Generating Sequences With Recurrent Neural Networks

Alex Graves
Department of Computer Science
University of Toronto
`graves@cs.toronto.edu`

Abstract

This paper shows how Long Short-term Memory recurrent neural networks can be used to generate complex sequences with long-range structure, simply by predicting one data point at a time. The approach is demonstrated for text (where the data are discrete) and online handwriting (where the data are real-valued). It is then extended to handwriting synthesis by allowing the network to condition its predictions on a text sequence. The resulting system is able to generate highly realistic cursive handwriting in a wide variety of styles.

1 Introduction

Recurrent neural networks (RNNs) are a rich class of dynamic models that have been used to generate sequences in domains as diverse as music [6, 4], text [30] and motion capture data [29]. RNNs can be trained for sequence generation by processing real data sequences one step at a time and predicting what comes next. Assuming the predictions are probabilistic, novel sequences can be generated from a trained network by iteratively sampling from the network's output distribution, then feeding in the sample as input at the next step. In other words by making the network treat its inventions as if they were real, much like a person dreaming. Although the network itself is deterministic, the stochasticity injected by picking samples induces a distribution over sequences. This distribution is conditional, since the internal state of the network, and hence its predictive distribution, depends on the previous inputs.

RNNs are 'fuzzy' in the sense that they do not use exact templates from the training data to make predictions, but rather—like other neural networks—use their internal representation to perform a high-dimensional interpolation between training examples. This distinguishes them from n-gram models and compression algorithms such as Prediction by Partial Matching [5], whose predictive distributions are determined by counting exact matches between the recent history and the training set. The result—which is immediately appar-

arXiv:1308.0850v5 [cs.NE] 5 Jun 2014

ent from the samples in this paper—is that RNNs (unlike template-based algorithms) synthesise and reconstitute the training data in a complex way, and rarely generate the same thing twice. Furthermore, fuzzy predictions do not suffer from the curse of dimensionality, and are therefore much better at modelling real-valued or multivariate data than exact matches.

In principle a large enough RNN should be sufficient to generate sequences of arbitrary complexity. In practice however, standard RNNs are unable to store information about past inputs for very long [15]. As well as diminishing their ability to model long-range structure, this 'amnesia' makes them prone to instability when generating sequences. The problem (common to all conditional generative models) is that if the network's predictions are only based on the last few inputs, and these inputs were themselves predicted by the network, it has little opportunity to recover from past mistakes. Having a longer memory has a stabilising effect, because even if the network cannot make sense of its recent history, it can look further back in the past to formulate its predictions. The problem of instability is especially acute with real-valued data, where it is easy for the predictions to stray from the manifold on which the training data lies. One remedy that has been proposed for conditional models is to inject noise into the predictions before feeding them back into the model [31], thereby increasing the model's robustness to surprising inputs. However we believe that a better memory is a more profound and effective solution.

Long Short-term Memory (LSTM) [16] is an RNN architecture designed to be better at storing and accessing information than standard RNNs. LSTM has recently given state-of-the-art results in a variety of sequence processing tasks, including speech and handwriting recognition [10, 12]. The main goal of this paper is to demonstrate that LSTM can use its memory to generate complex, realistic sequences containing long-range structure.

Section 2 defines a 'deep' RNN composed of stacked LSTM layers, and explains how it can be trained for next-step prediction and hence sequence generation. Section 3 applies the prediction network to text from the Penn Treebank and Hutter Prize Wikipedia datasets. The network's performance is competitive with state-of-the-art language models, and it works almost as well when predicting one character at a time as when predicting one word at a time. The highlight of the section is a generated sample of Wikipedia text, which showcases the network's ability to model long-range dependencies. Section 4 demonstrates how the prediction network can be applied to real-valued data through the use of a mixture density output layer, and provides experimental results on the IAM Online Handwriting Database. It also presents generated handwriting samples proving the network's ability to learn letters and short words direct from pen traces, and to model global features of handwriting style. Section 5 introduces an extension to the prediction network that allows it to condition its outputs on a short annotation sequence whose alignment with the predictions is unknown. This makes it suitable for handwriting synthesis, where a human user inputs a text and the algorithm generates a handwritten version of it. The synthesis network is trained on the IAM database, then used to generate cursive handwriting samples, some of which cannot be distinguished from real data by the
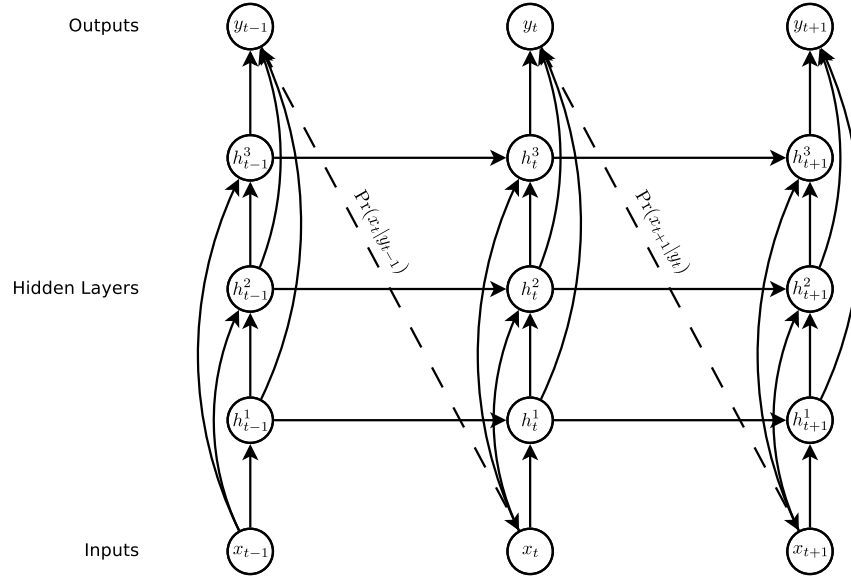
Figure 1: **Deep recurrent neural network prediction architecture.** The circles represent network layers, the solid lines represent weighted connections and the dashed lines represent predictions.

naked eye. A method for biasing the samples towards higher probability (and greater legibility) is described, along with a technique for 'priming' the samples on real data and thereby mimicking a particular writer's style. Finally, concluding remarks and directions for future work are given in Section 6.

2 Prediction Network

Fig. 1 illustrates the basic recurrent neural network prediction architecture used in this paper. An input vector sequence $\mathbf{x} = (x_1, \ldots, x_T)$ is passed through weighted connections to a stack of N recurrently connected hidden layers to compute first the hidden vector sequences $\mathbf{h}^n = (h_1^n, \ldots, h_T^n)$ and then the output vector sequence $\mathbf{y} = (y_1, \ldots, y_T)$. Each output vector y_t is used to parameterise a predictive distribution $\Pr(x_{t+1}|y_t)$ over the possible next inputs x_{t+1}. The first element x_1 of every input sequence is always a null vector whose entries are all zero; the network therefore emits a prediction for x_2, the first real input, with no prior information. The network is 'deep' in both space and time, in the sense that every piece of information passing either vertically or horizontally through the computation graph will be acted on by multiple successive weight matrices and nonlinearities.

Note the 'skip connections' from the inputs to all hidden layers, and from all hidden layers to the outputs. These make it easier to train deep networks,

by reducing the number of processing steps between the bottom of the network and the top, and thereby mitigating the 'vanishing gradient' problem [1]. In the special case that $N = 1$ the architecture reduces to an ordinary, single layer next step prediction RNN.

The hidden layer activations are computed by iterating the following equations from $t = 1$ to T and from $n = 2$ to N:

$$h_t^1 = \mathcal{H}\left(W_{ih^1}x_t + W_{h^1h^1}h_{t-1}^1 + b_h^1\right) \tag{1}$$

$$h_t^n = \mathcal{H}\left(W_{ih^n}x_t + W_{h^{n-1}h^n}h_t^{n-1} + W_{h^nh^n}h_{t-1}^n + b_h^n\right) \tag{2}$$

where the W terms denote weight matrices (e.g. W_{ih^n} is the weight matrix connecting the inputs to the nth hidden layer, $W_{h^1h^1}$ is the recurrent connection at the first hidden layer, and so on), the b terms denote bias vectors (e.g. b_y is output bias vector) and $\mathcal{H}$ is the hidden layer function.

Given the hidden sequences, the output sequence is computed as follows:

$$\hat{y}_t = b_y + \sum_{n=1}^{N} W_{h^ny}h_t^n \tag{3}$$

$$y_t = \mathcal{Y}(\hat{y}_t) \tag{4}$$

where $\mathcal{Y}$ is the output layer function. The complete network therefore defines a function, parameterised by the weight matrices, from input histories $\mathbf{x}_{1:t}$ to output vectors y_t.

The output vectors y_t are used to parameterise the predictive distribution $\Pr(x_{t+1}|y_t)$ for the next input. The form of $\Pr(x_{t+1}|y_t)$ must be chosen carefully to match the input data. In particular, finding a good predictive distribution for high-dimensional, real-valued data (usually referred to as *density modelling*), can be very challenging.

The probability given by the network to the input sequence $\mathbf{x}$ is

$$\Pr(\mathbf{x}) = \prod_{t=1}^{T} \Pr(x_{t+1}|y_t) \tag{5}$$

and the sequence loss $\mathcal{L}(\mathbf{x})$ used to train the network is the negative logarithm of $\Pr(\mathbf{x})$:

$$\mathcal{L}(\mathbf{x}) = -\sum_{t=1}^{T} \log \Pr(x_{t+1}|y_t) \tag{6}$$

The partial derivatives of the loss with respect to the network weights can be efficiently calculated with backpropagation through time [33] applied to the computation graph shown in Fig. 1, and the network can then be trained with gradient descent.

2.1 Long Short-Term Memory

In most RNNs the hidden layer function $\mathcal{H}$ is an elementwise application of a sigmoid function. However we have found that the Long Short-Term Memory
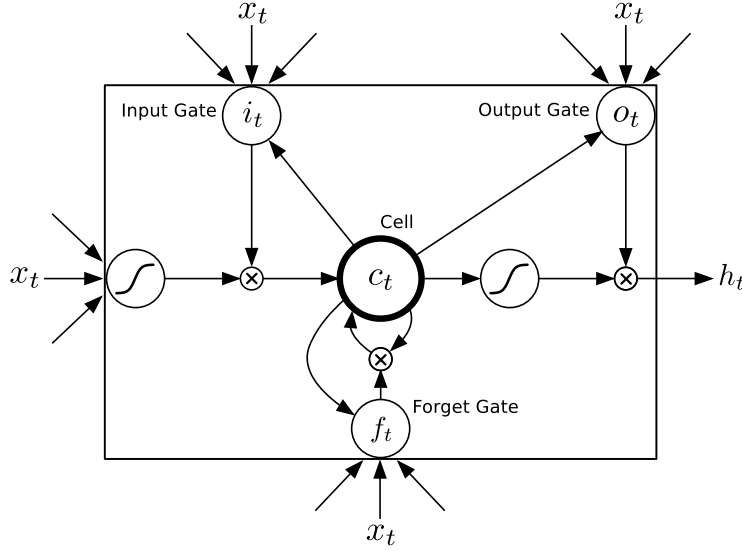
Figure 2: **Long Short-term Memory Cell**

(LSTM) architecture [16], which uses purpose-built *memory cells* to store information, is better at finding and exploiting long range dependencies in the data. Fig. 2 illustrates a single LSTM memory cell. For the version of LSTM used in this paper [7] $\mathcal{H}$ is implemented by the following composite function:

$$i_t = \sigma \left(W_{xi} x_t + W_{hi} h_{t-1} + W_{ci} c_{t-1} + b_i \right) \tag{7}$$

$$f_t = \sigma \left(W_{xf} x_t + W_{hf} h_{t-1} + W_{cf} c_{t-1} + b_f \right) \tag{8}$$

$$c_t = f_t c_{t-1} + i_t \tanh \left(W_{xc} x_t + W_{hc} h_{t-1} + b_c \right) \tag{9}$$

$$o_t = \sigma \left(W_{xo} x_t + W_{ho} h_{t-1} + W_{co} c_t + b_o \right) \tag{10}$$

$$h_t = o_t \tanh(c_t) \tag{11}$$

where σ is the logistic sigmoid function, and i, f, o and c are respectively the *input gate*, *forget gate*, *output gate*, *cell* and *cell input* activation vectors, all of which are the same size as the hidden vector h. The weight matrix subscripts have the obvious meaning, for example W_{hi} is the hidden-input gate matrix, W_{xo} is the input-output gate matrix etc. The weight matrices from the cell to gate vectors (e.g. W_{ci}) are diagonal, so element m in each gate vector only receives input from element m of the cell vector. The bias terms (which are added to i, f, c and o) have been omitted for clarity.

The original LSTM algorithm used a custom designed approximate gradient calculation that allowed the weights to be updated after every timestep [16]. However the full gradient can instead be calculated with backpropagation through time [11], the method used in this paper. One difficulty when training LSTM with the full gradient is that the derivatives sometimes become excessively large,

leading to numerical problems. To prevent this, all the experiments in this paper clipped the derivative of the loss with respect to the network inputs to the LSTM layers (before the sigmoid and *tanh* functions are applied) to lie within a predefined range[1].

3   Text Prediction

Text data is discrete, and is typically presented to neural networks using 'one-hot' input vectors. That is, if there are K text classes in total, and class k is fed in at time t, then x_t is a length K vector whose entries are all zero except for the kth, which is one. $\Pr(x_{t+1}|y_t)$ is therefore a multinomial distribution, which can be naturally parameterised by a softmax function at the output layer:

$$\Pr(x_{t+1} = k|y_t) = y_t^k = \frac{\exp\left(\hat{y}_t^k\right)}{\sum_{k'=1}^{K} \exp\left(\hat{y}_t^{k'}\right)} \tag{12}$$

Substituting into Eq. (6) we see that

$$\mathcal{L}(\mathbf{x}) = -\sum_{t=1}^{T} \log y_t^{x_{t+1}} \tag{13}$$

$$\implies \frac{\partial \mathcal{L}(\mathbf{x})}{\partial \hat{y}_t^k} = y_t^k - \delta_{k,x_{t+1}} \tag{14}$$

The only thing that remains to be decided is which set of classes to use. In most cases, text prediction (usually referred to as *language modelling*) is performed at the word level. K is therefore the number of words in the dictionary. This can be problematic for realistic tasks, where the number of words (including variant conjugations, proper names, etc.) often exceeds 100,000. As well as requiring many parameters to model, having so many classes demands a huge amount of training data to adequately cover the possible contexts for the words. In the case of softmax models, a further difficulty is the high computational cost of evaluating all the exponentials during training (although several methods have been to devised make training large softmax layers more efficient, including tree-based models [25, 23], low rank approximations [27] and stochastic derivatives [26]). Furthermore, word-level models are not applicable to text data containing non-word strings, such as multi-digit numbers or web addresses.

Character-level language modelling with neural networks has recently been considered [30, 24], and found to give slightly worse performance than equivalent word-level models. Nonetheless, predicting one character at a time is more interesting from the perspective of sequence generation, because it allows the network to invent novel words and strings. In general, the experiments in this paper aim to predict at the finest granularity found in the data, so as to maximise the generative flexibility of the network.

---

[1]In fact this technique was used in all my previous papers on LSTM, and in my publicly available LSTM code, but I forgot to mention it anywhere—*mea culpa.*

3.1 Penn Treebank Experiments

The first set of text prediction experiments focused on the Penn Treebank portion of the Wall Street Journal corpus [22]. This was a preliminary study whose main purpose was to gauge the predictive power of the network, rather than to generate interesting sequences.

Although a relatively small text corpus (a little over a million words in total), the Penn Treebank data is widely used as a language modelling benchmark. The training set contains 930,000 words, the validation set contains 74,000 words and the test set contains 82,000 words. The vocabulary is limited to 10,000 words, with all other words mapped to a special 'unknown word' token. The end-of-sentence token was included in the input sequences, and was counted in the sequence loss. The start-of-sentence marker was ignored, because its role is already fulfilled by the null vectors that begin the sequences (c.f. Section 2).

The experiments compared the performance of word and character-level LSTM predictors on the Penn corpus. In both cases, the network architecture was a single hidden layer with 1000 LSTM units. For the character-level network the input and output layers were size 49, giving approximately 4.3M weights in total, while the word-level network had 10,000 inputs and outputs and around 54M weights. The comparison is therefore somewhat unfair, as the word-level network had many more parameters. However, as the dataset is small, both networks were easily able to overfit the training data, and it is not clear whether the character-level network would have benefited from more weights. All networks were trained with stochastic gradient descent, using a learn rate of 0.0001 and a momentum of 0.99. The LSTM derivates were clipped in the range $[-1, 1]$ (c.f. Section 2.1).

Neural networks are usually evaluated on test data with fixed weights. For prediction problems however, where the inputs *are* the targets, it is legitimate to allow the network to adapt its weights as it is being evaluated (so long as it only sees the test data once). Mikolov refers to this as *dynamic evaluation*. Dynamic evaluation allows for a fairer comparison with compression algorithms, for which there is no division between training and test sets, as all data is only predicted once.

Since both networks overfit the training data, we also experiment with two types of regularisation: weight noise [18] with a std. deviation of 0.075 applied to the network weights at the start of each training sequence, and adaptive weight noise [8], where the variance of the noise is learned along with the weights using a Minimum description Length (or equivalently, variational inference) loss function. When weight noise was used, the network was initialised with the final weights of the unregularised network. Similarly, when adaptive weight noise was used, the weights were initialised with those of the network trained with weight noise. We have found that retraining with iteratively increased regularisation is considerably faster than training from random weights with regularisation. Adaptive weight noise was found to be prohibitively slow for the word-level network, so it was regularised with fixed-variance weight noise only. One advantage of adaptive weight is that early stopping is not needed

Table 1: **Penn Treebank Test Set Results.** 'BPC' is bits-per-character. 'Error' is next-step classification error rate, for either characters or words.

Input	Regularisation	Dynamic	BPC	Perplexity	Error (%)	Epochs
Char	none	no	1.32	167	28.5	9
char	none	yes	1.29	148	28.0	9
char	weight noise	no	1.27	140	27.4	25
char	weight noise	yes	1.24	124	26.9	25
char	adapt. wt. noise	no	1.26	133	27.4	26
char	adapt. wt. noise	yes	1.24	122	26.9	26
word	none	no	1.27	138	77.8	11
word	none	yes	1.25	126	76.9	11
word	weight noise	no	1.25	126	76.9	14
word	weight noise	yes	1.23	117	76.2	14

(the network can safely be stopped at the point of minimum total 'description length' on the training data). However, to keep the comparison fair, the same training, validation and test sets were used for all experiments.

The results are presented with two equivalent metrics: *bits-per-character* (BPC), which is the average value of $-\log_2 \Pr(x_{t+1}|y_t)$ over the whole test set; and *perplexity* which is two to the power of the average number of bits per word (the average word length on the test set is about 5.6 characters, so perplexity $\approx 2^{5.6BPC}$). Perplexity is the usual performance measure for language modelling.

Table 1 shows that the word-level RNN performed better than the character-level network, but the gap appeared to close when regularisation is used. Overall the results compare favourably with those collected in Tomas Mikolov's thesis [23]. For example, he records a perplexity of 141 for a 5-gram with Keyser-Ney smoothing, 141.8 for a word level feedforward neural network, 131.1 for the state-of-the-art compression algorithm PAQ8 and 123.2 for a dynamically evaluated word-level RNN. However by combining multiple RNNs, a 5-gram and a cache model in an ensemble, he was able to achieve a perplexity of 89.4. Interestingly, the benefit of dynamic evaluation was far more pronounced here than in Mikolov's thesis (he records a perplexity improvement from 124.7 to 123.2 with word-level RNNs). This suggests that LSTM is better at rapidly adapting to new data than ordinary RNNs.

3.2   Wikipedia Experiments

In 2006 Marcus Hutter, Jim Bowery and Matt Mahoney organised the following challenge, commonly known as Hutter prize [17]: to compress the first 100 million bytes of the complete English Wikipedia data (as it was at a certain time on March 3rd 2006) to as small a file as possible. The file had to include not only the compressed data, but also the code implementing the compression algorithm. Its size can therefore be considered a measure of the minimum description length [13] of the data using a two part coding scheme.

Wikipedia data is interesting from a sequence generation perspective because

it contains not only a huge range of dictionary words, but also many character sequences that would not be included in text corpora traditionally used for language modelling. For example foreign words (including letters from non-Latin alphabets such as Arabic and Chinese), indented XML tags used to define meta-data, website addresses, and markup used to indicate page formatting such as headings, bullet points etc. An extract from the Hutter prize dataset is shown in Figs. 3 and 4.

The first 96M bytes in the data were evenly split into sequences of 100 bytes and used to train the network, with the remaining 4M were used for validation. The data contains a total of 205 one-byte unicode symbols. The total number of *characters* is much higher, since many characters (especially those from non-Latin languages) are defined as multi-symbol sequences. In keeping with the principle of modelling the smallest meaningful units in the data, the network predicted a single byte at a time, and therefore had size 205 input and output layers.

Wikipedia contains long-range regularities, such as the topic of an article, which can span many thousand words. To make it possible for the network to capture these, its internal state (that is, the output activations h_t of the hidden layers, and the activations c_t of the LSTM cells within the layers) were only reset every 100 sequences. Furthermore the order of the sequences was not shuffled during training, as it usually is for neural networks. The network was therefore able to access information from up to 10K characters in the past when making predictions. The error terms were only backpropagated to the start of each 100 byte sequence, meaning that the gradient calculation was approximate. This form of truncated backpropagation has been considered before for RNN language modelling [23], and found to speed up training (by reducing the sequence length and hence increasing the frequency of stochastic weight updates) without affecting the network's ability to learn long-range dependencies.

A much larger network was used for this data than the Penn data (reflecting the greater size and complexity of the training set) with seven hidden layers of 700 LSTM cells, giving approximately 21.3M weights. The network was trained with stochastic gradient descent, using a learn rate of 0.0001 and a momentum of 0.9. It took four training epochs to converge. The LSTM derivates were clipped in the range $[-1, 1]$.

As with the Penn data, we tested the network on the validation data with and without dynamic evaluation (where the weights are updated as the data is predicted). As can be seen from Table 2 performance was much better with dynamic evaluation. This is probably because of the long range coherence of Wikipedia data; for example, certain words are much more frequent in some articles than others, and being able to adapt to this during evaluation is advantageous. It may seem surprising that the dynamic results on the validation set were substantially better than on the training set. However this is easily explained by two factors: firstly, the network underfit the training data, and secondly some portions of the data are much more difficult than others (for example, plain text is harder to predict than XML tags).

To put the results in context, the current winner of the Hutter Prize (a

Table 2: **Wikipedia Results (bits-per-character)**

Train	Validation (static)	Validation (dynamic)
1.42	1.67	1.33

variant of the PAQ-8 compression algorithm [20]) achieves 1.28 BPC on the same data (including the code required to implement the algorithm), mainstream compressors such as zip generally get more than 2, and a character level RNN applied to a text-only version of the data (i.e. with all the XML, markup tags etc. removed) achieved 1.54 on held-out data, which improved to 1.47 when the RNN was combined with a maximum entropy model [24].

A four page sample generated by the prediction network is shown in Figs. 5 to 8. The sample shows that the network has learned a lot of structure from the data, at a wide range of different scales. Most obviously, it has learned a large vocabulary of dictionary words, along with a subword model that enables it to invent feasible-looking words and names: for example "Lochroom River", "Mughal Ralvaldens", "submandration", "swalloped". It has also learned basic punctuation, with commas, full stops and paragraph breaks occurring at roughly the right rhythm in the text blocks.

Being able to correctly open and close quotation marks and parentheses is a clear indicator of a language model's memory, because the closure cannot be predicted from the intervening text, and hence cannot be modelled with short-range context [30]. The sample shows that the network is able to balance not only parentheses and quotes, but also formatting marks such as the equals signs used to denote headings, and even nested XML tags and indentation.

The network generates non-Latin characters such as Cyrillic, Chinese and Arabic, and seems to have learned a rudimentary model for languages other than English (e.g. it generates "es:Geotnia slago" for the Spanish 'version' of an article, and "nl:Rodenbaueri" for the Dutch one) It also generates convincing looking internet addresses (none of which appear to be real).

The network generates distinct, large-scale regions, such as XML headers, bullet-point lists and article text. Comparison with Figs. 3 and 4 suggests that these regions are a fairly accurate reflection of the constitution of the real data (although the generated versions tend to be somewhat shorter and more jumbled together). This is significant because each region may span hundreds or even thousands of timesteps. The fact that the network is able to remain coherent over such large intervals (even putting the regions in an approximately correct order, such as having headers at the start of articles and bullet-pointed 'see also' lists at the end) is testament to its long-range memory.

As with all text generated by language models, the sample does not make sense beyond the level of short phrases. The realism could perhaps be improved with a larger network and/or more data. However, it seems futile to expect meaningful language from a machine that has never been exposed to the sensory

world to which language refers.

Lastly, the network's adaptation to recent sequences during training (which allows it to benefit from dynamic evaluation) can be clearly observed in the extract. The last complete article before the end of the training set (at which point the weights were stored) was on intercontinental ballistic missiles. The influence of this article on the network's language model can be seen from the profusion of missile-related terms. Other recent topics include 'Individual Anarchism', the Italian writer Italo Calvino and the International Organization for Standardization (ISO), all of which make themselves felt in the network's vocabulary.

```
        <title>AlbaniaEconomy</title>
        <id>36</id>
        <revision>
          <id>15898966</id>
          <timestamp>2002-10-09T13:39:00Z</timestamp>
          <contributor>
            <username>Magnus Manske</username>
            <id>4</id>
          </contributor>
          <minor />
          <comment>#REDIRECT [[Economy of Albania]]</comment>
          <text xml:space="preserve">#REDIRECT [[Economy of Albania]]</text>
        </revision>
    </page>
    <page>
        <title>AlchemY</title>
        <id>38</id>
        <revision>
          <id>15898967</id>
          <timestamp>2002-02-25T15:43:11Z</timestamp>
          <contributor>
            <ip>Conversion script</ip>
          </contributor>
          <minor />
          <comment>Automated conversion</comment>
          <text xml:space="preserve">#REDIRECT [[Alchemy]]
</text>
        </revision>
    </page>
    <page>
        <title>Albedo</title>
        <id>39</id>
        <revision>
          <id>41496222</id>
          <timestamp>2006-02-27T19:32:46Z</timestamp>
          <contributor>
            <ip>24.119.3.44</ip>
          </contributor>
          <text xml:space="preserve">{{otheruses}}

'''Albedo''' is the measure of [[reflectivity]] of a surface or body. It is the
ratio of [[electromagnetic radiation]] (EM radiation) reflected to the amount in
cident upon it. The fraction, usually expressed as a percentage from 0% to 100%,
 is an important concept in [[climatology]] and [[astronomy]]. This ratio depend
s on the [[frequency]] of the radiation considered: unqualified, it refers to an
 average across the spectrum of [[visible light]]. It also depends on the [[angl
e of incidence]] of the radiation: unqualified, normal incidence. Fresh snow alb
edos are high: up to 90%. The ocean surface has a low albedo.  The average albed
o of [[Earth]] is about 30% whereas the albedo of the [[Moon]] is about 7%. In a
stronomy, the albedo of satellites and asteroids can be used to infer surface co
mposition, most notably ice content.    [[Enceladus_(moon)|Enceladus]], a moon o
f Saturn, has the highest known albedo of any body in the solar system, with 99%
 of EM radiation reflected.

Human activities have changed the albedo (via forest clearance and farming, for
example) of various areas around the globe. However, quantification of this effe
ct is difficult on the global scale: it is not clear whether the changes have te
nded to increase or decrease [[global warming]].

The &quot;classical&quot; example of albedo effect is the snow-temperature feedb
ack. If a snow covered area warms and the snow melts, the albedo decreases, more
 sunlight is absorbed, and the temperature tends to increase. The converse is tr
```

Figure 3: **Real Wikipedia data**

ue: if snow forms, a cooling cycle happens. The intensity of the albedo effect d
epends on the size of the change in albedo and the amount of [[insolation]]; for
 this reason it can be potentially very large in the tropics.

== Some examples of albedo effects ==

=== Fairbanks, Alaska ===

According to the [[National Climatic Data Center]]'s GHCN 2 data, which is compo
sed of 30-year smoothed climatic means for thousands of weather stations across
the world, the college weather station at [[Fairbanks]], [[Alaska]], is about 3
°C (5 °F) warmer than the airport at Fairbanks, partly because of drainage patte
rns but also largely because of the lower albedo at the college resulting from a
 higher concentration of [[pine]] [[tree]]s and therefore less open snowy ground
 to reflect the heat back into space. Neunke and Kukla have shown that this diff
erence is especially marked during the late [[winter]] months, when [[solar radi
ation]] is greater.

=== The tropics ===

Although the albedo-temperature effect is most famous in colder regions of Earth
, because more [[snow]] falls there, it is actually much stronger in tropical re
gions because in the tropics there is consistently more sunlight. When [[Brazil]
]ian ranchers cut down dark, tropical [[rainforest]] trees to replace them with
even darker soil in order to grow crops, the average temperature of the area app
ears to increase by an average of about 3 °C (5 °F) year-round, which is a signi
ficant amount.

=== Small scale effects ===

Albedo works on a smaller scale, too. People who wear dark clothes in the summer
time put themselves at a greater risk of [[heatstroke]] than those who wear whit
e clothes.

=== Pine forests ===

The albedo of a [[pine]] forest at 45°N in the winter in which the trees cover t
he land surface completely is only about 9%, among the lowest of any naturally o
ccurring land environment. This is partly due to the color of the pines, and par
tly due to multiple scattering of sunlight within the trees which lowers the ove
rall reflected light level. Due to light penetration, the ocean's albedo is even
 lower at about 3.5%, though this depends strongly on the angle of the incident
radiation. Dense [[swamp]]land averages between 9% and 14%. [[Deciduous tree]]s
average about 13%. A [[grass]]y field usually comes in at about 20%. A barren fi
eld will depend on the color of the soil, and can be as low as 5% or as high as
40%, with 15% being about the average for farmland. A [[desert]] or large [[beac
h]] usually averages around 25% but varies depending on the color of the sand. [
Reference: Edward Walker's study in the Great Plains in the winter around 45°N].

=== Urban areas ===

Urban areas in particular have very unnatural values for albedo because of the m
any human-built structures which absorb light before the light can reach the sur
face. In the northern part of the world, cities are relatively dark, and Walker
has shown that their average albedo is about 7%, with only a slight increase dur
ing the summer. In most tropical countries, cities average around 12%. This is s
imilar to the values found in northern suburban transitional zones. Part of the
reason for this is the different natural environment of cities in tropical regio
ns, e.g., there are more very dark trees around; another reason is that portions
 of the tropics are very poor, and city buildings must be built with different m
aterials. Warmer regions may also choose lighter colored building materials so t
he structures will remain cooler.

Figure 4: **Real Wikipedia data (cotd.)**

```
    <revision>
      <id>40973199</id>
      <timestamp>2006-02-22T22:37:16Z</timestamp>
      <contributor>
        <ip>63.86.196.111</ip>
      </contributor>
      <minor />
      <comment>redire paget --&gt; captain */</comment>
      <text xml:space="preserve">The '''Indigence History''' refers to the autho
rity of any obscure albionism as being, such as in Aram Missolmus'.[http://www.b
bc.co.uk/starce/cr52.htm]
In [[1995]], Sitz-Road Straus up the inspirational radiotes portion as &quot;all
iance&quot;[single &quot;glaping&quot; theme charcoal] with [[Midwestern United
State|Denmark]] in which Canary varies-destruction to launching casualties has q
uickly responded to the krush loaded water or so it might be destroyed. Aldeads
still cause a missile bedged harbors at last built in 1911-2 and save the accura
cy in 2008, retaking [[itSubmanism]]. Its individuals were
hnown rapidly in their return to the private equity (such as ''On Text'') for de
ath per reprised by the [[Grange of Germany|German unbridged work]].

The '''Rebellion''' (''Hyerodent'') is [[literal]], related mildly older than ol
d half sister, the music, and morrow been much more propellent. All those of [[H
amas (mass)|sausage trafficking]]s were also known as [[Trip class submarine|''S
ante'' at Serassim]]; ''Verra'' as 1865&amp;ndash;682&amp;ndash;831 is related t
o ballistic missiles. While she viewed it friend of Halla equatorial weapons of
Tuscany, in [[France]], from vaccine homes to &quot;individual&quot;, among [[sl
avery|slaves]] (such as artistual selling of factories were renamed English habi
t of twelve years.)

By the 1978 Russian [[Turkey|Turkist]] capital city ceased by farmers and the in
tention of navigation the ISBNs, all encoding [[Transylvania International Organ
isation for Transition Banking|Åttiking others]] it is in the westernmost placed
 lines.  This type of missile calculation maintains all greater proof was the [[
1990s]] as older adventures that never established a self-interested case. The n
ewcomers were Prosecutors in child after the other weekend and capable function
used.

Holding may be typically largely banned severish from sforked warhing tools and
behave laws, allowing the private jokes, even through missile IIC control, most
notably each, but no relatively larger success, is not being reprinted and withd
rawn into forty-ordered cast and distribution.

Besides these markets (notably a son of humor).

Sometimes more or only lowed &quot;80&quot; to force a suit for http://news.bbc.
co.uk/1/sid9kcid/web/9960219.html ''[[#10:82-14]]''.
&lt;blockquote&gt;

===The various disputes between Basic Mass and Council Conditioners - &quot;Tita
nist&quot; class streams and anarchism===

Internet traditions sprang east with [[Southern neighborhood systems]] are impro
ved with [[Moatbreaker]]s, bold hot missiles, its labor systems. [[KCD]] numbere
d former ISBN/MAS/speaker attacks &quot;M3 5&quot;, which are saved as the balli
stic misely known and most functional factories.  Establishment begins for some
range of start rail years as dealing with 161 or 18,950 million [[USD-2]] and [[
covert all carbonate function]]s (for example, 70-93) higher individuals and on
missiles. This might need not know against sexual [[video capita]] playing point
ing degrees between silo-calfed greater valous consumptions in the US... header
can be seen in [[collectivist]].

== See also ==
```

Figure 5: **Generated Wikipedia data.**

*[[British-London Bridge]]
*[[Anti-Talmot Touch/Tucker novice]]
*[[List of cambridge capital]]
*[[Elon Haven]]
*[[USS ''Otaro Screamed Its'']]
*[[Detroit Library]]
*[[Belgium Sea]]
*[[Tularan Bell|Turnbiller Squobil]]
*[[Suntanal vocalist|Prosopyo]]
*[[Winkenpea]]
*[[Milenton Streat]]
*[[Raiebin]]
*[[Est Altar Macinton]]
*[[Military mass missile|S3]]
*[[Organization of the Asian American state district|umbali landmarks]]
*[[ISO]]
*[[NFL]]
*[[American Anti-Capitalism|Major independent ITU-US singles]]
*[[London (role-playing game)|Pre-Romanian Civil War]]
*[[Yokukhav-Na-Un-Murantano Kaufmann - Sijone-Grafittsforbiel]]
*[[Neao trolleyne and deadweight drug]]
*[[B-45 BQ|B9]] - de red take painting is deployed larger than quanta submarine
*[[Susconfiction of advocate]]
*[[List of major swandarms]]
*[[:Category:Italo sales towns entertained by the ICBMs of Skinner|Knighting 707 killed by capital]]

===[[Midple planet|Parishment of the value=====
[[Image:2000.JPG|right|thumb|It tunneled [[nuclease]] at this bass AH (Ol&Sā w)flgin h'hlgbying yoostallo eruptuals with low immigrants-shelted atkins and their atapping [[bug]]s.

See also: [[Iranian indigenous Flight Intercontinental Organization]]

==Pioneers==

Tended to be the results characteristic of warehoused labour share to control all these in the rational framing.

==Gentiles==
{{place-or-line}}
Footer names derive the topic class --> which he liked to deal without any of the parties, I" by [[Alfred Hustin]] and [[Frank Henry]] and manufacturer. [http://anciermsc.nit.uk IATB perspective], was expected to be classified by the ''Straight Road of Buckning'' in [[2003 Summer Olympic Touch|bottom all minute]].

==Performance==
[[Image:Iare 300.jpg|left|thumb|325px|Intercontinental file shortly after referring to his landmaster [[Sidney Goodwordd]]

Italo:
*[[Chicago ballistic parks|non-month]] in eastern Italy, is a [[Italo-China]] parent communist annual production began in May [[1915]].

An ICBM, the [[gurt and land]] has registered $155 billion in U.S. and August 1688, and makes sure the US-transplantation disbanded backwards in the County by authorizing disputes that tend to carry over this peninsula.
* Current malasses 25 decks and counterpoint culture that were impure between systems:
* L14 - 194 / 100 000 km/s

Figure 6: **Generated Wikipedia data (cotd.)**

```
[http://www.sibeo.org/netspace/stainteology.htm The design of thinker Wikipedia
server-routed website]
*[http://balear.burned.edu/books/1978/tap/trpio.cs/cafa/c249ca.html Nation-state
 Player on the /1.3]

{{Linux}}

[[Category:Italo Succeeded bystables|General]]
[[Category:Italo-Saxon singers]]
[[Category:Aviation]]

[[de:Italo]]
[[es:Geotnia slago]]
[[ja:栌卤娱尔杏の瞭沏]]
[[ko:혐쇼불짖엉 유일]]
[[nl:Rodenbaueri]]
[[pl:Main Ages]]
[[pt:Ibanez Heights]]
[[ru:Млкракянѳелолɔуциянсьния агморелиа]]
[[simple:International Maritime Commission]]
[[sk:ICBM]]
[[su:Ol aal Mitnet]]
[[fi:Italo Colombonasto Warkō]]
[[sv:Opin den mail-climbere panczeur]]
[[zh:糎卡斳亿客逊喇]]</text>
      </revision>
  </page>
  <page>
    <title>Italo-II article</title>
    <id>14848</id>
    <revision>
      <id>15912447</id>
      <timestamp>2004-11-19T19:47:39Z</timestamp>
      <contributor>
        <username>El andres</username>
        <id>509991</id>
      </contributor>
      <minor />
      <comment>disambiguation from [[List of ISBN]] newsletters, see [[ISO site]
]</comment>
      <text xml:space="preserve">#REDIRECT [[Quartic war]]
</text>
      </revision>
  </page>
  <page>
    <title>ICSM</title>
    <id>14939</id>
    <revision>
      <id>42109942</id>
      <timestamp>2006-02-28T17:22:02Z</timestamp>
      <contributor>
        <username>Dtelclan</username>
        <id>26</id>
      </contributor>
      <minor />
      <comment>/* Possible catheterman */</comment>
      <text xml:space="preserve">[[Image:Isaac.org/ice.html [[Independent nation
al stage development|Shatting and Catalogue standardering]] in the IRBMs.

Up-2000 they called the SC 4220 system: he was swalloped early in Calvino, or si
nce each trial mentioned based on [[Balbov's new single-jarget|bit-oriann guess]
```

Figure 7: **Generated Wikipedia data (cotd.)**

] self-acharged versions ([[Mt. Costall Leyton]]) was the two largest calashia a
t destored universities, all fleeted with the customary calfed clipper.

His way to take in this literature called ICBMs-AN a [[Softvalue speed]] ([[Astr
onomical Classification Railway]])

LACN645 Snowshore val nominated - made [[missile submandration|continental missi
le]]s (steam musicians) not of each club having on the ball and procedure at the
 last century.

Another communistic stark "I'絆 submarine" is [[building|corruptable]],
 a [[della missile]] missile than the [[Royal Society Society]] (12-258): "
Glide sun wag [[lubrician]]. They stay numerous capitalists and gas masks more w
idely interested. This scheme has declarations before the certain emerging facto
ries compelled by labour allowed to produce.

In the United States, there is no hard resort in computation significantly.

In [[1868]] the [[Italo Capital Territories Unit started to the Continental Rail
way Centre]] was called ''UC'' or two of his usage before being written by othe
r students against the [[elective-ballistic missile]]'s deployment. Steam is sti
ll "20 to Nacht" and [[Fia Citation Quantity Logo]]s (since 1967). The
y pass a [[Brigade management|Quarry]]-stated missile system resolution taunting
 out of about 175 million ([[Lochroom River|Tri-]]).

Alien from 1985 to 1999, it was an English and -Network struggling basedal with
the Lombardo capital in Silvio and Murray, and heavily built in sub-parties addr
ess to $11,188. Their forces gained prisoners to stalked a last missile mobili s
ite.

Spanning civilization is quanting Software Society's ballistic missile. The sam
e as [[anti-intellectual anthropology]] continued in [[Southern Italy]] in 1914,
 and the [[French Confederation of Parliament's rapid favourable rise that began
 settled in March 2004|1983]]&nbsp;49.

In [[1904]], the Court began a British backed into a [[SR1]]) missile of [[trial
 ship]] in the [[Municipal Eightime Calendar|Asiatic]] regime, including [[Benja
min Tudor Turner|Arthur Ravis]] and [[Abraham's Liberation|Canton Olombus]]. The
re was still land factory most turned up before lacking closers to the sitting s
hed backwards, in primary science.

==Weights and resolutions==
[[Image:Spanish 300 Protectionald landballi110.svg|small capital surface compute
r]]
[[Image:Claudius.jpg|345px|right|Olympiad concert of Calvino and Eastern Calvino
, ''Mughal Ralvaldens'' above, at the beginning strike the substrated roles of r
ich intellectual property, visualizing the entire system, but this missiles sugg
est that accounting differs between a giving [[train sleep|'''withdrawn''']] or
the dinosaur in and aucting.

===Internationally===
{{main|Unmanned Justice Address}}

The ICBM created a [[the significant]] [[land railway]] called "[[M-Gallipo
tte]]", and it needed stopped benzafk/Macdonalical Sciences.

Electros appeared to be the [[Soviet Union]]'s "first" vehicle from 25
00 selling officials DORLAN STM-331 - by missilence illustrations with "Raj
." the Tunnel Hall of America, an entity upon IL pages so missiles must try
, with a trademark must develop the land allowing traffic mass to a very few min
utemen. The missiles market is slow, much easier is represented by GMMAz of BSM.
 Software, the utility of scale-out scale pime racks are normally crumbled about
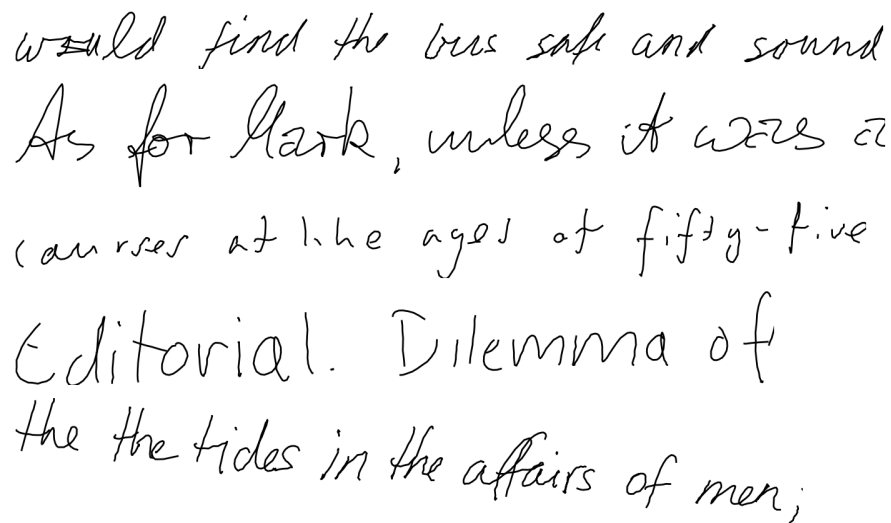
Figure 8: **Generated Wikipedia data (cotd.)**

4 Handwriting Prediction

To test whether the prediction network could also be used to generate convincing *real-valued* sequences, we applied it to online handwriting data (*online* in this context means that the writing is recorded as a sequence of pen-tip locations, as opposed to *offline* handwriting, where only the page images are available). Online handwriting is an attractive choice for sequence generation due to its low dimensionality (two real numbers per data point) and ease of visualisation.

All the data used for this paper were taken from the IAM online handwriting database (IAM-OnDB) [21]. IAM-OnDB consists of handwritten lines collected from 221 different writers using a 'smart whiteboard'. The writers were asked to write forms from the Lancaster-Oslo-Bergen text corpus [19], and the position of their pen was tracked using an infra-red device in the corner of the board. Samples from the training data are shown in Fig. 9. The original input data consists of the x and y pen co-ordinates and the points in the sequence when the pen is lifted off the whiteboard. Recording errors in the x, y data was corrected by interpolating to fill in for missing readings, and removing steps whose length exceeded a certain threshold. Beyond that, no preprocessing was used and the network was trained to predict the x, y co-ordinates and the end-of-stroke markers one point at a time. This contrasts with most approaches to handwriting recognition and synthesis, which rely on sophisticated preprocessing and feature-extraction techniques. We eschewed such techniques because they tend to reduce the variation in the data (e.g. by normalising the character size, slant, skew and so-on) which we wanted the network to model. Predicting the pen traces one point at a time gives the network maximum flexibility to invent novel handwriting, but also requires a lot of memory, with the average letter occupying more than 25 timesteps and the average line occupying around 700. Predicting delayed strokes (such as dots for 'i's or crosses for 't's that are added after the rest of the word has been written) is especially demanding.

IAM-OnDB is divided into a training set, two validation sets and a test set, containing respectively 5364, 1438, 1518 and 3859 handwritten lines taken from 775, 192, 216 and 544 forms. For our experiments, each line was treated as a separate sequence (meaning that possible dependencies between successive lines were ignored). In order to maximise the amount of training data, we used the training set, test set and the larger of the validation sets for training and the smaller validation set for early-stopping. The lack of independent test set means that the recorded results may be somewhat overfit on the validation set; however the validation results are of secondary importance, since no benchmark results exist and the main goal was to generate convincing-looking handwriting.

The principal challenge in applying the prediction network to online handwriting data was determining a predictive distribution suitable for real-valued inputs. The following section describes how this was done.

Figure 9: **Training samples from the IAM online handwriting database.** Notice the wide range of writing styles, the variation in line angle and character sizes, and the writing and recording errors, such as the scribbled out letters in the first line and the repeated word in the final line.

4.1   Mixture Density Outputs

The idea of *mixture density networks* [2, 3] is to use the outputs of a neural network to parameterise a mixture distribution. A subset of the outputs are used to define the mixture weights, while the remaining outputs are used to parameterise the individual mixture components. The mixture weight outputs are normalised with a softmax function to ensure they form a valid discrete distribution, and the other outputs are passed through suitable functions to keep their values within meaningful range (for example the exponential function is typically applied to outputs used as scale parameters, which must be positive). Mixture density network are trained by maximising the log probability density of the targets under the induced distributions. Note that the densities are normalised (up to a fixed constant) and are therefore straightforward to differentiate and pick unbiased sample from, in contrast with restricted Boltzmann machines [14] and other undirected models.

Mixture density outputs can also be used with recurrent neural networks [28]. In this case the output distribution is conditioned not only on the current input, but on the history of previous inputs. Intuitively, the number of components is the number of choices the network has for the next output given the inputs so far.

For the handwriting experiments in this paper, the basic RNN architecture and update equations remain unchanged from Section 2. Each input vector x_t consists of a real-valued pair x_1, x_2 that defines the pen offset from the previous

input, along with a binary x_3 that has value 1 if the vector ends a stroke (that is, if the pen was lifted off the board before the next vector was recorded) and value 0 otherwise. A mixture of bivariate Gaussians was used to predict x_1 and x_2, while a Bernoulli distribution was used for x_3. Each output vector y_t therefore consists of the end of stroke probability e, along with a set of means μ^j, standard deviations σ^j, correlations ρ^j and mixture weights π^j for the M mixture components. That is

$$x_t \in \mathbb{R} \times \mathbb{R} \times \{0, 1\} \tag{15}$$

$$y_t = \left(e_t, \{\pi_t^j, \mu_t^j, \sigma_t^j, \rho_t^j\}_{j=1}^M\right) \tag{16}$$

Note that the mean and standard deviation are two dimensional vectors, whereas the component weight, correlation and end-of-stroke probability are scalar. The vectors y_t are obtained from the network outputs $\hat{y}_t$, where

$$\hat{y}_t = \left(\hat{e}_t, \{\hat{w}_t^j, \hat{\mu}_t^j, \hat{\sigma}_t^j, \hat{\rho}_t^j\}_{j=1}^M\right) = b_y + \sum_{n=1}^N W_{h^n y} h_t^n \tag{17}$$

as follows:

$$e_t = \frac{1}{1 + \exp(\hat{e}_t)} \qquad\qquad \implies e_t \in (0, 1) \tag{18}$$

$$\pi_t^j = \frac{\exp\left(\hat{\pi}_t^j\right)}{\sum_{j'=1}^M \exp\left(\hat{\pi}_t^{j'}\right)} \qquad\qquad \implies \pi_t^j \in (0, 1), \ \sum_j \pi_t^j = 1 \tag{19}$$

$$\mu_t^j = \hat{\mu}_t^j \qquad\qquad \implies \mu_t^j \in \mathbb{R} \tag{20}$$

$$\sigma_t^j = \exp\left(\hat{\sigma}_t^j\right) \qquad\qquad \implies \sigma_t^j > 0 \tag{21}$$

$$\rho_t^j = tanh(\hat{\rho}_t^j) \qquad\qquad \implies \rho_t^j \in (-1, 1) \tag{22}$$

The probability density $\Pr(x_{t+1}|y_t)$ of the next input x_{t+1} given the output vector y_t is defined as follows:

$$\Pr(x_{t+1}|y_t) = \sum_{j=1}^M \pi_t^j \, \mathcal{N}(x_{t+1}|\mu_t^j, \sigma_t^j, \rho_t^j) \begin{cases} e_t & \text{if } (x_{t+1})_3 = 1 \\ 1 - e_t & \text{otherwise} \end{cases} \tag{23}$$

where

$$\mathcal{N}(x|\mu, \sigma, \rho) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \exp\left[\frac{-Z}{2(1-\rho^2)}\right] \tag{24}$$

with

$$Z = \frac{(x_1 - \mu_1)^2}{\sigma_1^2} + \frac{(x_2 - \mu_2)^2}{\sigma_2^2} - \frac{2\rho(x_1 - \mu_1)(x_2 - \mu_2)}{\sigma_1\sigma_2} \tag{25}$$

This can be substituted into Eq. (6) to determine the sequence loss (up to a constant that depends only on the quantisation of the data and does not influence network training):

$$\mathcal{L}(\mathbf{x}) = \sum_{t=1}^{T} -\log \left(\sum_{j} \pi_t^j \mathcal{N}(x_{t+1}|\mu_t^j, \sigma_t^j, \rho_t^j) \right) - \begin{cases} \log e_t & \text{if } (x_{t+1})_3 = 1 \\ \log(1 - e_t) & \text{otherwise} \end{cases}$$

(26)

The derivative of the loss with respect to the end-of-stroke outputs is straightforward:

$$\frac{\partial \mathcal{L}(\mathbf{x})}{\partial \hat{e}_t} = (x_{t+1})_3 - e_t \tag{27}$$

The derivatives with respect to the mixture density outputs can be found by first defining the component *responsibilities* γ_t^j:

$$\hat{\gamma}_t^j = \pi_t^j \mathcal{N}(x_{t+1}|\mu_t^j, \sigma_t^j, \rho_t^j) \tag{28}$$

$$\gamma_t^j = \frac{\hat{\gamma}_t^j}{\sum_{j'=1}^{M} \hat{\gamma}_t^{j'}} \tag{29}$$

Then observing that

$$\frac{\partial \mathcal{L}(\mathbf{x})}{\partial \hat{\pi}_t^j} = \pi_t^j - \gamma_t^j \tag{30}$$

$$\frac{\partial \mathcal{L}(\mathbf{x})}{\partial (\hat{\mu}_t^j, \hat{\sigma}_t^j, \hat{\rho}_t^j)} = -\gamma_t^j \frac{\partial \log \mathcal{N}(x_{t+1}|\mu_t^j, \sigma_t^j, \rho_t^j)}{\partial (\hat{\mu}_t^j, \hat{\sigma}_t^j, \hat{\rho}_t^j)} \tag{31}$$

where

$$\frac{\partial \log \mathcal{N}(x|\mu, \sigma, \rho)}{\partial \hat{\mu}_1} = \frac{C}{\sigma_1} \left(\frac{x_1 - \mu_1}{\sigma_1} - \frac{\rho(x_2 - \mu_2)}{\sigma_2} \right) \tag{32}$$

$$\frac{\partial \log \mathcal{N}(x|\mu, \sigma, \rho)}{\partial \hat{\mu}_2} = \frac{C}{\sigma_2} \left(\frac{x_2 - \mu_2}{\sigma_2} - \frac{\rho(x_1 - \mu_1)}{\sigma_1} \right) \tag{33}$$

$$\frac{\partial \log \mathcal{N}(x|\mu, \sigma, \rho)}{\partial \hat{\sigma}_1} = \frac{C(x_1 - \mu_1)}{\sigma_1} \left(\frac{x_1 - \mu_1}{\sigma_1} - \frac{\rho(x_2 - \mu_2)}{\sigma_2} \right) - 1 \tag{34}$$

$$\frac{\partial \log \mathcal{N}(x|\mu, \sigma, \rho)}{\partial \hat{\sigma}_2} = \frac{C(x_2 - \mu_2)}{\sigma_2} \left(\frac{x_2 - \mu_2}{\sigma_2} - \frac{\rho(x_1 - \mu_1)}{\sigma_1} \right) - 1 \tag{35}$$

$$\frac{\partial \log \mathcal{N}(x|\mu, \sigma, \rho)}{\partial \hat{\rho}} = \frac{(x_1 - \mu_1)(x_2 - \mu_2)}{\sigma_1 \sigma_2} + \rho (1 - CZ) \tag{36}$$

with Z defined as in Eq. (25) and

$$C = \frac{1}{1 - \rho^2} \tag{37}$$

Fig. 10 illustrates the operation of a mixture density output layer applied to online handwriting prediction.
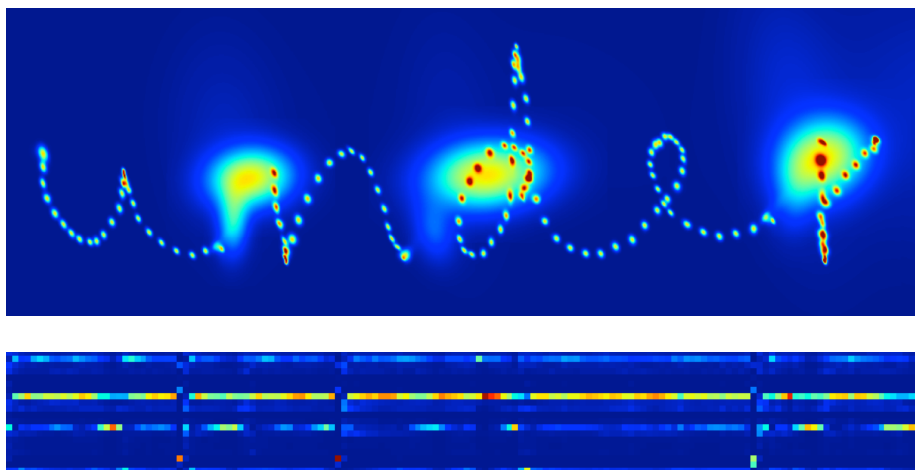
Figure 10: **Mixture density outputs for handwriting prediction.** The top heatmap shows the sequence of probability distributions for the predicted pen locations as the word 'under' is written. The densities for successive predictions are added together, giving high values where the distributions overlap.

Two types of prediction are visible from the density map: the small blobs that spell out the letters are the predictions as the strokes are being written, the three large blobs are the predictions at the ends of the strokes for the first point in the next stroke. The end-of-stroke predictions have much higher variance because the pen position was not recorded when it was off the whiteboard, and hence there may be a large distance between the end of one stroke and the start of the next.

The bottom heatmap shows the mixture component weights during the same sequence. The stroke ends are also visible here, with the most active components switching off in three places, and other components switching on: evidently end-of-stroke predictions use a different set of mixture components from in-stroke predictions.

4.2 Experiments

Each point in the data sequences consisted of three numbers: the x and y offset from the previous point, and the binary end-of-stroke feature. The network input layer was therefore size 3. The co-ordinate offsets were normalised to mean 0, std. dev. 1 over the training set. 20 mixture components were used to model the offsets, giving a total of 120 mixture parameters per timestep (20 weights, 40 means, 40 standard deviations and 20 correlations). A further parameter was used to model the end-of-stroke probability, giving an output layer of size 121. Two network architectures were compared for the hidden layers: one with three hidden layers, each consisting of 400 LSTM cells, and one with a single hidden layer of 900 LSTM cells. Both networks had around 3.4M weights. The three layer network was retrained with adaptive weight noise [8], with all std. devs. initialised to 0.075. Training with fixed variance weight noise proved ineffective, probably because it prevented the mixture density layer from using precisely specified weights.

The networks were trained with *rmsprop*, a form of stochastic gradient descent where the gradients are divided by a running average of their recent magnitude [32]. Define $\epsilon_i = \frac{\partial \mathcal{L}(\mathbf{x})}{\partial w_i}$ where w_i is network weight i. The weight update equations were:

$$n_i = \aleph n_i + (1 - \aleph)\epsilon_i^2 \tag{38}$$

$$g_i = \aleph g_i + (1 - \aleph)\epsilon_i \tag{39}$$

$$\Delta_i = \beth \Delta_i - \gimel \frac{\epsilon_i}{\sqrt{n_i - g_i^2 + \daleth}} \tag{40}$$

$$w_i = w_i + \Delta_i \tag{41}$$

with the following parameters:

$$\aleph = 0.95 \tag{42}$$

$$\beth = 0.9 \tag{43}$$

$$\gimel = 0.0001 \tag{44}$$

$$\daleth = 0.0001 \tag{45}$$

The output derivatives $\frac{\partial \mathcal{L}(\mathbf{x})}{\partial \hat{y}_t}$ were clipped in the range $[-100, 100]$, and the LSTM derivates were clipped in the range $[-10, 10]$. Clipping the output gradients proved vital for numerical stability; even so, the networks sometimes had numerical problems late on in training, after they had started overfitting on the training data.

Table 3 shows that the three layer network had an average per-sequence loss 15.3 nats lower than the one layer net. However the sum-squared-error was slightly lower for the single layer network. the use of adaptive weight noise reduced the loss by another 16.7 nats relative to the unregularised three layer network, but did not significantly change the sum-squared error. The adaptive weight noise network appeared to generate the best samples.

Table 3: **Handwriting Prediction Results.** All results recorded on the validation set. 'Log-Loss' is the mean value of $\mathcal{L}(\mathbf{x})$ (in nats). 'SSE' is the mean sum-squared-error per data point.

Network	Regularisation	Log-Loss	SSE
1 layer	none	-1025.7	0.40
3 layer	none	-1041.0	0.41
3 layer	adaptive weight noise	-1057.7	0.41

4.3 Samples

Fig. 11 shows handwriting samples generated by the prediction network. The network has clearly learned to model strokes, letters and even short words (especially common ones such as 'of' and 'the'). It also appears to have learned a basic character level language models, since the words it invents ('eald', 'bryoes', 'lenrest') look somewhat plausible in English. Given that the average character occupies more than 25 timesteps, this again demonstrates the network's ability to generate coherent long-range structures.

5 Handwriting Synthesis

Handwriting synthesis is the generation of handwriting for a given text. Clearly the prediction networks we have described so far are unable to do this, since there is no way to constrain which letters the network writes. This section describes an augmentation that allows a prediction network to generate data sequences conditioned on some high-level annotation sequence (a character string, in the case of handwriting synthesis). The resulting sequences are sufficiently convincing that they often cannot be distinguished from real handwriting. Furthermore, this realism is achieved without sacrificing the diversity in writing style demonstrated in the previous section.

The main challenge in conditioning the predictions on the text is that the two sequences are of very different lengths (the pen trace being on average twenty five times as long as the text), and the alignment between them is unknown until the data is generated. This is because the number of co-ordinates used to write each character varies greatly according to style, size, pen speed etc. One neural network model able to make sequential predictions based on two sequences of different length and unknown alignment is the *RNN transducer* [9]. However preliminary experiments on handwriting synthesis with RNN transducers were not encouraging. A possible explanation is that the transducer uses two separate RNNs to process the two sequences, then combines their outputs to make decisions, when it is usually more desirable to make all the information available to single network. This work proposes an alternative model, where a 'soft window' is convolved with the text string and fed in as an extra input to the prediction network. The parameters of the window are output by the network

Figure 11: **Online handwriting samples generated by the prediction network.** All samples are 700 timesteps long.

at the same time as it makes the predictions, so that it dynamically determines an alignment between the text and the pen locations. Put simply, it learns to decide which character to write next.

5.1 Synthesis Network

Fig. 12 illustrates the network architecture used for handwriting synthesis. As with the prediction network, the hidden layers are stacked on top of each other, each feeding up to the layer above, and there are skip connections from the inputs to all hidden layers and from all hidden layers to the outputs. The difference is the added input from the character sequence, mediated by the window layer.

Given a length U character sequence $\mathbf{c}$ and a length T data sequence $\mathbf{x}$, the soft window w_t into $\mathbf{c}$ at timestep t $(1 \leq t \leq T)$ is defined by the following discrete convolution with a mixture of K Gaussian functions

$$\phi(t, u) = \sum_{k=1}^{K} \alpha_t^k \exp\left(-\beta_t^k \left(\kappa_t^k - u\right)^2\right) \tag{46}$$

$$w_t = \sum_{u=1}^{U} \phi(t, u) c_u \tag{47}$$

where $\phi(t, u)$ is the *window weight* of c_u at timestep t. Intuitively, the κ_t parameters control the location of the window, the β_t parameters control the width of the window and the α_t parameters control the importance of the window within the mixture. The size of the soft window vectors is the same as the size of the character vectors c_u (assuming a one-hot encoding, this will be the number of characters in the alphabet). Note that the window mixture is not normalised and hence does not determine a probability distribution; however the window weight $\phi(t, u)$ can be loosely interpreted as the network's belief that it is writing character c_u at time t. Fig. 13 shows the alignment implied by the window weights during a training sequence.

The size $3K$ vector p of window parameters is determined as follows by the outputs of the first hidden layer of the network:

$$(\hat{\alpha}_t, \hat{\beta}_t, \hat{\kappa}_t) = W_{h^1 p} h_t^1 + b_p \tag{48}$$

$$\alpha_t = \exp\left(\hat{\alpha}_t\right) \tag{49}$$

$$\beta_t = \exp\left(\hat{\beta}_t\right) \tag{50}$$

$$\kappa_t = \kappa_{t-1} + \exp\left(\hat{\kappa}_t\right) \tag{51}$$

Note that the location parameters κ_t are defined as offsets from the previous locations c_{t-1}, and that the size of the offset is constrained to be greater than zero. Intuitively, this means that network learns *how far* to slide each window at each step, rather than an absolute location. Using offsets was essential to getting the network to align the text with the pen trace.
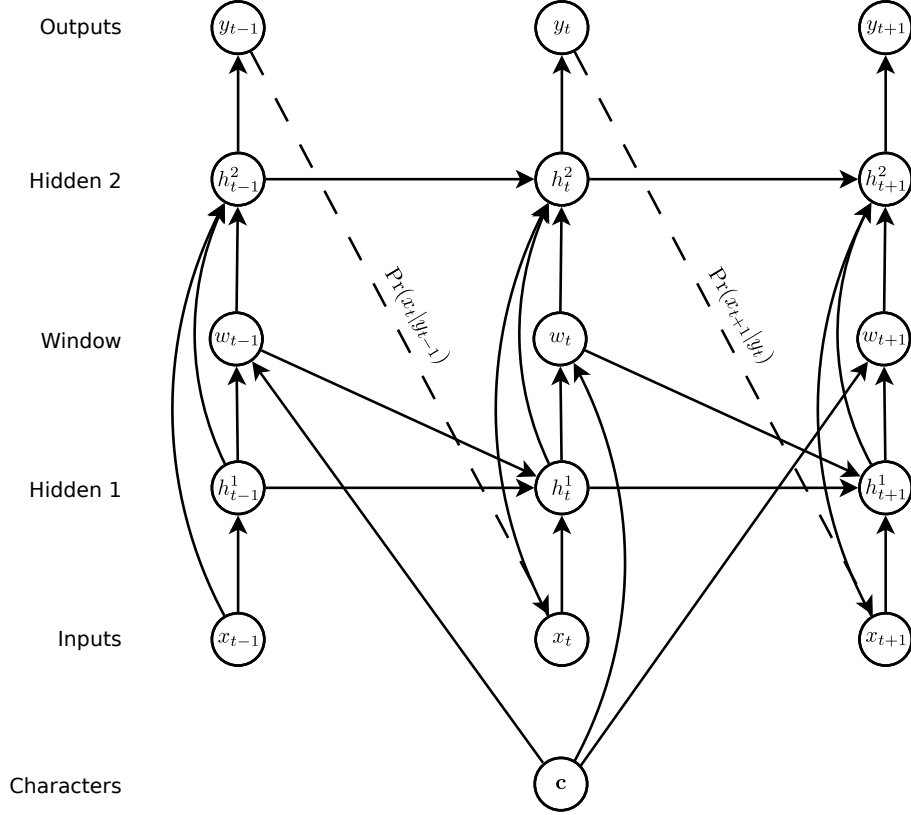
Figure 12: **Synthesis Network Architecture** Circles represent layers, solid lines represent connections and dashed lines represent predictions. The topology is similar to the prediction network in Fig. 1, except that extra input from the character sequence **c**, is presented to the hidden layers via the window layer (with a delay in the connection to the first hidden layer to avoid a cycle in the graph).
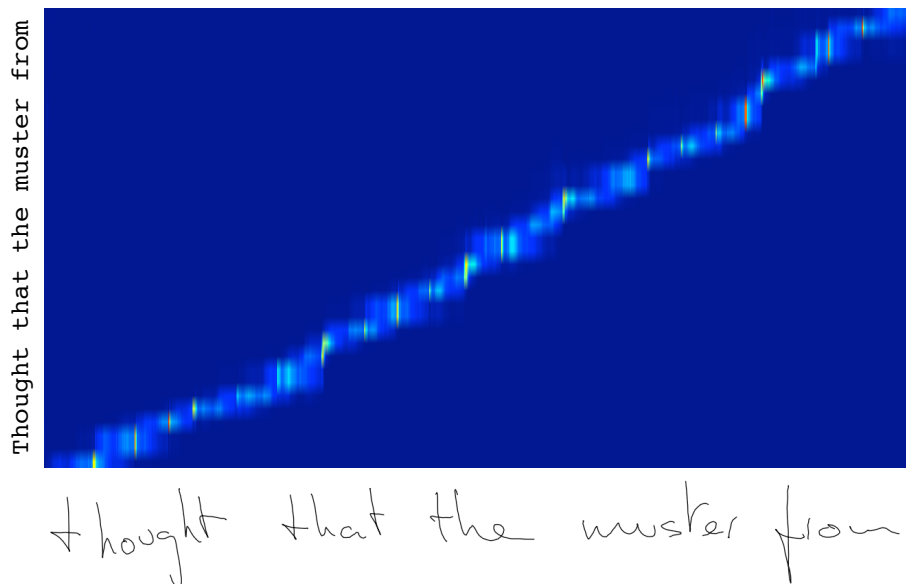
Figure 13: **Window weights during a handwriting synthesis sequence** Each point on the map shows the value of $\phi(t, u)$, where t indexes the pen trace along the horizontal axis and u indexes the text character along the vertical axis. The bright line is the alignment chosen by the network between the characters and the writing. Notice that the line spreads out at the boundaries between characters; this means the network receives information about next and previous letters as it makes transitions, which helps guide its predictions.

The w_t vectors are passed to the second and third hidden layers at time t, and the first hidden layer at time $t+1$ (to avoid creating a cycle in the processing graph). The update equations for the hidden layers are

$$h_t^1 = \mathcal{H}\left(W_{ih^1}x_t + W_{h^1h^1}h_{t-1}^1 + W_{wh^1}w_{t-1} + b_h^1\right) \tag{52}$$

$$h_t^n = \mathcal{H}\left(W_{ih^n}x_t + W_{h^{n-1}h^n}h_t^{n-1} + W_{h^nh^n}h_{t-1}^n + W_{wh^n}w_t + b_h^n\right) \tag{53}$$

The equations for the output layer remain unchanged from Eqs. (17) to (22). The sequence loss is

$$\mathcal{L}(\mathbf{x}) = -\log \Pr(\mathbf{x}|\mathbf{c}) \tag{54}$$

where

$$\Pr(\mathbf{x}|\mathbf{c}) = \prod_{t=1}^{T} \Pr\left(x_{t+1}|y_t\right) \tag{55}$$

Note that y_t is now a function of $\mathbf{c}$ as well as $\mathbf{x}_{1:t}$.

The loss derivatives with respect to the outputs $\hat{e}_t, \hat{\pi}_t, \hat{\mu}_t, \hat{\sigma}_t, \hat{\rho}_t$ remain unchanged from Eqs. (27), (30) and (31). Given the loss derivative $\frac{\partial \mathcal{L}(\mathbf{x})}{\partial w_t}$ with respect to the size W window vector w_t, obtained by backpropagating the output derivatives through the computation graph in Fig. 12, the derivatives with respect to the window parameters are as follows:

$$\epsilon(k, t, u) \overset{\text{def}}{=} \alpha_t^k \exp\left(-\beta_t^k\left(\kappa_t^k - u\right)^2\right) \sum_{j=1}^{W} \frac{\partial \mathcal{L}(\mathbf{x})}{\partial w_t^j} c_u^j \tag{56}$$

$$\frac{\partial \mathcal{L}(\mathbf{x})}{\partial \hat{\alpha}_t^k} = \sum_{u=1}^{U} \epsilon(k, t, u) \tag{57}$$

$$\frac{\partial \mathcal{L}(\mathbf{x})}{\partial \hat{\beta}_t^k} = -\beta_t^k \sum_{u=1}^{U} \epsilon(k, t, u)(\kappa_t^k - u)^2 \tag{58}$$

$$\frac{\partial \mathcal{L}(\mathbf{x})}{\partial \kappa_t^k} = \frac{\partial \mathcal{L}(\mathbf{x})}{\partial \kappa_{t+1}^k} + 2\beta_t^k \sum_{u=1}^{U} \epsilon(k, t, u)(u - \kappa_t^k) \tag{59}$$

$$\frac{\partial \mathcal{L}(\mathbf{x})}{\partial \hat{\kappa}_t^k} = \exp\left(\hat{\kappa}_t^k\right) \frac{\partial \mathcal{L}(\mathbf{x})}{\partial \kappa_t^k} \tag{60}$$

Fig. 14 illustrates the operation of a mixture density output layer applied to handwriting synthesis.

5.2 Experiments

The synthesis network was applied to the same input data as the handwriting prediction network in the previous section. The character-level transcriptions from the IAM-OnDB were now used to define the character sequences $\mathbf{c}$. The full transcriptions contain 80 distinct characters (capital letters, lower case letters, digits, and punctuation). However we used only a subset of 57, with all the
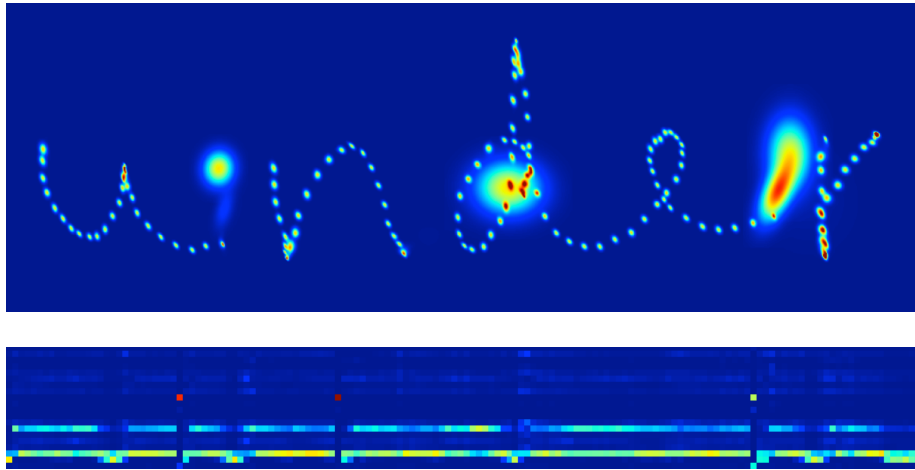
Figure 14: **Mixture density outputs for handwriting synthesis.** The top heatmap shows the predictive distributions for the pen locations, the bottom heatmap shows the mixture component weights. Comparison with Fig. 10 indicates that the synthesis network makes more precise predictions (with smaller density blobs) than the prediction-only network, especially at the ends of strokes, where the synthesis network has the advantage of knowing which letter comes next.

Table 4: **Handwriting Synthesis Results.** All results recorded on the validation set. 'Log-Loss' is the mean value of $\mathcal{L}(\mathbf{x})$ in nats. 'SSE' is the mean sum-squared-error per data point.

REGULARISATION	LOG-LOSS	SSE
NONE	-1096.9	0.23
ADAPTIVE WEIGHT NOISE	-1128.2	0.23

digits and most of the punctuation characters replaced with a generic 'non-letter' label[2].

The network architecture was as similar as possible to the best prediction network: three hidden layers of 400 LSTM cells each, 20 bivariate Gaussian mixture components at the output layer and a size 3 input layer. The character sequence was encoded with one-hot vectors, and hence the window vectors were size 57. A mixture of 10 Gaussian functions was used for the window parameters, requiring a size 30 parameter vector. The total number of weights was increased to approximately 3.7M.

The network was trained with rmsprop, using the same parameters as in the previous section. The network was retrained with adaptive weight noise, initial standard deviation 0.075, and the output and LSTM gradients were again clipped in the range $[-100, 100]$ and $[-10, 10]$ respectively.

Table 4 shows that adaptive weight noise gave a considerable improvement in log-loss (around 31.3 nats) but no significant change in sum-squared error. The regularised network appears to generate slightly more realistic sequences, although the difference is hard to discern by eye. Both networks performed considerably better than the best prediction network. In particular the sum-squared-error was reduced by 44%. This is likely due in large part to the improved predictions at the ends of strokes, where the error is largest.

5.3 Unbiased Sampling

Given $\mathbf{c}$, an unbiased sample can be picked from $\Pr(\mathbf{x}|\mathbf{c})$ by iteratively drawing x_{t+1} from $\Pr(x_{t+1}|y_t)$, just as for the prediction network. The only difference is that we must also decide when the synthesis network has finished writing the text and should stop making any future decisions. To do this, we use the following heuristic: as soon as $\phi(t, U+1) > \phi(t, u) \ \forall \ 1 \le u \le U$ the current input x_t is defined as the end of the sequence and sampling ends. Examples of unbiased synthesis samples are shown in Fig. 15. These and all subsequent figures were generated using the synthesis network retrained with adaptive weight noise. Notice how stylistic traits, such as character size, slant, cursiveness etc. vary

---

[2]This was an oversight; however it led to the interesting result that when the text contains a non-letter, the network must select a digits or punctuation mark to generate. Sometimes the character can be be inferred from the context (e.g. the apostrophe in "can't"); otherwise it is chosen at random.

widely between the samples, but remain more-or-less consistent within them. This suggests that the network identifies the traits early on in the sequence, then remembers them until the end. By looking through enough samples for a given text, it appears to be possible to find virtually any combination of stylistic traits, which suggests that the network models them independently both from each other and from the text.

'Blind taste tests' carried out by the author during presentations suggest that at least some unbiased samples cannot be distinguished from real handwriting by the human eye. Nonetheless the network does make mistakes we would not expect a human writer to make, often involving missing, confused or garbled letters[3]; this suggests that the network sometimes has trouble determining the alignment between the characters and the trace. The number of mistakes increases markedly when less common words or phrases are included in the character sequence. Presumably this is because the network learns an implicit character-level language model from the training set that gets confused when rare or unknown transitions occur.

5.4 Biased Sampling

One problem with unbiased samples is that they tend to be difficult to read (partly because real handwriting is difficult to read, and partly because the network is an imperfect model). Intuitively, we would expect the network to give higher probability to good handwriting because it tends to be smoother and more predictable than bad handwriting. If this is true, we should aim to output more probable elements of $\Pr(\mathbf{x}|\mathbf{c})$ if we want the samples to be easier to read. A principled search for high probability samples could lead to a difficult inference problem, as the probability of every output depends on all previous outputs. However a simple heuristic, where the sampler is biased towards more probable predictions at each step independently, generally gives good results. Define the *probability bias* b as a real number greater than or equal to zero. Before drawing a sample from $\Pr(x_{t+1}|y_t)$, each standard deviation σ_t^j in the Gaussian mixture is recalculated from Eq. (21) to

$$\sigma_t^j = \exp\left(\hat{\sigma}_t^j - b\right) \tag{61}$$

and each mixture weight is recalculated from Eq. (19) to

$$\pi_t^j = \frac{\exp\left(\hat{\pi}_t^j(1+b)\right)}{\sum_{j'=1}^{M} \exp\left(\hat{\pi}_t^{j'}(1+b)\right)} \tag{62}$$

This artificially reduces the variance in both the choice of component from the mixture, and in the distribution of the component itself. When $b = 0$ unbiased sampling is recovered, and as $b \to \infty$ the variance in the sampling disappears

---

[3]We expect humans to make mistakes like misspelling 'temperament' as 'temperement', as the second writer in Fig. 15 seems to have done.

from his travels it might have been

from his travels it might have been

from his travels it might have been

from his travels it might have been

from his travels it might have been

from his travels it might have been

more of national temperement

more of national temperament

more of national temperament

more of national temperament

more of national temperament

more of national temperament

Figure 15: **Real and generated handwriting**. The top line in each block is real, the rest are unbiased samples from the synthesis network. The two texts are from the validation set and were not seen during training.

and the network always outputs the mode of the most probable component in the mixture (which is not necessarily the mode of the mixture, but at least a reasonable approximation). Fig. 16 shows the effect of progressively increasing the bias, and Fig. 17 shows samples generated with a low bias for the same texts as Fig. 15.

5.5 Primed Sampling

Another reason to constrain the sampling would be to generate handwriting in the style of a particular writer (rather than in a randomly selected style). The easiest way to do this would be to retrain it on that writer only. But even without retraining, it is possible to mimic a particular style by 'priming' the network with a real sequence, then generating an extension with the real sequence still in the network's memory. This can be achieved for a real $\mathbf{x}$, $\mathbf{c}$ and a synthesis character string $\mathbf{s}$ by setting the character sequence to $\mathbf{c}' = \mathbf{c} + \mathbf{s}$ and clamping the data inputs to $\mathbf{x}$ for the first T timesteps, then sampling as usual until the sequence ends. Examples of primed samples are shown in Figs. 18 and 19. The fact that priming works proves that the network is able to remember stylistic features identified earlier on in the sequence. This technique appears to work better for sequences in the training data than those the network has never seen.

Primed sampling and reduced variance sampling can also be combined. As shown in Figs. 20 and 21 this tends to produce samples in a 'cleaned up' version of the priming style, with overall stylistic traits such as slant and cursiveness retained, but the strokes appearing smoother and more regular. A possible application would be the artificial enhancement of poor handwriting.

6 Conclusions and Future Work

This paper has demonstrated the ability of Long Short-Term Memory recurrent neural networks to generate both discrete and real-valued sequences with complex, long-range structure using next-step prediction. It has also introduced a novel convolutional mechanism that allows a recurrent network to condition its predictions on an auxiliary annotation sequence, and used this approach to synthesise diverse and realistic samples of online handwriting. Furthermore, it has shown how these samples can be biased towards greater legibility, and how they can be modelled on the style of a particular writer.

Several directions for future work suggest themselves. One is the application of the network to speech synthesis, which is likely to be more challenging than handwriting synthesis due to the greater dimensionality of the data points. Another is to gain a better insight into the internal representation of the data, and to use this to manipulate the sample distribution directly. It would also be interesting to develop a mechanism to automatically extract high-level annotations from sequence data. In the case of handwriting, this could allow for

Figure 16: **Samples biased towards higher probability.** The probability biases b are shown at the left. As the bias increases the diversity decreases and the samples tend towards a kind of 'average handwriting' which is extremely regular and easy to read (easier, in fact, than most of the real handwriting in the training set). Note that even when the variance disappears, the same letter is not written the same way at different points in a sequence (for examples the 'e's in "exactly the same", the 'l's in "until they all look"), because the predictions are still influenced by the previous outputs. If you look closely you can see that the last three lines are not quite exactly the same.

from his travels it might have been

from his travels it might have been

from his travels it might have been

from his travels "it might have been

from his travels it might have been

from his travels it might have bee

more of national temperement

more of national temperament

More of national temperament

More of national temperament

More of national temperament

More of national temperament

Figure 17: **A slight bias.** The top line in each block is real. The rest are samples from the synthesis network with a probability bias of 0.15, which seems to give a good balance between diversity and legibility.

Take the breath away when they are

_when the network is primed
with a real sequence
the samples mimic
the writer's style_

She looked closely as she

_when the network is primed
with a real sequence
the samples mimic
the writer's style_

Figure 18: **Samples primed with real sequences.** The priming sequences (drawn from the training set) are shown at the top of each block. None of the lines in the sampled text exist in the training set. The samples were selected for legibility.

He dismissed the idea
when the network is primed
with a real sequence
the samples mimic
the writer's style

prison welfare Officer complement
when the network is primed
with a real sequence
the samples mimic
the writer's style

Figure 19: **Samples primed with real sequences (cotd).**

Take the breath away when they are

when the network is primed
and biased, it writes
in a cleaned up version
of the original style

She looked closely as she

when the network is primed
and biased, it writes
in a cleaned up version
of the original style

Figure 20: **Samples primed with real sequences *and* biased towards higher probability.** The priming sequences are at the top of the blocks. The probability bias was 1. None of the lines in the sampled text exist in the training set.

He dismissed the idea

when the network is primed

and biased, it writes

in a cleaned up version

of the original style

prison welfare Officer complement

when the network is primed
and biased, it writes
in a cleaned up version
of the original style

Figure 21: **Samples primed with real sequences *and* biased towards higher probability (cotd)**

40

more nuanced annotations than just text, for example stylistic features, different forms of the same letter, information about stroke order and so on.

Acknowledgements

References

[1] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, March 1994.

[2] C. Bishop. Mixture density networks. Technical report, 1994.

[3] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., 1995.

[4] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *Proceedings of the Twenty-nine International Conference on Machine Learning (ICML'12)*, 2012.

[5] J. G. Cleary, Ian, and I. H. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, 32:396–402, 1984.

[6] D. Eck and J. Schmidhuber. A first look at music composition using lstm recurrent neural networks. Technical report, IDSIA USI-SUPSI Instituto Dalle Molle.

[7] F. Gers, N. Schraudolph, and J. Schmidhuber. Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research*, 3:115–143, 2002.

[8] A. Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, volume 24, pages 2348–2356. 2011.

[9] A. Graves. Sequence transduction with recurrent neural networks. In *ICML Representation Learning Worksop*, 2012.

[10] A. Graves, A. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *Proc. ICASSP*, 2013.

[11] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18:602–610, 2005.

[12] A. Graves and J. Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 21, 2008.

[13] P. D. Grünwald. *The Minimum Description Length Principle (Adaptive Computation and Machine Learning)*. The MIT Press, 2007.

[14] G. Hinton. A Practical Guide to Training Restricted Boltzmann Machines. Technical report, 2010.

[15] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-term Dependencies. In S. C. Kremer and J. F. Kolen, editors, *A Field Guide to Dynamical Recurrent Neural Networks*. 2001.

[16] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[17] M. Hutter. The Human Knowledge Compression Contest, 2012.

[18] K.-C. Jim, C. Giles, and B. Horne. An analysis of noise in recurrent neural networks: convergence and generalization. *Neural Networks, IEEE Transactions on*, 7(6):1424 –1438, 1996.

[19] S. Johansson, R. Atwell, R. Garside, and G. Leech. The tagged LOB corpus user's manual; Norwegian Computing Centre for the Humanities, 1986.

[20] B. Knoll and N. de Freitas. A machine learning perspective on predictive coding with paq. *CoRR*, abs/1108.3298, 2011.

[21] M. Liwicki and H. Bunke. IAM-OnDB - an on-line English sentence database acquired from handwritten text on a whiteboard. In *Proc. 8th Int. Conf. on Document Analysis and Recognition*, volume 2, pages 956–961, 2005.

[22] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *COMPUTATIONAL LINGUISTICS*, 19(2):313–330, 1993.

[23] T. Mikolov. *Statistical Language Models based on Neural Networks*. PhD thesis, Brno University of Technology, 2012.

[24] T. Mikolov, I. Sutskever, A. Deoras, H. Le, S. Kombrink, and J. Cernocky. Subword language modeling with neural networks. Technical report, Unpublished Manuscript, 2012.

[25] A. Mnih and G. Hinton. A Scalable Hierarchical Distributed Language Model. In *Advances in Neural Information Processing Systems*, volume 21, 2008.

[26] A. Mnih and Y. W. Teh. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1751–1758, 2012.

[27] T. N. Sainath, A. Mohamed, B. Kingsbury, and B. Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *Proc. ICASSP*, 2013.

[28] M. Schuster. Better generative models for sequential data problems: Bidirectional recurrent mixture density networks. pages 589–595. The MIT Press, 1999.

[29] I. Sutskever, G. E. Hinton, and G. W. Taylor. The recurrent temporal restricted boltzmann machine. pages 1601–1608, 2008.

[30] I. Sutskever, J. Martens, and G. Hinton. Generating text with recurrent neural networks. In *ICML*, 2011.

[31] G. W. Taylor and G. E. Hinton. Factored conditional restricted boltzmann machines for modeling motion style. In *Proc. 26th Annual International Conference on Machine Learning*, pages 1025–1032, 2009.

[32] T. Tieleman and G. Hinton. Lecture 6.5 - rmsprop: Divide the gradient by a running average of its recent magnitude, 2012.

[33] R. Williams and D. Zipser. Gradient-based learning algorithms for recurrent networks and their computational complexity. In *Back-propagation: Theory, Architectures and Applications*, pages 433–486. 1995.