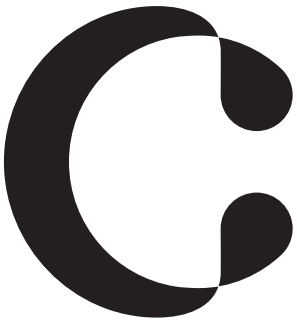

Corti

Paper Summaries

Some Joke About VAEs

Written @ Corti

Authors:
Magnus Berg Sletfjerding
{ms}@corti.ai
May 17, 2021

Abstract

This was written for me to understand papers in my thesis better. Don't be alarmed if you don't understand it 100%, I probably don't either.

Contents

Distribution	Equation
Generative model	$p(\mathbf{x}) = p_\theta(\mathbf{x}\|\mathbf{z})p_\theta(\mathbf{z})$
Prior over $\mathbf{z}$	$p_\theta(\mathbf{z})$
Likelihood over $\mathbf{x}$	$p_\theta(\mathbf{x}\|\mathbf{z})$
Posterior over $\mathbf{z}$	$p_\theta(\mathbf{z}\|\mathbf{x})$
Posterior approximation over $\mathbf{z}$ $q_\psi(\mathbf{z}\|\mathbf{x})$	

Table 1: Overview of the different definitions in the VAE

0 Auto Encoding Variational Bayes

Authors: **Diederik P. Kingma, Max Welling** [3]

Main contributions of this paper is as follows:

1. The reparametrization trick for SDG methods
2. The first ”vanilla” VAE

0.1 Method and setup

Assume a directed graphical model able to do both inference and running a pure generative process.

We define the problem as the following: Given a generative model $p_\theta(x)$, we want to find the optimal parameters for θ.

$$p(\mathbf{x}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z}) \tag{1}$$

1 WaveNet

The WaveNet paper presents a CNN-based approach to generating audio samples. [5] Instead of using RNNs as a recurrent architecture, the generative model only conditions on past samples, and as such does not include any hidden "state".

The probability of a waveform $\mathbf{x} \in \mathbb{R}^T$ is expressed purely as:

$$p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t | x_1, ..., x_t) \tag{2}$$

where $p(x_t|x_1, ..., x_t)$ is parametrized only by the weights in the network.

1.1 Architecture and design

The WaveNet Architecture draws advantage from three developments: quantized output spaces (as shown in PixelRNN), dilated causal convolutions and gated activation units,

Quantized Output Space with μ law companding transformation Given an audio waveform $\mathbf{x} \in [-1, 1]^T$, transform the audio according to :

$$f(x_t) = \text{sign}(x_t) \frac{\ln(1 - \mu|x_t|)}{\ln(1 + \mu)} \tag{3}$$

with $\mu = 255$.

Dilated Causal Convolutions A Causal Convolution is a fancy way of saying that audio convolutions only work forward in time, not backward. This is to enforce the forward dependency in eq. (2).

A Dilated Convolution is a convolution where the convolution kernel skips over a dimension, increasing the receptive field and observing more of the surrounding environment. For an image the simplest dilated convolutional is illustrated in fig. 1

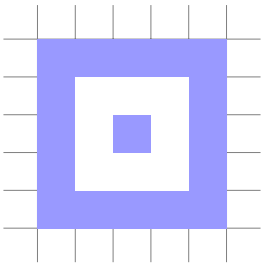

Figure 1: A Simple Pixel Dilated Convolution

Accordingly, for an audio signal, it would look like what we see in fig. 2

Gated Activation Units Each Convolution layer, Insteadof just having a filter weight, also has a **gating weight**. Hence the weights $\mathbf{W} \in \mathbb{R}^{K \times 2}$, with K as the number of layers. The operation of layer $k \in [0, K]$, is parametrized as:

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

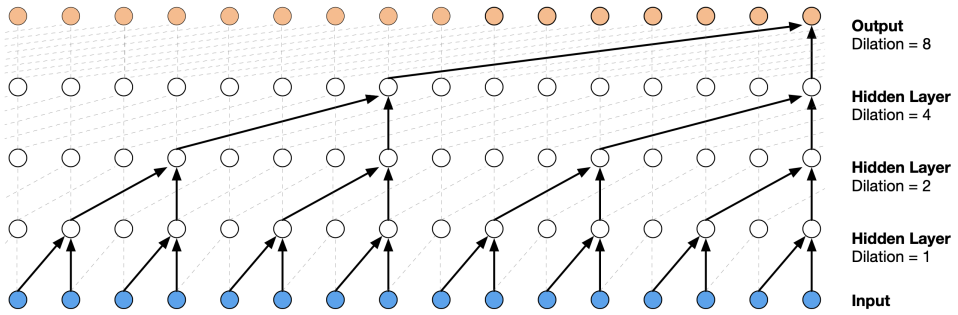

Figure 2: The Dilated Causal Convolution in WaveNet

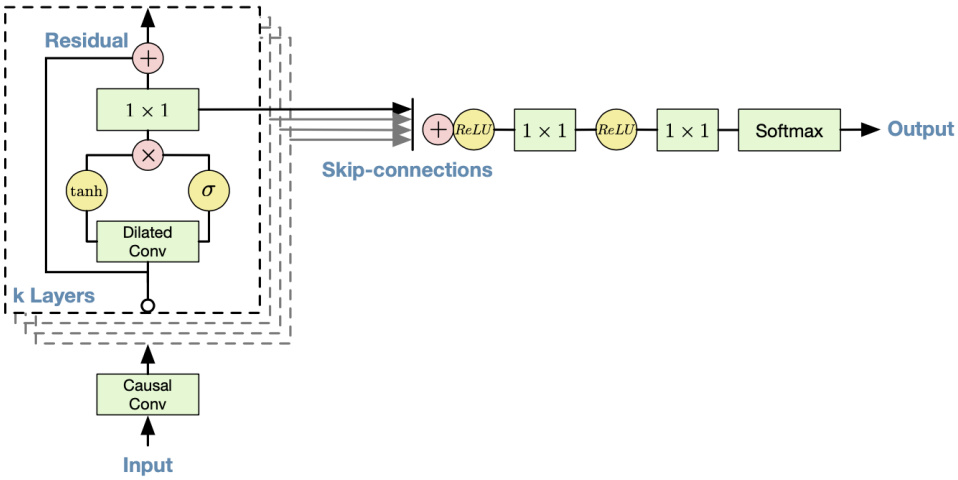

Figure 3: Overall Residual Architecture of WaveNet. Skip connections happen from every Convolutional Layer to the final softmax.

Summary of architechture The architecture is summed up in fig. 3. It's important to note that the Causal Convolution setup as described in fig. 2 only is applied once, as the first layer. This makes the entire rest of the network a simple convolutional network with dilation, as the **first (causal) convolutional stack ensures that the rest of the network will only see samples from the past.** In all other respects we can consider this a standard CNN architecture.

1.1.1 Extending architecture to include latent representations of speaker

It's possible to add a latent representation $\mathbf{h}$, extending eq. (2) to:

$$p(\mathbf{x}|\mathbf{h}) = \prod_{t=1}^{T} p(x_t|x_1, \ldots, x_t, \mathbf{h}) \tag{5}$$

There are two ways to represent this:

Global Conditioning (Speaker, Accent, Noise level) Here we set $\mathbf{h}$ to a single global latent, representing a constant over the entire sequence. The activation from eq. (4) then becomes

$$\mathbf{z} = \tanh(\mathbf{x} * W_{k,f} + V_{k,f}^T \mathbf{h}) \odot \sigma(\mathbf{x} * W_{k,g} + V_{k,g}^T \mathbf{h}) \tag{6}$$

With $V_{k,*}$ is a linear projection , and the resulting vector $V_{k,f}^T \mathbf{h}$ is broadcast over time T.

Local Conditioning (Tone of voice, changing noise levels over the call Here we define h_t, and use a ConvNet to upsample h_t to $\mathbf{y} = f(\mathbf{h})$, so eq. (4) becomes:

$$\mathbf{z} = \tanh(\mathbf{x} * W_{k,f} + V_{k,f} * \mathbf{y}) \odot \sigma(\mathbf{x} * W_{k,g} + V_{k,g} * \mathbf{y}) \tag{7}$$

1.2 Results

The main results here are evaluated on "subjective naturalness" by human evaluators. As such, WaveNets have outperformed previos TTS methods. That's not really that interesting but it makes for a cool listen: here

2 Vector Quantized VAE (VQ-VAE)

Title: Neural Discrete Representation Learning [6].

Main points:

- Avoiding Posterior Collapse
- Discrete Latent
- With the right prior, generates speech/audio well
- Language Learning through raw speech
- Speaker Conversion

The main point of the VQ-VAE lies in that it uses a discrete (i.e. categorical) embedding space as its latent space.

2.1 Model Components and Architecture

The model is described in fig. 4. What's very important to realize is that **the VQ-VAE is deterministic, not stochastic!**

2.2 Discrete Latent Embedding Space

The VQ VAE uses a D-dimensional latent space with K embedding vectors. This means that the latent space **does not sample from a latent space** (so it's not a real VAE) but instead **does a nearest-neighbor embedding lookup**. We define the embedding vectors as

$$e_i \in \mathbb{R}^D, i \in \{1..K\}, \therefore e \in \mathbb{R}^{K \times D}$$

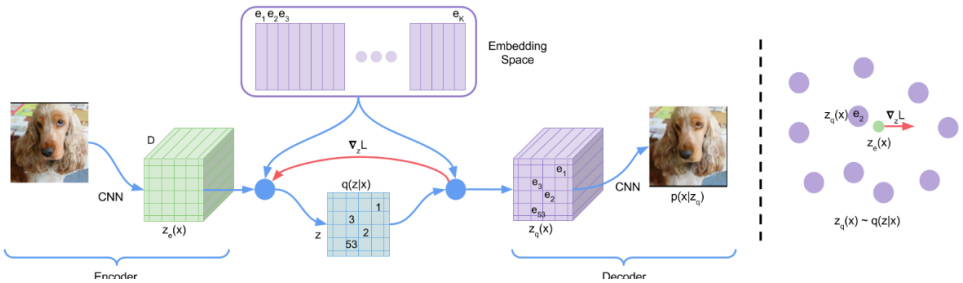

Figure 4: VQ-VAE Architecture. Right: Embedding space. Left: Overall Architecture. Note that the CNN works as a down/upsampling CNN, so as to avoid identity operations all the way through.

2.3 Loss function

The loss function eq. (8) is composed of 3 parts, here covered in detail:

$$L = \log p(x|z_q(x)) + ||\text{sg}[z_e(x)] - e||_2^2 + ||z_e(x) - \text{sg}[e]||_2^2 \tag{8}$$

We denote $z_e(x)$ as the **encoder output** and $z_q(x)$ as the **quantized encoding.** We also use sg to denote the stopgradient operator (identity function forward but 0 partial derivatives).

Reconstruction Loss

$$\log p(x|z_q(x)) +$$

This refers to the probability of the data x given the embedding $z_q(x)$.

The reconstruction loss is optimized by the **encoder** and **decoder**.

Embedding Loss

$$||\text{sg}[z_e(x)] - e||_2^2$$

The embedding loss quantifies how far away from the samples the embeddings are. This loss comes from Vector Quantization (VQ), a dictionary learning algorithm. The VQ objective seen here moves the discretized embedding vectors $e_i \in \mathbb{R}^D$ towards the continuous encoder outputs $z_e(x)$. This means that the model is able to learn embeddings and update them at need.

The embedding loss is optimized by the **embedding**.

Commitment Loss

$$||z_e(x) - \text{sg}[e]||_2^2$$

The commitment loss quantifies how far away from the embeddings a sample is. The embedding space is dimensionless in volume, and therefore the output of the encoder can grow arbitrarily large, while embeddings can't keep up. This equates to the encoder seeing an out-of-distribution sample and encoding it as very far away from the latent space embeddings.

The commitment loss is optimized by the **encoder**.

2.4 Experiments and Results

2.4.1 Images

Downsampling from $128 \times 128 \times 3$ to a $32 \times 32 \times 1$ discretized latent space on the ImageNet ($128 \times 128 \times 3$) dataset.

For images, the encoder/decoder is the PixelCNN.

2.4.2 Audio

For audio, the VQ-VAE is trained on the VCTK dataset, which has 109 different speakers. The encoder is **6 strided convolutions with stride 2 and window-size**

4, corresponding to a downsampling of 64x. The latent space is a single 512-imensional discretized space. In addition to the latent space, the decoder is conditioned on a 1-hot speaker embedding.

In order to make a prior for the latent space distribution, the authors trained a WaveNet model on the latent variables and used it as a prior on the latent space. [5]

Results The VQ-VAE manages to learn to interchange speakers very well. To cite the paper:

> This means that the VQ-VAE has, without any form of linguistic supervision, learned a high-level abstract space that is invariant to low-level features and only encodes the content of the speech.

The authors also ran an experiment where they mapped a 128-dimensional discrete latent space to 41 phonemes. Using this simple mapping they found the accuracy to be 49.3%, without further mappings.

3 Clockwork Variational Autoencoders

Authors Vaibhav Saxena, Jimmy Ba, Danijar Hafner. [8]

The clockwork VAE aims to learn higher-level, abstract prediction timelines without needing to predict the actual images going forward. They term this "Temporally Abstract Latent Dynamics Models".

3.1 CW-VAE Architecture and components

The CW-VAE is composed of a hierarchy of "states" as seen in fig. 5

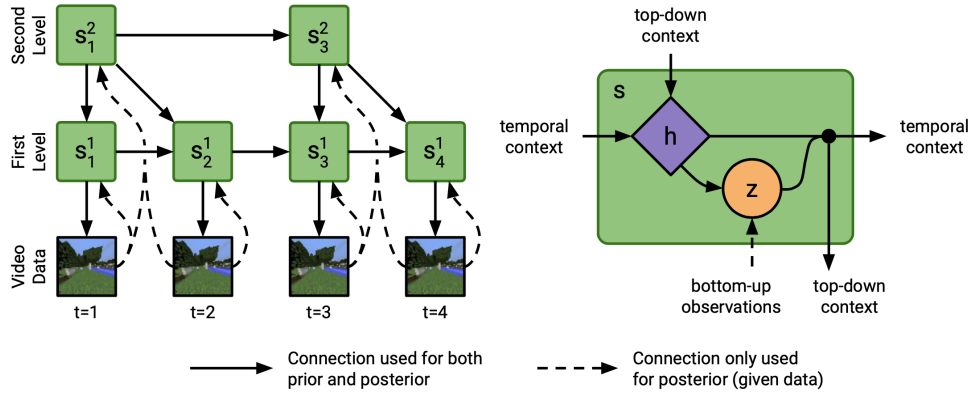

Figure 5: **Arrows:** Solid is the generative model, dashed is the inference model. **Left:** The general setup of the CW-VAE, with a temporal abstraction factor $k = 2$. The upper state only updates every kth timestep, and this would compound in higher hierarchies. As such we see that the video frame with $t = 2$ still will feed into s_1^2, and likewise the frame at $t = 4$ feeds into s_3^2 **Right:** The internals of states s_l^t.

3.1.1 The latent states s_t^l

The latent state is composed of a deterministic variable h_t and a stochastic variable z_t.

Updates to latent states during inference The latent states are updated at the "active" timesteps $\mathcal{T}_l = \{t \in [1, T] | t \bmod k^{l-1} = 1\}$, where k is the dilation factor. This means that latent states will only be updated at each k_{l-1} timestep, where l is the "level" of the latent variable. Otherwise, the states are copied from the previous timestep.

During inference, all "active" latents will receive a CNN image embedding (in fig. 5 this is the "bottom-up observations"). The posterior q_t^l for that latent is calculated "as a function (what function?) of the input features, (ASK JAKOB: Is this the arrow from z to the combination node?) the posterior sample at the previous step (temporal context), and the posterior sample above (top-down context)" The posterior / "Gaussian Belief" q_t^l, which is a diagonal Gaussians with means and variances predicted from the deterministic variable.

The deteministic variable is updated with a GRU at every active step.

3.1.2 Embeddings and layers in between.

The authors (Appendix C) claim to have used ”architectures very similar to the DCGAN”.

The DCGAN paper’s decoder architecture is shown in fig. 6. [7]

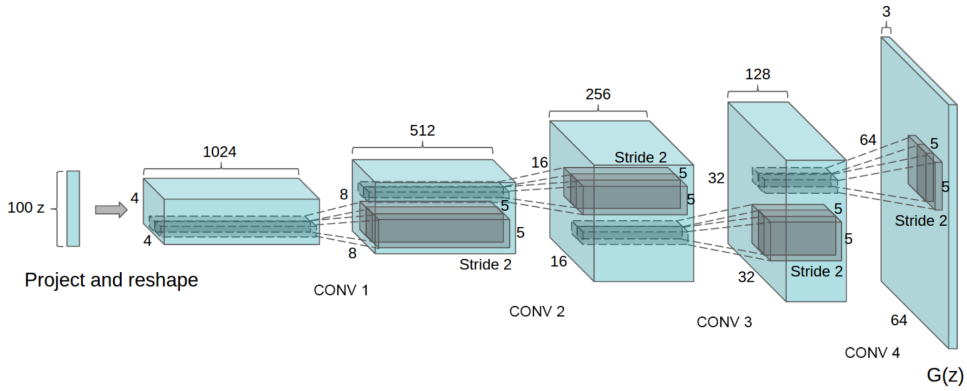

Figure 6: From DCGAN paper: A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions) then convert this high level representation into a 64 × 64 pixel image. Notably, no fully connected or pooling layers are used.

3.2 General Comments

The CWVAE paper manages to show very promising results in terms of temporal abstraction. They also conjecture that they would be able to do similar things with ”more data more GPU” but that we lack good validation metrics.

NB This paper has a good appendix to keep in mind.

4 Hierarchical Multiscale Recurrent Neural Networks

Authors : Junyoung Chung, Sungjin Ahn, Yoshua Bengio [1]

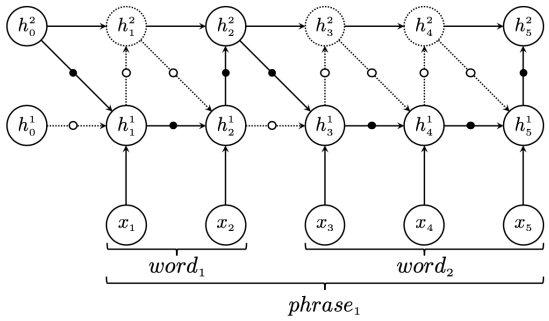

Figure 7: Overall Architecture of HMRNN. HM-RNN should learn and discover the structure of the data by itself, textbfwithout the use of boundary tokens.

4.1 Problem

The HMRNN addresses the question:

> Can an RNN discover hierarchical multiscale structure without explicit hierarchical boundary information?

Temporal-hierarchical information learning Temporal-Hierarchical information exist in many kinds of sequential data, and can easily be explained as a hierarchy of character, word, sentence, and paragraph context.

Unsupervised boundary detection The authors see the unsupervised boundary objective as necessary to learn a temporal-hierarchical information representation from data. In the past, there have been attempts at setting up a hierarchy of RNNs with the time hierarchies as hyperparameters, but allowing the model to learn this unsupervised has advantages:

1. The model is robust to variable length words/sentences. Example: `hello` and `hi` contain roughly the same word-level information, although one is almost 3 times as long as the other.
2. When progressing past a word and sentence-level abstraction, the ground truth information available quickly decreases. Example: Line breaks, paragraph breaks, hyphens, and chapter breaks are very sparsely represented in text datasets.

$$\mathbf{c}_t^\ell = \begin{cases} \mathbf{f}_t^\ell \odot \mathbf{c}_{t-1}^\ell + \mathbf{i}_t^\ell \odot \mathbf{g}_t^\ell & \text{if } z_{t-1}^\ell = 0 \text{ and } z_t^{\ell-1} = 1 \text{ (UPDATE)} \\ \mathbf{c}_{t-1}^\ell & \text{if } z_{t-1}^\ell = 0 \text{ and } z_t^{\ell-1} = 0 \text{ (COPY)} \\ \mathbf{i}_t^\ell \odot \mathbf{g}_t^\ell & \text{if } z_{t-1}^\ell = 1 \text{ (FLUSH)}, \end{cases}$$

Figure 8: HM-LSTM cell state update. Here (f, i, o) are gates (forget, input, output) and g is a proposal vector for the cell state.

4.2 HMRNN model and components

The model is summarized in fig. 7.

The HMRNN Model includes a binary boundary detector at each layer, which triggers the different operations.

COPY, FLUSH, and UPDATE operations in the HM-RNN are implemented at each abstraction level.

- COPY operation copies the state of the current abstraction level to the next cell
- UPDATE sparsely nupdates the cell in the next level
- FLUSH happens when a boundary is detected, so the current state representation is ejected to the higher level cell, and the state is reinitialized.

Boundary detection gradients and the Slope Annealing Trick are used with the straight-through estimator, to account for the non-differentiability of the discrete variable. Hence, during the backwards pass, the step function used in the forward pass is replaces with a hard sigmoid. The Slope Annealing trick includes starting the slope of the hard sigmoid as $a = 1$ and increasing it during training for the network to slowly learn to pass information through the sigmoid.

The HM-LSTM is proposed, as an LSTM model with $l \in [0...L]$, performing the following update at step t for layer l

$$\mathbf{h}_t^l, \mathbf{c}_t^l, z_t^l = f_{HM-LSTM}^l(\mathbf{c}_{t-1}^l, \mathbf{h}_{t-1}^l, \mathbf{h}_t^{l-1}, \mathbf{h}_{t-1}^{l+1}, z_{t-1}^l, z_t^{l-t}) \tag{9}$$

with $f_{HM_LSTM}^l$ being implemented s.t.

Model setup for experiments have the following:

1. A continuous input embedding layer
2. HM-LSTM recurrent unit (3 layers for CL-LM)
3. Output module of a FFNN, an output embedding layer and a softmax for probability calculation

Here the output module will receive the hidden states of all RNN layers as inputs, adding a gating layer

$$g_t^l = \texttt{sigm}(\mathbf{w}^l, [\mathbf{h}_t^1 ... \mathbf{h}_t^L]) \tag{10}$$

And an output embedding as

$$\mathbf{h}_t^e = \texttt{ReLu}\left(\sum_{l=1}^{L} g_t^l W_t^l \mathbf{h}_t^l\right) \tag{11}$$

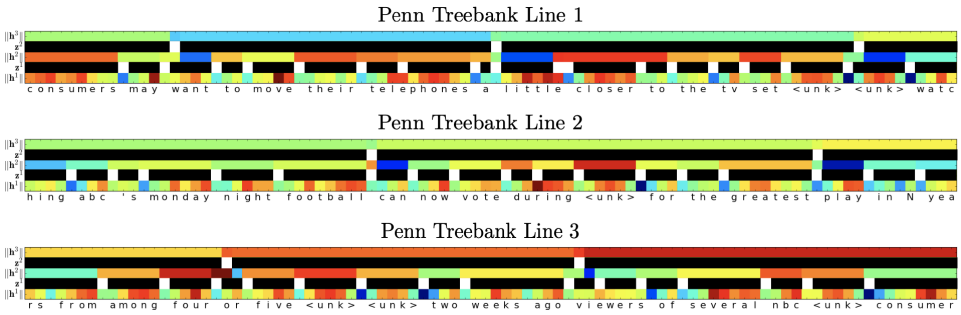

Figure 9: l^2 norm of hidden states along with boundary detection states for PTB.I

4.3 Experiments and results

Character-level language modelling providing raw text as input, is an ideal task for evaluating raw boundary detection. They train the CLLM task using the NLL sequence objective:

$$\min_{\theta} -\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T^n}\log p(x_t^n|x_{<t}^n,\theta) \tag{12}$$

The results on boundary detection are shown in **??**

5 Variational Temporal Abstraction

Authors : Taesup Kim, Sungjin Ahn, Yoshua Bengio [2]

Overview This paper is among the first to combine two recurrent state space model with latent variables.

They frame the problem as a temporal imaginative structure that should be probed, and reference the HMRNN model as a good first step. The entire point of the paper is trying to find an "imaginative" model that can also use the latent stochastic variables to model future possibilities.

5.1 Model

Given a sequence $X = x_{1:T}$ we can split it into subsequences $X = (X_1, \dots, X_N)$ with lengths $L = l_i$ such that $T = \sum_{i=1}^{N} l_i$. The VTA authors split the subsequences based on N and L as discrete latent variables.

Furthermore, they introduce the terms temporal abstraction z_i which generates X_i and an observation abstraction s_t which generates observation x_t. As such, an observation $x_t \in X_i$ depends on both z_i and s_t, while the transitions of each level happen at subsequence scale for z_i and at each time step for s_t.

The joint generative model can be written as:

$$p(X, S, L, Z, N) = p(N) \prod_{i=1}^{N} p(X_i, S_i | z_i, l_i) p(l_i | z_i) p(z_i | z_{<i}) \tag{13}$$

With the subsequence joint distribution:

$$p(X_i, S_i | z_i, l_i) = \prod_{j=1}^{l_i} p(x_j^i | s_j^i) p(s_j^i | s_{<j}^i, z_i) \tag{14}$$

The authors argue that even if it is possible to use a Markovian SSM or HMM in eq. (13) and eq. (14), using a RSSM type approach solves this better. This is because the RSSM includes a deterministic RNN path which can encode complex long-term dependencies within the model, which is a weakness of HMM type models.

6 A Clockwork RNN

Authors : Jan Koutník, Klaus Greff, Faustino Gomez, Jürgen Schmidhuber [4]

7 Towards Generating Long and Coherent Text with Multi-Level Latent Variable Models

Authors Dinghan Shen, Asli Celikyilmaz, Yizhe Zhang, Liqun Chen, Xin Wang, Jianfeng Gao, Lawrence Carin [9]

8 Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context

Authors Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, Ruslan Salakhutdinov

9 TacoTron

Authors: Yuxuan Wang,..., Rob Clark, Rif A. Saurous [10]

9.1 Problem to solve

End2End Text-To-Speech is a remaining frontier for ML systems, typically requiring extensive in-subject technical knowledge to set up. The goal of a end2end TTS system that can model $p(\texttt{Audio}|\texttt{Text})$ directly, is therefore very useful. The main issue here, is that TTS encompasses an Inverse Problem, i.e. a problem where the input dimensionality is much LOWER than the output dimensionality.

Tacotron is proposed as an integrated end-to-end generative TTS model that takes a character sequence as input and outputs the corresponding spectrogram.

9.2 Architecture

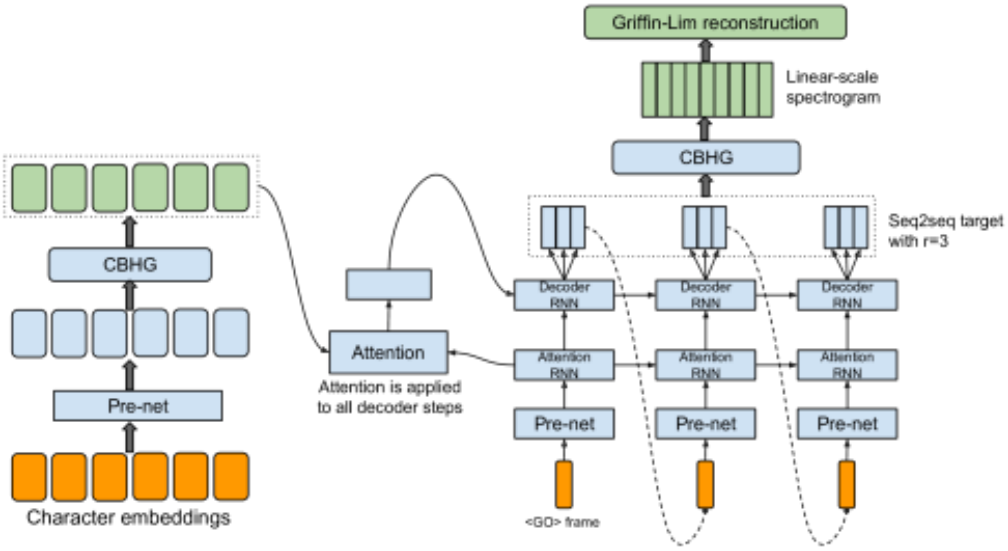

Figure 10: Overall architecture of TacoTron TTS system. The CBHG module is described in fig. 11

Convolution Bank + Highway Net + Bidirectional GRU feature (CBHG) consists of a bank of 1-D convolutional filters, followed by highway networks and a bidirectional gated recurrent unit (GRU). The input sequence is first convolved with K sets of 1-D convolutional filters, where the k-th set contains C_k filters, like modelling unigrams, bigrams, ...K-grams.

Convolution outputs are fed into a multi-layer highway network to extract high-evel features. Finally, we stack a bidirectional GRU RNN on top to extract sequential features from both forward and backward context.

See fig. 11 for a visual.

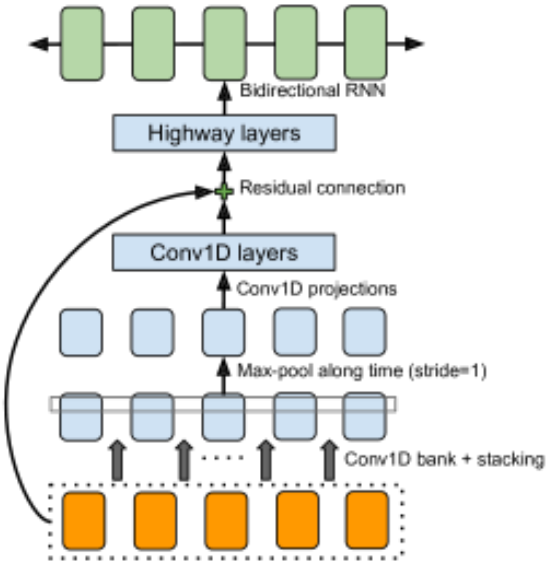

Figure 11: The CBHG module in the TacoTron architecture.

The Encoder on the left side of fig. 10 passes the input character embedding sequence to a bottleneck layer and through a CBHG to an encoded representation of the entire sequence.

The Decoder consists of a 2-layer FC pre-net, an Attention GRU (RNN), and a Decoder stack of GRUs with vertical residual connections.

References

[1] J. Chung, S. Ahn, and Y. Bengio. Hierarchical Multiscale Recurrent Neural Networks. arXiv:1609.01704 [cs], Mar. 2017. arXiv: 1609.01704.

[2] T. Kim, S. Ahn, and Y. Bengio. Variational Temporal Abstraction. arXiv:1910.00775 [cs, stat], Oct. 2019. arXiv: 1910.00775.

[3] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. arXiv:1312.6114 [cs, stat], May 2014. arXiv: 1312.6114.

[4] J. Koutník, K. Greff, F. Gomez, and J. Schmidhuber. A Clockwork RNN. arXiv:1402.3511 [cs], Feb. 2014. arXiv: 1402.3511.

[5] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. arXiv:1609.03499 [cs], Sept. 2016. arXiv: 1609.03499.

[6] A. v. d. Oord, O. Vinyals, and K. Kavukcuoglu. Neural Discrete Representation Learning. arXiv:1711.00937 [cs], May 2018. arXiv: 1711.00937.

[7] A. Radford, L. Metz, and S. Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. arXiv:1511.06434 [cs], Jan. 2016. arXiv: 1511.06434.

[8] V. Saxena, J. Ba, and D. Hafner. Clockwork Variational Autoencoders. arXiv:2102.09532 [cs], Feb. 2021. arXiv: 2102.09532.

[9] D. Shen, A. Celikyilmaz, Y. Zhang, L. Chen, X. Wang, J. Gao, and L. Carin. Towards Generating Long and Coherent Text with Multi-Level Latent Variable Models. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 2079–2089, Florence, Italy, July 2019. Association for Computational Linguistics.

[10] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. V. Le, Y. Agiomyrgiannakis, R. Clark, and R. Saurous. Tacotron: Towards End-to-End Speech Synthesis. undefined, 2017.