LEARNING STOCHASTIC RECURRENT NETWORKS

Justin Bayer
Lehrstuhl für Echtzeitsysteme und Robotik
Fakultät für Informatik
Technische Universität München
bayer.justin@googlemail.com

Christian Osendorfer
Lehrstuhl für Echtzeitsysteme und Robotik
Fakultät für Informatik
Technische Universität München
osendorf@in.tum.de

ABSTRACT

Leveraging advances in variational inference, we propose to enhance recurrent neural networks with latent variables, resulting in Stochastic Recurrent Networks (STORNs). The model i) can be trained with stochastic gradient methods, ii) allows structured and multi-modal conditionals at each time step, iii) features a reliable estimator of the marginal likelihood and iv) is a generalisation of deterministic recurrent neural networks. We evaluate the method on four polyphonic musical data sets and motion capture data.

1 INTRODUCTION

Recurrent Neural Networks (RNNs) are flexible and powerful tools for modeling sequences. While only bearing marginal existence in the 1990's, recent successes in real world applications (Graves, 2013; Graves et al., 2013; Sutskever et al., 2014; Graves et al., 2008; Cho et al., 2014) have resurged interest. This is partially due to architectural enhancements (Hochreiter & Schmidhuber, 1997), new optimisation findings (Martens & Sutskever, 2011; Sutskever et al., 2013; Bengio et al., 2012) and the increased computional power available to researchers. RNNs can be employed for a wide range of tasks as they inherit their flexibility from plain neural networks. This includes universal approximation capabilities, since RNNs are capable of approximating any measureable sequence to sequence mapping and have been shown to be Turing complete (Hammer, 2000; Siegelmann & Sontag, 1991).

One typical application is to let an RNN model a probability distribution over sequences, i.e. $p(\mathbf{x}_{1:T})$. This is done by writing the distribution in cascade form,

$$p(\mathbf{x}_{1:T}) = \prod_{t=0}^{T-1} p(x_{t+1}|\mathbf{x}_{1:t}),$$

where $\mathbf{x}_{1:0} = \emptyset$. Each $p(x_{t+1}|\mathbf{x}_{1:t})$ is then represented by the output of an RNN at a single time step, identifying each of its components with the statistics of the distribution. A simple example is that of a Bernoulli, i.e.

$$p(x_{t+1,k} = 1|\mathbf{x}_{1:t}) = \eta_k(\mathbf{x}_{1:t}) \tag{1}$$

where $x_{t+1,k}$ corresponds to the k'th component of the $t+1$'th time step of $\mathbf{x}$ with $k = 1, \ldots, \omega$ and $t = 1, \ldots, T$. Each $\eta_k(\mathbf{x}_{1:t})$ is the k'th output of some RNN at time step t, constrained to lie in the interval $(0, 1)$. Learning such an RNN then boils down to minimising the negative log-likelihood of the data with respect to the parameters of the network.

This framework gives practitioners a powerful tool to model rich probability distributions over sequences. A common simplification is a naïve Bayes assumption that the individual components

factorise:

$$p(x_{t+1}|\mathbf{x}_{1:t}) = \prod_k p(x_{t+1,k}|\mathbf{x}_{1:t}).$$

While sufficient for many applications, reintroduction of dependency among the components of x_t leaves room for improvement. This is especially true for sequences over spaces which are high dimensional and tightly coupled. The approach taken by Graves (2013) is to use a mixture distribution for $p(x_t|\mathbf{x}_{1:t-1})$. Arguably powerful enough to model any dependency between the components of x_t, a drawback is that the number of parameters scales at least linearly with the number of chosen mixture components.

Models based on restricted Boltzmann machines and variations (Boulanger-Lewandowski et al., 2012; 2013; Sutskever et al., 2008) provide a solution to this as well, yet come with tighter restrictions on the assumptions that can be made. E.g. RBMs are restricted to model data using posteriors from the exponential family (Welling et al., 2004), make use of an intractable objective function and require costly MCMC steps for learning and sampling.

In this work, we propose to consider adding latent variables similar to Tang & Salakhutdinov (2013) to the network. Using stochastic gradient variational Bayes (SGVB) (Rezende et al., 2014; Kingma & Welling, 2013) as an estimator, we train RNNs to model high dimensional sequences.

2 PRELIMINARIES

In this section we will recap the basis of our method. We will first describe the used model family, that of recurrent neural networks and then the estimator, stochastic gradient variational Bayes (SGVB).

2.1 RECURRENT NEURAL NETWORKS

Given an input sequence $\mathbf{x} = (x_1, \dots, x_T), x_t \in \mathbb{R}^\kappa$ we compute the output sequence of a simple Recurrent Neural Network (sRNN) $\mathbf{y} = (y_1, \dots, y_T), y_t \in \mathbb{R}^\omega$ via an intermediary hidden state layer $\mathbf{h} = (h_1, \dots, h_T), h_t \in \mathbb{R}^\gamma$ by recursive evaluation of the following equations:

$$h_t = f_h(x_t\mathbf{W}_{\text{in}} + h_{t-1}\mathbf{W}_{\text{rec}} + \mathbf{b}_{\text{hidden}}),, \tag{2}$$

$$y_t = f_y(h_t\mathbf{W}_{\text{out}} + \mathbf{b}_{\text{out}}). \tag{3}$$

The set of adaptable parameters is given by $\theta = \{\mathbf{W}_{\text{in}}, \mathbf{W}_{\text{rec}}, \mathbf{W}_{\text{out}}, \mathbf{b}_{\text{hidden}}, \mathbf{b}_{\text{out}}\}$. f_h and f_y are transfer functions introducing nonlinearity into the computation.

Adaptation of the network's behaviour can be done by optimising a loss function with respect to the network's parameters with gradient-based schemes. Consider a data set of finite size, i.e. $\mathcal{D} = \{(\mathbf{x}^{(i)}_{1:T})\}_{i=1}^I$ on which the loss operates. In a setting as in Equation (1) a reasonable choice is the negative log-likelihood given by $\mathcal{L}_{\text{NLL}}(\theta) = -\sum_{i=1}^I \sum_{t=1}^T \log p(x_t|\mathbf{x}_{1:t-1})$.

2.2 STOCHASTIC GRADIENT VARIATIONAL BAYES

SGVB was introduced independently by Rezende et al. (2014) and Kingma & Welling (2013). For this paper, we will review the method briefly in order to introduce notation. We are interested in modelling the data distribution $p(\mathbf{x})$ with the help of unobserved latent variable $\mathbf{z}$ represented as a directed graphical model, i.e. $p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$. The integral is in general intractable, which is why we will use a variational upper bound on the negative log-likelihood for learning.

$$\begin{aligned}
-\log p(\mathbf{x}) &= -\log \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})dz \\
&= -\log \int \frac{q(\mathbf{z}|\mathbf{x})}{q(\mathbf{z}|\mathbf{x})} p(\mathbf{x}|\mathbf{z})p(\mathbf{z})dz \\
&\leq KL(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) - \mathbb{E}_{z\sim q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z})] =: \mathcal{L}.
\end{aligned}$$

where $KL(q||p)$ denotes the Kullback-Leibler divergence of p from q. In this case, we call q the *recognition model* since it allows for fast approximate inference of the latent variables $\mathbf{z}$ given the

observed variables $\mathbf{x}$. Note that q is a variational approximation of $p(\mathbf{z}|\mathbf{x})$, which is the inverse of the *generating model*[1] $p(\mathbf{x}|\mathbf{z})$ that cannot be found in general.

Both the recognition and the generating model can be chosen arbitrarily in their computational form with the possibility to represent probability distributions as outputs and stochastic training being the only requirements. In order to minimise the upper bound of the negative log-likelihood $\mathcal{L}$ with numerical means, it is convenient to choose parametric models. In that case we write $p(\mathbf{x}|\mathbf{z}, \theta^g)$ and $q(\mathbf{z}|\mathbf{x}, \theta^r)$ to make the dependency on the respective parameter sets explicit. Learning good parameters can then be done by performing stochastic optimization of $\mathcal{L}$ with respect to both θ^r and θ^g, where the expectation term is approximated by single draws from q in each training step.

Designing a model is then done by the following steps: (1) Choice of a prior $p(\mathbf{z})$ over the latent variables. (2) Choice of a recognition model $q(\mathbf{z}|\mathbf{x}, \theta^r)$. The Kullback-Leibler divergence between the prior and the recognition model has to be tractable and efficient to compute. (3) Choice of a generating model $p(\mathbf{x}|\mathbf{z}, \theta^g)$, which is often given by the type of data under investigation.

An important question is that of the representation capabilities of such a model. It turns out that if the distribution $p(x|z)$ is a universal *function* approximator, so is the overall model. An argument for the one-dimensional case is as follows. Assume random variables x and z with respective distribution functions F_x and F_z. According to the inverse transform technique theorem (Grimmett & Stirzaker, 1992), $u = F_x^{-1}(x)$ will be uniformly distributed over the interval $[0, 1]$ and so will be $u' = F_z^{-1}(z)$. Equating gives $F_z^{-1}(z) = F_x^{-1}(x) \Rightarrow F_x(F_z^{-1}(z)) = x$. Therefore setting $p(x|z) := \delta(x = f(z))$ with $F = F_x \circ F_z^{-1}$ makes $p(x) = \int_z p(x|z)p(z)dz$. An extension to the multidimensional case can be done by applying the above to the individual factors of a cascade decomposition and requiring x and z to be of the same dimensionality. The burden is then on the learning algorithm to find a good approximation for F.

3 METHODS

We propose to combine SGVB and RNNs by making use of an sRNN for both the recognition model $q(z_t|\mathbf{x}_{1:t-1})$ and the generating model $p(x_t|\mathbf{z}_{1:t})$.

3.1 THE GENERATING MODEL

More specifically, the generating model is an sRNN where the latent variables form additional inputs:

$$h_t = f_h(x_t \mathbf{W}^g_{\text{in}} + z_t \mathbf{W}'^g_{\text{in}} + h_{t-1}\mathbf{W}_{\text{rec}} + \mathbf{b}_{\text{hidden}}) \tag{4}$$

which replaces Eq. (2). We let y_t from Eq. (3) represent the necessary statistics to fully determine $p(x_{t+1}|\mathbf{x}_{1:t})$.

Note that the model reduces to an sRNN as soon as we remove any latent variables, e.g. by setting $\mathbf{W}'^g_{\text{in}} = \mathbf{0}$. Hence, such a model generalises sRNNs.

The only quantities bearing uncertainty in the calculation of $\mathbf{h}_{1:T}$ are the latent variables $\mathbf{z}_{1:T}$, as $\mathbf{x}_{1:T}$ stems from the data set and for all t, h_t is a deterministic function of $\mathbf{x}_{1:t}$ and $\mathbf{z}_{1:t}$. The resulting factorisation of the data likelihood of a single sequence $p(\mathbf{x}_{1:T})$ is then

$$\begin{aligned} p(\mathbf{x}_{1:T}) &= \prod_{t=0}^{T-1} p(x_{t+1}|\mathbf{x}_{1:t}) \\ &= \int_{\mathbf{z}_{1:T}} p(\mathbf{z}_{1:T}) \prod_{t=0}^{T-1} p(x_{t+1}|\mathbf{x}_{1:t}, \mathbf{z}_{1:t}, \cancel{\mathbf{z}_{t+1:T}}) d\mathbf{z}_{1:T} \\ &= \int_{\mathbf{z}_{1:T}} p(\mathbf{z}_{1:T}) \prod_{t=0}^{T-1} \int_{h_t} p(x_{t+1}|\mathbf{x}_{1:t}, \mathbf{z}_{1:t}, h_t) p(h_t|\mathbf{x}_{1:t}, \mathbf{z}_{1:t}) dh_t d\mathbf{z}_{1:T}, \end{aligned}$$

[1] We use the non standard term "generating model" for $p(\mathbf{x}|\mathbf{z})$ to distinguish it more clearly from the generative model $p(\mathbf{x})$.

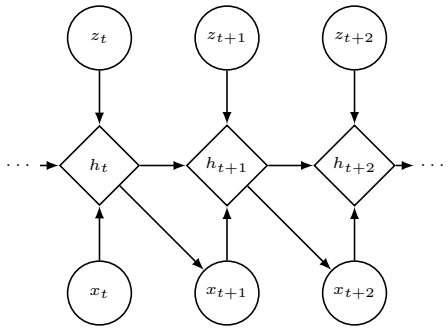

Figure 1: Graphical model corresponding to the factorisation given in Eq. (5). The hidden states h_t are shown as diamonds to stress that they are no source of stochasticity. Despite of this, marginalising out $\mathbf{z}_{1:T}$ makes $\mathbf{h}_{1:T}$ stochastic.

where we have made use of the fact that x_{t+1} is independent of $\mathbf{z}_{t+1:T}$. Since h_t is a deterministic function of $\mathbf{x}_{1:t}$ and $\mathbf{z}_{1:t}$, we note that $p(h_t|\mathbf{x}_{1:t}, \mathbf{z}_{1:t})$ follows a Dirac distribution with its mode given by Eq. (4). Thus, the integral over the hidden states is replaced by a single point; we make the dependency of h_t on both $\mathbf{z}_{1:t}$ and $\mathbf{x}_{1:t}$ explicit.

$$p(\mathbf{x}_{1:T}) = \int_{\mathbf{z}_{1:T}} p(\mathbf{z}_{1:T}) \prod_{t=0}^{T-1} p(x_{t+1}|h_t(\mathbf{x}_{1:t}, \mathbf{z}_{1:t}))d\mathbf{z}_{1:T}. \tag{5}$$

The corresponding graphical model is shown in Figure 3.1. Even though the determinism of h_t might seem restrictive at first, we will argue that it is not. Let $\mathbf{h}_{1:T}$ be the sequence of hidden layer activations as given by Eq. (4). This sequence is deterministic given $\mathbf{x}_{1:T}$ and $\mathbf{z}_{1:T}$ and consequently, $p(\mathbf{h}_{1:T}|\mathbf{x}_{1:T}, \mathbf{z}_{1:T})$ will follow a Dirac distribution. Marginalising out $\mathbf{z}_{1:T}$ will however lead to a universal approximator of probability distributions over sequences, analoguously to the argument given in Section 2.2.

An additional consequence is, that we can restrict ourselves to prior distributions over the latent variables that factorise over time steps, i.e. $p(\mathbf{z}_{1:T}) = \prod_t p(z_t)$. This is much easier to handle in practice, as calculating necessary quantities such as the KL-divergence can be done independently over all time steps and components of z_t.

Despite of this, the distribution over $\mathbf{h}_{1:T}$ will be a Markov chain and can exhibit stochastic behaviour, if necessary for modelling the data distribution.

3.2 Variational Inference for Latent State Sequences

The derivation of the training criterion is done by obtaining a variational upper bound on the negative log-likelihood via Jensen's inequality, where we use a variational approximation $q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) \approx p(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})$.

$$\begin{aligned}
-\log p(\mathbf{x}_{1:T}) &= -\log \int_{\mathbf{z}_{1:T}} \frac{q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})}{q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})} p(\mathbf{z}_{1:T}) \prod_{t=0}^{T-1} p(x_{t+1}|h_t(\mathbf{x}_{1:t}, \mathbf{z}_{1:t}))d\mathbf{z}_{1:T} \\
&\leq KL(q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})|p(\mathbf{z}_{1:T})) - \mathbb{E}_{\mathbf{z}_{1:T}\sim q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})}[\sum_{t=0}^{T-1} \log p(x_t|h_{t-1}, \mathbf{z}_{1:t})] \qquad (6) \\
&:= \mathcal{L}_{\text{STORN}}
\end{aligned}$$

In this work, we restrict ourselves to a standard Normal prior[2] of the form

$$p(\mathbf{z}_{1:T}) = \prod_{t,k} \mathcal{N}(z_{t,k}|0, 1),$$

[2]In a preliminary report, we proposed the use of a Wiener process for a prior. However, the presented results were invalid due to implementation errors and the paper has been withdrawn.

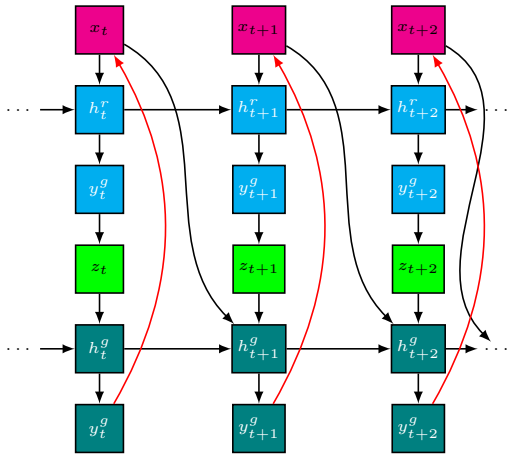

Figure 2: Diagram of the *computational* dependencies of STORNs. Each node of the graph corresponds to a vectorial quantity. The different types of nodes shown are data (magenta), the recognition model (cyan), samples (green) and the generating model (teal). Note that the outputs of the recognition model y^r_t depict the statistics of $q(z_t|\mathbf{x}_{1:t})$, from which the sample z_t (green) is drawn. The output of the generating model, y^g_t is used to represent $p(x_{t+1}|\mathbf{x}_{1:t})$. The red arrow expresses that this prediction is used to evaluate the loss, i.e. the negative log-likelihood.

where $z_{t,k}$ is the value of the k'th latent sequence at time step t.

The recognition model q will in this case be parameterised by a single mean $\mu_{t,k}$ and variance $\sigma^2_{t,k}$ for each time step and latent sequence. Both will be represented by the output of a recurrent net, which thus has 2ω outputs of which the first ω (representing the mean) will be unconstrained, while the second ω (representing the variance) need to be strictly positive. Given the output $\mathbf{y}_{1:T} = f^r(\mathbf{x}_{1:T})$ of the recognition RNN f^r, we set

$$\mu_{t,k} = y_{t,k},$$
$$\sigma^2_{t,k} = y^2_{t,k+\omega}.$$

Note that the square ensures positiveness.

Going along with the reparametrisation trick of Kingma & Welling (2013), we will sample from a standard Normal at each time step, i.e. $\epsilon_{t,k} \sim \mathcal{N}(0,1)$ and use it to sample from q via $z_{t,k} = \mu_{t,k} + \sigma_{t,k}\epsilon_{t,k}$. Given the complete sample sequence $\mathbf{z}_{1:T}$ we calculate the two terms of Equation (6). The KL-divergence can be readily computed, while we need to pass $\mathbf{z}_{1:T}$ through the generating model f^g which gives $-\log p(\mathbf{x}_{1:T}|\mathbf{z}_{1:T})$. The computational flow is illustrated in Figure 3.2.

3.3 Comparison to RNNs

An important question is whether the proposed model offers any theoretical improvements over RNNs with no latent variables. The approximation capabilities (with respect to probability distributions) of RNNs result from the choice of likelihood function, i.e. the way the density of the observations at time step t is determined by the outputs of the network, y_t. See Eq. (1). We have argued in Section 1 that a naïve Bayes assumption reduces the approximation capabilities. One way to circumvent this is to use mixture distributions (Graves, 2013). The number of parameters of the latter scales poorly, though: linear in the number of modes, hidden units in the last layer and output dimensions.

Both approaches also share the drawback that the stochasticity entering the computation is not represented in the hidden layers: drawing a sample is determined by a random process invisible to the network.

STORN overcomes both of these issues. Introducing an additional mode merely requires an additional change of curvature in the approximation of F (compare Section 2.2). This can be obtained by additional hidden units, for which the number of parameters scales linearly in the number of hidden

Table 1: Results on the midi data sets. All numbers are average negative log-likelihoods on the test set, where "FD-RNN" represents the work from Bayer et al. (2013a); "sRNN" and "RNN-NADE" results are from Bengio et al. (2012) while "Deep RNN" shows the best results from Pascanu et al. (2013). The results of our work are shown as "STORN" and have been obtained by means of the importance sampler described in Rezende et al. (2014).

Data set	STORN	FD-RNN	sRNN	RNN-NADE	Deep RNN
Piano-midi.de	7.13	7.39	7.58	7.05	–
Nottingham	2.85	3.09	3.43	2.31	2.95
MuseData	6.16	6.75	6.99	5.60	6.59
JSBChorales	6.91	8.01	8.58	5.19	7.92

units in the incoming and outgoing layer. Further, the stochasticity in the network is stemming from z of which the hidden layer is a function.

4 EXPERIMENTS

For evaluation we trained the proposed model on a set of midi music, which was used previously (Bengio et al., 2012; Pascanu et al., 2013; Bayer et al., 2013a; Boulanger-Lewandowski et al., 2012) to evaluate RNNs. We also investigated modelling human motion in the form of motion capture data (Boulanger-Lewandowski et al., 2012; Sutskever et al., 2008; Taylor et al., 2006). We employ Fast Dropout Recurrent Networks (FD-RNNs) (Bayer et al., 2013a) for both the recognition and the generating model. While we determine the dropout rates for the generating model via model selection on a validation set, we include them into the parameter set for the recognition model. In a manner similar to Bayer et al. (2013b), we exploit fast dropout's natural inclusion of variance as the variance for the recognition model, i.e. $\sigma^2_{t,k}$. We used Adadelta (Zeiler, 2012) enhanced with Nesterov momentum (Sutskever et al., 2013) for optimisation.

4.1 POLYPHONIC MUSIC GENERATION

All experiments were done by performing a random search (Bergstra & Bengio, 2012) over the hyper parameters, where 128 runs were performed for each data set. Both the recognition and the generating model used 300 hidden units with the logistic sigmoid as the transfer function. We report the estimated negative log-likelihood (obtained via the optimiser proposed in (Rezende et al., 2014)) on the test set of the parameters which yielded the best bound on the validation set.

As expected, STORN improves over the models assuming a factorised output distribution (FD-RNN, sRNN, Deep RNN) in all cases. Still, RNN-NADE has a competitive edge. The reasons for this remain unclear from the results alone, but the stochastic training and resulting noisy gradients are a viable hypothesis, since RNN-NADE does not suffer from those.

4.2 MOTION CAPTURE DATA

The motion capture data set (Hsu et al., 2005; Taylor et al., 2006) is a sequence of kinematic quantities obtained from a human body during walking. It consists of 3128 time steps of 49 angular quantities each. The experimental protocol of previous studies of this data set is to report the mean squared error on the training set , which we comply with. [3]

For motion capture data, we chose a Gaussian likelihood with a fixed standard deviation for the generating model. The recognition model was chosen to be a bidirectional RNN. While the standard deviation was fixed to 1 during training, we performed a binary search for a better value after training; the resulting estimate of the negative log-likelihood on the validation set was then used for model selection.

[3]The use of the MSE on the trainig set is debatable for this task. First, there is the danger of overfitting the training set. Second, the metric only captures a single moment of the residual distribution. We go forward with this protocol nonetheless to make our results comparable to previous works. Additionally, we report the negative log-likelihood, which is the right metric for the task.

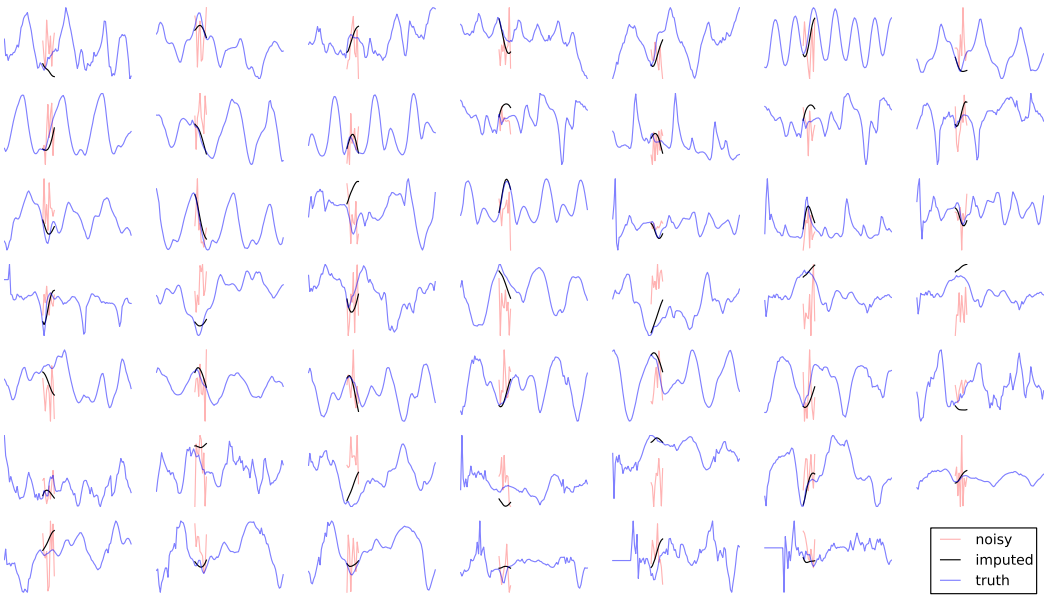

Figure 3: Illustration of missing value imputation on the motion capture data set. We show the first 48 of the 49 channels of a random sample, where time steps 30 to 40 were initialised with random noise. Subsequently, a maximum a posteriori point estimate of the latent variables was used to reconstruct the missing parts of the signals.

The estimated negative log-likelihood of the data was 15.99. Other models trained on this data set, namely the RNN-RBM, RTRBM and cRBM do not offer a tractable way of estimating the log-likelihood of the data, which is why there is no direct mean of comparison respecting the probabilistic nature of the models. In the case of the former two, the mean squared prediction error is reported instead, which is 20.1 and 16.2 respectively. Our method achieved an average MSE of 4.94, which is substantially less than previously reported results.For additional means of comparison, we performed approximate missing value imputation of motion capture data. We picked random sequences of length 60 and replaced all of the 49 channels from time steps 30 to 40 with standard normal noise. We then performed a *maximum a posteriori* point selection of the recognition model, i.e. $\operatorname{argmax}_{\hat{\mathbf{z}}_{1:T}} q(\hat{\mathbf{z}}_{1:T}|\mathbf{x}_{1:T})$, from which we reconstructed the output via $\operatorname{argmax}_{\hat{\mathbf{x}}_{30:40}} \log p(\mathbf{x}_{1:T}|\hat{\mathbf{z}}_{1:T})$. Note that this method is different from the one proposed in (Rezende et al., 2014), where an iterative scheme is used. We also experimented with that method, but did not find it to yield substantially better results. The results of the imputations are shown in Figure 3.

To demonstrate the generative capabilities of the method, we drew 50 samples from the model after initialising it with a stimulus prefix. The stimulus had a length of 20, after which we ran the model in "generating mode" for another 80 time steps. This was done by feeding the mean of the model's output at time step t into the generating model at time step $t + 1$. Additionally, we drew $\mathbf{z}_{20:80}$ from the prior. The results are visualised in Figure 4.

5 Discussion and Future Work

We have presented a model class of stochastic RNNs that can be trained with a recently proposed estimator, SGVB. The resulting model fulfills the expectation to greatly improve over the performance of sRNNs erroneously assuming a factorisation of output variables. An important take away message of this work is that the performance of RNNs can greatly benefit from more sophisticated methods that greatly improve the representative capabilities of the model.

While not shown in this work, STORNs can be readily extended to feature computationally more powerful architectures such as LSTM or deep transition operators (Hochreiter & Schmidhuber, 1997; Pascanu et al., 2013).

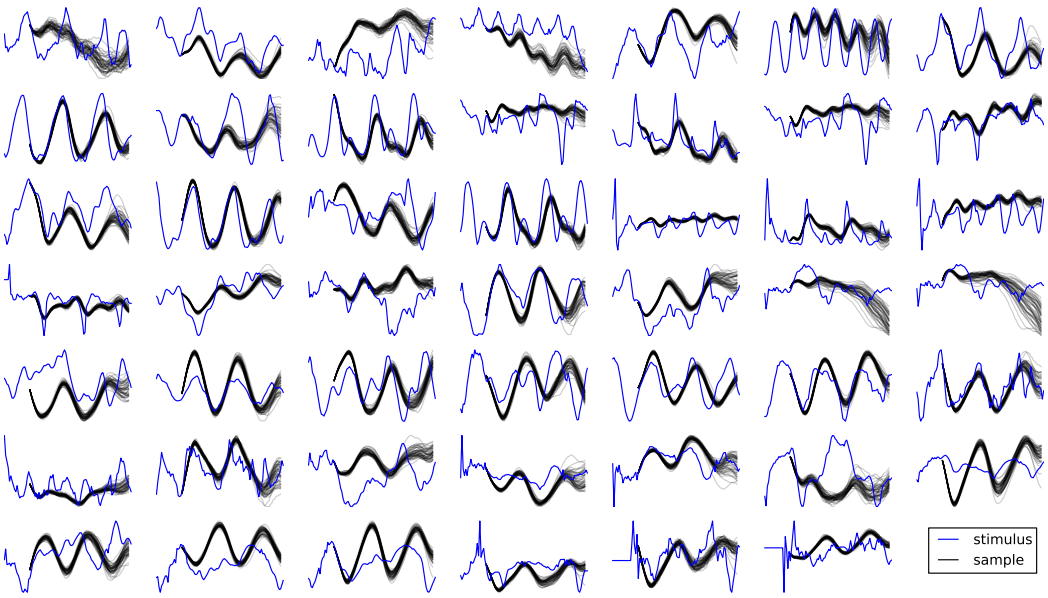

Figure 4: Samples from the model trained on motion capture data after providing a stimulus prefix sequence of 20 time steps. The uncertainty of the learned distribution is visible by the diversity of the samples; nevertheless, the distribution is rather unimodal.

Still, an apparent weakness seems to be the stochastic objective function. Thankfully, research in optimisation of stochastic objective functions has far from halted and we believe STORN to benefit from any advances in that area.

6 Acknowledgements

Part of this work has been supported by the TACMAN project, EC Grant agreement no. 610967, within the FP7 framework programme.

References

Bayer, Justin, Osendorfer, Christian, Korhammer, Daniela, Chen, Nutan, Urban, Sebastian, and van der Smagt, Patrick. On fast dropout and its applicability to recurrent networks. *arXiv preprint arXiv:1311.0701*, 2013a.

Bayer, Justin, Osendorfer, Christian, Urban, Sebastian, et al. Training neural networks with implicit variance. In *Proceedings of the 20th International Conference on Neural Information Processing , ICONIP-2013*, 2013b.

Bengio, Y., Boulanger-Lewandowski, N., and Pascanu, R. Advances in optimizing recurrent networks. *arXiv preprint arXiv:1212.0901*, 2012.

Bergstra, James and Bengio, Yoshua. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13:281–305, 2012.

Boulanger-Lewandowski, N., Bengio, Y., and Vincent, P. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*, 2012.

Boulanger-Lewandowski, Nicolas, Bengio, Yoshua, and Vincent, Pascal. High-dimensional sequence transduction. In *ICASSP*, 2013.

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

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

Graves, Alex, Fernández, Santiago, Liwicki, Marcus, Bunke, Horst, and Schmidhuber, Jurgen. Unconstrained online handwriting recognition with recurrent neural networks. *Advances in Neural Information Processing Systems*, 20:1–8, 2008.

Graves, Alex, Mohamed, Abdel-rahman, and Hinton, Geoffrey. Speech recognition with deep recurrent neural networks. *arXiv preprint arXiv:1303.5778*, 2013.

Grimmett, Geoffrey and Stirzaker, David. *Probability and random processes*, volume 2. Oxford Univ Press, 1992.

Hammer, Barbara. On the approximation capability of recurrent neural networks. *Neurocomputing*, 31(1):107–123, 2000.

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

Hsu, Eugene, Pulli, Kari, and Popović, Jovan. Style translation for human motion. *ACM Transactions on Graphics (TOG)*, 24(3):1082–1089, 2005.

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

Martens, J. and Sutskever, I. Learning recurrent neural networks with hessian-free optimization. *Proc. 28th Int. Conf. on Machine Learning*, 2011.

Pascanu, Razvan, Gulcehre, Caglar, Cho, Kyunghyun, and Bengio, Yoshua. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*, 2013.

Rezende, Danilo Jimenez, Mohamed, Shakir, and Wierstra, Daan. Stochastic back-propagation and variational inference in deep latent gaussian models. *arXiv preprint arXiv:1401.4082*, 2014.

Siegelmann, Hava T and Sontag, Eduardo D. Turing computability with neural nets. *Applied Mathematics Letters*, 4(6):77–80, 1991.

Sutskever, I., Hinton, G., and Taylor, G. The recurrent temporal restricted boltzmann machine. *Advances in Neural Information Processing Systems*, 21, 2008.

Sutskever, Ilya, Martens, James, Dahl, George, and Hinton, Geoffrey. On the importance of initialization and momentum in deep learning. 2013.

Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*, 2014.

Tang, Yichuan and Salakhutdinov, Ruslan. A new learning algorithm for stochastic feedforward neural nets. 2013.

Taylor, Graham W, Hinton, Geoffrey E, and Roweis, Sam T. Modeling human motion using binary latent variables. In *Advances in neural information processing systems*, pp. 1345–1352, 2006.

Welling, Max, Rosen-Zvi, Michal, and Hinton, Geoffrey E. Exponential family harmoniums with an application to information retrieval. In *Advances in neural information processing systems*, pp. 1481–1488, 2004.

Zeiler, Matthew D. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.