{(L-1)} + b^{(L)} + \varepsilon_{t+1}, \quad \varepsilon_{t+1} \sim N(0, \sigma_\varepsilon^2).$$

Here

$$H_t^{(\ell)} := \begin{bmatrix} h_{t-(p-1)d^{(\ell)}}^{(\ell)} & \cdots & h_{t-d^{(\ell)}}^{(\ell)} & h_t^{(\ell)} \end{bmatrix}^\top$$

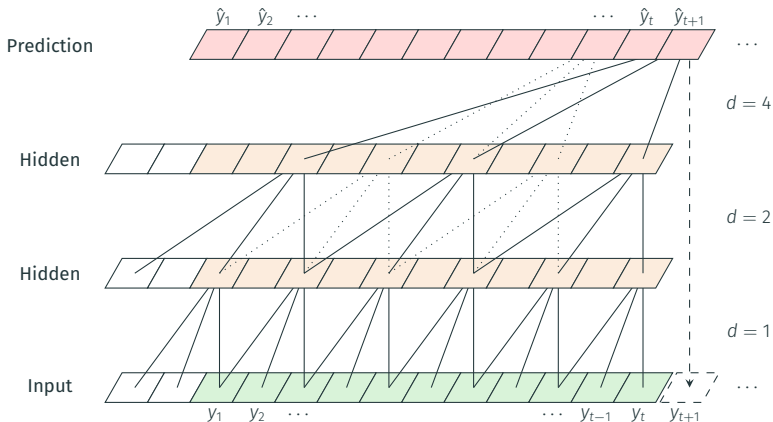
and $d^{(\ell)}$ is the **dilation factor used in layer ℓ** .

Often we let the dilation increase with the depth of the network, e.g.,

$$d^{(\ell)} := 2^\ell.$$

TCN with dilated convolutions

By using **dilated convolutions** we can increase receptive field **exponentially** with depth.



Multi-channel TCN

All the tricks of the trade from deep learning can be used to extend the TCN model!

In particular:

Multi-dimensional hidden signals:

Let $\dim h_t^{(\ell)} = c^{(\ell)} \geq 1$.

- 1-dimension signal of length $n \iff$ “image” of size $n \times 1 \times 1$
- $c^{(\ell)}$ -dimensional signal of length $n \iff$ “image” of size $n \times 1 \times c^{(\ell)}$
- Dimension of convolutional filter $W^{(\ell)} : p \times c^{(\ell-1)} \times c^{(\ell)}$

Residual connections

Residual connections: Direct linear term relating $h_t^{(\ell-1)}$ to $h_t^{(\ell)}$.

$$h_t^{(\ell)} = \sigma(W^{(\ell)} H_t^{(\ell-1)} + b^{(\ell)}) + V^{(\ell)} h_t^{(\ell-1)}.$$

Useful for training deep models — enables “easier flow of gradient information”.

General TCN formulation

More generally, a TCN can be written as:

$$h_t^{(0)} = y_t,$$

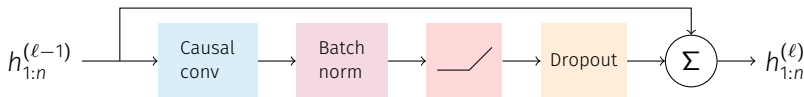
$$h_t^{(\ell)} = g_{\theta}^{(\ell)}(H_t^{(\ell-1)}), \quad \ell = 1, \dots, L-1,$$

$$y_{t+1} = g_{\theta}^{(L)}(H_t^{(L-1)}) + \varepsilon_{t+1}, \quad \varepsilon_{t+1} \sim N(0, \sigma_{\varepsilon}^2).$$

with $H_t^{(\ell)} := \begin{bmatrix} h_{t-(p-1)d^{(\ell)}}^{(\ell)} & \dots & h_{t-d^{(\ell)}}^{(\ell)} & h_t^{(\ell)} \end{bmatrix}^T$ for

arbitrary nonlinear functions $g_{\theta}^{(1)}, \dots, g_{\theta}^{(L)}$.

ex) One possible residual block of a TCN





WaveNet by DeepMind powers Google's text-to-speech technology.



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

WAVENET: A GENERATIVE MODEL FOR RAW AUDIO

Aaron van den Oord Sander Dieleman Heiga Zen
Karen Simonyan Oriol Vinyals Alex Graves
Nal Kalchbrenner Andrew Senior Koray Kavukcuoglu

Google DeepMind, London, UK
Google, London, UK

ABSTRACT

This paper introduces WaveNet, a deep neural network for generating raw audio waveforms. The model is fully probabilistic and autoregressive, with the predictive distribution for each audio sample conditioned on all previous samples. Additionally, we show that it can be efficiently trained on data with rates of thousands of samples per second of audio. When applied to text-to-speech, it yields state-of-the-art performance, with human listeners rating it as significantly more natural sounding than the best previous neural and concatenative systems for both English and Mandarin. A single WaveNet can capture the characteristics of many different speakers with equal fidelity, and can switch between them by conditioning on the speaker identity. When trained to model music, we find that it generates novel and often highly realistic musical fragments. We also show that it can be employed as a discriminative model, returning processing results for phoneme recognition.

1 INTRODUCTION

This work explores raw audio generation techniques, inspired by recent advances in neural autoregressive generative models that model complex distributions such as images (van den Oord et al., 2016) and text (Burgun et al., 2016). Modeling raw audio waveforms presents a particularly challenging task, as natural audio is composed of complex conditional distributions, yielding state-of-the-art performance.

Remarkably, these architectures are able to model distributions over thousands of variables (e.g. 48-kHz audio) as in PixelRNN (van den Oord et al., 2016a). The question this paper addresses is whether similar approaches can succeed in generating realistic raw audio waveforms, which are signals with very high temporal resolution, at least 16,000 samples per second (see Fig. 1).



Figure 1: A second of generated speech.

This paper introduces WaveNet, an audio generative model based on the PixelRNN (van den Oord et al., 2016b) architecture. The main contributions of this work are as follows:

- We show that WaveNet can generate raw speech signals with subjective naturalness never before reported in the field of text-to-speech (TTS), as assessed by human scores.

- In order to deal with long-range temporal dependencies needed for raw audio generation, we develop new architectures based on dilated causal convolutions, which exhibit very large receptive fields.
- We show that when conditioned on a speaker identity, a single model can be used to generate different voices.
- The same architecture shows strong results when trained on a small speech recognition dataset, and is promising when used to generate other audio modalities such as music.

We believe that WaveNet provides a generic and flexible framework for tackling many applications that rely on audio generation (e.g. TTS, music, speech enhancement, voice conversion, source separation).

2 WAVENET

In this paper we introduce a new generative model operating directly on the raw audio waveform. The joint probability of a waveform $x = (x_1, \dots, x_N)$ is factored as a product of conditional probabilities as follows:

$$p(x) = \prod_{n=1}^N p(x_n | x_{1:n-1}) \quad (1)$$

Each audio sample x_n is therefore conditioned on the samples at all previous timesteps.

Statistically, in PixelRNN (van den Oord et al., 2016b), the conditional probability distribution is modeled by a stack of convolutional layers. There are no pooling layers in the network, and the output of the model has the same time dimensionality as the input. The model outputs a categorical distribution over the next value x_n , with a softmax layer and it is optimized to maximize the log-likelihood of the data x . Additionally, because log-likelihoods are intractable, we use log-likelihoods as a validation set and can easily measure if the model is overfitting or underfitting.

2.1 DILATED CAUSAL CONVOLUTIONS

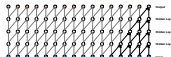


Figure 2: Visualization of a stack of causal convolutions layers.

The main ingredient of WaveNet are causal convolutions. By using causal convolutions, we make sure the model cannot violate the ordering in which we model the data: the prediction of x_{n+1} is only conditioned on the samples x_1, \dots, x_n and not on any of the future samples x_{n+2}, \dots, x_N as shown in Fig. 2. For instance, the equivalent of a causal convolution in a neural convolution (van den Oord et al., 2016b) can be implemented by constructing a mask matrix and doing an elementwise multiplication of this mask with the convolution kernel before applying it. For 1-D data such as audio one can more easily implement this by shifting the output of a neural convolution a few time steps.

By training time, the conditional predictions for all timesteps can be made in parallel because all timesteps of generated audio x are known. When generating with the model, the predictions are sequential: after each sample x_n is predicted, it is fed back into the network to predict the next sample.

Because models with causal convolutions do not have recurrent connections, they are typically faster to train than RNNs, especially when applied to very long sequences. One of the problems of causal convolutions is that they require many layers, or large filters to increase the receptive field. For example, as in Fig. 2 the receptive field is only 5 in Figure 2 (filter length = 5). In this paper we use dilated convolutions to increase the receptive field by orders of magnitude, which greatly reduces the computational cost.

A dilated convolution (also called a 2D or, convolution with holes) is a convolution where the filter is applied over a sub-sampled set of inputs by skipping pixels with a certain step. It is equivalent to convolution with a larger filter derived from the original filter by dilating it with zeros, but the filter is more efficient. A dilated convolution (also called a 2D or, convolution with holes) is a convolution where the filter is applied over a sub-sampled set of inputs by skipping pixels with a certain step. It is equivalent to convolution with a larger filter derived from the original filter by dilating it with zeros, but the filter is more efficient. A dilated convolution (also called a 2D or, convolution with holes) is a convolution where the filter is applied over a sub-sampled set of inputs by skipping pixels with a certain step. It is equivalent to convolution with a larger filter derived from the original filter by dilating it with zeros, but the filter is more efficient.

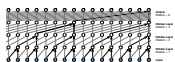


Figure 3: Visualization of a stack of dilated causal convolutions layers.

Stacked dilated convolutions enable networks to have very large receptive fields with just a few layers, while preserving the input resolution throughout the network as well as computational efficiency. In this paper, the dilation is doubled for every layer up to a limit and then kept constant.

1, 2, 4, ..., 32, 64, 128, ..., 512, 1024, ..., 512.

The intuition behind this configuration is two-fold. First, exponentially increasing the dilation factor results in a very efficient receptive field growth (see Graves & Kallman, 2016). For example, steps 1, 2, 4, ..., 512 block the receptive field of size 1024 and can be used as a more efficient and faster alternative (see Graves & Kallman, 2016). Second, stacking these blocks further increases the model capacity and makes it more robust to overfitting.

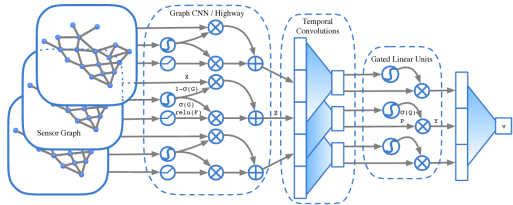
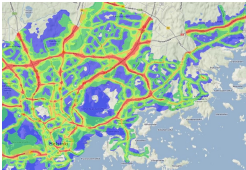
2.2 STOCHASTIC DISTRIBUTIONS

One approach to modeling the conditional distribution $p(x_n | x_{1:n-1})$ over the individual audio samples would be to use a mixture model such as a mixture density network (Burgun, 1990) or a mixture of conditional Gaussian scale mixtures (MCMCM) (Doris & Bagnall, 2015). However, van den Oord et al. (2016b) showed that a mixture density network is not flexible enough to model the distribution of raw audio samples. In this paper, we use a mixture model to model the distribution of raw audio samples. In this paper, we use a mixture model to model the distribution of raw audio samples.

Because raw audio is typically stored as a sequence of 16-bit integer values (one per time-step), a softmax layer would need to output 65,536 probabilities per time-step to model all possible values. To make this more tractable, we first apply a p -log companding transformation (ITU-T, 1988) to the data, and then quantize it to 256 possible values:

$$f(x_n) = \text{sign}(x_n) \cdot \frac{\log(1 + p|x_n|)}{\log(1 + p)}$$

ex) Traffic prediction



 Spatio-temporal Graph Convolutional Neural Network: A Deep Learning Framework for Traffic Forecasting. **Bing Yu, Haoteng Yin, Zhanxing Zhu** *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

Deep Learning for time series analysis

So is this what we should always do for modeling sequential data?!

No!

- Only makes sense to use something as complex as TCN when classical methods fail — **Try Simple Things First!**
- Methods based on deep learning work best if we have multiple sequences, or one long sequence that can be split into segments
- For a single univariate time series, classical methods (AR, state space, NAR, ...) often work better.