Autoregressive Models

The *WaveNet* Architecture; `with code*`

Christoph Heindl

12/2021

*https://github.com/cheind/autoregressive

WaveNet*

Wavenet: A generative model for raw audio.
Aaron van den Oord, et al.
@deepmind, 2016

Contributions

- Generative model for wave-form forms
- Capable of capturing important audio structure at many time-scales
- Conditioning support

Led to the **most natural-sounding** speech/audio synthesis at the time.

*https://arxiv.org/abs/1609.03499

Content

This talk covers

- an introduction to autoregressive models and some of their limitations,
- the architectural ideas to overcome those limitations, and
- few of existing improvements.

This talk is not

- about audio/speech (we use time series instead),
- a comprehensive state-of-the-art presentation on generative models.

Accompanying code: https://github.com/cheind/autoregressive

Background

Generative Models

Generative models build a distribution over the data itself. Consider a set of random variables

$$\mathbf{X} = \{X_1, X_2, X_3\},$$

then a generative model estimates

$$p(\mathbf{X}).$$

Given the joint distribution, we can generate *new* data via sampling

$$\mathbf{x} \sim p(\mathbf{X}).$$

In contrast, a discriminative models models conditional distributions, e.g. $p(X_3 \mid X_2, X_1)$.

Chain Rule of Probability

Allows us to break down $p(\mathbf{X})$ into a product of single-variable conditional distributions

$$\begin{aligned} p(\mathbf{X}) &= p(X_3 \mid X_2, X_1)p(X_2 \mid X_1)p(X_1) \\ &= p(X_1 \mid X_2, X_3)p(X_3 \mid X_2)p(X_2) \\ &\dots \end{aligned}$$

Autoregressive Models

Autoregressive Models

Given a set of (time-)ordered random variables $\mathbf{X} = \{X_1, X_2, X_3..., X_T\}$, we represent their joint distribution as

$$\begin{aligned} p(\mathbf{X}) &= \prod_{i=1}^{T} p(X_i \mid \mathbf{X}_{j<i}) \\ &= p(X_1)p(X_2 \mid X_1)p(X_3 \mid X_2, X_1)\ldots. \end{aligned}$$

This induces a form of **causality**, as the distribution over a future variable depends on all previous observations. It also allows us to generate *new* data one sample point at a time (conditioned on all the previous ones).

Lagged Autoregressive Models

For computational reasons, one usually limits the number of past observations influencing future predictions. An autoregressive model of order/lag/receptive-field R is defined as

$$X_t \,|\, \mathbf{X}_{j<t} = \theta_0 + \sum_{i=1}^{R} \theta_i X_{t-i} + \epsilon_t,$$

where $\theta = \{\theta_0, ..., \theta_R\}$ are the parameters of the model and ϵ_t is (white) noise.

Translation to Neural Networks

The definition of autogressive models can be captured by a single fully connected neural layer

$$\begin{aligned} X_t \mid \mathbf{X}_{j<t} &= \theta_0 + \sum_{i=1}^{R} \theta_i X_{t-i} + \epsilon_t \\ &= \theta^T \mathbf{h}_t + \epsilon_t, \end{aligned}$$

where $\theta = \begin{pmatrix} \theta_0 & \theta_1 & \dots & \theta_R \end{pmatrix}$ are the weights including the bias, and $\mathbf{h}_t = \begin{pmatrix} 1 & X_{t-1} & \dots & X_{t-R} \end{pmatrix}$.

Deep models

For more model capacity, one might stack layers having multiple features, in which case we get something along the following line

$$\mathbf{H}_t^l = \sigma \left(\mathbf{\Theta}^l \mathbf{H}_t^{l-1} + \mathrm{E}_t^l \right),$$

where σ is a non-linearity and subscript l denotes the l-th layer.

Limitations

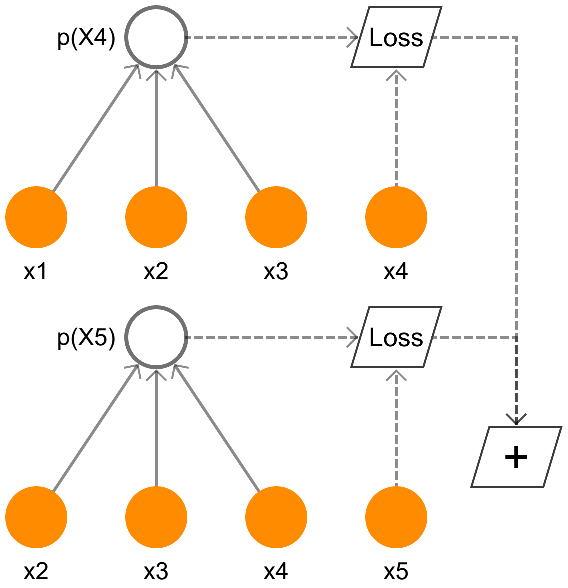
1. **Training** with linear layers is **inefficient** as autoregressive value needs to be computed for every possible window of size R.
2. The **number of weights** grows linearily with the receptive field of the model. For multi-time scale models (speech, audio) this becomes quickly an issue.

WaveNet

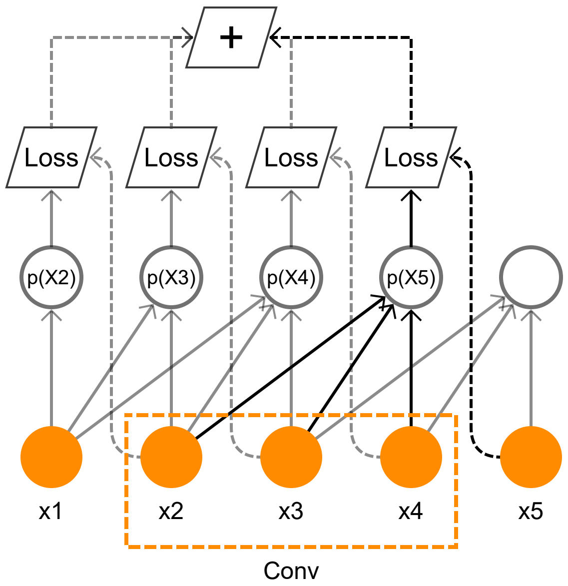
Convolutions: Improving Training Efficiency

Interpret $X_t \mid \mathbf{X}_{j<t}$ in terms of convolution. Allows for a fully-convolutional computation of all X_t in one sweep. Below illustration is for a model of $R = 3$.

Fully Connected Approach



Convolutional Approach

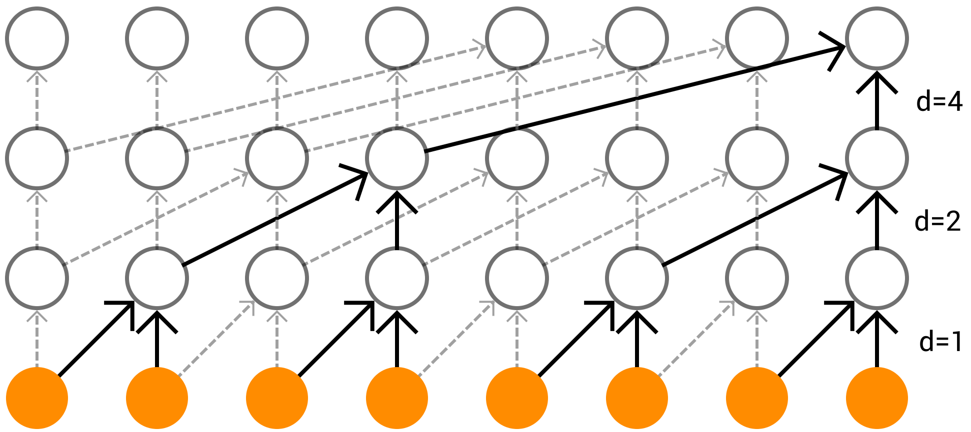


Need to be careful about (see Causal Padding slides)

- Ensure no data leakage happens (i.e input restricted to $\mathbf{x}_{j<t}$)
- How to handle variables X_t, where $t < R$

Dilated Convolutions: Exponential Receptive Fields

Receptive field of dilated convolutions grows exponentially while parameters increase only linearly. Figure below uses kernel size $K_i = 2$.



In general, each layer with dilation factor D_i and kernel size K_i adds

$$r_i = (K_i - 1)D_i$$

to the receptive field $R = \sum_i r_i + 1$.

Note, how each input (orange) within the receptive field is used exactly once.

Dilated Convolutions: Number of parameters

Assume kernel size $K_i = 2$ and a receptive field of $R = 512$. Then a vanilla convolution requires

$$R_{\text{vanilla}} = 512 \text{ parameters}$$

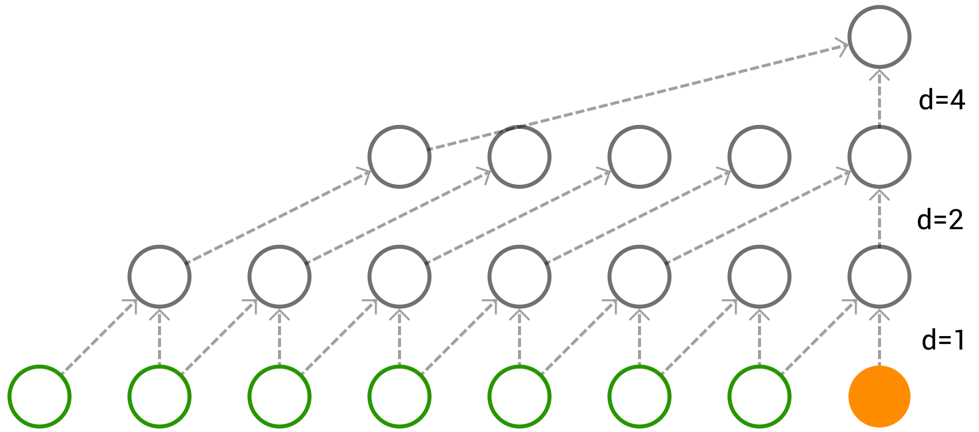
(without a bias), while a stacked dilated convolution requires

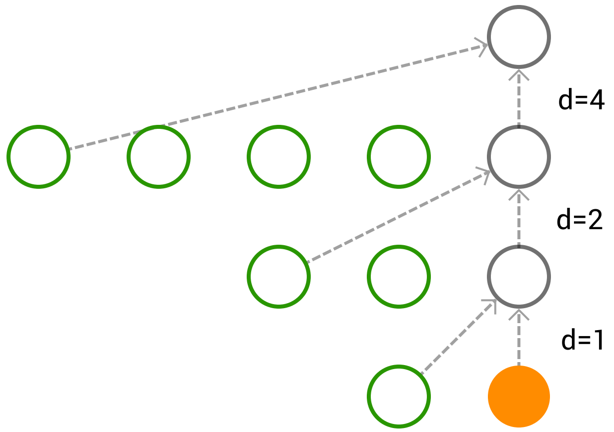
$$R_{\text{dilated}} = 2 * 9 = 18 \text{ parameters.}$$

Note: stacked dilated convolutions make use of all 512 inputs.

Causal Padding

Causal padding (left-padding) ensures that convoluted features do not depend on future values and allows us to compute predictions for X_t, where $t < R$. Two possibilities: input-padding (left), layer-padding (right)



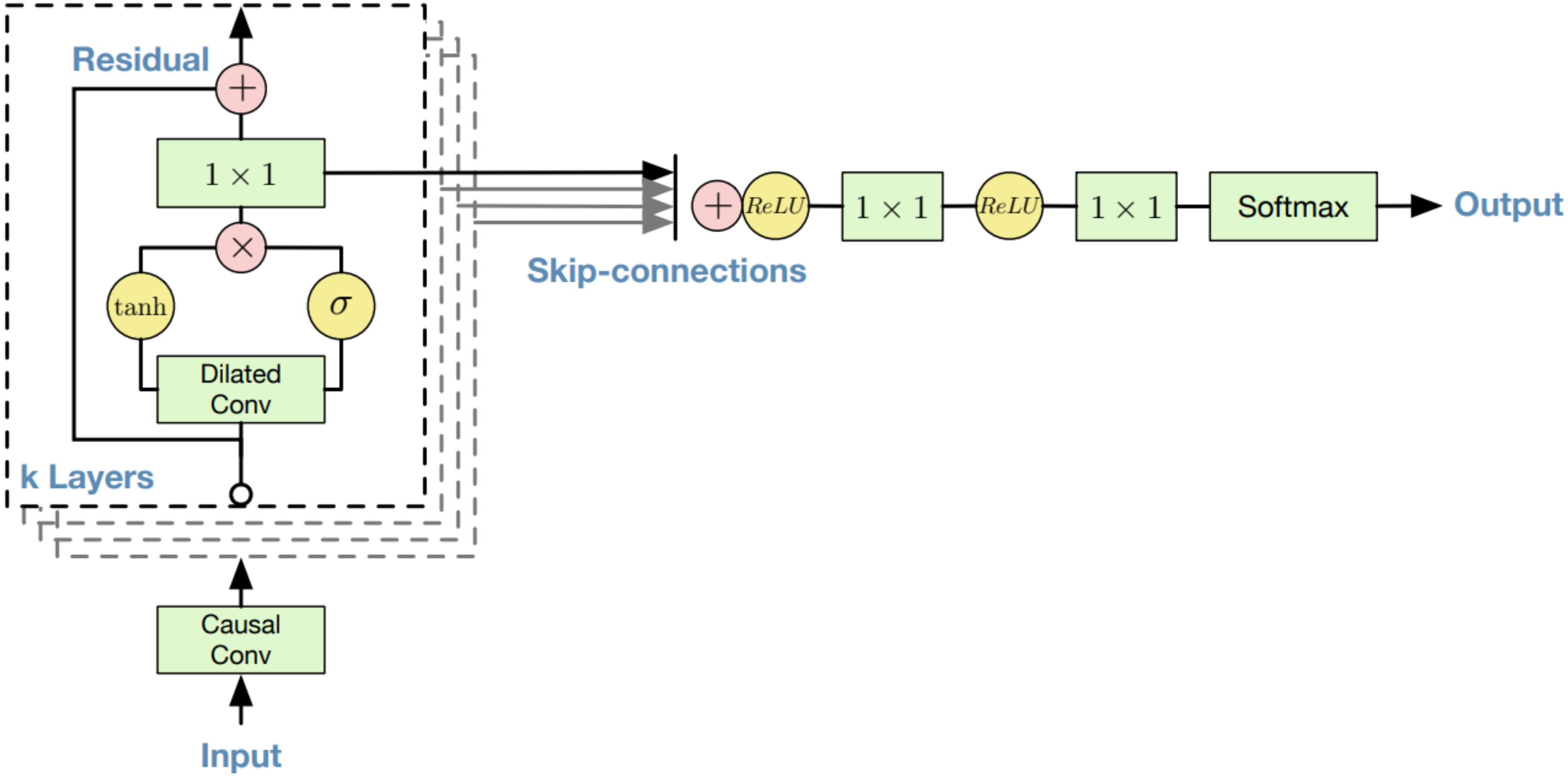


In general, a total of $P = R - 1$ padding values are required.

Autoregressive library uses layer-padding, WaveNet paper suggest input padding.

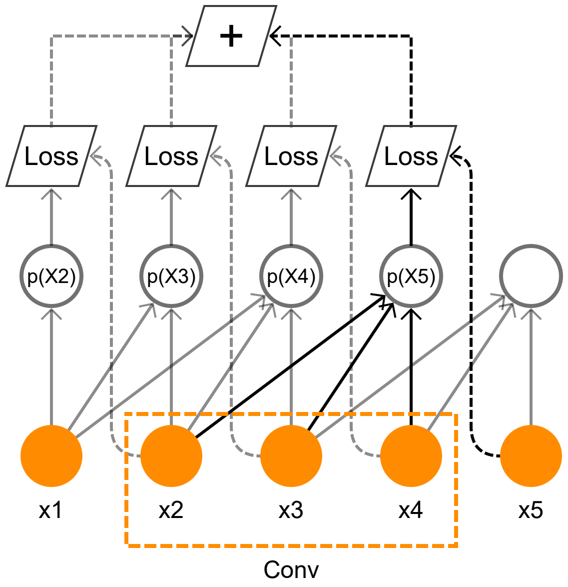
Full Architecture

WaveNet combines stacked dilated convolutions, causal padding and gated activation functions to predict a categorical distribution for $X_t|\mathbf{X}_{j<t}$ in parallel.



Training

Paper performs a one-step rolling origin training routine using cross entropy as loss function.

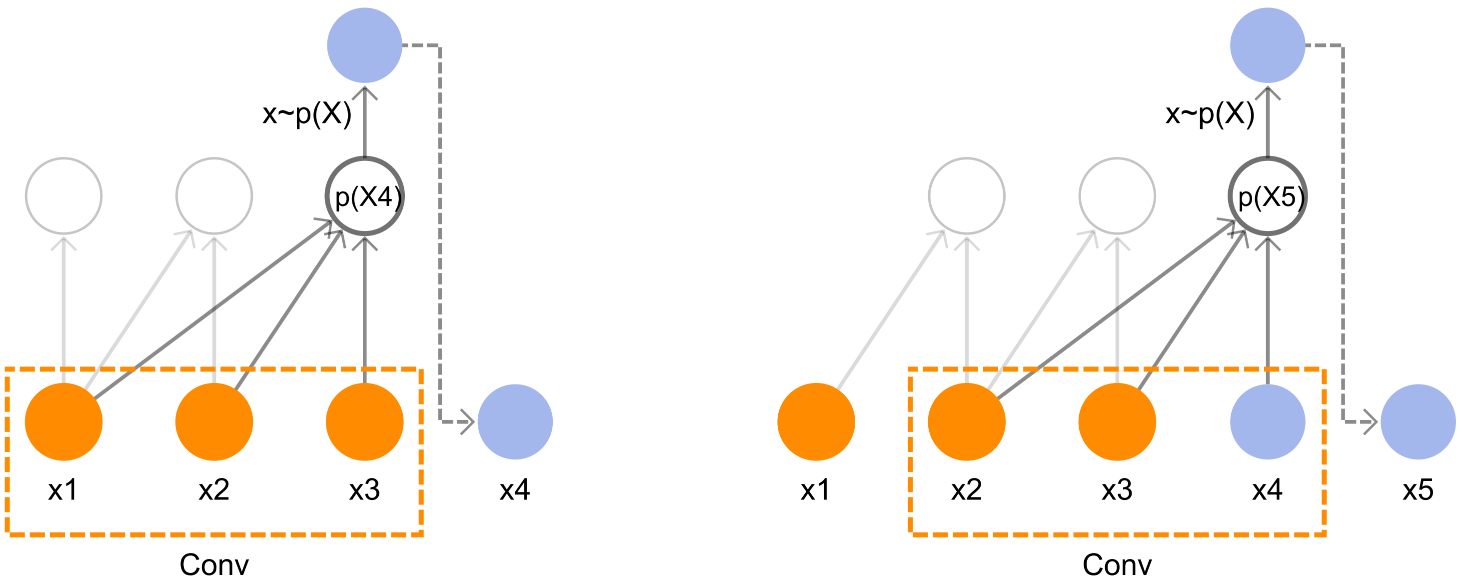


Raw audio data is quantized to 256 bins and one-hot encoded ($X_t \sim Cat(\pi_1, ..., \pi_{256})$).

Side note: one-step loss does not account for generative n-step drift (which is probably ok for audio synthesis).

Data Generation

New data is generated one sample at a time. The figure below shows two steps for a model with $R = 3$



Remarks:

- Generation is inefficient - requires R inputs but uses only the last output.
- Generation involves sampling from the distribution.

Extensions

Conditional WaveNets*

Condition the model on additional external input

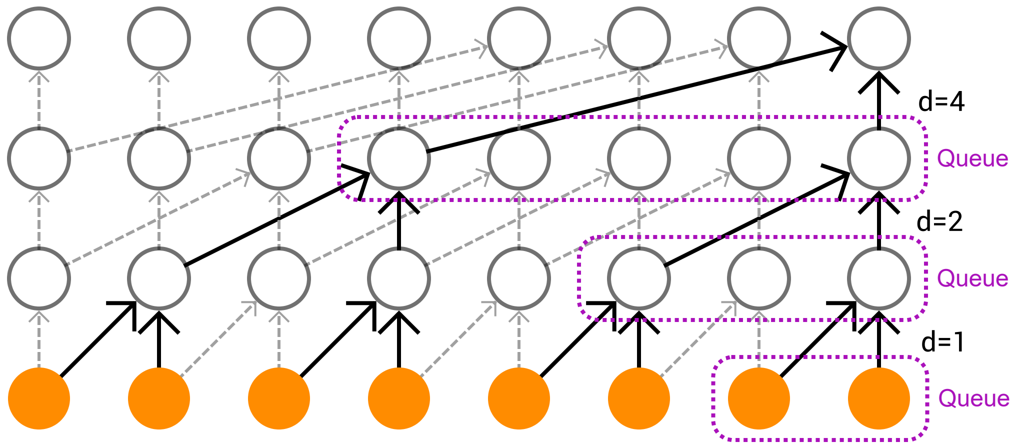
$$
\begin{aligned}
p(\mathbf{X} \mid \mathbf{y}) &= \prod_{i=1}^{T} p(X_i \mid \mathbf{X}_{j<i}, \mathbf{y}) \\
&= p(X_1 \mid \mathbf{y}) p(X_2 \mid X_1, \mathbf{y}) p(X_3 \mid X_2, X_1, \mathbf{y}) \ldots .
\end{aligned}
$$

to change generative behavior. For example $\mathbf{y}$ might represent speaker identity in which case the model would generate data wrt. the given speaker.

*Wavenet: A generative model for raw audio, Aaron van den Oord, et al., 2016

Faster Generation*

Relies on sparsity of access during computation. Introduce *queues* (i.e rolling buffers of size $r_i + 1$) to store intermediate outputs. During generation only use oldest in queue and update queue.



Similar to updates in recurrent neural nets.

*Fast WaveNet Generation Algorithm, Tom le Paine et al., 2016.
For even faster generation check Parallel WaveNet: Fast High-Fidelity Speech Synthesis, Aaron van den Oord et al., 2017.

Train Unrolling*

In training, WaveNet uses a one-step rolling-origin loss which can causes substantial drift.

Idea

A n-step loss would allow the model to correct its own drift.
I.e we want to apply n-step generation and backprop through all samples.

Issue

How to backprop through a random sample from a categorical distribution?

*TBD, Christoph Heindl, 2021 (unpublished)

Fully Differentiable Train Unrolling

Reparametrization Idea

Note if $X_t \sim \mathcal{N}(\mu, \sigma)$, which we can express as $X_t \sim \mathcal{N}(0,1)\sigma + \mu$.

Now $\frac{\partial}{\partial \mu}$, $\frac{\partial}{\partial \sigma}$ exist and randomness becomes an input (for which we do not require gradients).

Reparametrization of Categorical Distributions

Similar reparametrization exists for $X_t \sim \mathcal{C}at(\pi_1, \ldots, \pi_C)$ using Gumbel distribution*, which allows us to write

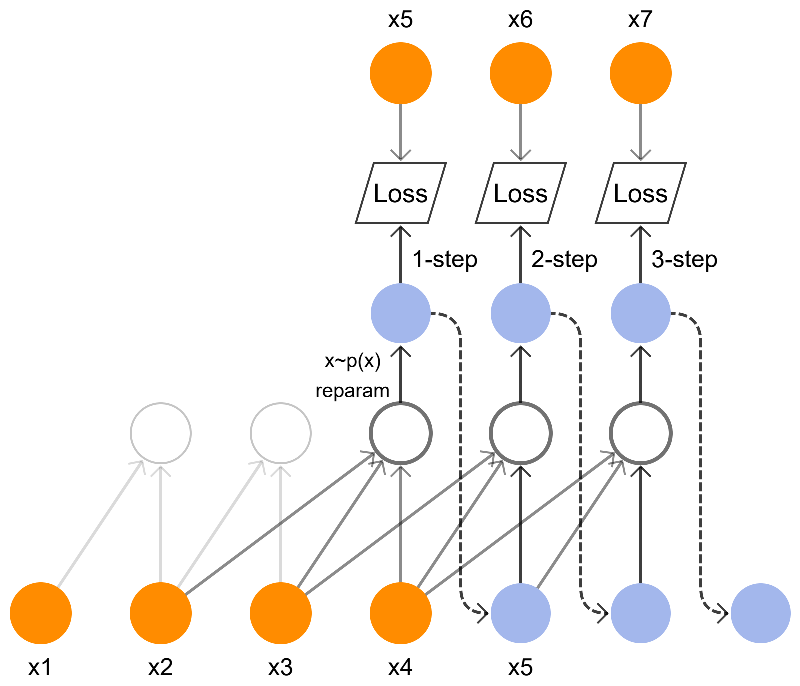
$$X_t \sim g(Gumbel(0,1), \pi_1, \ldots, \pi_C, \tau),$$

such that $\frac{\partial g}{\partial \pi_i}$ exists. Here τ is a temperature scaling parameter.

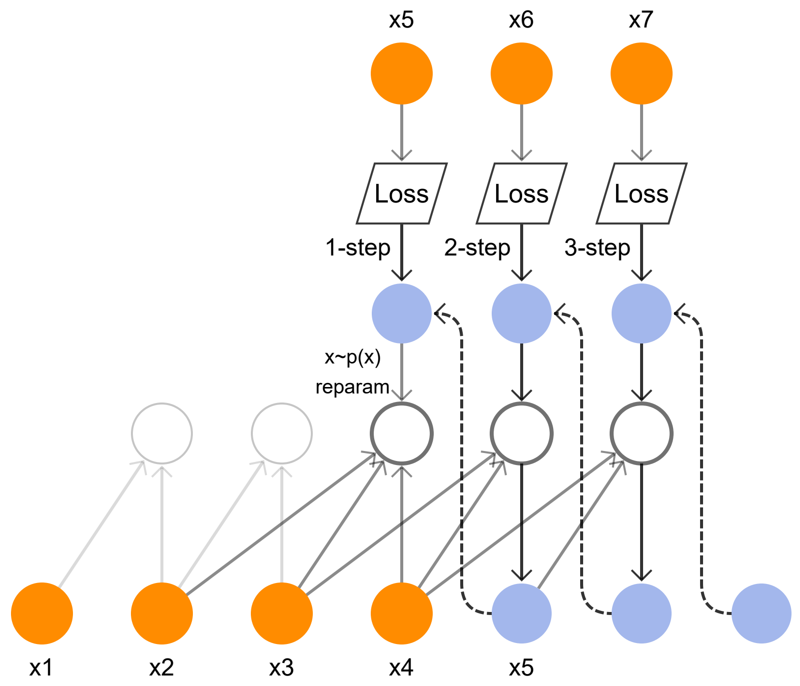
Categorical Reparameterization with Gumbel-Softmax, Eric Jang et al., 2017
The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables, C. Maddison et al., 2017 -

Fully Differentiable Train Unrolling

Forward pass



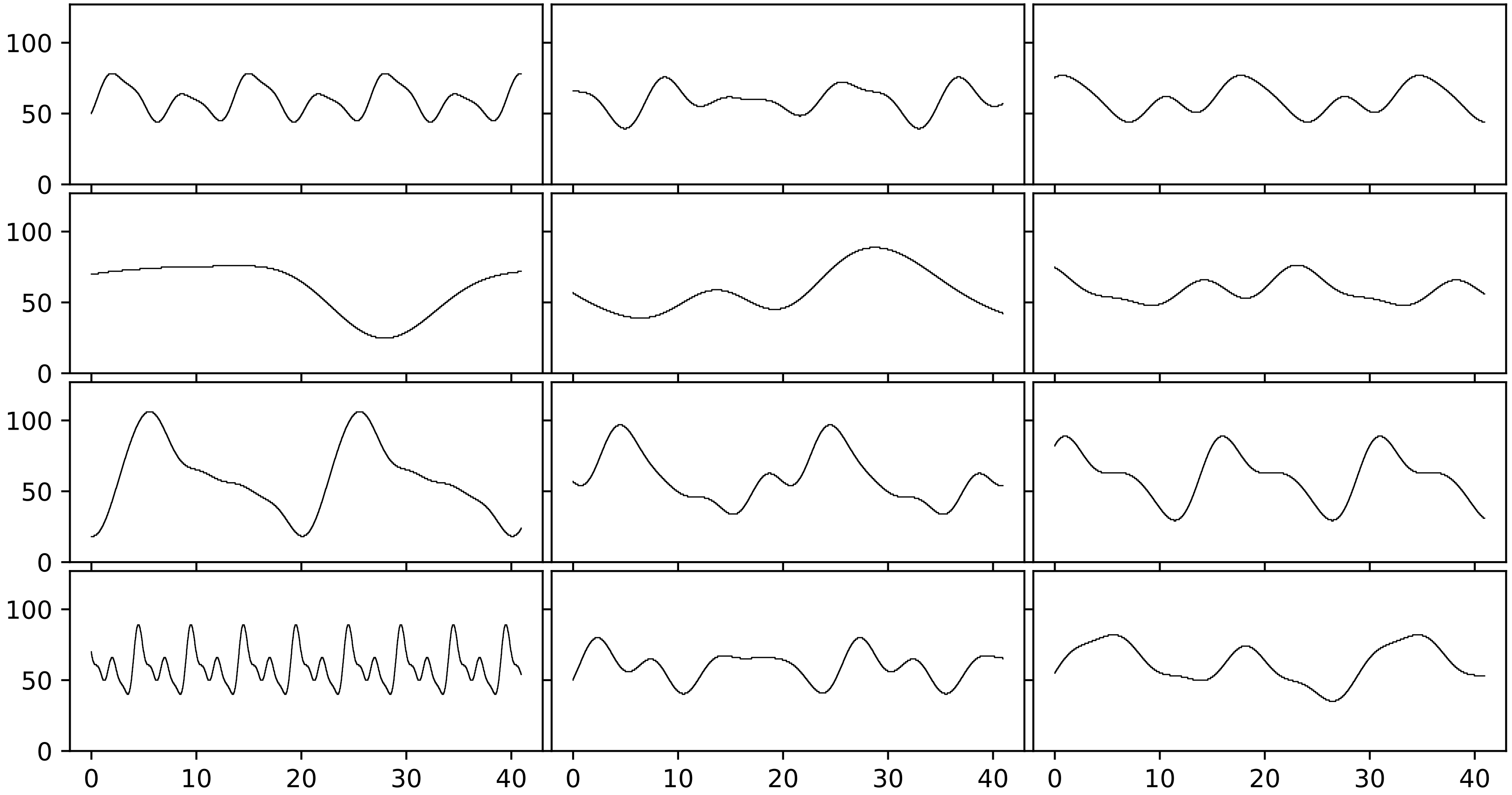
Backward pass



Experiments

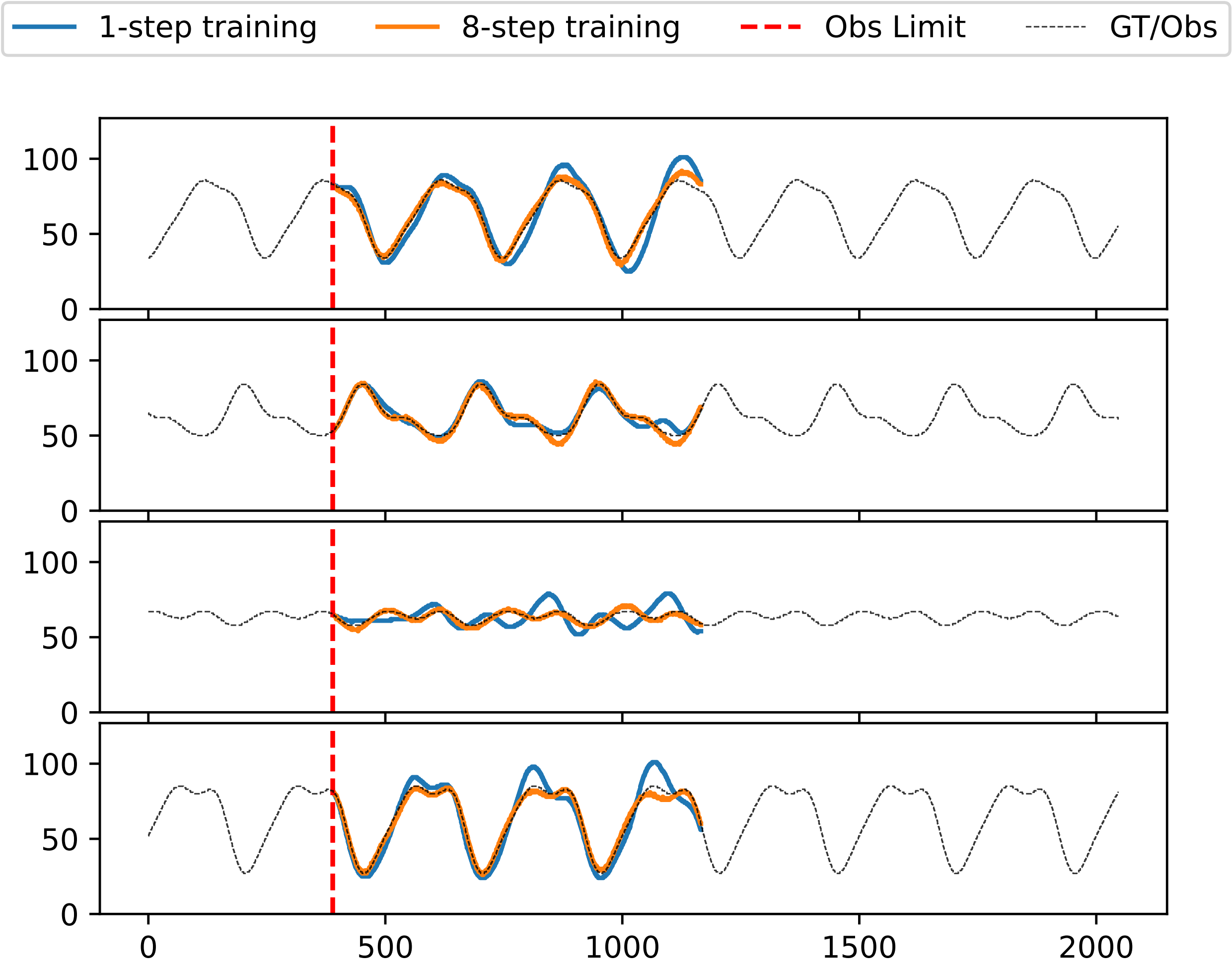
Setup

Instead of audio waveforms as input, using a Fourier dataset with randomized coefficients, number of terms and periodicity (sampling: 50Hz, quantization: 127 bins, encoding one-hot)



Train-Unrolling Results

N-step forecast comparison between two models trained with and without unrolling on Fourier-series dataset with up to 4 terms.



Conclusion

(+) Decreases generative drift

(+) Improves recreation of higher frequency patterns

(-) Increases training time (rolling origin)

(-) Sparser losses

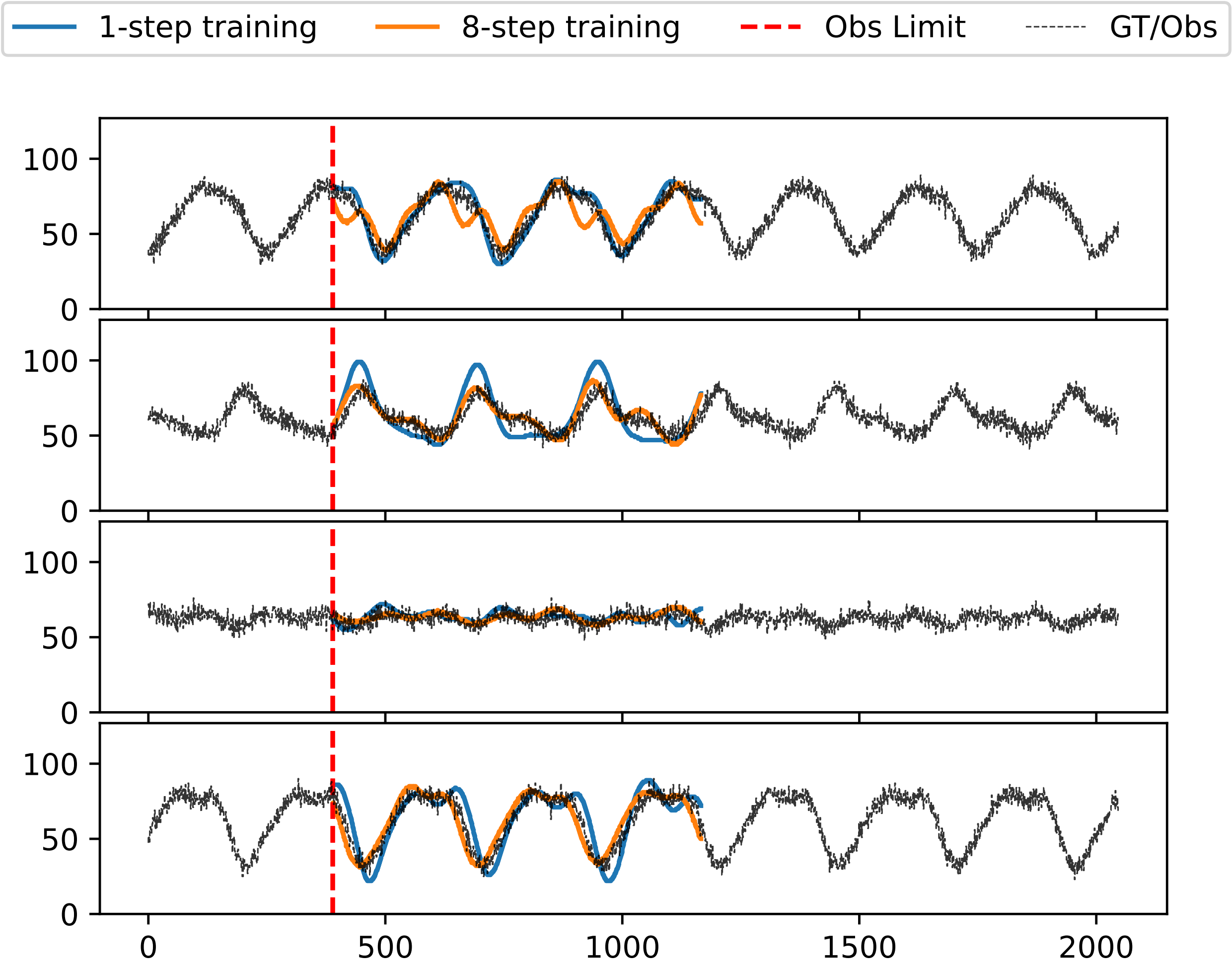
Noisy Train-Unrolling Results

N-step prediction based on noisy observations - comparison between two models trained with and without unrolling on a clean Fourier series dataset with up to 4 terms.

Conclusion

(+) Both models capture global trends

(-) Accuracy of both modes decreases

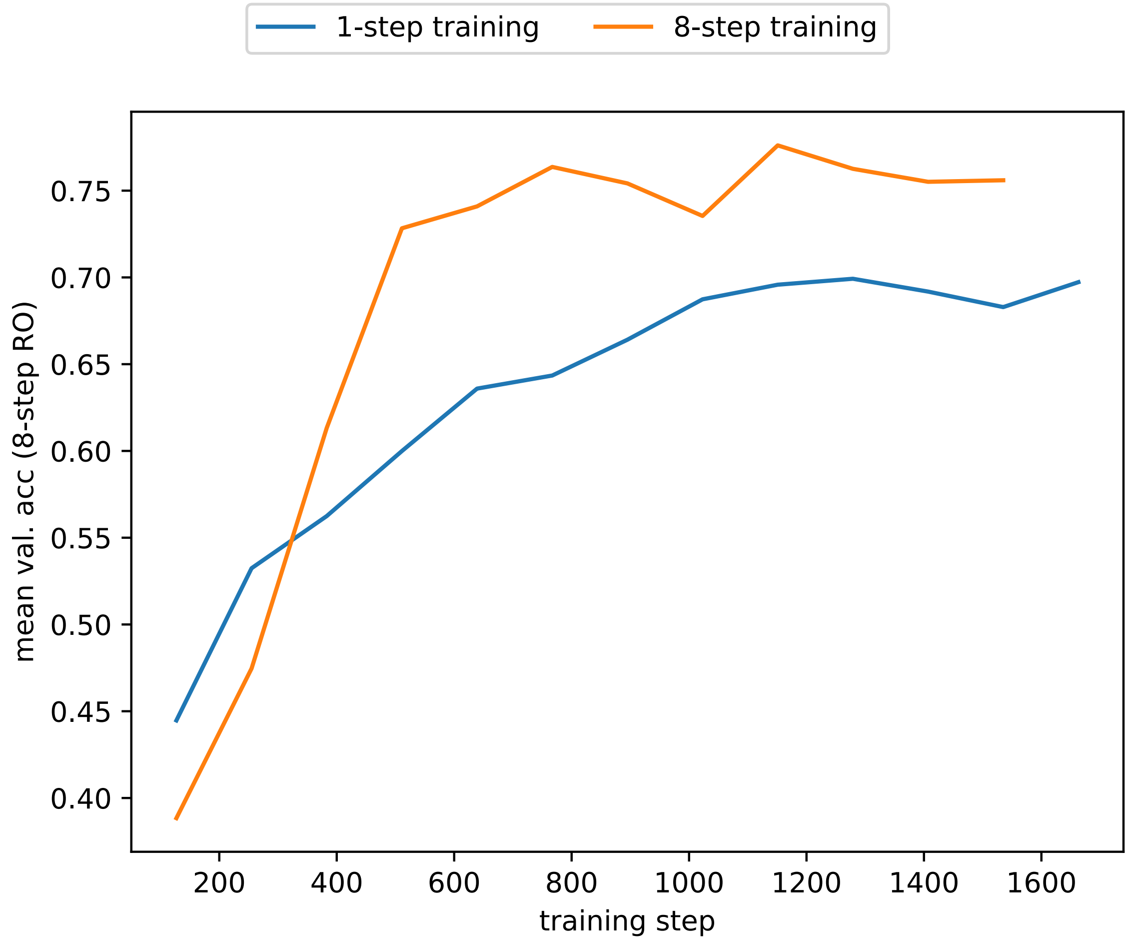


Train-Unrolling Validation Acc. Results

8-step rolling origin validation comparison between models trained with and without unrolling on Fourier-series dataset with up to 4 terms.

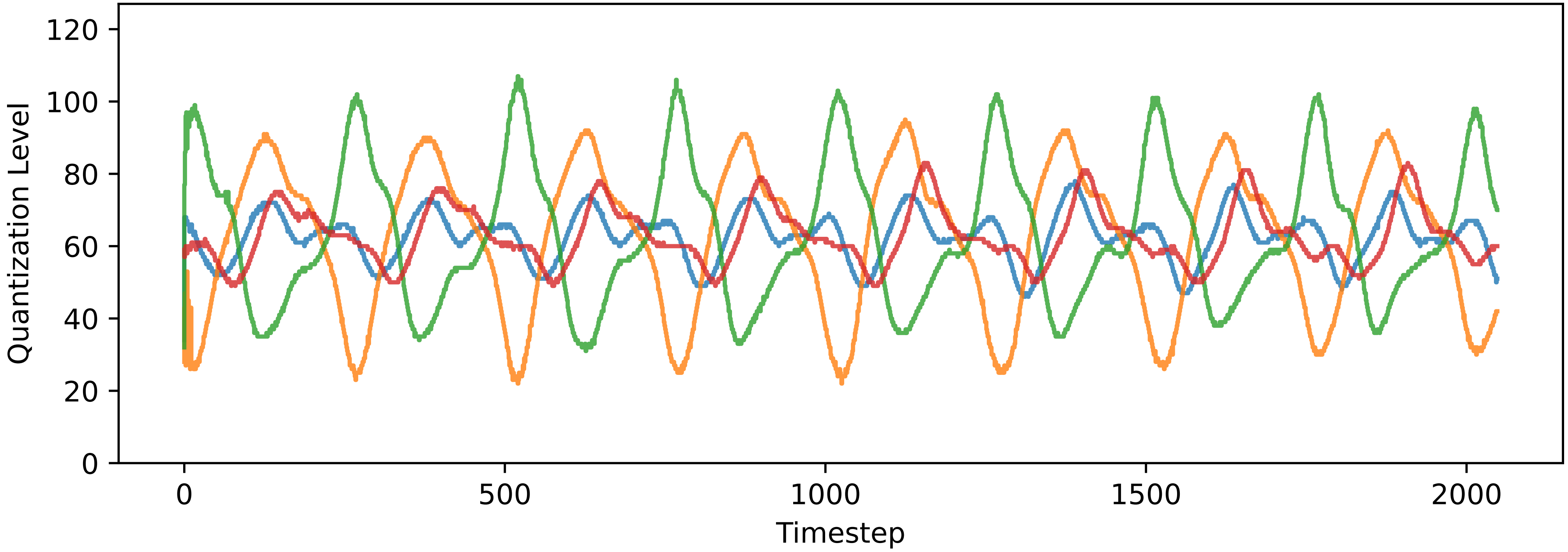
Conclusion

(+) Generally higher validation acc. at earlier training epochs.
(+) Similar picture if validation unrolling > train unrolling steps.



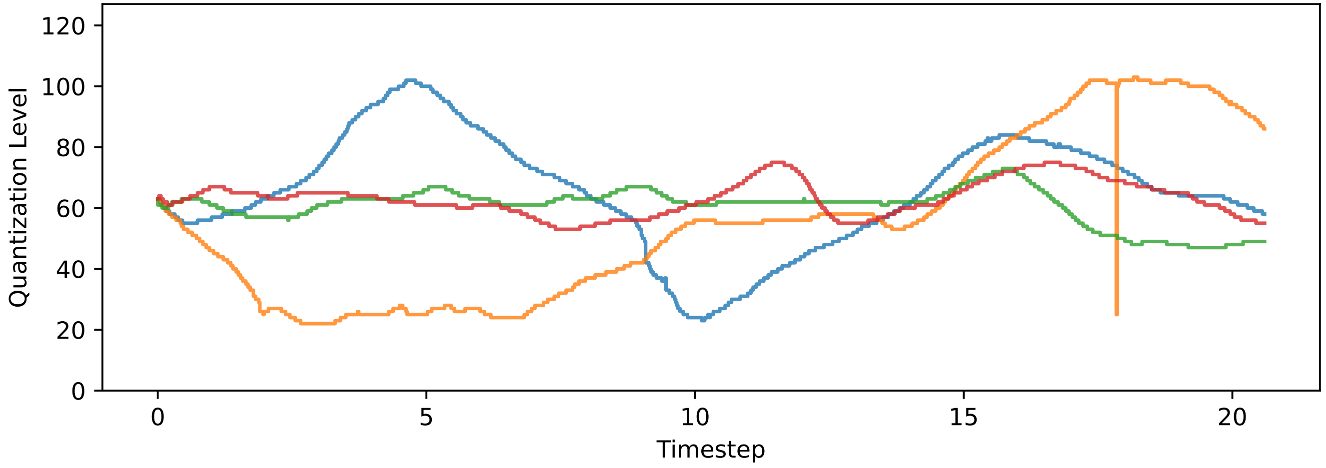
Generative Results

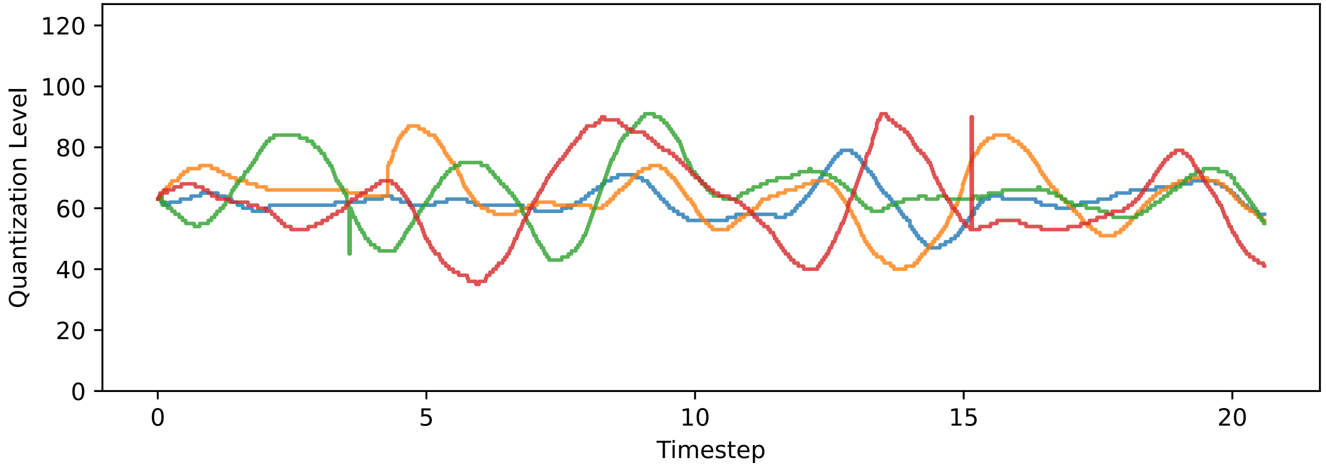
The following graph shows four samples drawn from the models' prior distribution (periodicity fixed in training).



Conditional Generative Results

The following graphs depict samples using different periodicity conditions: Large period (~20secs), short periods (~5secs). Model trained without unrolling.



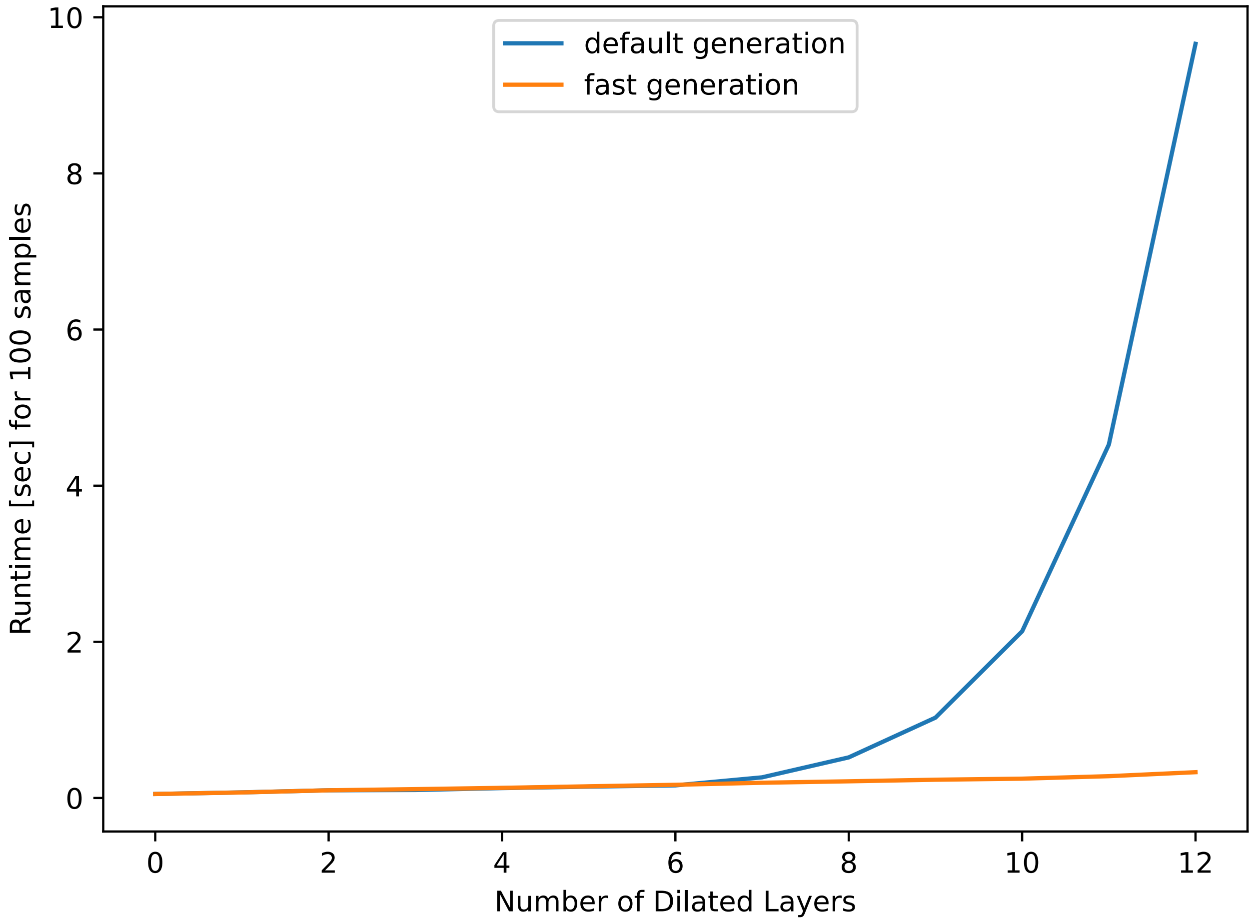


Runtime Performance Results

The plot to the left shows default (blue) and fast (orange) sample generation* using 64 wave-channels, 8 quantization levels and 32 batch-size.

Conclusion

(+) Fast method avoids exponential inference time as layer depth increases.
(-) Code overhead is considerable.



*Performed on a 1080 Ti