# HYPERNETWORKS

**David Ha**[*], **Andrew Dai, Quoc V. Le**
Google Brain
{hadavid,adai,qvl}@google.com

## ABSTRACT

This work explores hypernetworks: an approach of using one network, also known as a hypernetwork, to generate the weights for another network. We apply hypernetworks to generate adaptive weights for recurrent networks. In this case, hypernetworks can be viewed as a relaxed form of weight-sharing across layers. In our implementation, hypernetworks are are trained jointly with the main network in an end-to-end fashion. Our main result is that hypernetworks can generate non-shared weights for LSTM and achieve state-of-the-art results on a variety of sequence modelling tasks including character-level language modelling, handwriting generation and neural machine translation, challenging the weight-sharing paradigm for recurrent networks.

## 1 INTRODUCTION

In this work, we consider an approach of using a small network (called a "hypernetwork") to generate the weights for a larger network (called a main network). The behavior of the main network is the same as with any usual neural network: it learns to map some raw inputs to their desired targets; whereas the hypernetwork takes a set of inputs that contain information about the structure of the weights and generates the weights for that layer.

The focus of this work is to use hypernetworks to generate weights for recurrent networks (RNN). In this case, the weights $W_t$ for the main RNN at step $t$ is a function of the input to the hidden state of the main RNN at the previous step $h_{t-1}$ and the input at the current time step $x_t$. This weight-generation scheme allows approximate weight-sharing across layers of the main RNN.

We perform experiments to investigate the behaviors of hypernetworks in a range of contexts and find that hypernetworks mix well with other techniques such as batch normalization and layer normalization. Our main result is that hypernetworks can generate non-shared weights for LSTM that work better than the standard version of LSTM (Hochreiter & Schmidhuber, 1997). On language modelling tasks with character Penn Treebank, Hutter Prize Wikipedia datasets, hypernetworks for LSTM achieve near state-of-the-art results. On a handwriting generation task with IAM handwriting dataset, hypernetworks for LSTM achieves good quantitative and qualitative results. On machine translation, hypernetworks for LSTM also obtain state-of-the-art performance on the WMT'14 en$\rightarrow$fr benchmark.

## 2 RELATED WORK

Our approach is inspired by methods in evolutionary computing, where it is difficult to directly operate in large search spaces consisting of millions of weight parameters. A more efficient method is to evolve a smaller network to generate the structure of weights for a larger network, so that the search is constrained within the much smaller weight space. An instance of this approach is the work on the HyperNEAT framework (Stanley et al., 2009). In the HyperNEAT framework, Compositional Pattern-Producing Networks (CPPNs) are evolved to define the weight structure of the much larger main network. Closely related to our approach is a simplified variation of HyperNEAT, where the structure is fixed and the weights are evolved through Discrete Cosine Transform (DCT), called Compressed Weight Search (Koutnik et al., 2010). Even more closely related to our

[*]Work done as a member of the Google Brain Residency program (g.co/brainresidency).

approach are Differentiable Pattern Producing Networks (DPPNs), where the structure is evolved but the weights are learned (Fernando et al., 2016), and ACDC-Networks (Moczulski et al., 2015), where linear layers are compressed with DCT and the parameters are learned. Most reported results using these methods, however, are in small scales, perhaps because they are both slow to train and require heuristics to be efficient. The main difference between our approach and HyperNEAT is that hypernetworks in our approach are trained end-to-end with gradient descent together with the main network, and therefore are more efficient.

Another closely related idea to hypernetworks is the concept of fast weights Schmidhuber (1992; 1993) in which one network can produce context-dependent weight changes for a second network. Small scale experiments were conducted to demonstrate fast weights for feed forward networks at the time, but perhaps due to the lack of modern computational tools, the recurrent network version was mentioned mainly as a thought experiment (Schmidhuber, 1993). A subsequent work demonstrated practical applications of fast weights (Gomez & Schmidhuber, 2005), where a generator network is learnt through evolution to solve an artificial control problem.

The focus of this work is to apply our method to recurrent networks. In this context, our method has a connection to second-order or multiplicative networks (Goudreau et al., 1994; Sutskever et al., 2011; Wu et al., 2016), where the hidden state of the last step and the input vector of the current time step interact in a multiplicative fashion. The key difference between our approach and second-order networks is that our approach is more memory efficient because we only learn the scaling factors in the interaction matrix. Furthermore, in second-order or multiplicative networks, the weights of the RNN are not fixed, but a linear function of the previous hidden state. In our work, we explore the use of a smaller RNN, rather than a linear function, to produce the weights of the main RNN.

The concept of a network interacting with another network is central to the work of (Jaderberg et al., 2016; Andrychowicz et al., 2016), and especially (Denil et al., 2013; Yang et al., 2015; Bertinetto et al., 2016; De Brabandere et al., 2016), where certain parameters in a convolutional network are predicted by another network. These studies however did not explore the use of this approach to recurrent networks, which is a main contribution of our work.

## 3 METHODS

Though it is possible to use hypernetworks to generate weights for feedforward or convolutional networks, in this paper, our focus is on using hypernetworks with recurrent networks. As can be seen below, when they are applied to recurrent networks, hypernetworks can be seen as a form of relaxed weight-sharing in the time dimension.

### 3.1 HYPERRNN

Our hypernetworks can be used to generate weights for the RNN and LSTM. When a hypernetwork is used to generate the weights for an RNN, we refer to it as the HyperRNN. At every time step $t$, a HyperRNN takes as input the concatenated vector of input $x_t$ and the hidden states of the main RNN $h_{t-1}$, it then generates as output the vector $\hat{h}_t$. This output vector is then used to generate the weights for the main RNN at the same timestep. Both the HyperRNN and the main RNN are trained jointly with backpropagation and gradient descent. In the following, we will give a more formal description of the model.

The standard formulation of a Basic RNN is given by:

$$h_t = \phi(W_h h_{t-1} + W_x x_t + b) \tag{1}$$

where $h_t$ is the hidden state, $\phi$ is a non-linear operation such as $tanh$ or $relu$, and the weight matrices and bias $W_h \in \mathbb{R}^{N_h \times N_h}, W_x \in \mathbb{R}^{N_h \times N_x}, b \in \mathbb{R}^{N_h}$ is fixed each timestep for an input sequence $X = (x_1, x_2, \ldots, x_T)$.

In HyperRNN, we allow $W_h$ and $W_x$ to float over time by using a smaller hypernetwork to generate these parameters of the main RNN at each step (see Figure 1). More concretely, the parameters $W_h, W_x, b$ of the main RNN are different at different time steps, so that $h_t$ can now be computed as:

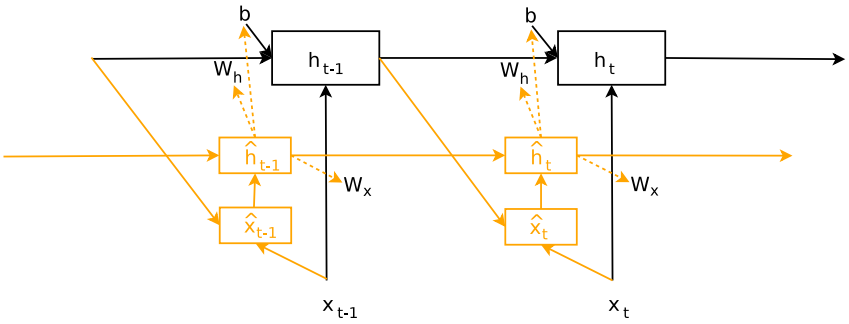


Figure 1: An overview of HyperRNNs. Black connections and parameters are associated basic RNNs. Orange connections and parameters are introduced in this work and associated with HyperRNNs. Dotted arrows are for parameter generation.

$$
\begin{aligned}
h_t &= \phi\big(W_h(z_h)h_{t-1} + W_x(z_x)x_t + b(z_b)\big), \text{ where} \\
W_h(z_h) &= \langle W_{hz}, z_h \rangle \\
W_x(z_x) &= \langle W_{xz}, z_x \rangle \\
b(z_b) &= W_{bz}z_b + b_0
\end{aligned}
\tag{2}
$$

Where $W_{hz} \in \mathbb{R}^{N_h \times N_h \times N_z}, W_{xz} \in \mathbb{R}^{N_h \times N_x \times N_z}, W_{bz} \in \mathbb{R}^{N_h \times N_z}, b_0 \in \mathbb{R}^{N_h}$ and $z_h, z_x, z_z \in \mathbb{R}^{N_z}$. We use a recurrent hypernetwork to compute $z_h, z_x$ and $z_b$ as a function of $x_t$ and $h_{t-1}$:

$$
\begin{aligned}
\hat{x}_t &= \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \\
\hat{h}_t &= \phi(W_{\hat{h}}\hat{h}_{t-1} + W_{\hat{x}}\hat{x}_t + \hat{b}) \\
z_h &= W_{\hat{h}h}\hat{h}_{t-1} + b_{\hat{h}h} \\
z_x &= W_{\hat{h}x}\hat{h}_{t-1} + b_{\hat{h}x} \\
z_b &= W_{\hat{h}b}\hat{h}_{t-1}
\end{aligned}
\tag{3}
$$

Where $W_{\hat{h}} \in \mathbb{R}^{N_{\hat{h}} \times N_{\hat{h}}}, W_{\hat{x}} \in \mathbb{R}^{N_{\hat{h}} \times (N_h + N_z)}, b \in \mathbb{R}^{N_{\hat{h}}}$, and $W_{\hat{h}h}, W_{\hat{h}x}, W_{\hat{h}b} \in \mathbb{R}^{N_z \times N_{\hat{h}}}$ and $b_{\hat{h}h}, b_{\hat{h}x} \in \mathbb{R}^{N_z}$. This *HyperRNN cell* has $N_{\hat{h}}$ hidden units. Typically $N_{\hat{h}}$ is much smaller than $N_h$.

As the embeddings $z_h$, $z_x$ and $z_b$ are of dimensions $N_z$, which is typically smaller than the hidden state size $N_{\hat{h}}$ of the HyperRNN cell, a linear network is used to project the output of the HyperRNN cell into the embeddings in Equation 3. After the embeddings are computed, they will be used to generate the full weight matrix of the main RNN.

The above is a general formulation of HyperRNN. However, Equation 2 is not practical because the memory usage becomes too large for real problems. We modify the HyperRNN described in Equation 2 so that it can be more memory efficient. We will use an intermediate hidden vector $d(z) \in \mathbb{R}^{N_h}$ to parametrize each weight matrix, where $d(z)$ will be a linear function of $z$. To dynamically modify a weight matrix $W$, we will allow each row of this weight matrix to be scaled linearly by an element in vector $d$. We refer $d$ as a *weight scaling vector*. Below is the modification to $W(z)$:

$$
W(z) = W\big(d(z)\big) = \begin{pmatrix} d_0(z)W_0 \\ d_1(z)W_1 \\ \dots \\ d_{N_h}(z)W_{N_h} \end{pmatrix}
\tag{4}
$$

While we sacrifice the ability to construct an entire weight matrix from a linear combination of $N_z$ matrices of the same size, we are able to linearly scale the rows of a single matrix with $N_z$ degrees of

freedom. We find this change to have a good trade-off, as this formulation of converting $W(z)$ into $W(d(z))$ decreases the amount of memory required by the HyperRNN. Rather than requiring $N_z$ times the memory of a Basic RNN, we will only be using memory in the order $N_z$ times the number of hidden units, which is an acceptable amount of extra memory usage that is often available in many applications. In addition, the row-level operation in Equation 4 can be shown to be equivalent to an element-wise multiplication operator and hence computationally much more efficient in practice. Below is the more memory efficient version of the setup of Equation 2:

$$\begin{aligned} h_t &= \phi\big(d_h(z_h) \odot W_h h_{t-1} + d_x(z_x) \odot W_x x_t + b(z_b)\big), \text{ where} \\ d_h(z_h) &= W_{hz} z_h \\ d_x(z_x) &= W_{xz} z_x \\ b(z_b) &= W_{bz} z_b + b_0 \end{aligned} \tag{5}$$

In our experiments, we focus on the use of hypernetworks with the Long Short-Term Memory (LSTM) architecture (Hochreiter & Schmidhuber, 1997) because LSTM often works better than the Basic RNN. In such case, an LSTM will have more weight matrices and biases, and thus our main change is to have many more $d$'s, each $d$ is being associated with each weight matrix or bias.

### 3.2 Related Approaches

The formulation of the HyperRNN in Equation 5 has similarities to Recurrent Batch Normalization (Cooijmans et al., 2016) and Layer Normalization (Ba et al., 2016). The central idea for the normalization techniques is to calculate the first two statistical moments of the inputs to the activation function, and to linearly scale the inputs to have zero mean and unit variance. After the normalization, an additional set of fixed parameters are learned to unscale the inputs if required.

Since the HyperRNN cell can indirectly modify the rows of each weight matrix and also the bias of the main RNN, it is implicitly also performing a linear scaling to the inputs of the activation function. The difference here is that the linear scaling parameters will be learned by the HyperRNN cell, and not based on statistical-moments. We note that the existing normalization approaches can work together with the HyperRNN approach, where the HyperRNN cell will be tasked with discovering a better dynamical scaling policy to complement normalization. We also explore this combination in our experiments.

The element-wise operation also has similarities to the Multiplicative RNN and its extensions (mRNN, mLSTM) (Sutskever et al., 2011; Krause et al., 2016) and Multiplicative Integration RNN (MI-RNN) (Wu et al., 2016). In the case of the mRNN, the hidden-to-hidden weight matrix is replaced with a factorized matrix, to allow the weights to be input dependent. The factorization is described below in Equation 6 (Krause et al., 2016).

$$h_t = \phi(W_{hm}(W_{mx}x_t) \odot (W_{mh}h_{t-1}) + W_x x_t + b) \tag{6}$$

For the MI-RNN approach, a second order term is added to the Basic RNN formulation, along with scaling vectors for each term, as described in Equation 7. The addition of the scaling vectors allow parameters to be shared more efficiently.

$$h_t = \phi(\alpha \odot W_x x_t \odot W_h h_{t-1} + \beta_1 \odot W_h h_{t-1} + \beta_2 \odot W_x x_t + b) \tag{7}$$

In the HyperRNN approach, the weights are also input dependent. However, unlike mRNN, both weight matrices and also the bias term will be dependent to not only to the inputs, but also to the hidden states. In the MI-RNN approach, the weights are also be augmented by both the input and hidden states, via the second order term in Equation 7. In both mRNN and MI-RNN approaches, the weight augmentation terms are produced by a linear operation, while in the HyperRNN approach, the weight scaling vectors $d$ are dynamically produced by another RNN with its own hidden states and non-linearities.

# 4 EXPERIMENTS

In the following experiments, we will benchmark the performance of HyperLSTM on language modelling with Penn Treebank, and Hutter Prize Wikipedia. We will also benchmark the method on the tasks of handwriting generation with IAM On-Line Handwriting Database, and machine translation with WMT'14 en→fr.

## 4.1 CHARACTER-LEVEL PENN TREEBANK LANGUAGE MODELLING

We first evaluation the HyperLSTM model on a character level prediction task with the Penn Treebank corpus (Marcus et al., 1993) using the train/validation/test split outlined in (Mikolov et al., 2012). As the dataset is quite small, we apply dropout on both input and output layers with a keep probability of 90%. Unlike previous approaches (Graves, 2013; Ognawala & Bayer, 2014) of applying weight noise during training, we instead also apply dropout to the recurrent layer (Henaff et al., 2016) with the same dropout probability.

We compare our model to the basic LSTM cell, stacked LSTM cells (Graves, 2013), and LSTM with layer normalization applied. In addition, we also experimented with applying layer normalization to HyperLSTM. Using the setup in (Graves, 2013), we use networks with 1000 units and train the network to predict the next character. In this task, the HyperLSTM cell has 128 units and an embedding size of 4. As the HyperLSTM cell has more trainable parameters compared to the basic LSTM cell, we also experimented with an LSTM cell with 1250 units.

For character-level Penn Treebank, we use mini-batches of size 128, to train on sequences of length 100. We train the model using Adam (Kingma & Ba, 2015) with a learning rate of 0.001 and gradient clipping of 1.0. During evaluation, we generate the entire sequence, and do not use information about previous test errors for prediction, e.g., dynamic evaluation (Graves, 2013; Rocki, 2016b). As mentioned earlier, we apply dropout to the input and output layers, and also apply recurrent dropout with a keep probability of 90%. For baseline models, orthogonal initialization (Henaff et al., 2016) is used for all weights.

We also experiment with a version of the model using a larger embedding size of 16, and also with a lower dropout keep probability of 85%, and report results with this "Large Embedding" model in Table 1. Lastly, we stack two layers of this "Large Embedding" model together to measure the benefits of a multi-layer version of HyperLSTM, with a dropout keep probability of 80%.

| **Model**[1] | **Test** | **Validation** | **Param Count** |
|---|---|---|---|
| ME n-gram (Mikolov et al., 2012) | 1.37 | | |
| Batch Norm LSTM (Cooijmans et al., 2016) | 1.32 | | |
| Recurrent Dropout LSTM (Semeniuta et al., 2016) | 1.301 | 1.338 | |
| Zoneout RNN (Krueger et al., 2016) | 1.27 | | |
| HM-LSTM[3] (Chung et al., 2016) | 1.27 | | |
| LSTM, 1000 units [2] | 1.312 | 1.347 | 4.25 M |
| LSTM, 1250 units[2] | 1.306 | 1.340 | 6.57 M |
| 2-Layer LSTM, 1000 units[2] | 1.281 | 1.312 | 12.26 M |
| Layer Norm LSTM, 1000 units[2] | 1.267 | 1.300 | 4.26 M |
| HyperLSTM (ours), 1000 units | 1.265 | 1.296 | 4.91 M |
| Layer Norm HyperLSTM, 1000 units (ours) | 1.250 | 1.281 | 4.92 M |
| Layer Norm HyperLSTM, 1000 units, Large Embedding (ours) | 1.233 | 1.263 | 5.06 M |
| 2-Layer Norm HyperLSTM, 1000 units | 1.219 | 1.245 | 14.41 M |

Table 1: Bits-per-character on the Penn Treebank test set.

Our results are presented in Table 1. The key observation here is that 1) HyperLSTM outperforms standard LSTM and 2) HyperLSTM also achieves similar improvements compared to Layer Normalization. The combination of Layer Normalization and Hyper LSTM achieves the best test perplexity so far on this dataset.

### 4.2 HUTTER PRIZE WIKIPEDIA LANGUAGE MODELLING

We train our model on the larger and more challenging Hutter Prize Wikipedia dataset, also known as `enwik8` (Hutter, 2012) consisting of a sequence of 100M characters composed of 205 unique characters. Unlike Penn Treebank, `enwik8` contains some foreign words (Latin, Arabic, Chinese), indented XML, metadata, and internet addresses, making it a more realistic and practical dataset to test character language models.

Our setup is similar in the previous experiment, using the same mini-batch size, learning rate, weight initialization, gradient clipping parameters and optimizer. We do not use dropout for the input and output layers, but still apply recurrent dropout with a keep probability of 90%. Similar to (Chung et al., 2015), we train on the first 90M characters of the dataset, use the next 5M as a validation set for early stopping, and the last 5M characters as the test set.

As `enwik8` is a bigger dataset compared to Penn Treebank, we will use 1800 units for our networks. We also perform training on sequences of length 250. Our normal HyperLSTM cell consists of 256 units, and we use an embedding size of 64. To improve results, we also experiment with a larger model where both HyperLSTM and main network both have 2048 hidden units. The HyperLSTM cell consists of 512 units with an embedding size of 64. We also apply recurrent dropout to this larger model, with dropout keep probability of 85%, and train on a longer sequence length of 300.

| **Model**[1] | **`enwik8`** | **Param Count** |
|---|---|---|
| Stacked LSTM (Graves, 2013) | 1.67 | 27.0 M |
| MRNN (Sutskever et al., 2011) | 1.60 | |
| Grid-LSTM (Kalchbrenner et al., 2016) | 1.47 | 16.8 M |
| LSTM (Rocki, 2016b) | 1.45 | |
| MI-LSTM (Wu et al., 2016) | 1.44 | |
| MLSTM (Krause et al., 2016) | 1.42 | |
| Recurrent Highway Networks (Zilly et al., 2016) | 1.42 | 8.0 M |
| Recurrent Memory Array Structures (Rocki, 2016a) | 1.40 | |
| HM-LSTM[3] (Chung et al., 2016) | 1.40 | |
| Surprisal Feedback LSTM[4] (Rocki, 2016b) | 1.37 | |
| LSTM, 1800 units, no recurrent dropout[2] | 1.470 | 14.81 M |
| LSTM, 2000 units, no recurrent dropout[2] | 1.461 | 18.06 M |
| Layer Norm LSTM, 1800 units[2] | 1.402 | 14.82 M |
| HyperLSTM (ours), 1800 units | 1.391 | 18.71 M |
| Layer Norm HyperLSTM, 1800 units (ours) | 1.353 | 18.78 M |
| Layer Norm HyperLSTM, 2048 units (ours) | 1.340 | 26.54 M |

Table 2: Bits-per-character on the `enwik8` test set.

The results are summarized in Table 2. As can be seen from the table, HyperLSTM is once again competitive to Layer Norm LSTM, and if we combine both techniques, the Layer Norm HyperLSTM achieves respectable results. The large version of HyperLSTM with normalization that uses 2048 hidden units achieve near state-of-the-art performance for this task. In addition, HyperLSTM converges more quickly compared to LSTM and Layer Norm LSTM (see Figure 2).

We perform additional analysis to understand the behavior of HyperLSTM by visualizing how the weight scaling vectors of the main LSTM change during the character sampling process. In Figure 3, we examine a sample text passage generated by HyperLSTM after training on `enwik8` along with the weight differences below the text. We see that in regions of low intensity, where the weights of the main LSTM are relatively static, the types of phrases generated seem more deterministic. For example, the weights do not change much during the words `Europeans`, `possessions` and `reservation`. The regions of high intensity is when the HyperLSTM cell is making relatively large changes to the weights of the main LSTM.

[1] We do not compare against methods that use dynamic evaluation.

[2] Our implementation.

[3] Based on results of version 2 at the time of writing. `http://arxiv.org/abs/1609.01704v2`

[4] This method uses information about test errors during inference for predicting the next characters, hence it is not directly comparable to other methods that do not use this information.

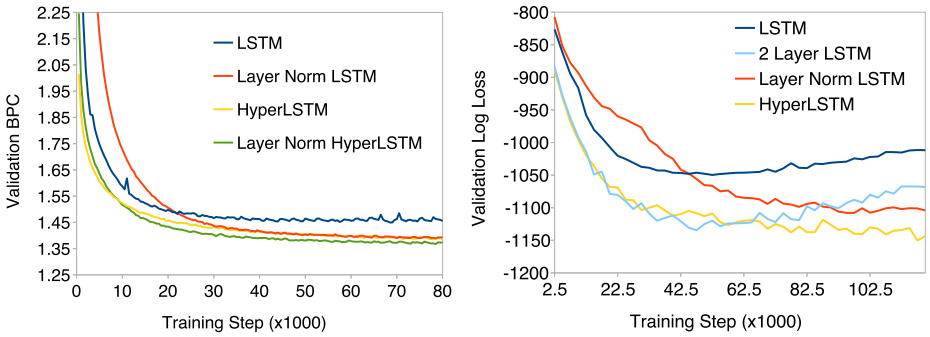


Figure 2: Loss graph for `enwik8` (left). Loss graph for Handwriting Generation (right)

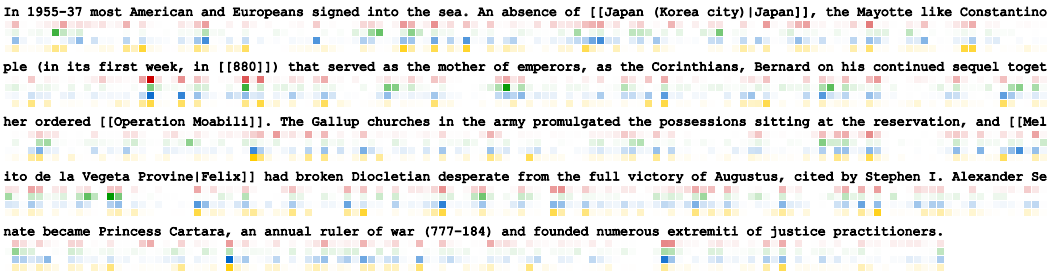


Figure 3: Example text generated from HyperLSTM model. We visualize how four of the main RNN's weight matrices ($W_h^i$, $W_h^g$, $W_h^f$, $W_h^o$) effectively change over time by plotting the norm of the changes below each generated character. High intensity represent large changes being made to weights of main RNN.

### 4.3 Handwriting Sequence Generation

In addition to modelling discrete sequential data, we want to see how the model performs when modelling sequences of real valued data. We will train our model on the IAM online handwriting database (Liwicki & Bunke, 2005) and have our model predict pen strokes as per Section 4.2 of (Graves, 2013). The dataset has contains 12179 handwritten lines from 221 writers, digitally recorded from a tablet. We will model the (x, y) coordinate of the pen location at each recorded time step, along with a binary indicator of pen-up/pen-down. The average sequence length is around 700 steps and the longest around 1900 steps, making the training task particularly challenging as the network needs to retain information about both the stroke history and also the handwriting style in order to predict plausible future handwriting strokes.

We will use the same model architecture described in (Graves, 2013) and use a Mixture Density Network layer (Bishop, 1994) to generate a mixture of bi-variate Gaussian distributions to model at each time step to model the pen location. We normalize the data and use the same train/validation split as per (Graves, 2013) in this experiment. We remove samples less than length 300 as we found these samples contain a lot of recording errors and noise. After the pre-processing, as the dataset is small, we introduce data augmentation of chosen uniformly from +/- 10% and apply a this random scaling a the samples used for training.

For model training, will apply recurrent dropout and also dropout to the output layer with a keep probability of 0.95. The model is trained on mini-batches of size 32 containing sequences of variable length. We trained the model using Adam (Kingma & Ba, 2015) with a learning rate of 0.0001 and gradient clipping of 5.0. Our HyperLSTM cell consists of 128 units and a signal size of 4. For baseline models, orthogonal initialization (Henaff et al., 2016) is performed for all weights.

[1]Our implementation, to replicate setup of (Graves, 2013).

[2]Our implementation, with data augmentation, dropout and recurrent dropout.

| Model | Log-Loss | Param Count |
|---|---|---|
| LSTM, 900 units (Graves, 2013) | -1,026 | |
| 3-Layer LSTM, 400 units (Graves, 2013) | -1,041 | |
| 3-Layer LSTM, 400 units, adaptive weight noise (Graves, 2013) | -1,058 | |
| LSTM, 900 units, no dropout, no data augmentation.[1] | -1,026 | 3.36 M |
| 3-Layer LSTM, 400 units, no dropout, no data augmentation.[1] | -1,039 | 3.26 M |
| LSTM, 900 units[2] | -1,055 | 3.36 M |
| LSTM, 1000 units[2] | -1,048 | 4.14 M |
| 3-Layer LSTM, 400 units[2] | -1,068 | 3.26 M |
| 2-Layer LSTM, 650 units[2] | -1,135 | 5.16 M |
| Layer Norm LSTM, 900 units[2] | -1,096 | 3.37 M |
| Layer Norm LSTM, 1000 units[2] | -1,106 | 4.14 M |
| Layer Norm HyperLSTM, 900 units (ours) | -1,067 | 3.95 M |
| HyperLSTM (ours), 900 units | -1,162 | 3.94 M |

Table 3: Log-Loss of IAM Online DB validation set.

The results are summarized in Table 3. Our main result is that HyperLSTM with 900 units, without Layer Norm, achieves best log loss on the validation set across all models in published work and in our experiment. HyperLSTM also converges more quickly compared to other models (see Figure 2).

Similar to the earlier character generation experiment, we show a generated handwriting sample from the HyperLSTM model in Figure 4, along with a plot of how the weight scaling vectors of the main RNN is changing over time below the sample. For a more detailed interactive demonstration of handwriting generation using HyperLSTM, visit `http://blog.otoro.net/2016/09/28/hyper-networks/`.

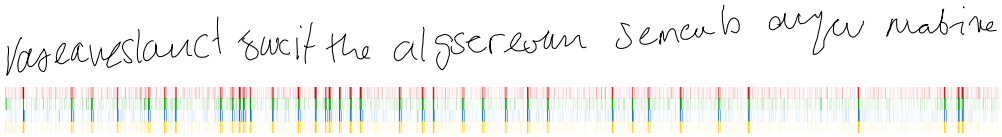

Figure 4: Handwriting sample generated from HyperLSTM model. We visualize how four of the main RNN's weight matrices ($W_h^i$, $W_h^g$, $W_h^f$, $W_h^o$) effectively change over time, by plotting norm of changes made to them over time.

We observe that the regions of high intensity is concentrated at many discrete instances, rather than slowly varying over time. This implies that the weights experience regime changes rather than gradual slow adjustments. We can see that many of these weight changes occur at the boundaries between words, and between characters. While the LSTM model alone already does a decent job of generating time-varying parameters of a Mixture Gaussian distribution used to generate realistic handwriting samples, the ability to go one level deeper, and to dynamically generate the generative model is one of the key advantages of HyperLSTM over a normal LSTM.

### 4.4 Neural Machine Translation

Finally, we experiment with the Neural Machine Translation task using the same experimental setup outlined in (Wu et al., 2016). Our model is the same wordpiece model architecture with a vocabulary size of 32k, but we replace the LSTM cells with HyperLSTM cells. We benchmark both models on WMT'14 En→Fr using the same test/validation set split described in the GNMT paper (Wu et al., 2016). The GNMT network has 8 layers in the encoder, 8 layers in the decoder. The first layer of the encoder has bidirectional connections. The attention module is a neural network with 1 hidden layer. When a LSTM cell is used, the number of hidden units in each layer is 1024. The model is trained in a distributed setting with a parameter sever and 12 workers. Additionally, each worker uses 8 GPUs and a minibatch of 128.

Our experimental setup is similar to that in the GNMT paper (Wu et al., 2016), with two simplifications. First, we use only Adam without SGD at the end. Adam was used with the same same hyperparameters described in the GNMT paper: learning rate of 0.0002 for 1M training steps.

We apply the HyperLSTM cell with Layer Norm to the GNMT architecture that uses a vocabulary of 32K wordpieces. We keep the same number of hidden units, which means that our model will have 16% more parameters.

| **Model** | **Test BLEU** | **Log Perplexity** | **Param Count** |
|---|---|---|---|
| Deep-Att + PosUnk (Zhou et al., 2016) | 39.2 | | |
| GNMT WPM-32K, LSTM (Wu et al., 2016) | 38.95 | 1.027 | 280.7 M |
| GNMT WPM-32K, ensemble of 8 LSTMs (Wu et al., 2016) | 40.35 | | 2,246.1 M |
| GNMT WPM-32K, HyperLSTM (ours) | 40.03 | 0.993 | 325.5 M |

Table 4: Single model results on WMT En→Fr (newstest2014)

The results are reported in Table 4, which shows that the HyperLSTM cell improves the performance of the existing GNMT model, achieving state-of-the-art single model results for this dataset. In addition, we demonstrate the applicability of hypernetworks to large-scale models used in production systems.

## 5 Conclusion

In this paper, we presented a method to use one network to generate weights for another neural network. Our hypernetworks are trained end-to-end with backpropagation and therefore are efficient and scalable. We focused on applying hypernetworks to generate weights for recurrent networks. On language modelling and handwriting generation, hypernetworks are competitive to or sometimes better than state-of-the-art models. On machine translation, hypernetworks achieve a significant gain on top of a state-of-the-art production-level model.

### Acknowledgments

We thank Jeff Dean, Geoffrey Hinton, Mike Schuster and the Google Brain team for their help with the project.

## References

M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas. Learning to learn by gradient descent by gradient descent. *arXiv preprint arXiv:1606.04474*, 2016.

Jimmy L. Ba, Jamie R. Kiros, and Geoffrey E. Hinton. Layer normalization. *NIPS*, 2016.

Luca Bertinetto, João F. Henriques, Jack Valmadre, Philip H. S. Torr, and Andrea Vedaldi. Learning feed-forward one-shot learners. In *NIPS*, 2016.

Christopher M. Bishop. Mixture density networks. Technical report, 1994.

Junyoung Chung, Caglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. *arXiv preprint arXiv:1502.02367*, 2015.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704*, 2016.

Tim Cooijmans, Nicolas Ballas, Cesar Laurent, and Caglar Gulcehre. Recurrent Batch Normalization. *arXiv:1603.09025*, 2016.

Bert De Brabandere, Xu Jia, Tinne Tuytelaars, and Luc Van Gool. Dynamic filter networks. In *NIPS*, 2016.

Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando de Freitas. Predicting Parameters in Deep Learning. In *NIPS*, 2013.

Chrisantha Fernando, Dylan Banarse, Malcolm Reynolds, Frederic Besse, David Pfau, Max Jaderberg, Marc Lanctot, and Daan Wierstra. Convolution by evolution: Differentiable pattern producing networks. In *GECCO*, 2016.

Faustino Gomez and Jürgen Schmidhuber. Evolving modular fast-weight networks for control. In *ICANN*, 2005.

Mark W Goudreau, C Lee Giles, Srimat T Chakradhar, and D Chen. First-order versus second-order single-layer recurrent neural networks. *IEEE Transactions on Neural Networks*, 1994.

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

Mikael Henaff, Arthur Szlam, and Yann LeCun. Orthogonal RNNs and long-memory tasks. In *ICML*, 2016.

Sepp Hochreiter and Juergen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

Marcus Hutter. The human knowledge compression contest. 2012. URL `http://prize.hutter1.net/`.

Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Decoupled Neural Interfaces using Synthetic Gradients. *arXiv preprint arXiv:1608.05343*, 2016.

Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. Grid long short-term memory. In *ICLR*, 2016.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Jan Koutnik, Faustino Gomez, and Jürgen Schmidhuber. Evolving neural networks in compressed weight space. In *GECCO*, 2010.

B. Krause, L. Lu, I. Murray, and S. Renals. Multiplicative LSTM for sequence modelling. *ArXiv e-prints*, September 2016.

David Krueger, Tegan Maharaj, János Kramár, Mohammad Pezeshki, Nicolas Ballas, Nan Rosemary Ke, Anirudh Goyal, Yoshua Bengio, Hugo Larochelle, Aaron Courville, et al. Zoneout: Regularizing RNNs by randomly preserving hidden activations. *arXiv preprint arXiv:1606.01305*, 2016.

Marcus Liwicki and Horst Bunke. IAM-OnDB - an on-line English sentence database acquired from handwritten text on a whiteboard. In *ICDAR*, 2005.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and Jan Cernocky. Subword language modeling with neural networks. *preprint*, 2012.

Marcin Moczulski, Misha Denil, Jeremy Appleyard, and Nando de Freitas. ACDC: A Structured Efficient Linear Layer. *arXiv preprint arXiv:1511.05946*, 2015.

Saahil Ognawala and Justin Bayer. Regularizing recurrent networks-on injected noise and norm-based methods. *arXiv preprint arXiv:1410.5684*, 2014.

Kamil Rocki. Recurrent memory array structures. *arXiv preprint arXiv:1607.03085*, 2016a.

Kamil Rocki. Surprisal-driven feedback in recurrent networks. *arXiv preprint arXiv:1608.06027*, 2016b.

Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

Jürgen Schmidhuber. A 'self-referential' weight matrix. In *ICANN*, 1993.

Stanislaw Semeniuta, Aliases Severyn, and Erhardt Barth. Recurrent dropout without memory loss. *arXiv:1603.05118*, 2016.

Kenneth O. Stanley, David B. D'Ambrosio, and Jason Gauci. A hypercube-based encoding for evolving large-scale neural networks. *Artificial Life*, 15(2):185–212, 2009.

Ilya Sutskever, James Martens, and Geoffrey E. Hinton. Generating text with recurrent neural networks. In *ICML*, 2011.

Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Ł. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *ArXiv e-prints*, 2016.

Yuhuai Wu, Saizheng Zhang, Ying Zhang, Yoshua Bengio, and Ruslan Salakhutdinov. On multiplicative integration with recurrent neural networks. *NIPS*, 2016.

Z. Yang, M. Moczulski, M. Denil, N. de Freitas, A. Smola, L. Song, and Z. Wang. Deep Fried Convnets. In *ICCV*, 2015.

Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. Deep recurrent models with fast-forward connections for neural machine translation. *CoRR*, abs/1606.04199, 2016. URL `http://arxiv.org/abs/1606.04199`.

Julian Zilly, Rupesh Srivastava, Jan Koutník, and Jürgen Schmidhuber. Recurrent highway networks. *arXiv preprint arXiv:1607.03474*, 2016.

# A APPENDIX

## A.1 EXAMPLES OF GENERATED WIKIPEDIA TEXT

```
The eastern half of Russia varies from Modern to Central Europe. Due to
    similar lighting and the extent of the combination of long
    tributaries to the [[Gulf of Boston]], it is more of a private
    warehouse than the [[Austro-Hungarian Orthodox Christian and Soviet
    Union]].

==Demographic data base==

[[Image:Auschwitz controversial map.png|frame|The ''Austrian Spelling'']]
[[Image:Czech Middle East SSR chief state 103.JPG|thumb|Serbian Russia
    movement]] [[1593]]&amp;ndash;[[1719]], and set up a law of [[
    parliamentary sovereignty]] and unity in Eastern churches.

In medieval Roman Catholicism Tuba and Spanish controlled it until the
    reign of Burgundian kings and resulted in many changes in
    multiculturalism, though the [[Crusades]], usually started following
    the [[Treaty of Portugal]], shored the title of three major powers,
    only a strong part.

[[French Marines]] (prompting a huge change in [[President of the Council
     of the Empire]], only after about [[1793]], the Protestant church,
    fled to the perspective of his heroic declaration of government and,
    in the next fifty years, [[Christianity|Christian]] and [[Jutland]].
    Books combined into a well-published work by a single R. (Sch. M.
    ellipse poem) tradition in St Peter also included 7:1, he dwell upon
    the apostle, scripture and the latter of Luke; totally unknown, a
    distinct class of religious congregations that describes in number of
     [[remor]]an traditions such as the [[Germanic tribes]] (Fridericus
    or Lichteusen and the Wales). Be introduced back to the [[14th
    century]], as related in the [[New Testament]] and in its elegant [[
    Anglo-Saxon Chronicle]], although they branch off the characteristic
    traditions which Saint [[Philip of Macedon]] asserted.

Ae also in his native countries.

In [[1692]], Seymour was barged at poverty of young English children,
    which cost almost the preparation of the marriage to him.

Burke's work was a good step for his writing, which was stopped by clergy
     in the Pacific, where he had both refused and received a position of
     successor to the throne. Like the other councillors in his will, the
     elder Reinhold was not in the Duke, and he was virtually non-father
    of Edward I, in order to recognize [[Henry II of England|Queen Enrie
    ]] of Parliament.

The Melchizedek Minister Qut]] signed the [[Soviet Union]], and forced
    Hoover to provide [[Hoover (disambiguation)|hoover]]s in [[1844]],
    [[1841]].

His work on social linguistic relations is divided to the several times
    of polity for educatinnisley is 760 Li Italians.  After Zaiti's death
    , and he was captured August 3, he witnessed a choice better by
    public, character, repetitious, punt, and future.
```

Figure 5: `enwik8` sample generated from 2048-unit Layer Norm HyperLSTM

```
== Quatitis==
:''Main article: [[sexagesimal]]''

Sexual intimacy was traditionally performed by a male race of the [[
    mitochondria]] of living things. The next geneme is used by ''
    Clitoron'' into short forms of [[sexual reproduction]]. When a
    maternal suffeach-Lashe]] to the myriad of a &amp;quot;master's character
    &amp;quot;. He recognizes the associated reflection of [[force call|
    carriers]], the [[Battle of Pois except fragile house and by
    historians who have at first incorporated his father.

==Geography==
The island and county top of Guernsey consistently has about a third of
    its land, centred on the coast subtained by mountain peels with
    mountains, squares, and lakes that cease to be links with the size
    and depth of sea level and weave in so close to lowlands.
    Strategically to the border of the country also at the southeast
    corner of the province of Denmark do not apply, but sometimes west of
     dense climates of coastal Austria and west Canada, the Flemish area
    of the continent actually inhabits [[tropical geographical transition
    ]] and transitions from [[soil]] to [[snow]] residents.]]

==Definition==
The symbols are ''quotational'' and '''distinct''' or advanced. {{ref|
    no_1}} Older readings are used for [[phrase]]s, especially, [[ancient
     Greek]], and [[Latin]] in their development process. Several
    varieties of permanent systems typically refer to [[primordial
    pleasure]] (for example, [[Pleistocene]], [[Classical antenni|Ctrum
    ]]), but its claim is that it holds the size of the coci, but is
    historically important both for import: brewing and commercial use.

A majority of cuisine specifically refers to this period, where the
    southern countries developed in the 19th century. Scotland had a
    cultural identity of or now a key church who worked between the 8th
    and 60th through 6 (so that there are small single authors of
    detailed recommendations for them and at first) rather than appearing
    , [[Adoptionism|adoptionists]] often started inscribed with the words
     distinct from two types.  On the group definition the adjective ''
    fighting'' is until Crown Violence Association]], in which the higher
     education [[motto]] (despite the resulting attack on [[medical
    treatment]]) peaked on [[15 December]], [[2005]]. At 30 percent, up
    to 50% of the electric music from the period was created by Voltaire,
     but Newton promoted the history of his life.

Publications in the Greek movie ''[[The Great Theory of Bertrand Russell
    ]]'', also kept an important part into the inclusion of ''[[The Beast
     for the Passage of Study]]'', began in [[1869]], opposite the
    existence of racial matters.  Many of Mary's religious faiths (
    including the [[Mary Sue Literature]] in the United States)
    incorporated much of Christianity within Hispanic [[Sacred text]]s.

But controversial belief must be traced back to the 1950s stated that
    their anticolonial forces required the challenge of even lingering
    wars tossing nomon before leaves the bomb in paint on the South
    Island, known as [[Quay]], facing [[Britain]], though he still holds
    to his ancestors a strong ancestor of Orthodoxy. Others explain that
    the process of reverence occurred from [[Common Hermitage]], when the
     [[Crusade|Speakers]] laid his lifespan in [[Islam]] into the north
    of Israel. At the end of the [[14th century BCE]], the citadel of [[
    Israel]] set Eisenace itself in the [[Abyssinia]]n islands, which was
     Faroe's Dominican Republic claimed by the King.
```

Figure 6: `enwik8` sample generated from 2048-unit Layer Norm HyperLSTM

### A.2 EXAMPLES OF RANDOMLY CHOSEN GENERATED HANDWRITING SAMPLES

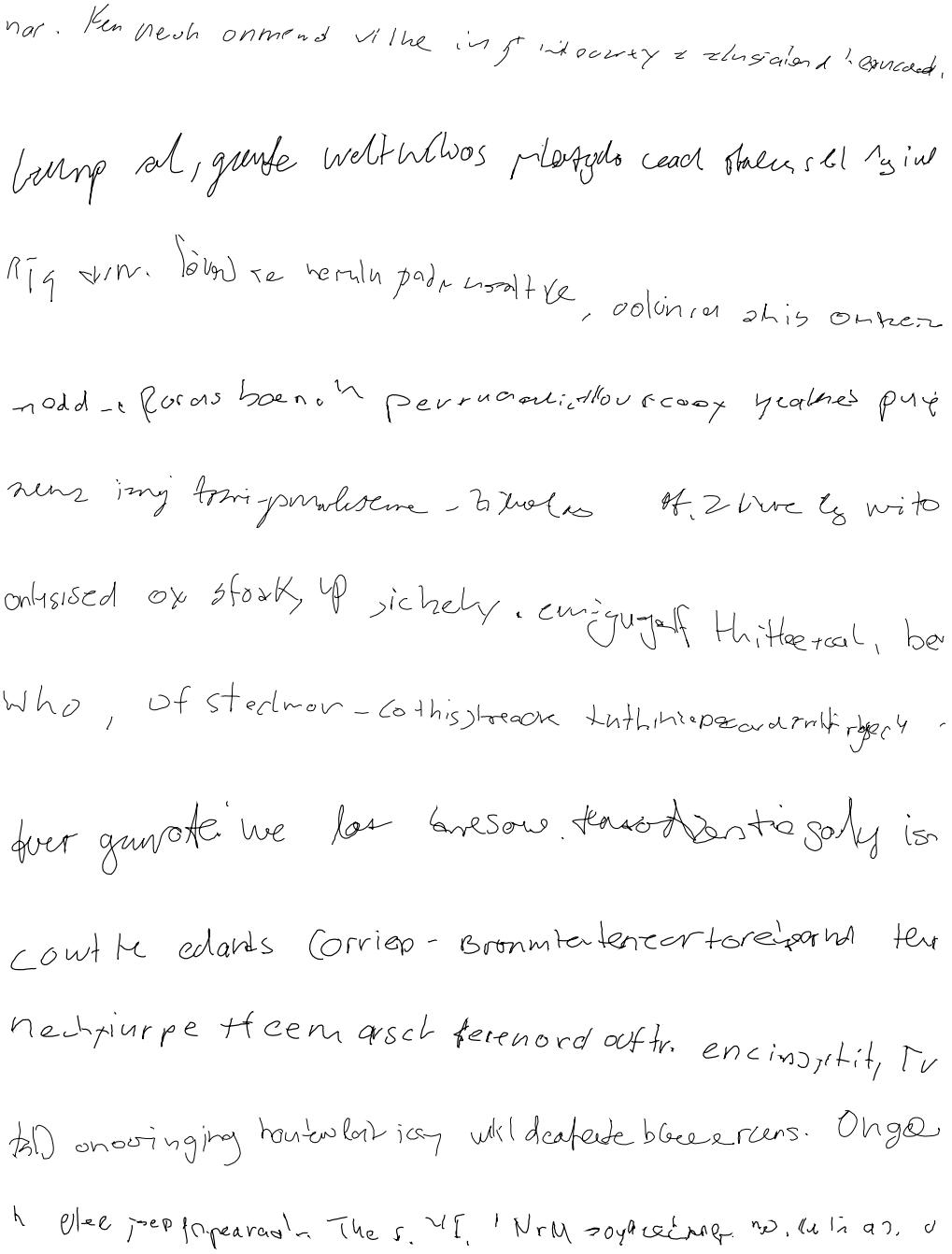

Figure 7: Handwriting samples generated from LSTM

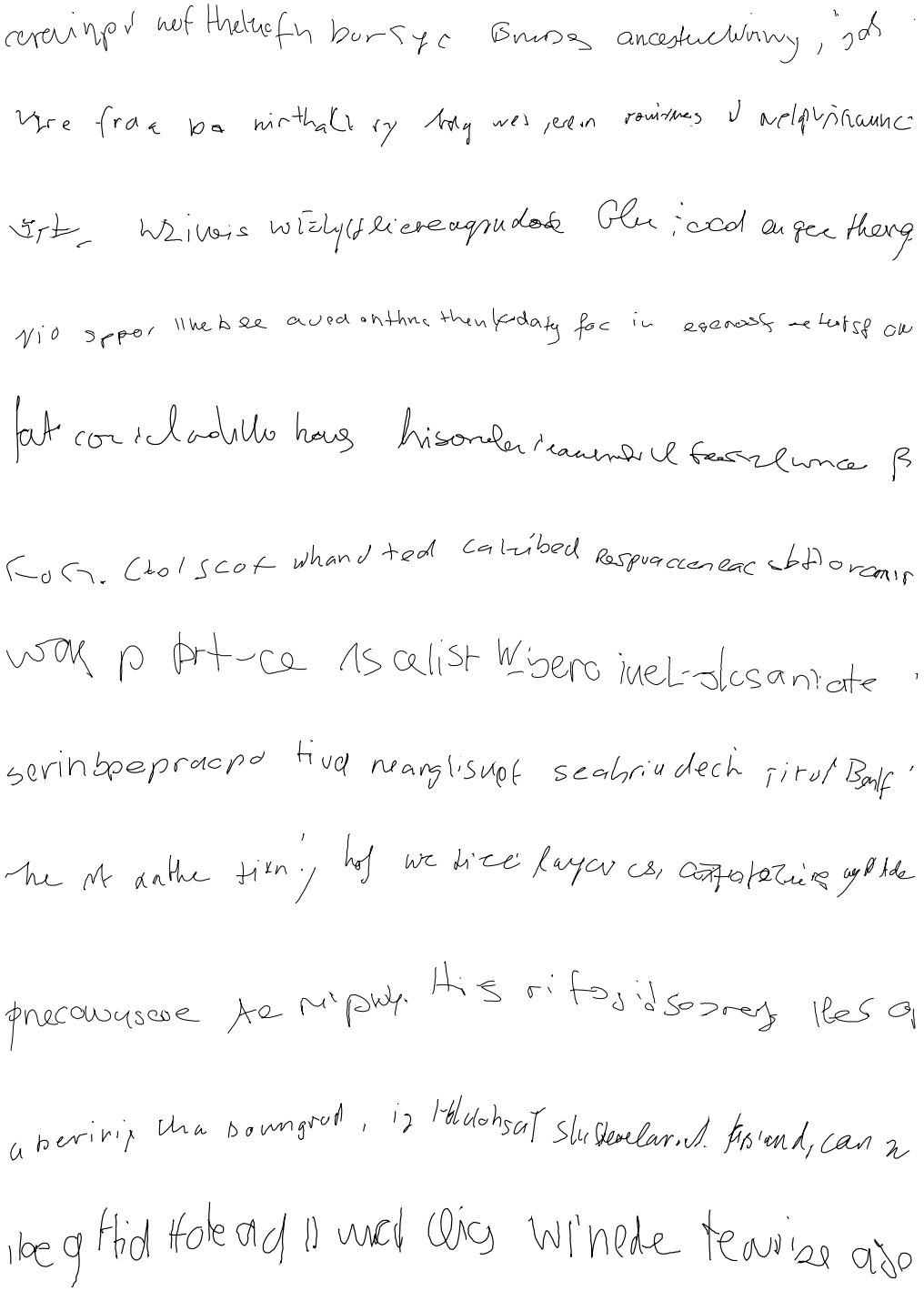

Figure 8: Handwriting samples generated from Layer Norm LSTM

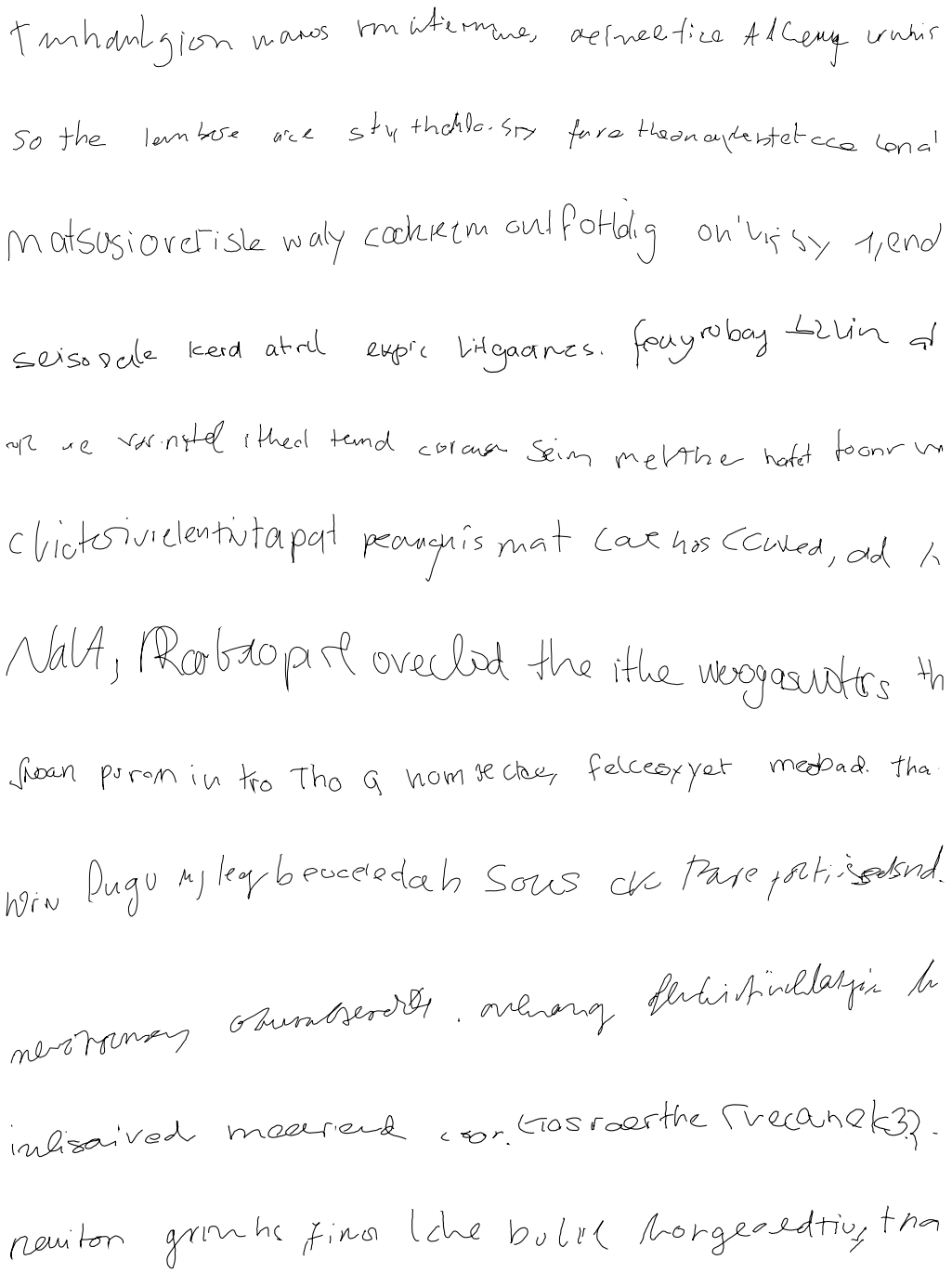

Figure 9: Handwriting samples generated from HyperLSTM

### A.3 EXAMPLES OF RANDOMLY CHOSEN MACHINE TRANSLATION SAMPLES

We randomly selected translation samples generated from both LSTM baseline and HyperLSTM models from the WMT'14 En→Fr Test Set. Given an English phrase, we can compare between the correct French translation, the LSTM translation, and the HyperLSTM translation.

English Input

```
I was expecting to see gnashing of teeth and a fight breaking
out at the gate .
```

French (Ground Truth)

```
Je m' attendais à voir des grincements de dents et une
bagarre éclater à la porte .
```

LSTM Translation

```
Je m' attendais à voir des larmes de dents et un combat à la
porte .
```

HyperLSTM Translation

```
Je m' attendais à voir des dents grincer des dents et une
bataille éclater à la porte .
```

English Input

```
Prosecuting , Anne Whyte said : " If anyone should know not
to the break the law , it is a criminal solicitor . "
```

French (Ground Truth)

```
Le procureur Anne Whyte a déclaré : « Si quelqu' un doit
savoir qu' il ne faut pas violer la loi , c' est bien un
avocat pénaliste . »
```

LSTM Translation

```
Prosecuting , Anne Whyte a dit : « Si quelqu' un doit savoir
qu' il ne faut pas enfreindre la loi , c' est un solicitor
criminel .
```

HyperLSTM Translation

```
En poursuivant , Anne Whyte a dit : « Si quelqu' un doit
savoir ne pas enfreindre la loi , c' est un avocat criminel .
```

English Input

```
According to her , the CSRS was invited to a mediation and she
asked for an additional period for consideration .
```

French (Ground Truth)

```
Selon elle , la CSRS a été invitée à une médiation et elle a
demandé un délai supplémentaire pour y réfléchir .
```

LSTM Translation

```
Selon elle , le SCRS a été invité à une médiation et elle a demandé
un délai supplémentaire .
```

HyperLSTM Translation

```
Selon elle , le SCRS a été invité à une médiation et elle a demandé
une période de réflexion supplémentaire .
```

English Input

```
Relations between the US and Germany have come under strain
following claims that the NSA bugged Chancellor Angela 's
Merkel 's phone .
```

French (Ground Truth)

```
Les relations entre les États-Unis et l' Allemagne ont été
mises à rude épreuve à la suite de plaintes selon lesquelles
la NSA avait mis sur écoute le téléphone portable de la
chancelière allemande Angela Merkel .
```

LSTM Translation

```
Les relations entre les Etats-Unis et l' Allemagne ont été
mises à rude épreuve suite aux affirmations selon lesquelles
la NSA aurait pris le téléphone de Merkel de la chancelière
Angela .
```

HyperLSTM Translation

```
Les relations entre les États-Unis et l' Allemagne ont été
mises à rude épreuve après que la NSA a attaqué le téléphone
de la chancelière Angela Angela .
```

English Input

```
Germany 's BfV advises executives to consider using simple prepaid
mobiles when on foreign trips because of the risk that smart phones
are compromised .
```

French (Ground Truth)

```
Le BfV d' Allemagne conseille à ses dirigeants d' envisager
d' utiliser de simples téléphones portables prépayés lors de leurs
voyages à l' étranger en raison du risque d' atteinte à
l' intégrité des smartphones .
```

LSTM Translation

```
Le BfV allemand conseille aux dirigeants d' envisager
l' utilisation de mobiles prépayés simples lors de voyages à
l' étranger en raison du risque de compromission des téléphones
intelligents .
```

HyperLSTM Translation

```
Le BfV allemand conseille aux dirigeants d' envisager
l' utilisation de téléphones mobiles prépayés simples lors de
voyages à l' étranger en raison du risque que les téléphones
intelligents soient compromis .
```

English Input

```
I was on the mid-evening news that same evening , and on TV the
following day as well .
```

French (Ground Truth)

```
Le soir-même , je suis au 20h , le lendemain aussi je suis à la
télé .
```

LSTM Translation

```
J' étais au milieu de l' actualité le soir même , et à la
télévision le lendemain également .
```

HyperLSTM Translation

```
J' étais au milieu de la soirée ce soir-là et à la télévision le
lendemain .
```