STCN: STOCHASTIC TEMPORAL CONVOLUTIONAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Convolutional architectures have recently been shown to be competitive on many sequence modelling tasks when compared to the de-facto standard of recurrent neural networks (RNNs), while providing computational and modeling advantages due to inherent parallelism. However, currently there remains a performance gap to more expressive stochastic RNN variants, especially those with several layers of dependent random variables. In this work, we propose stochastic temporal convolutional networks (STCNs), a novel architecture that combines the computational advantages of temporal convolutional networks (TCN) with the representational power and robustness of stochastic latent spaces. In particular, we propose a hierarchy of stochastic latent variables that captures temporal dependencies at different time-scales. The architecture is modular and flexible due to decoupling of deterministic and stochastic layers. We show that the proposed architecture achieves state of the art log-likelihoods across several tasks. Finally, the model is capable of predicting high-quality synthetic samples over a long-range temporal horizon in a variety of tasks including modeling of handwritten text and digits.

1 INTRODUCTION

Generative modeling of sequence data requires capturing long-term dependencies and learning of correlations between output variables at the same time-step. Recurrent neural networks (RNNs) and its variants have been very successful in a vast number of problem domains which rely on sequential data. Recent work in in audio synthesis, language modeling and machine translation tasks (Dauphin et al., 2016; Van Den Oord et al., 2016; Dieleman et al., 2018; Gehring et al., 2017) has demonstrated that temporal convolutional networks (TCNs) can also achieve at least competitive performance without relying on recurrence, and hence reducing the computational cost for training.

Both RNNs and TCNs model the joint probability distribution over sequences by decomposing the distribution over discrete time-steps. In other words, such models are trained to predict the next step, given all previous time-steps. RNNs are able to model long-term dependencies by propagating information through their deterministic hidden state, acting as an internal memory. In contrast, TCNs leverage large receptive fields by stacking many dilated convolutions, allowing them to model even longer time scales up to the entire sequence length. It is noteworthy that there is no explicit temporal dependency between the model outputs and hence the computations can be performed in parallel. The TCN architecture also introduces a temporal hierarchy: the upper layers have access to longer input sub-sequences and learn representations at a larger time scale. The local information from the lower layers is propagated through the hierarchy by means of residual and skip connections (Van Den Oord et al., 2016; Bai et al., 2018)

However, while TCN architectures have been shown to perform similar or better than standard recurrent architectures on particular tasks (Van Den Oord et al., 2016; Bai et al., 2018), there currently remains a performance gap to more recent stochastic RNN variants (Bayer & Osendorfer, 2014; Chung et al., 2015; Fabius & van Amersfoort, 2014; Fraccaro et al., 2016; Goyal et al., 2017; Shabanian et al., 2017). Following a similar approach to stochastic RNNs, Lai et al. (2018) present a significant improvement in the log-likelihood when a TCN model is coupled with latent variables, albeit at the cost of limited receptive field size. In this work we propose a new approach for augmenting the TCN models with random latent variables, that decouples deterministic and stochastic structures yet leverages the increased modeling capacity efficiently.

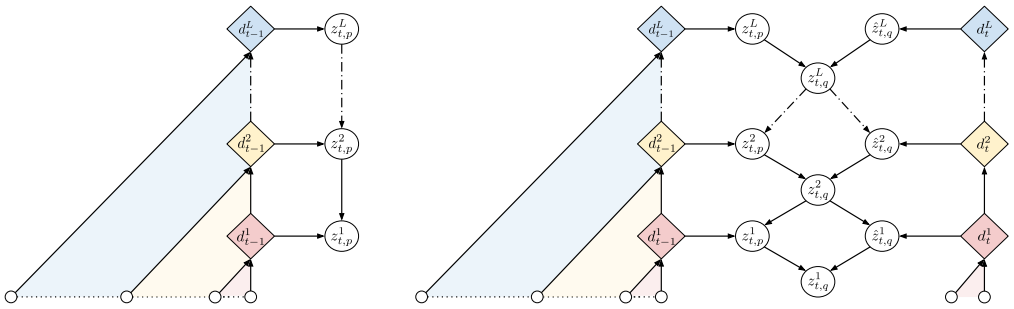


Figure 1: The computational graph of generative (left) and inference (right) models of STCN. The approximate posterior q is conditioned on $\mathbf{d}_t$ and is updated by the prior p which is conditioned on the TCN representations of the previous time-step $\mathbf{d}_{t-1}$. The random latent variables at the upper layers have access to a longer history while the lower layers receive more recent input steps.

Motivated by the simplicity and computational advantages of TCNs and the robustness and performance of stochastic RNNs, we introduce stochastic temporal convolutional networks (STCN) by incorporating a hierarchy of stochastic latent variables into TCNs which enables learning of representations at many timescales. However, due to the absence of an internal state in TCNs, introducing latent random variables analogously to stochastic RNNs is not feasible. Furthermore, defining conditional random variables across time-steps would result in breaking the parallelism of TCNs and is hence undesirable.

In STCN the latent random variables are arranged in correspondence to the temporal hierarchy of the TCN blocks which effectively distributes the random variables over the various timescales (see figure 1). Crucially, our hierarchical latent structure is designed to be a modular add-on for any temporal convolutional network architecture. Separating the deterministic and stochastic layers allows us to build STCNs without requiring modifications to the base TCN architecture, and hence retains the scalability of TCNs with respect to the receptive field.

We propose two different inference networks. In the canonical configuration, samples from each latent variable are passed down from layer to layer and only one sample from the lowest layer is used to condition the prediction of the output. In the second configuration, called STCN-dense, we take inspiration from recent CNN architectures (Huang et al., 2017) and utilize samples from all latent random variables via concatenation before computing the final prediction.

Our contributions can thus be summarized as: 1) We present a modular and scalable approach to augment temporal convolutional network models with effective stochastic latent variables. 2) We empirically show that the STCN-dense design prevents the model from ignoring latent variables in the upper layers Zhao et al. (2017). 3) We achieve state-of-the-art log-likelihood performance, measured by ELBO, on IAM-OnDB, Deepwriting, TIMIT and binarized sequential MNIST datasets. On Blizzard our model gives marginally lower result. 4) Finally we show that the quality of the synthetic samples matches the significant quantitative improvements.

2 Background

Auto-regressive models such as RNNs and TCNs factorize the joint probability of a variable-length sequence $\mathbf{x} = \{x_1, \ldots, x_T\}$ as a product of conditionals as follows:

$$p_\theta(\mathbf{x}) = \prod_{t=1}^{T} p_\theta(x_t | x_{1:t-1}) \quad , \tag{1}$$

where the joint distribution is parametrized by θ. The prediction at each time-step is conditioned on all previous observations. The observation model is frequently chosen to be a Gaussian or Gaussian mixture model (GMM) for real-valued data, and a categorical distribution for discrete-valued data.

2.1 TEMPORAL CONVOLUTIONAL NETWORKS

In TCNs the joint probabilities in Eq. (1) are parametrized by a stack of convolutional layers. *Causal convolutions* are the central building block of such models and are designed to be asymmetric such that the model has no access to future information. In order to produce outputs of the same size as the input, zero-padding is applied at every layer.

In the absence of a state transition function, a large receptive field is crucial in capturing long-range dependencies. To avoid the need for vast numbers of causal convolution layers, typically *dilated* convolutions are used. Exponentially increasing the dilation factor results in an exponential growth of the receptive field size with depth (Yu & Koltun, 2015; Van Den Oord et al., 2016; Bai et al., 2018). In this work, without loss of generality, we use the building blocks of Wavenet (Van Den Oord et al., 2016) as gated activation units (van den Oord et al., 2016) have been reported to perform better.

A deterministic TCN representation d_t^l at time-step t and layer l summarizes the input sequence $x_{1:t}$:

$$d_t^l = \text{Conv}^{(l)}(d_t^{l-1}, d_{t-j}^{l-1}) \quad \text{and} \quad d_t^1 = \text{Conv}^{(1)}(x_t, x_{t-j}) \quad , \tag{2}$$

where the filter width is 2 and j denotes the dilation step. In our work, the stochastic variables $z^l, l = 1 \dots L$ are conditioned on TCN representations d^l that are constructed by stacking K Wavenet blocks over the previous d^{l-1} (for details see Figure 5 in Appendix).

2.2 NON-SEQUENTIAL LATENT VARIABLE MODELS

VAEs (Kingma & Welling, 2013; Rezende et al., 2014) introduce a latent random variable $\mathbf{z}$ to learn the variations in the observed non-sequential data where the generation of the sample $\mathbf{x}$ is conditioned on the latent variable $\mathbf{z}$. The joint probability distribution is defined as:

$$p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z}) \quad , \tag{3}$$

and parametrized by θ. Optimizing the marginal likelihood is intractable due to the non-linear mappings between $\mathbf{z}$ and $\mathbf{x}$ and the integration over $\mathbf{z}$. Instead the VAE framework introduces an approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ and optimizes a lower-bound on the marginal likelihood:

$$\log p_\theta(\mathbf{x}) \geq -KL(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] \quad , \tag{4}$$

where KL denotes the Kullback-Leibler divergence. Typically the prior $p_\theta(\mathbf{z})$ and the approximate $q_\phi(\mathbf{z}|\mathbf{x})$ are chosen to be in simple parametric form, such as a Gaussian distribution with diagonal covariance, which allows for an analytical calculation of the KL-term in Eq. (4).

2.3 STOCHASTIC RNNS

An RNN captures temporal dependencies by recursively processing each input, while updating an internal state h_t at each time-step via its state-transition function:

$$h_t = f^{(h)}(x_t, h_{t-1}) \quad , \tag{5}$$

where $f^{(h)}$ is a deterministic transition function such as LSTM (Hochreiter & Schmidhuber, 1997) or GRU (Cho et al., 2014) cells. The computation has to be sequential because h_t depends on h_{t-1}.

The VAE framework has been extended for sequential data, where a latent variable z_t augments the RNN state h_t at each sequence step. The joint distribution $p_\theta(\mathbf{x}, \mathbf{z})$ is modeled via an auto-regressive model which results in the following factorization:

$$p_\theta(\mathbf{x}, \mathbf{z}) = \prod_{t=1}^{T} p_\theta(x_t|z_{1:t}, x_{1:t-1})p_\theta(z_t|x_{1:t-1}, z_{1:t-1}) \quad . \tag{6}$$

In contrast to the fixed prior of VAEs, $\mathcal{N}(\mathbf{0}, \mathbf{I})$, sequential variants define prior distributions conditioned on the RNN hidden state $\mathbf{h}$ and implicitly on the input sequence $\mathbf{x}$ (Chung et al., 2015).

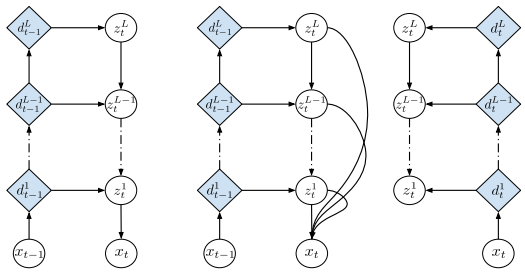


Figure 2: Graphical model view of generative models of STCN (*left*) and STCN-dense (*middle*), and the inference model (*right*), which is shared by both variants. Diamonds represent the outputs of deterministic dilated convolution blocks where the dependence of d_t on the past inputs is not shown for clarity (see Eq. (2)). x_t and z_t are observable inputs and latent random variables, respectively. The generative task is to predict the next step in the sequence, given all past steps. Note that in the STCN-dense variant the next step is conditioned on all latent variables z_t^l for $l = 1 \dots L$.

3 Stochastic Temporal Convolutional Networks

The mechanics of STCNs are related to those of VRNNs and LVAEs. Intuitively, the RNN state h_t is replaced by temporally independent TCN layers d_t^l. In the absence of an internal state, we define hierarchical latent variables z_t^l that are conditioned *vertically*, i.e., in the same time-step, but independent *horizontally*, i.e., across time-steps. We follow a similar approach to LVAEs (Sønderby et al., 2016) in defining the hierarchy in a *top-down* fashion and in how we estimate the approximate posterior. The inference network first computes the approximate likelihood, and then this estimate is corrected by the prior, resulting in the approximate posterior. The TCN layers **d** are shared between the inference and generator networks, analogous to VRNNs (Chung et al., 2015).

Figure 2 depicts the proposed STCN as a graphical model. STCNs consist of two main modules: the deterministic temporal convolutional network and the stochastic latent variable hierarchy. For a given input sequence $\mathbf{x} = \{x_t\}, t = 1 \dots T$ we first apply dilated convolutions over the entire sequence to compute a set of deterministic representations $d_t^l, l = 1 \dots L$. Here, d_t^l corresponds to the output of a block of dilated convolutions at layer l and time-step t. The output d_t^l is then used to update a set of random latent variables z_t^l arranged to correspond with different time-scales.

To preserve the *parallelism* of TCNs, we do not introduce an explicit dependency between different time-steps. However, we suggest that conditioning a latent variable z_t^{l-1} on the preceding variable z_t^l implicitly introduces temporal dependencies. Importantly, the random latent variables in the upper layer have access to a larger receptive field due to its deterministic input d_{t-1}^l, whereas latent random variables in lower layers are updated with different, more local information. However, the latent variable z_t^{l-1} may receive longer-range information from z_t^l.

The generative and inference models are jointly trained by optimizing a step-wise variational lower bound on the log-likelihood (Kingma & Welling, 2013; Rezende et al., 2014). In the following sections we describe these components and build up the lower-bound for a single time-step t.

3.1 Generative Model

Each sequence step x_t is generated from a set of latent variables z_t, split into layers as follows:

$$p_\theta(z_t|x_{1:t-1}) = p_\theta(z_t^L|d_{t-1}^L) \prod_{l=1}^{L-1} p_\theta(z_t^l|z_t^{l+1}, d_{t-1}^l) \quad , \tag{7}$$

$$\text{where} \quad p_\theta(z_t^l|z_t^{l+1}, d_{t-1}^l) = \mathcal{N}(\mu_{t,p}^l, \sigma_{t,p}^l) \quad \text{and} \quad [\mu_{t,p}^l, \sigma_{t,p}^l] = f_p^{(l)}(z_t^{l+1}, d_{t-1}^l) \quad . \tag{8}$$

Here the prior is modeled by a Gaussian distribution with diagonal covariance, as is common in the VAE framework. The subscript p denotes items of the generative distribution. For the inference distribution we use the subscript q. The distributions are parameterized by a neural network $f_p^{(l)}$ and conditioned on: (1) the d_{t-1}^l computed by the dilated convolutions from the previous time-step, and

(2) a sample from the preceding level at the same time-step z_t^{l+1}. Please note that at inference time we draw samples from the approximate posterior distribution $z_t^{l+1} \sim q_\phi(z_t^{l+1}|\cdot)$. The generative model, on the other hand, uses the prior $z_t^{l+1} \sim p_\theta(z_t^{l+1}|\cdot)$.

We propose two variants of the observation model. In the non-sequential scenario, the observations are defined to be conditioned on only the last latent variable in the hierarchy, i.e., $p_\theta(x_t|z_t^1)$, following Sønderby et al. (2016); Gulrajani et al. (2016) and Rezende et al. (2014) our STCN variant uses the same observation model, allowing for an efficient optimization. However, latent units are likely to become inactive during training in this configuration (Burda et al., 2015; Bowman et al., 2015; Zhao et al., 2017) resulting in a loss of representational power.

The latent variables at different layers are conditioned on different contexts due to the inputs d_t^l. Hence, the latent variables are expected to capture complementary aspects of the temporal context. To propagate the information all the way to the final prediction and to ensure that gradients flow through all layers, we take inspiration from Huang et al. (2017) and directly condition the output probability on samples from *all* latent variables. We call this variant of our architecture *STCN-dense*.

The final predictions are then computed by the respective observation functions:

$$p_\theta(x_t|z_t) = f^{(o)}(z_t^1) \quad \text{and} \quad p_\theta^{dense}(x_t|z_t) = f^{(o)}(z_t^1, z_t^2 \dots z_t^L) \quad , \tag{9}$$

where $f^{(o)}$ corresponds to the output layer constructed by stacking 1D convolutions or Wavenet blocks depending on the dataset.

3.2 Inference Model

In the original VAE framework the inference model is defined as a bottom-up process, where the latent variables are conditioned on the stochastic layer below. Furthermore, the parameterization of the prior and approximate posterior distributions are computed separately (Burda et al., 2015; Rezende et al., 2014). In contrast, Sønderby et al. (2016) propose a top-down dependency structure shared across the generative and inference models. From a probabilistic point of view, the approximate Gaussian likelihood, computed bottom-up by the inference model, is combined with the Gaussian prior, computed top-down from the generative model. We follow a similar procedure in computing the approximate posterior.

First, the parameters of the approximate likelihood are computed for each stochastic layer l:

$$[\hat{\mu}_{t,q}^l, \hat{\sigma}_{t,q}^l] = f_q^{(l)}(z_t^{l+1}, d_t^l) \quad , \tag{10}$$

followed by the downward pass, recursively computing the prior and approximate posterior by precision-weighted addition:

$$\begin{aligned} \sigma_{t,q}^l &= \frac{1}{(\hat{\sigma}_{t,q}^l)^{-2} + (\sigma_{t,p}^l)^{-2}} \quad , \\ \mu_{t,q}^l &= \sigma_{t,q}^l(\hat{\mu}_{t,q}^l(\hat{\sigma}_{t,q}^l)^{-2} + \mu_{t,p}^l(\sigma_{t,p}^l)^{-2}) \quad . \end{aligned} \tag{11}$$

Finally, the approximate posterior has the same decomposition as the prior (see Eq. (7)):

$$q_\phi(z_t|x_{1:t}) = q_\phi(z_t^L|d_t^L) \prod_{l=1}^{L-1} q_\phi(z_t^l|z_t^{l+1}, d_t^l) \quad , \tag{12}$$

$$q_\phi(z_t^l|z_t^{l+1}, d_t^l) = \mathcal{N}(\mu_{t,q}^l, \sigma_{t,q}^l) \quad . \tag{13}$$

Note that the inference and generative network share the parameters of dilated convolutions $\text{Conv}^{(l)}$.

3.3 Learning

The variational lower-bound on the log-likelihood at time-step t can be defined as follows:

$$\begin{aligned} \log p(x_t) &\geq \mathbb{E}_{q_\phi(z_t|x_t)}[\log p_\theta(x_t|z_t)] - D_{KL}(q_\phi(z_t|x_{1:t})||p_\theta(z_t|x_{1:t-1})) \\ &= \mathbb{E}_{q_\phi(z_t^1 \dots z_t^L|x_t)}[\log p_\theta(x_t|z_t^1 \dots z_t^L)] - D_{KL}(q_\phi(z_t^1 \dots z_t^L|x_{1:t})||p_\theta(z_t^1 \dots z_t^L|x_{1:t-1})) \\ \mathcal{L}_t(\theta, \phi; x_t) &= \mathcal{L}_t^{Recon} + \mathcal{L}_t^{KL}. \end{aligned} \tag{14}$$

Using the decompositions from Eq. (7) and (12), the Kullback-Leibler divergence term becomes:

$$\begin{aligned}\mathcal{L}_t^{KL} = &- D_{KL}(q_\phi(z_t^L|d_t^L)||p_\theta(z_t^L|d_{t-1}^L)) \\ &- \sum_{l=1}^{L-1} \mathbb{E}_{q_\phi(z_t^l|\cdot)}[D_{KL}(q_\phi(z_t^l|z_t^{l+1}, d_t^l)||p_\theta(z_t^l|z_t^{l+1}, d_{t-1}^l))] \quad .\end{aligned} \tag{15}$$

The KL term is the same for the STCN and STCN-dense variants. The reconstruction term $\mathcal{L}_t^{Recon}$ however differs between the variants. In STCN we only use samples from the lowest layer of the hierarchy, whereas in STCN-dense we use all latent samples in the observation model:

$$\mathcal{L}_t^{Recon} = \mathbb{E}_{q_\phi(z_t^1 \dots z_t^L|x_t)}[\log p_\theta(x_t|z_t^1)] \quad , \tag{16}$$

$$\mathcal{L}_t^{Recon-dense} = \mathbb{E}_{q_\phi(z_t^1 \dots z_t^L|x_t)}[\log p_\theta(x_t|z_t^1 \dots z_t^L] \quad . \tag{17}$$

In the dense variant, samples drawn from the latent variables z_t^l are carried over the dense connections. Similar to Maaløe et al. (2016), the expectation over z_t^l variables are computed by Monte Carlo sampling using the reparameterization trick (Kingma & Welling, 2013; Rezende et al., 2014).

Please note that the computation of $\mathcal{L}_t^{Recon-dense}$ does not introduce any additional computational cost. In STCN, all latent variables have to be visited in terms of ancestral sampling in order to draw the latent sample z_t^1 for the observation x_t. Similarly in STCN-dense, the same intermediate samples z_t^l are used in the prediction of x_t.

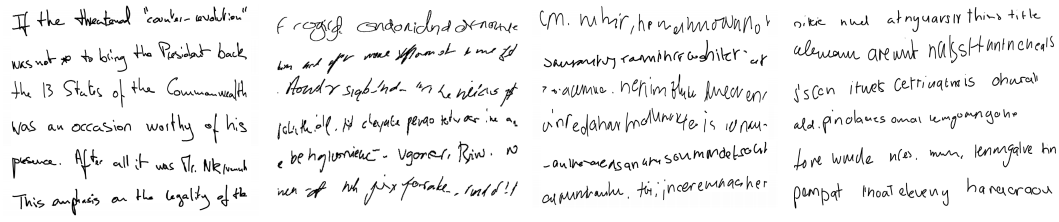
One alternative option to use the latent samples could be to sum individual samples before feeding them into the observation model, i.e., $sum([z_t^1 \dots z_t^L])$, (Maaløe et al., 2016). We empirically found that this does not work well in STCN-dense. Instead, we concatenate all samples $[z_t^1 \circ \dots \circ z_t^L]$ analogously to DenseNet (Huang et al., 2017) and (Kaiser et al., 2018).

4 EXPERIMENTS

We evaluate the proposed variants STCN and STCN-dense both quantitatively and qualitatively on three sequence modelling tasks: 1) modeling of digital handwritten text, 2) speech modeling and, 3) modeling of images of handwritten digits (sequential MNIST). We compare with vanilla TCNs, RNNs, VRNNs and state-of-the art models on the corresponding tasks.

In our experiments we use two variants of the Wavenet model: (1) the original model proposed in (Van Den Oord et al., 2016) and (2) a variant that we augment with skip connections analogously to STCN-dense. This additional baseline evaluates the benefit of learning *multi-scale* representations in the deterministic setting. Details of the experimental setup are provided in the Appendix (7.1).

Handwritten text: The IAM-OnDB and Deepwriting datasets consist of digital handwriting sequences where each time-step contains real-valued (x, y) pen coordinates and a binary *pen-up* event. The IAM-OnDB data is split and pre-processed as done in (Chung et al., 2015). Aksan et al. (2018) extend this dataset with additional samples and better pre-processing.



(a) Ground truth (b) VRNN (c) SwaveNet (d) STCN-dense

Figure 3: (a) Handwriting samples from IAM-OnDB dataset. Generated samples from (b) VRNN, (c) SWaveNet and (d) our model STCN-dense. Each line corresponds to one sample.

Models	TIMIT	Blizzard	IAM-OnDB	Deepwriting
Wavenet (GMM)	30188	8190	1381	612
Wavenet-dense (GMM)	30636	8212	1380	642
RNN (GMM) Chung et al. (2015)	26643	7413	1358	528 *
VRNN (Normal) Chung et al. (2015)	$\approx$ 30235	$\approx$ 9516	$\approx$ 1354	$\geq$ 495 *
VRNN (GMM) Chung et al. (2015)	$\approx$ 29604	$\approx$ 9392	$\approx$ 1384	$\geq$ 673 *
SRNN (Normal) Fraccaro et al. (2016)	$\geq$ 60550	$\geq$ 11991	n/a	n/a
Z-forcing (Normal) Goyal et al. (2017)	$\geq$ 70469	$\geq$ 15430	n/a	n/a
Var. Bi-LSTM (Normal) Shabanian et al. (2017)	$\geq$ 73976	$\geq$ **17319**	n/a	n/a
SWaveNet (Normal) Lai et al. (2018)	$\geq$ 72463	$\geq$ 15708	$\geq$ 1301	n/a
STCN (GMM)	$\geq$ 69195	$\geq$ 15800	$\geq$ 1338	$\geq$ 605
STCN-dense (GMM)	$\geq$ 71386	$\geq$ 16288	$\geq$ **1796**	$\geq$ **797**
STCN-dense-large (GMM)	$\geq$ **77438**	$\geq$ 17128	n/a	n/a

Table 1: Average log-likelihood per sequence on TIMIT, Blizzard, IAM-OnDB and Deepwriting datasets. (Normal) and (GMM) stand for unimodal Gaussian or multi-modal Gaussian Mixture Model (GMM) as the observation model (Graves, 2013; Chung et al., 2015). Asterisks * indicate that we used our re-implementation only for the Deepwriting dataset.

Table 1 reveals that again both our variants outperform the vanilla variants of TCNs and RNNs on IAM-OnDB. While the stochastic VRNN and SWaveNet are competitive wrt to the STCN variant, both are outperformed by the STCN-dense version. The same relative ordering is maintained on the Deepwriting dataset, indicating that the proposed architecture is robust across datasets.

Fig. 3 compares generated handwriting samples. While all models produce consistent style, our model generates more natural looking samples. Note that the spacing between words is clearly visible and most of the letters are distinguishable.

Speech modeling: TIMIT and Blizzard are standard benchmark dataset in speech modeling. The models are trained and tested on 200 dimensional real-valued amplitudes. We apply the same pre-processing as Chung et al. (2015). For this task we introduce STCN-dense-large, with increased model capacity. Here we use 512 instead of 256 convolution filters. Note that the total number of model parameters is comparable to SWaveNet and other SOA models.

On TIMIT, STCN-dense (Table 1) significantly outperforms the vanilla TCN and RNN, and stochastic models. On the Blizzard dataset, however, the Variational Bi-LSTM is marginally better. Note that the inference models of SRNN (Fraccaro et al., 2016), Z-forcing (Goyal et al., 2017), and Variational Bi-LSTM (Shabanian et al., 2017) receive future information by using backward RNN cells. Similarly, SWaveNet (Lai et al., 2018) uses causal convolutions in the backward direction. Hence, the latent variable can be expected to model future dynamics of the sequence as well. In contrast, our models have only access to information up to the current time-step. Considering this, we suggest that the STCN variants perform very well in the speech modeling task.

Sequential MNIST contains binary images of handwritten, individual digits and is commonly used in evaluations of sequence modeling approaches. The task typically consists of predicting the next pixel given all pixels up to the current observation, or to generate entirely novel digits, while maintaining the visual appearance.

Models	MNIST
Wavenet	80.42
Wavenet-dense	80.35
RNN	80.39
VRNN	$\leq$ 80.22
STCN	$\leq$ 82.96
STCN-dense	$\leq$ **60.75**

Models	MNIST
Pixel VAE Gulrajani et al. (2016)	$\approx$ 79.02
P-Forcing Goyal et al. (2016)	79.58
PixelRNN Van Den Oord et al. (2016)	79.20
MatNets Bachman (2016)	78.50
Z-forcing Goyal et al. (2017)	$\leq$ 80.09
Var. Bi-LSTM Shabanian et al. (2017)	$\leq$ 79.78

Table 2: Negative log-likelihood performance on binarized MNIST dataset. We use our re-implementation of RNN and VRNN.

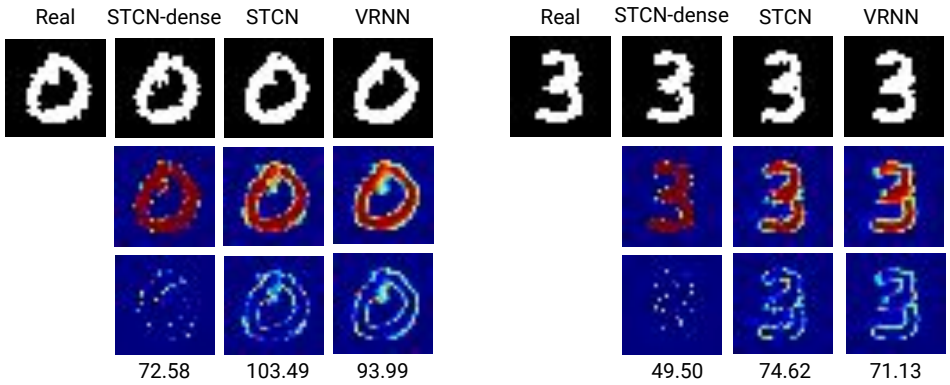


Figure 4: Reconstruction performance of STCN-dense, STCN and VRNN models on sequential MNIST. Negative log-likelihood (in ELBO) of the particular test sample is provided at the bottom. (Top row) A test sample and its reconstructions in binarized form. (Middle row) probability values, i.e., non-binarized image. (Bottom row) Residual images between the real and probability images. Red and blue colors correspond to 1 and 0, respectively. STCN-dense makes more precise predictions and almost binary probabilities, leading to state-of-the-art log-likelihood performance.

Table 2 summarizes the average negative $\log$-likelihood per sequence of our approach and a number of state of the art baselines. On this dataset, the STCN performs worse than the baselines. However, the STCN-dense model outperforms the other models by a large margin, most likely because it can leverage the increased model capacity more effectively.

Fig. 4 also shows that the lower $\log$-likelihood score translates to preservation of high frequency details in the reconstructed samples, while the VRNN and STCN produce slightly more blurry images. Moreover, the non-binarized predictions (see Fig. 4, middle row) illustrate that STCN-dense makes very precise predictions while STCN and VRNN have more uncertainty around the digit boundary.

Latent Space Analysis: Zhao et al. (2017) observe that in hierarchical latent variable models the upper layers have a tendency to become inactive, indicated by a low KL loss (Sønderby et al., 2016; Dieng et al., 2018). Table 3 shows the KL loss per latent variable and the corresponding log-likelihood (NLL for MNIST) measured by ELBO in our models. Across the datasets it can be observed that our models make use of many of the latent variables which may explain the strong performance across tasks in terms of log-likelihoods. Note that STCN uses a standard hierarchical structure. However, individual latent variables have different information context due to the corresponding TCN block's receptive field. This observation suggests that the proposed combination of TCNs and stochastic variables is indeed effective. Furthermore, in STCN we see a similar utilization pattern of the z variables across tasks, whereas STCN-dense may have more flexibility in modeling the data and the temporal dependencies within due to its dense connections to the output layer.

Finally, on MNIST we compare the KL loss of SKIP-VAE (Dieng et al., 2018) to our models. Note that SKIP-VAE is not a hierarchical model, but introduces skip connections from the latent variable to the decoder layers. STCN-dense effectively uses high modeling capacity of the stochastic latent variables at the upper layers. One may worry that such high discrepancy between the prior and approximate posterior may harm the generative modeling capacity in return for strong log-likelihood performance. However, the generated handwritten (Figure 3) and MNIST samples (in Figure 6 in Appendix) show that STCN-dense is able to generate high-quality synthetic samples.

5 Related Work

Rezende et al. (2014) propose Deep Latent Gaussian Models (DLGM) and Sønderby et al. (2016) propose the Ladder Variational Autoencoder (LVAE). In both models the latent variables are hierarchically defined and conditioned on the preceding stochastic layer. LVAEs improve upon DLGMs via implementation of a top-down hierarchy both in the generative and inference model. The approximate posterior is computed via a precision-weighted update of the approximate likelihood (i.e.,

Dataset (Model)	ELBO	KL	KL1	KL2	KL3	KL4	KL5	KL6
IAM-OnDB (STCN-dense)	1796.3	1653.9	17.9	1287.4	305.3	41.0	2.4	n/a
IAM-OnDB (STCN)	1339.2	964.2	846.0	105.2	12.9	0.1	0.0	n/a
TIMIT (STCN-dense)	71385.9	22297.5	16113.0	5641.6	529.0	8.3	5.7	n/a
TIMIT (STCN)	69194.9	23118.3	22275.5	487.2	355.5	0.0	0.0	n/a
MNIST (STCN-dense)	60.7	39.7	0.0	0.0	0.0	0.5	2.0	37.2
MNIST (STCN)	83.0	14.1	6.3	3.1	2.6	2.1	0.0	0.0
MNIST (Skip-VAE)	82.6	9.3	9.3	n/a	n/a	n/a	n/a	n/a

Table 3: KL-loss per latent variable in the hierarchy computed over the entire test split.

the inference model) and prior (i.e., the generative model). Similarly, the PixelVAE (Gulrajani et al., 2016) incorporates a hierarchical latent space decomposition and uses an autoregressive decoder. Zhao et al. (2017) show under mild conditions that hierarchical models constructed by stacking latent variables, i.e., LVAE and PixelVAE, struggle in disentangling representations, because the latent variables that are not directly conditioned on by the observation variable often become ineffective.

Due to the nature of the sequential problem domain, our approach differs in the crucial aspects that STCNs use dynamic, i.e., conditional, priors (Chung et al., 2015) at every level. Moreover, the hierarchy is not only implicitly defined by the network architecture but also explicitly defined by the information content, i.e., receptive field size. Dieng et al. (2018) both theoretically and empirically show that using skip connections from the latent variable to every layer of the decoder increases mutual information between the latent and observation variables. Similar to (Dieng et al., 2018) in STCN-dense, we introduce skip connections from all latent variables to the output layer. In STCN the model is expected to encode and propagate the information through its hierarchy.

Yang et al. (2017) suggest using autoregressive TCN decoders to remedy the posterior collapse problem observed in language modeling with LSTM decoders (Bowman et al., 2015). van den Oord et al. (2017) and Dieleman et al. (2018) use TCN decoders conditioned on discrete latent variables to model audio signals.

Stochastic RNN architectures mostly vary in the way they employ the latent variable and parametrize the approximate posterior for variational inference. Chung et al. (2015) and Bayer & Osendorfer (2014) use the latent random variable to capture high-level information causing the variability observed in sequential data. Particularly Chung et al. (2015) shows that using a conditional prior rather than a standard Gaussian distribution is very effective in sequence modeling. In (Fraccaro et al., 2016; Goyal et al., 2017; Shabanian et al., 2017), the inference model, i.e., the approximate posterior, receives both the past and future summaries of the sequence from the hidden states of forward and backward RNN cells. The KL-divergence term in the objective enforces the model to learn predictive latent variables in order to capture the future states of the sequence.

The SWaveNet architecture (Lai et al., 2018) is most closely related to ours. Like STCN SWaveNet introduces latent variables into TCNs. However, in SWaveNet, the deterministic and stochastic units are coupled, which may prevent stacking of larger numbers of TCN blocks. Since, the number of stacked dilated convolutions determines the receptive field size, it directly correlates with the model capacity. For example, the performance of SWaveNet on the IAM-OnDB dataset degrades after stacking more than 3 stochastic layers (Lai et al., 2018), limiting the model to a small receptive field. In contrast, we aim to preserve the flexibility of stacking dilated convolutions in the base TCNs. In STCNs, the deterministic TCN units do not have any dependency on the stochastic variables (see Figure 1) and the ratio of stochastic to deterministic units can be adjusted, depending on the task.

6 Conclusion

In this paper we proposed STCNs, a novel auto-regressive model, combining the computational benefits of convolutional architectures and expressiveness of hierarchical stochastic latent spaces. We have shown the effectivness of the approach across several sequence modelling tasks and datasets. The proposed models optimize ELBO objective. Tighter lower bounds such as IWAE (Burda et al., 2015) or FIVO (Maddison et al., 2017) may further improve modeling performance, which remains as future work.

REFERENCES

Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 265–283, 2016. URL `https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf`.

Emre Aksan, Fabrizio Pece, and Otmar Hilliges. DeepWriting: Making Digital Ink Editable via Deep Generative Modeling. In *SIGCHI Conference on Human Factors in Computing Systems*, CHI '18, New York, NY, USA, 2018. ACM.

Philip Bachman. An architecture for deep, hierarchical generative models. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 29*, pp. 4826–4834. Curran Associates, Inc., 2016.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Justin Bayer and Christian Osendorfer. Learning stochastic recurrent networks. *arXiv preprint arXiv:1411.7610*, 2014.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pp. 2980–2988, 2015.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. *arXiv preprint arXiv:1612.08083*, 2016.

Sander Dieleman, Aäron van den Oord, and Karen Simonyan. The challenge of realistic music generation: modelling raw audio at scale. *arXiv preprint arXiv:1806.10474*, 2018.

Adji B Dieng, Yoon Kim, Alexander M Rush, and David M Blei. Avoiding latent variable collapse with generative skip models. *arXiv preprint arXiv:1807.04863*, 2018.

Otto Fabius and Joost R van Amersfoort. Variational recurrent auto-encoders. *arXiv preprint arXiv:1412.6581*, 2014.

Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. In *Advances in neural information processing systems*, pp. 2199–2207, 2016.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*, 2017.

Anirudh Goyal ALIAS PARTH Goyal, Alex M Lamb, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 29*, pp. 4601–4609. Curran Associates, Inc., 2016.

Anirudh Goyal ALIAS PARTH Goyal, Alessandro Sordoni, Marc-Alexandre Côté, Nan Ke, and Yoshua Bengio. Z-forcing: Training stochastic recurrent networks. In *Advances in Neural Information Processing Systems*, pp. 6713–6723, 2017.

Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

Ishaan Gulrajani, Kundan Kumar, Faruk Ahmed, Adrien Ali Taiga, Francesco Visin, David Vazquez, and Aaron Courville. Pixelvae: A latent variable model for natural images. *arXiv preprint arXiv:1611.05013*, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, volume 1, pp. 3, 2017.

Łukasz Kaiser, Aurko Roy, Ashish Vaswani, Niki Pamar, Samy Bengio, Jakob Uszkoreit, and Noam Shazeer. Fast decoding in sequence models using discrete latent variables. *arXiv preprint arXiv:1803.03382*, 2018.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Guokun Lai, Bohan Li, Guoqing Zheng, and Yiming Yang. Stochastic wavenet: A generative latent variable model for sequential data, 2018.

Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Auxiliary deep generative models. *arXiv preprint arXiv:1602.05473*, 2016.

Chris J Maddison, John Lawson, George Tucker, Nicolas Heess, Mohammad Norouzi, Andriy Mnih, Arnaud Doucet, and Yee Teh. Filtering variational objectives. In *Advances in Neural Information Processing Systems*, pp. 6573–6583, 2017.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Samira Shabanian, Devansh Arpit, Adam Trischler, and Yoshua Bengio. Variational bi-lstms. *arXiv preprint arXiv:1711.05717*, 2017.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *Advances in neural information processing systems*, pp. 3738–3746, 2016.

Aäron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *SSW*, pp. 125, 2016.

Aäron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pp. 4790–4798, 2016.

Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 1747–1756. JMLR.org, 2016.

Aaron van den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pp. 6306–6315, 2017.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. Improved variational autoencoders for text modeling using dilated convolutions. *arXiv preprint arXiv:1702.08139*, 2017.

Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. Learning hierarchical features from generative models. *arXiv preprint arXiv:1702.08396*, 2017.

7 APPENDIX

7.1 NETWORK DETAILS

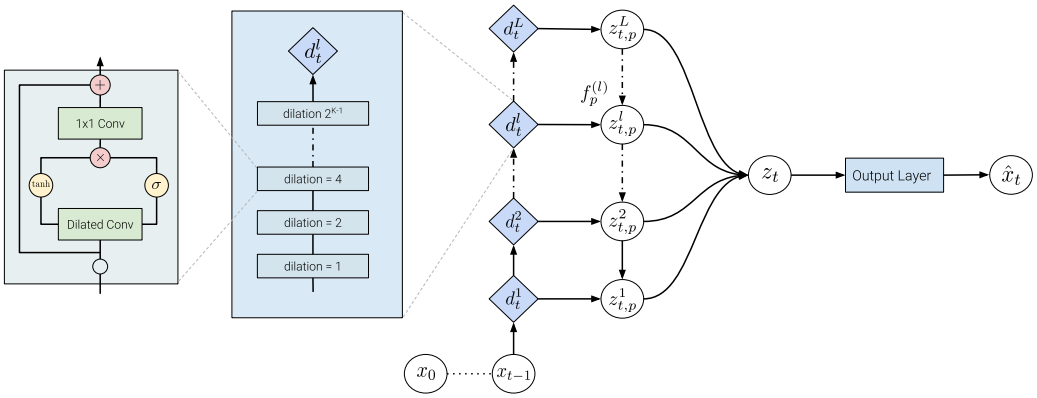


Figure 5: Generative model of STCN-dense architecture. Building blocks are highlighted. Note that the dependence of $d_t^l, l = 1 \cdots L$ on past inputs is not visualized for clarity.

Network architecture of the proposed model is illustrated in Fig. 5. We make a small modification to vanilla Wavenet architecture. Instead of using the skip connections from Wavenet blocks, we only use the latent sample z_t in order to make a prediction of x_t. In STCN-dense configuration, z_t is the concatenation of all latent variables in the hierarchy, i.e., $z_t = [z_t^1 \circ \cdots \circ z_t^L]$, whereas in STCN only z_t^1 is fed to the output layer.

Each stochastic latent variable z_t^l (except the top-most z_t^L) is conditioned on a deterministic TCN representation d_t^l and the preceding random variable z_t^{l+1}. The latent variables are calculated by using the latent layers $f_p^{(l)}$ or $f_q^{(l)}$ which are neural networks.

We do not define a latent variable per TCN layer. Instead, the stochastic layers are uniformly distributed where each random variable is conditioned on a number of stacked TCN layers d_t^l. We stack K Wavenet blocks (see Fig.. 5 left) with exponentially increasing dilation size.

Observation Model: We use Normal or GMM distributions with 20 components to model real-valued data. All Gaussian distributions have diagonal covariance matrix.

Output layer $f^{(o)}$: For IAM-OnDB, Deepwriting and MNIST datasets we use 1D convolutions with ReLU nonlinearity and filter size 1. In MNIST task 2 convolution operations with 128 filters are stacked. More complex handwriting datasets require 5 layers with 256 filters.

For TIMIT and Blizzard datasets Wavenet blocks in the output layer perform significantly better. We stack 5 Wavenet blocks with dilation size 1. For each convolution operation in the block we use 256 filters. The filter size of the dilated convolution is set to 2. The STCN-dense-large model is constructed by using 512 filters instead of 256.

TCN blocks d_t^l: The number of Wavenet blocks is usually determined by the receptive field size we want to have.

- For handwriting datasets $K = 6$ and $L = 5$. In total we have 30 Wavenet blocks where each convolution operation has 256 filters with size 2.
- For speech datasets $K = 5$ and $L = 5$. In total we have 25 Wavenet blocks where each convolution operation has 256 filters with size 2. The large model configuration uses 512 filters.
- For MNIST dataset $K = 7$ and $L = 6$. In total we have 42 Wavenet blocks where each convolution operation has 128 filters with size 2.

Latent layers $f_p^{(l)}$ and $f_q^{(l)}$: The number of stochastic layers per task is given by L. We used $[32, 16, 8, 5, 2]$ dimensional latent variables for handwriting tasks. It is $[256, 128, 64, 32, 16]$ for

speech datasets. Note that the first entry of the list corresponds to z^1. The number of units is 2 at every stochastic layer of MNIST models.

The mean and sigma parameters of Normal distributions modeling the latent variables are calculated by the $f_p^{(l)}$ and $f_q^{(l)}$ networks. We stack 2 1D convolutions with ReLU nonlinearity and filter size 1. The number of filters are the same with the number of Wavenet block filters for the corresponding task.

We did not observe any instabilities. However, we clamped the latent sigma predictions between 0.001 and 5.

7.2 Training Details

In all STCN experiments we applied KL annealing. In all tasks, the weight of KL term is initialized with 0 and increased by $1 \times e^{-4}$ every step until it reaches 1.

The batch size was 20 for all datasets except Blizzard where it was 128.

We use ADAM optimizer with its default parameters and exponentially decay the learning rate. For handwriting and MNIST datasets the learning rate was initialized with 5^{-4} and followed a decay rate of 0.94 and 1000 decay steps. On speech models it was initialized with 1^{-3} and decayed with a rate of 0.98. We applied early stopping by measuring ELBO performance on the validation dataset.

We implement STCN models in Tensorflow (Abadi et al., 2016). The code will be made publicly available.

7.3 Generated MNIST Samples

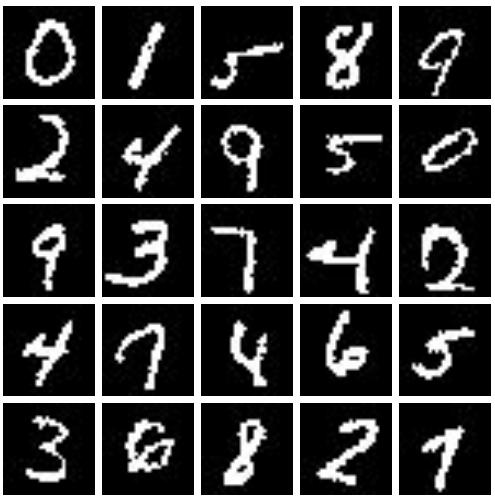


Figure 6: MNIST samples generated by STCN-dense model.

7.4 DETAILED RESULTS

Here we provide the extended results table with Normal observation model entries for available models.

Models	TIMIT	Blizzard	IAM-OnDB	Deepwriting
Wavenet (Normal)	-7443	3784	1053	337
Wavenet (GMM)	30188	8190	1381	612
Wavenet-dense (Normal)	-8579	3712	1030	323
Wavenet-dense (GMM)	30636	8212	1380	642
RNN (Normal) Chung et al. (2015)	-1900	3539	1016	363 *
RNN (GMM) Chung et al. (2015)	26643	7413	1358	528 *
VRNN (Normal)Chung et al. (2015)	$\approx$ 30235	$\approx$ 9516	$\approx$ 1354	$\geq$ 495 *
VRNN (GMM) Chung et al. (2015)	$\approx$ 29604	$\approx$ 9392	$\approx$ 1384	$\geq$ 673 *
SRNN (Normal) Fraccaro et al. (2016)	$\geq$ 60550	$\geq$ 11991	n/a	n/a
Z-forcing (Normal)Goyal et al. (2017)	$\geq$ 70469	$\geq$ 15430	n/a	n/a
Var. Bi-LSTM (Normal)Shabanian et al. (2017)	$\geq$ 73976	$\geq$ **17319**	n/a	n/a
SWaveNet (Normal)Lai et al. (2018)	$\geq$ 72463	$\geq$ 15708	$\geq$ 1301	n/a
STCN(Normal)	$\geq$ 64913	$\geq$ 13273	$\geq$ 1327	$\geq$ 575
STCN(GMM)	$\geq$ 69195	$\geq$ 15800	$\geq$ 1338	$\geq$ 605
STCN-dense(Normal)	$\geq$ 70294	$\geq$ 15950	$\geq$ 1729	$\geq$ 740
STCN-dense(GMM)	$\geq$ 71386	$\geq$ 16288	$\geq$ **1796**	$\geq$ **797**
STCN-dense-large (GMM)	$\geq$ **77438**	$\geq$ 17128	n/a	n/a

Table 4: Average log-likelihood per sequence on TIMIT, Blizzard, Iamondb and Deepwriting datasets. (Normal) and (GMM) stand for unimodal Gaussian or multi-modal Gaussian Mixture Model (GMM) as the observation model (Graves, 2013; Chung et al., 2015). Asterisks * indicate that we used our re-implementation only for the Deepwriting dataset.