To Remember Or Not To Remember: Shakespeare Text Generation with Recurrent Neural Networks and Long Short Term Memory

Anthony Tong A17720195
Brandon Park A17428364
Chi Zhang A16346955
Christopher Rebollar-Ramirez A16982224

Abstract

In this report, we discuss the setup and training of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) architectures, two that have seen significant success in modeling temporal dependencies. We test the effectiveness of teacher forcing on the rate of convergence in our models, and finally analyze our trained models on qualitative text generation. We report the best performance on the LSTM architecture with higher sequence lengths and teacher forcing.

1 Introduction

The introduction of temporal dependencies dates back to Elman [2], which opened the door to foundational progress in various domains, including speech recognition, time series forecasting, and natural language processing. We explore the latter in this report, employing a recurrent neural network (RNN) to perform text generation on sources from William Shakespeare. We use the TinyShakespeare dataset [5], a collection of 40,000 lines aggregated from Shakespeare's plays, with the goal of training our models to produce verses in his literary voice, finding remarkably successful performance on both architectures, almost unreasonably, with only character level inputs and outputs. We additionally explore the improvements upon RNNs in the form of a core architectural change by Hochreiter and Schmidhuber, Long Short-Term Memory (LSTM) networks [6]. We detail our training architectures in Section 3, results in Section 4, and discuss our findings in Section 5.

2 Related Work

Elman orginally proposed the concept of Simple Recurrent Networks (SRNs), which introduced the recurrent hidden layer to model sequential dependencies, namely time [2]. However, these SRNs were prone to the vanishing gradient problem, as discussed by Hochreiter [4]. Namely, the network had significant trouble propagating gradients back in time, and as such, early SRNs struggled with capturing long term dependencies. LSTM networks were therefore introduced, which had a significant architectural improvement over SRNs: LSTMs included memory cells and gating mechanisms that filter the information flow in and out of these cells. Due to these gate mechanisms, gradients could now pass through back in time arbitrarily deep, which addressed the vanishing gradient problem.

Later, Gers et al. [3] introduced the forget gate, so that the LSTM could dispose of irrelevant information stored in the cell, whereas the original LSTM would break down if the model had not reset its internal state for long enough. In addition to these architectural improvements of the LSTM, significant progress in training methods was made. In particular, teacher forcing, proposed by Williams and Zipser [7], is a technique which vastly accelerated the rate of convergence during training. Teacher forcing is a technique that feeds the model with purely groundtruth inputs as opposed to the model's own outputs. While the technique accelerates model convergence rather quickly, it suffers from biases introduced during autoregressive predictions during inference time.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

Bengio et al. [1] sought to remedy this via scheduled sampling, a technique where the model gradually replaced the groundtruth inputs with predicted tokens over time as to slowly acclimatize the model to its own outputs. For the purposes of this project, we leave out the scheduled sampling technique to focus our analysis on the base RNN and LSTM architectures.

3 Methods

3.1 Baseline Models

Our baseline RNN model is a standard character-level RNN, with one 150 neuron hidden layer and embeddings of size 100, followed by a fully connected layer mapping all the hidden neurons to the vocabulary size to generate the final token. By default, the RNN has a tanh nonlinearity.

Our baseline LSTM model is identical to the RNN, simply using an LSTM rather than an RNN, but with the same hyperparameters.

To train, we use a teacher forcing approach and pass the entire input sequence at once to the model, then calculate cross entropy loss only on the additional output token of each sequence. This forces the model to use the ground truth inputs for its token generation, discarding its own outputs at each timestep. We train for 10 epochs without early stopping, with a sequence length of 16 and batch size of 64, using an AdamW optimizer of learning rate 0.0001.

We incrementally train both the LSTM and RNN models with a sequence length of 16, 128, and 512. For the sequence length of 128, we also train an additional LSTM model with a larger hidden layer of 300 neurons.

3.2 Training Procedure without Teacher Forcing

We also implement a training procedure without teacher forcing, with the key difference being that instead of passing in all inputs at once, we pass in only one input at a time, starting with the first token and letting the model generate each subsequent token based on its own previous output. This ensures the model learns to generate tokens with consideration for long term dependencies. However, we continue to only calculate loss on the final generated token rather then the entire sequence.

Now, we incrementally train only the LSTM model with lower sequence lengths of 4, 8, 16, and 32. We decrease the sequence lengths from the baseline for faster training.

3.3 Text Generation

To generate text, we begin with an initial primer sequence: "Macbeth\n by William Shakespeare\n Edited by Barbara A. Mowat and Paul Werstine". At each timestep, we use the model's output logits to sample the next token using a multinomial distribution. The logits are scaled down by a temperature parameter before sampling: higher temperature values flatten the probability distribution, increasing randomness and diversity in the output, while lower values make the model more deterministic by sharpening the distribution. This process continues iteratively until the desired length is reached.

We generate text with the best performing model with temperature 0.5, 1, and 2, and a max sequence length of 1000.

3.4 Increased Hidden Layers

Finally, we experiment with increasing the number of hidden layers, specifically for the LSTM and RNN models with a sequence length of 16.

In an LSTM network, the total number of trainable parameters is determined by the weights and biases associated with its four gates: the input gate, forget gate, cell gate, and output gate. Each gate has its own weight matrices for both input-to-hidden connections and hidden-to-hidden connections, as well as bias terms. The number of parameters for a single LSTM layer with an input size d and hidden size h is given by:

$$\text{Params}_{\text{LSTM}} = 4 \times \left(h^2 + d \times h + h\right) \tag{1}$$

where the first term represents the recurrent weights, the second term represents the input weights, and the last term accounts for the biases. When increasing the number of layers in an LSTM network,

Table 1: Test losses. Best stats are bolded, second best are underlined.

Model	Sequence Length	Loss
RNN (Baseline)	16	1.5309
RNN	128	1.5473
RNN	512	1.5267
LSTM (Baseline)	16	1.4054
LSTM	128	1.3932
LSTM (Larger Hidden)	128	**1.3237**
<u>LSTM</u>	512	<u>1.3773</u>
LSTM (No Teacher Forcing)	4	3.1663
LSTM (No Teacher Forcing)	8	3.1266
LSTM (No Teacher Forcing)	16	3.3049
LSTM (No Teacher Forcing)	32	3.3174
RNN (2 Hidden Layers)	16	1.5242
LSTM (2 Hidden Layers)	16	1.3857

each additional layer introduces a new set of parameters following the same formula. For a model with L layers, the total number of parameters is:

$$\text{Params}_{\text{Total}} = L \times 4 \times \left(h^2 + d \times h + h\right) \tag{2}$$

If the goal is to increase the number of layers while keeping the total number of parameters approximately constant, then reducing the hidden size is necessary. To achieve this, we equate the total parameter count before and after increasing the number of layers:

$$1 \times 4 \times \left(h^2 + d \times h + h\right) = 2 \times 4 \times \left(h'^2 + d \times h' + h'\right) \tag{3}$$

Solving for the new hidden size h', we obtain:

$$h'^2 + d \times h' + h' = \frac{h^2 + d \times h + h}{2} \tag{4}$$

which ensures that the parameter count remains balanced. Additionally, in LSTM architectures with a fully connected output layer, the number of parameters in this layer is:

$$\text{Params}_{\text{FC}} = h \times V + V \tag{5}$$

where V is the vocabulary size. This contributes to the overall model complexity but does not significantly affect the hidden size adjustment when modifying the number of layers. By applying the derived formula, we determine a hidden size of 100 is needed to maintain a similar parameter count when increasing the number of layers from 1 to 2 in an LSTM network.

4 Results

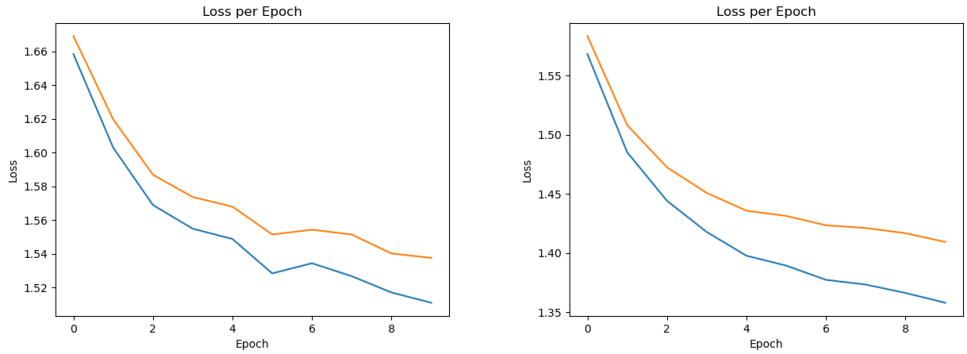
The final test losses for each model are summarized in Table 1

Training and validation loss over time for the baseline models are shown in Figure 1. Similarly, results for the 128 sequence length and 512 sequence length models are displayed in Figure 2. The increased hidden layer size LSTM results are displayed in Figure 3, and LSTM results without teacher forcing are displayed in Figure 4. Finally, the results for the LSTM and RNN with an increased number of hidden layers is displayed in Figure 5.

The text generation results are as follows:

- Temperature = 0.5:

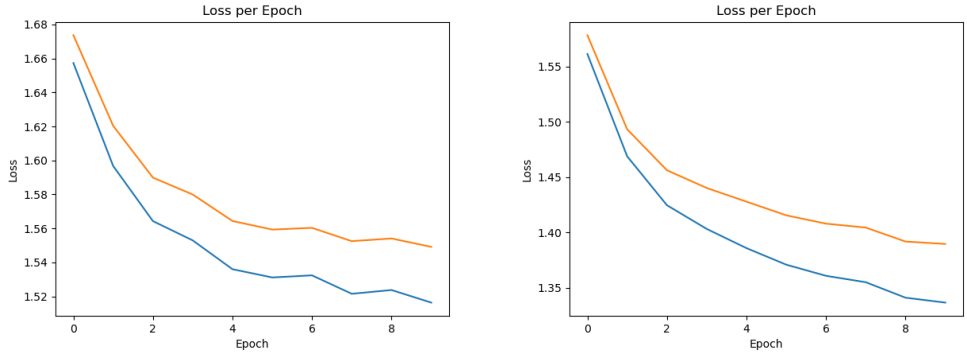
```
Macbeth
 by William Shakespeare
 Edited by Barbara A. Mowat and Paul Werstine,
The sun that hath he honest, so said him in a
mile and years his made, but shall be reason
```



(a) Training and validation loss of the baseline RNN model.

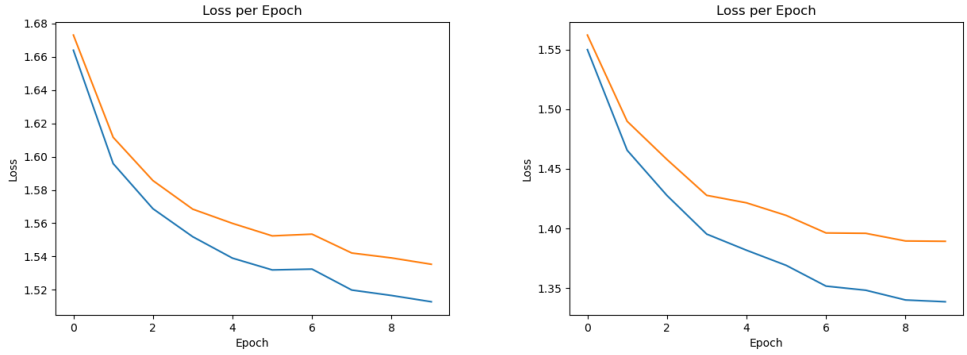
(b) Training and validation loss of the baseline LSTM model.

Figure 1: Baseline model performance on the Shakespeare text dataset.



(a) Training and validation loss of the RNN model with sequence length 128.

(b) Training and validation loss of the LSTM model with sequence length 128.



(c) Training and validation loss of the RNN model with sequence length 512.

(d) Training and validation loss of the LSTM model with sequence length 512.

Figure 2: Performance with increased sequence length on the Shakespeare text dataset.

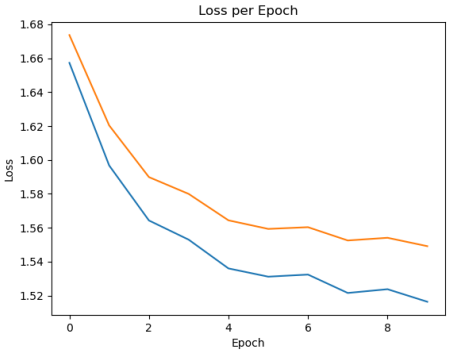
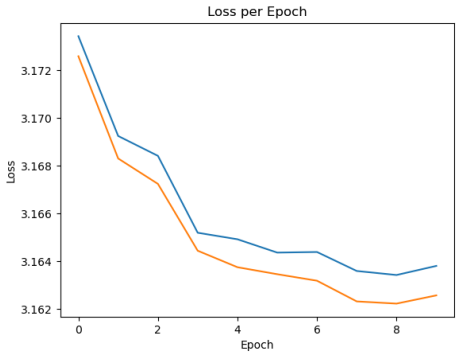
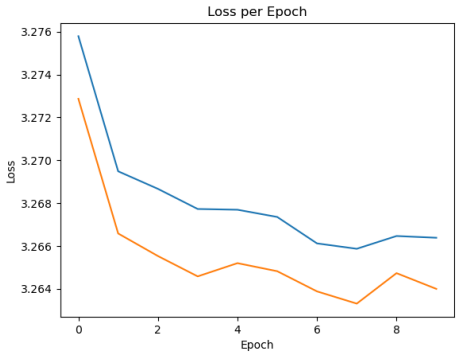


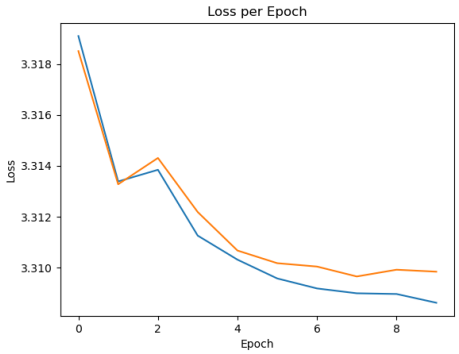
Figure 3: Training and validation loss of the LSTM model with sequence length 128 and 300 hidden neurons.



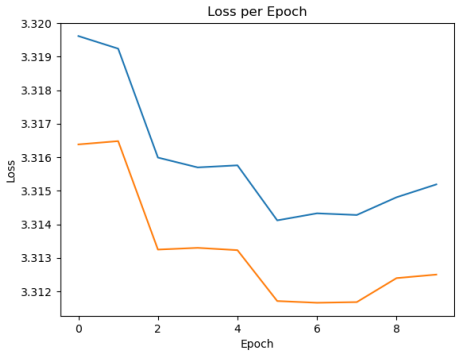
(a) Training and validation loss of the LSTM model with sequence length 4 and no teacher forcing.



(b) Training and validation loss of the LSTM model with sequence length 8 and no teacher forcing.

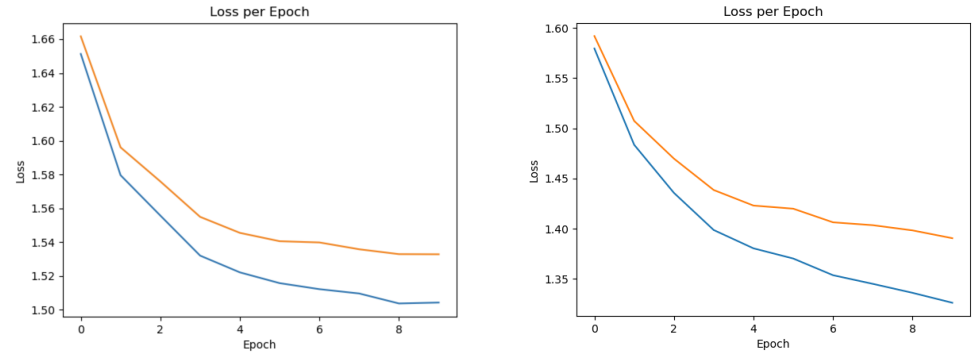


(c) Training and validation loss of the LSTM model with sequence length 16 and no teacher forcing.



(d) Training and validation loss of the LSTM model with sequence length 32 and no teacher forcing.

Figure 4: Performance without teacher forcing on the Shakespeare text dataset.



(a) Training and validation loss of the 2 layer RNN model.

(b) Training and validation loss of the 2 layer LSTM model.

Figure 5: Baseline model performance on the Shakespeare text dataset.

```
The sectual all, and bring them to me.

First Citizen:
Have you art the suprement, where is the word,
Where saw the sunster of stew the lands.

RIVERS:
Why, then, the senses are in a fair compance
That I mean to shall succession to his power
And she will not speak to make all the field.

TRANIO:
Where is my father, thou wilt say that should speak,
That he is all the scares, the sake is all the country,
To make we seem to the cold, when the people
And speaks himself and his brown should be too.

BIANCA:
No, sir, a word to the last, and forthwith their brother
will stay to the people of the house.

POMPHIO:
I pray you, my lord.

BRAKENBURY:
I do beseech you to you, good my lord?

CATESBY:
My lord, where is the people!

MENENIUS:
Has not so far then, here is the state of the world
To them as the truth, where by the very strong
Of the senate blessed but their leaves,
And back to see their last, and
```

- Temperature = 1:

```
Macbeth
 by William Shakespeare
```

```
 Edited by Barbara A. Mowat and Paul Werstines,
do all displeased with a spirit.

DUKE VINCENTIO:
Thou shalt, therefore.'

BUSHY:
I have offered!

First Lunce:
Then, God!
Then, took; and seem'd not wround itself in many master,
If they leived so. Neise is strange; and, I promive,
Therefore then the champiname have stell as
An man would know thou art put should have,
My noble wit not bive as at himself
To anon world: strike us to-day! Thou canst that!
O, let their best two?

SICINIUS:
Yes, O!
My heart with some staals, look to
comfort, whose fathal blashes would do note pain:
The breth, let me hold, madam: God, sir,
For a horse! Where is the fight by.
If he, Warwick! Bones, chastians then,
Which shall have, so it shortly, when I shall
queen and wenting so, but speaks entold; what made me hither?
My boy-heaveness!' defoly queen, let to spoil
That nock not have real men.

BRUTUS:
Yes do harm, yet, my old moants! the villain will truck,
so--whom you,--goes of the waters bear
That that's him and to grant, but the foul,
My heart is I f
```

- Temperature = 2:

```
Macbeth
 by William Shakespeare
 Edited by Barbara A. Mowat and Paul Werstiness,.
F,--quot?, -entle, beind! sho yitas,,
NuEsClow; now
hidswealyyy's witrain word woewe deser-dmmid'st
Trus'ses! nelve its
Was, howonescly ryKne?
Zehen unCl, and, expeessions,
noon-daci'ng, it.io, heaven.
'halshly Vo-bicainVs?  revis Maste!
tAy: JoH, pucy weak, comf!
You'll amakmany are prMXon UYe thitKeyn
it namel;
peis'd
Frty mbaoun mal me- 'Visten o'! Assetm
My asht! I am!neth out; 'Whelco?'kpen foot!
Nay, lav,'? hugh Pumpoinot-side!- asseste,
Viravonius licar'd; no; ouc a cholturd;Bine,s.'
```

```
QUEEN;
BliM!y.
Tunl yor's uintof's ejoacheht my did. Fare we not!
Celdered,
Vosue, likeD he! Gdo SymaS
So!made hou t,ry:-ybaciefy foot Cllorra! Acmzet'm,
Julit; chafedbluy,-bless he is; Mjewekn'st spead!'
kisteed Glour, not,
'Ey lew gloscling, darel, I inj,ctain,
PMyngbmice!' Scrly fave,l Liguar
Plum, wus with, sgormilocany all.

GObed:
Ther?

QUREET:
Hot swhe-Pad. WhRet Sweel, Ellib:; poot did!s
Verhold'st blowkBy; Is he, Georgily,
If, if my choken, syour York's users is
lacibazily; Brutf.
Lu
```

5 Discussion

The LSTM models consistently outperformed the RNN models in terms of test loss, particularly when teacher forcing was applied. As shown in Table 1, the baseline LSTM model achieved a lower loss of 1.4054 compared to the baseline RNN's loss of 1.5309. Increasing the sequence length further improved performance for the LSTM, with the sequence length of 512 having the lowest loss of 1.3773. Then, increasing the hidden layer size in the LSTM further reduced loss to 1.3237 at sequence length 128 resulting in our best model, indicating that model capacity plays a significant role in capturing long-range dependencies.

Similarly, the RNN models were more performant with larger sequence length, though not achieving results as strong as the LSTM models. The strongest RNN with the base architecture achieved a loss of 1.5267, with a sequence length of 512.

The results also highlight the impact of teacher forcing. Removing teacher forcing significantly degraded LSTM performance, with a sequence length of 8 yielding a much higher loss (3.1266) compared to the teacher-forced models. At greater sequence lengths, the loss further increased. This suggests that the model struggles to generate sequences without being conditioned on previous ground-truth tokens, likely due to accumulated errors over time. Since we only calculate loss on the final token of each sequence, any previous incorrect tokens are not well penalized and the model thus fails to train meaningfully. This is clearly displayed in Figure 4, with the longer sequence lengths having hardly any decrease in loss over the 10 training epochs compared to the models with teacher forcing.

The generated text with a temperature of 0.5 is highly structured and grammatically sound, closely resembling Shakespearean writing. Sentences maintain logical coherence, though they exhibit some repetition and lack expressive diversity. Character dialogues remain readable, but the model heavily favors high-probability word choices, making the text mechanical in tone. While linguistically stable, this setting produces text that is too rigid and lacks spontaneity. With the middle temperature, the model strikes a balance between coherence and creativity. The text retains grammatical correctness while incorporating lexical diversity. Some invented phrases emerge, enriching the narrative without significantly compromising readability. The generated dialogue maintains a Shakespearean tone with moderate unpredictability, making it the most aesthetically and structurally compelling setting. Finally, at the highest temperature of 2, the text becomes highly erratic and largely unintelligible. The model generates nonsensical word combinations, deviating from conventional linguistic structures. The excessive randomness destabilizes the generative process, making the text impractical for literary applications. Overall, the middle temperature produces the most optimal results, balancing

grammatical coherence and creative variation. It ensures a dynamic yet intelligible output, preserving the model's ability to generate text that is both readable and stylistically compelling.

Finally, increasing the number of layers to 2 while reducing the hidden layer size to 100 resulted in a slightly lower test loss for the RNN model (1.5242) and a more significant improvement for the LSTM (1.3857) model. These results suggest that LSTMs benefit more from increased depth, whereas traditional RNNs exhibit marginal improvements due to vanishing gradients. Thus, increasing the number of layers while keeping parameters constant benefits LSTMs more than RNNs, reinforcing the importance of gating mechanisms in deeper recurrent architectures.

6 Contributions

6.1 Anthony Tong

Anthony implemented the baseline models, dataloading, and the training pipeline. He also ran the 512 sequence length experiments, and contributed to discussions about the results. He wrote the abstract, introduction, related work, and proofread and edited the final report.

6.2 Brandon Park

Brandon completed and debugged the models and training pipeline implementation, and implemented the model without teacher forcing. He also ran the majority of training runs and experiments, and wrote the methods, results, and discussion sections.

6.3 Chi Zhang

Chi implemented, ran the experiments, and wrote the relevant portions of the report for the RNN and LSTM models with increased hidden layers. Chi actively contributed to discussions and analysis of the results, and also aided in proofreading the final report.

6.4 Christopher Rebollar-Ramirez

Christopher assisted with experimentation and bugfixing with Chi, and wrote and proofread parts of the report. Christopher also contributed to conversations about different experiments in particular and pointed out various implementation details that needed to be refined.

7 References

References

[1] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks, 2015.

[2] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[3] Felix Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural Computation*, 12:2451–2471, 10 2000.

[4] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 06(02):107–116, 1998.

[5] Andrej Karpathy. char-rnn. `https://github.com/karpathy/char-rnn`, 2015.

[6] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning internal representations by error propagation*, page 318–362. MIT Press, Cambridge, MA, USA, 1986.

[7] Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280, 1989.