NLU+ 2025 Assignment 1

2 - a)

The model was initially evaluated using the given parameters and the default values for anneal (5), batch size (100) and epochs (10), as shown in Table 1. In these experiments, a learning rate of 0.5 outperformed both 0.1 and 0.01, using 50 hidden units resulted in a lower loss than nearly all experiments with 25 hidden units, and the number of back steps caused minimal variation in the loss, with the largest occurring between 0 and 2 steps. From these experiments, **50 hidden units, 0.5 learning rate** and **0 back steps** presented the best performance, with a **loss of 4.973139** and a **perplexity of 144.480**.

Hidden units	Learning rate	Back steps	Best Loss	Perplexity
25	0.5	0	5.016976	150.954
25	0.5	2	5.035141	153.721
25	0.5	5	5.018009	151.110
25	0.1	0	5.218973	184.744
25	0.1	2	5.214506	183.921
25	0.1	5	5.214641	183.946
25	0.05	0	5.407678	223.113
25	0.05	2	5.403825	222.255
25	0.05	5	5.403781	222.245
50	**0.5**	**0**	**4.973139**	**144.480**
50	0.5	2	5.029222	152.814
50	0.5	5	5.021099	151.578
50	0.1	0	5.132932	169.513
50	0.1	2	5.136560	170.130
50	0.1	5	5.136824	170.174
50	0.05	0	5.312958	202.950
50	0.05	2	5.314184	203.199
50	0.05	5	5.314224	203.207

Table 1: Performance for suggested parameter settings.

Following this result, combinations of batch size (1 - 100) and hidden units (30 - 70) were tested, as well as pairs of epochs (20, 30, 40) and anneal (0, 2, 5). Fixing their best results - **50 hidden units, batch size 5, anneal 2** and **30 epochs** -, a higher **learning rate of 0.8** was tested, outperforming 0.5. With this new value, the model was trained using 0 to 8 back steps, achieving the best loss of **4.530544** for **3 back steps**. These results are in the Appendix.

The model is being trained on a small dataset (1000 sentences). This explains why 50 outperformed higher numbers of hidden units, as a large amount of parameters leads to excessive overfitting, and lower values such as 25 and 30 hidden units, as these are not enough to effectively represent the data. Additionally, a smaller batch size like 5 performs better because it allows for more weight updates per epoch, and because it relies on fewer samples, it also introduces noise in the estimated gradients, improving generalization and avoiding sharp minima.

Furthermore, a higher learning rate like 0.8 enables larger updates, preventing the model with a small batch size from getting stuck in sharp local minima due to noise, ensuring it still moves towards convergence. The anneal value 2 effectively reduces the learning rate per epoch, to avoid overshooting the minima in later epochs, unlike higher anneal values like 5, which result in smaller adjustments and a higher learning rate in later stages, hindering convergence.

Finally, a small value of 3 back steps balances the RNN's ability to capture prior information (compared to 0 or fewer steps) with its inability to capture long-range dependencies effectively, due to vanishing gradients, which causes higher step values, such as 5 or 8, to perform worse.

2 - b)

After training the model with our best set of parameters from the previous question - hidden units: 50, learning rate: 0.8, back steps: 3, number of epochs: 30, anneal value: 2, batch size: 5 -, we evaluated it on the test set, achieving the following metrics:

- Mean Loss: 4.026935 (rounded from 4.026935277103141)
- Perplexity: 56.089

Question 3 - d

Hidden Layers	Avg per Epoch (sec)		Final Loss		Final Accuracy	
	RNN	GRU	RNN	GRU	RNN	GRU
10	0.75	1.60	0.6161	0.5917	0.659	0.672
25	1.70	3.63	0.6097	0.5467	0.688	0.734
50	1.57	4.00	0.5975	0.5233	0.687	0.751

Table 2: Comparison of RNN and GRU Performance

As per table 2, below is the comparison between the two models:

- **Computation Time:** The GRU model required significantly more time per epoch to train as compared to the RNN model. This difference increased with the increase in the number of hidden dimensions. This increased computation time is due to the presence of the extra gated units (reset and update) in the GRU model. Both of these gates require extra computation, thereby increasing the total time to train the model.

- **Best Loss:** The GRU model achieves a lower loss across all configurations compared to the RNN model. Initially, both models start with similar losses, but the GRU reduces loss more effectively over epochs. The improvement in loss reduction is due to the gating mechanism and superior activation functions in GRU. While RNN use only sigmoid activations, which restrict hidden state values between 0 and 1, GRU leverages both sigmoid (gates) and tanh (candidate hidden state), enabling a wider and more expressive range (-1 to 1). This richer transformation allows GRU to extract better input representations even when no past dependencies are considered. Additionally, GRU regulates gradient flow through the update gate, preventing excessive updates and stabilizing early training. RNN, lacking this regulation, produce noisier and less effective weight updates, leading to higher initial loss.

- **Accuracy:** GRU achieves higher accuracy than the RNN model across all configurations of hidden dimensions. The superior accuracy of the GRU can be explained by its more robust architecture.

- **Effect of Hidden Dimensions:** As we increase the number of hidden dimensions the performance of both the models increase because with more neurons, we are able to capture more complex patterns and therefore reduce the loss. However, beyond a certain point, the improvements become marginal because excessive dimensions allows the model to remember redundant information and may even lead to overfitting. Also, the computation cost increases significantly with the increase in the hidden dimensions, especially for GRU because of the increased computation of additional gates. For this task, we observe that while increasing the hidden dimensions improves accuracy, the gains become less significant as we move from 25 to 50 hidden layers, suggesting an optimal range for maximizing performance while keeping computation feasible.

Question 4

Hypothesis

Model Behaviours: As Backpropagation Through Time (BPTT) steps grow, the RNN will increasingly suffer from vanishing gradients, whereas the GRU can mitigate this and maintain stronger long-term dependencies due to the presence of its gating mechanism. Both models will improve within small BPTT ranges (1–5), but while the RNN will stagnate beyond this range, the GRU will continue to show gains up to a higher BPTT step. Unlike the GRU, we don't expect the RNN to show loss deterioration at higher BPTT values, only stagnation, as there will essentially be no weight updates.

Training Time & Computation: We expect to see an increased training time for both models at higher BPTT steps because of deeper unrolling, but GRU requires more time per step due to additional gating overhead, making their training-time increase more pronounced.

The **impact of the hyperparameters** as the BPTT steps increase is expected to be:

- For **training size**, **epochs** and **hidden dimensions**: both models will benefit from an increase of these parameters at lower BPTT steps. More training data, epochs and hidden dimensions should improve short-term dependencies for both RNN and GRU. However, increasing the hidden dimensions or epochs beyond a certain point can lead to overfitting or learning noise.
  As BPTT grows, the models will act differently:
  - The RNN model is expected to stagnate because, regardless of changes in these hyperparameters, their impact is nullified by vanishing gradients.
  - Conversely, the GRU will continue to see performance improvement at higher BPTT steps because its gating mechanism allows it to use the extra training data, the additional updates (epochs) and the higher capacity for learning patterns (more hidden units) effectively, by mitigating the problem of vanishing gradients.
- For **batch size** and **learning rate**, we expect their effect, on training stability and convergence speed, to be independent of the BPTT steps. A well-tuned learning rate should ensure efficient training and a smaller batch size should lead to improved performance in both models due to more frequent updates and enhanced generalization for both models.

Experimental Setup

To validate our hypothesis, we conduct controlled experiments by varying the Backpropagation Through Time (BPTT) steps while adjusting one hyperparameter at a time.

Experiment 1: Vanishing Gradient Analysis

Since vanishing gradients are a key limitation of the RNN at high BPTT, this experiment tests whether RNN gradient norms decay while the GRU maintains stable gradients due to its gating mechanisms.
Setup: We fix *Hdim = 50, learning rate = 0.5, batch size = 100, epochs = 10, training data = 10K* and the gradient norm of ΔU is tracked for BPTT (1–20).

Experiment 2: General Performance Validation

Since the RNN is expected to stagnate at higher BPTT while the GRU continues improving, this experiment aims to track how loss, accuracy, and training time evolve with increasing

BPTT. It provides an overall comparison of model performance across different BPTT steps. **Setup:** We fix *Hdim = 50, learning rate = 0.5, batch size = 100, epochs = 10, training data = 10K* and track loss, accuracy, and computation time for BPTT (1–20).

Experiment 3: Effect of Hyperparameters

Used to analyse the effects of different hyperparameters on models at different BPTT steps *Setup:* We test one hyperparameter (training size, epochs, hidden dimensions, batch size and learning rate) at a time by keeping everything else constant and just varying that particular parameter across BPTT (1–15).

Results

Experiment 1: Vanishing Gradient Analysis

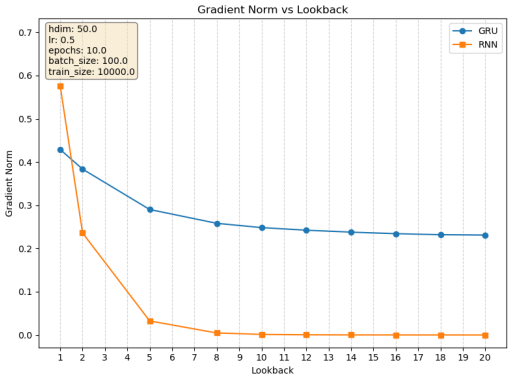
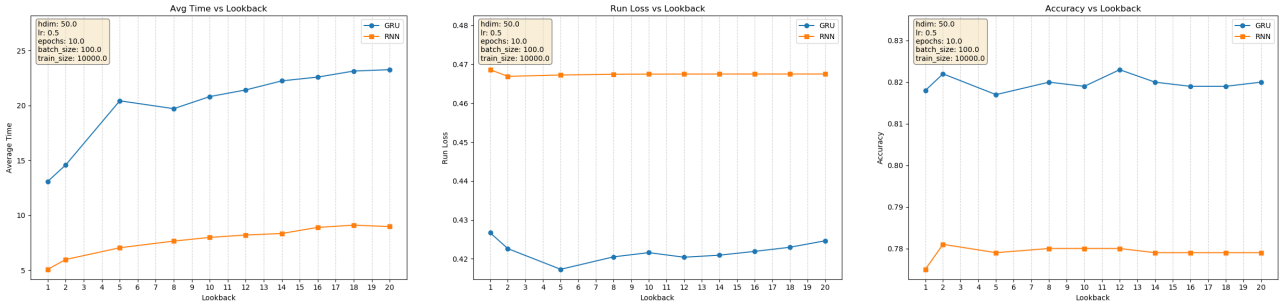


Figure 1: Gradient Norm vs. Lookback (BPTT) for RNN and GRU.
(Hdim = 50, lr = 0.5, epochs = 10, batch_size = 100, train_size = 10K)

Analysis:
Figure 1 illustrates the gradient norm of ΔU (y-axis) as the lookback (BPTT steps) increases from 1 to 20 (x-axis). The **RNN** (orange) shows a rapid decrease in gradient norm, approaching near-zero values by around 5-10 lookback steps. This behaviour indicates a strong *vanishing gradient* effect in RNN architecture, limiting its ability to learn long-range dependencies. In contrast, the **GRU** (blue) maintains a relatively higher and more stable gradient norm across all lookback steps, demonstrating that its *gating mechanisms* mitigate vanishing gradients. These findings align with our hypothesis about both models' architecture.

Experiment 2: General Performance Validation



(a) Avg Time vs. Lookback (b) Run Loss vs. Lookback (c) Accuracy vs. Lookback

Figure 2: RNN (orange) vs GRU (blue) performance across different BPTT steps.
(Hdim = 50, lr = 0.5, epochs = 10, batch_size = 100, train_size = 10K)

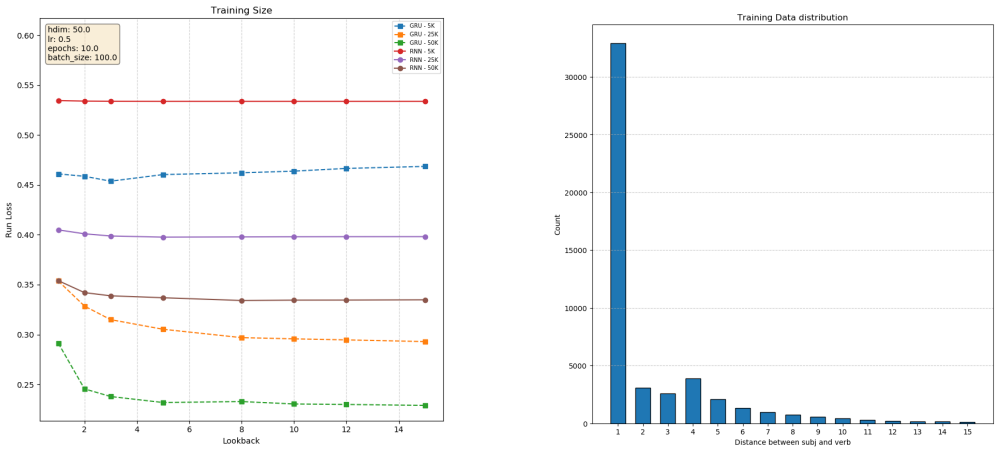
Analysis:

- **Training Time (Figure 2a).** The GRU consistently takes longer time to train per epoch than the RNN, and both models' training times increase with higher BPTT. This was expected as per the hypothesis because longer unrolling leads to more computations. The additional overhead in the GRU arises from its gating mechanisms, which require extra parameter updates compared to the simpler RNN architecture.
- **Run Loss (Figure 2b).** The RNN's loss remains flat after 2-3 BPTT steps (around 0.467), indicating it does not effectively leverage increased BPTT. In contrast, the GRU shows a noticeably lower loss (dropping to around 0.415) at around BPTT 5. Although the GRU loss slightly fluctuates at higher BPTT, it remains well below the RNN loss. This fluctuation shows that the GRU model is still trying to learn because of the presence of some meaningful gradient at even higher BPTT steps. This aligns with the findings from **Experiment 1 (Figure 1)**, where the RNN gradients vanish rapidly at BPTT > 5, making the performance stagnant at higher BPTT steps.
- **Accuracy (Figure 2c).** The RNN accuracy hovers around 0.78 and does not improve with more lookback steps. Meanwhile, the GRU achieves higher accuracy (around 0.82), demonstrating that it continues to benefit from additional BPTT steps. Its gating mechanism allows it to capture longer dependencies without suffering from vanishing gradients, thus translating into higher accuracy. Additionally, the accuracy in the GRU is higher at 12 BPTT steps than at where the loss was minimal (5 steps). This apparent discrepancy between loss and accuracy might be explained by the class imbalance present in the training data, where the count of class 0 (34151) is much grater than the count of class 1 (15849). For this reason, accuracy is not a reliable performance metric. Therefore, we will use loss in future experiments.

Overall, these results confirm our hypothesis that the **RNN stagnates at higher BPTT** steps due to vanishing gradients, as evidenced in Experiment 1, while the **GRU leverages increased lookback** to reduce loss and improve accuracy, at a higher computational cost. Thus, the GRU achieves **better performance** in terms of both loss and accuracy, but **requires more training time** compared to the RNN.

Experiment 3: Effect of Hyperparameters

Effect of Training Size



(a) Run Loss vs. Lookback (BPTT) for different training size

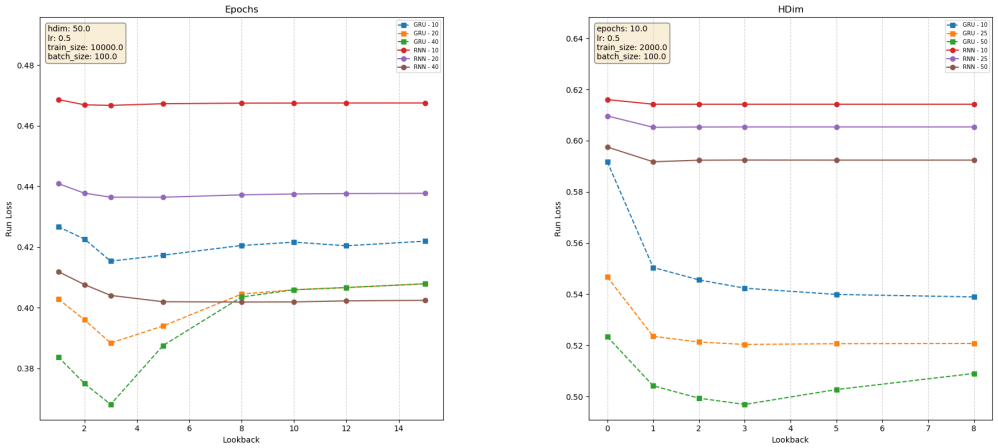
(b) Count vs Distance between subj and verb.

Figure 3: Run Loss for varying training size and training data distribution

Analysis:
Figure 3a compares the **RNN** (solid) and **GRU** (dashed) run loss as *lookback* (BPTT) increases from 1 to 15 under training sizes of 5K, 25K, and 50K. Larger training sets (e.g., 50K vs. 5K) provide an initial loss advantage for both models, but the RNN stagnates beyond a lookback of roughly 3–4. In contrast, the GRU improves more substantially with bigger datasets, maintaining a gradual loss decline up to lookback 14, thanks to its update and reset gates. The limited gains can be explained by the data distribution, as Figure 3b shows few instances with subject–verb distances beyond 7.

Overall, these findings support our hypothesis: the **GRU outperforms the RNN** at higher BPTT steps with sufficient training data, as **vanishing gradients restrict RNN gains** beyond short lookbacks, whereas the GRU **leverages larger datasets** more effectively over a wider range of BPTT steps.

Effect of Epochs and Hidden Dimensions



(a) Run Loss vs. BPTT - Epochs (b) Run Loss vs. BPTT - Hdim

Figure 4: Run Loss for varying epochs (10, 20, 40) and Hdim (10, 25, 50) across BPTT steps

Analysis:
Our hypothesis expected GRU to improve at longer lookback steps, while RNN would plateau due to short-range dependencies. As seen in Figure 4a, RNN followed the expected curve, improving at low BPTT (2–3 steps) but stagnating at higher values.

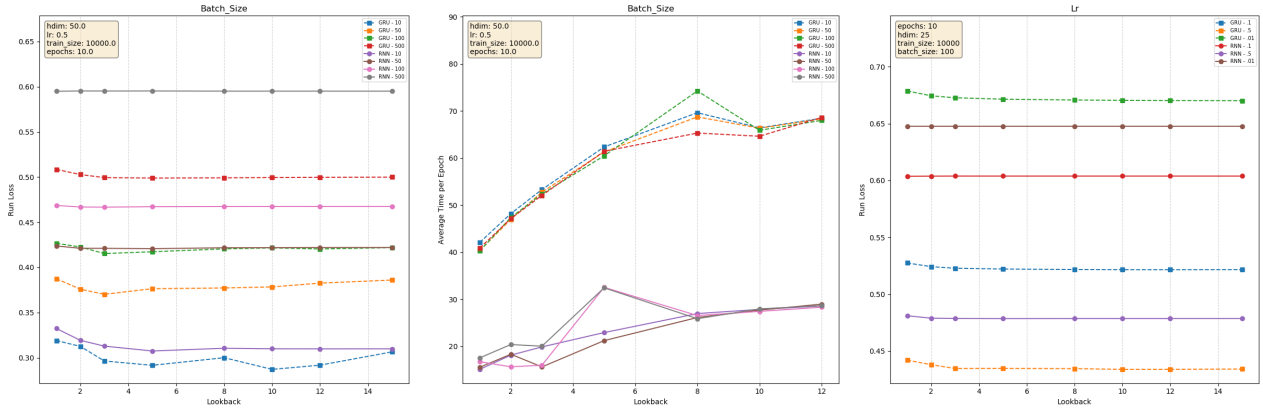
However, contrary to our expectations, the GRU exhibited increased loss at high BPTT (10–15 steps) when trained for more epochs. This could have happened due to **insufficient training data**. For this experiment, the GRU probably overfits by memorizing noise instead of learning meaningful long-term dependencies, which degrades its generalisation capability. Thus, while the GRU is theoretically better at leveraging additional training for long-range dependency learning, excessive epochs in the context of high BPTT actually push the model into an overfitting regime, highlighting the need for careful tuning.

Figure 4b shows how run loss varies with hidden units, revealing that:

- Despite limited training data, larger hidden layers improve both RNN and GRU, leading to lower final loss.
- While RNN performance remains stable across hidden dimensions, GRU benefits for short BPTT (1–3 steps) but plateaus or deteriorates beyond this. This may be due to inadequate training data or suboptimal hyperparameters, causing it to learn noise.

Nonetheless, GRU can exploit larger hidden dimensions for long-term dependencies if properly tuned with sufficient data.

Effect of Batch Size and Learning Rate



(a) Run Loss vs. Lookback (BPTT) - Batch Size

(b) Avg. Time vs. Lookback (BPTT) - Batch Size

(c) Run Loss vs. Lookback (BPTT) - Learning rate

Figure 5: Run Loss and Avg. Time vs. Lookback for varying Batch Size and Learning Rate

Analysis:

Batch Sizes affects training stability, Not BPTT Trends: Figure 5a shows that as batch size decreases (e.g., from 500 to 10), the RNN's loss transitions from flat to improving with more BPTT steps, with a similar but less pronounced effect in the GRU. However, this does not imply that BPTT steps depend on batch size. Instead, the difference arises because larger batch sizes result in less frequent updates, preventing convergence to local optima and limiting the model's ability to exploit additional steps. Thus, the model's behaviour with increasing steps remains unchanged but is constrained by a suboptimal batch size. We believe this supports our hypothesis that while batch size does not directly affect changes in BPTT steps, more frequent, though noisier, gradient updates can often improve generalization.

Learning Rate Affects Convergence, Not BPTT Trends: The absolute loss values shift depending on how quickly (and stably) the models converge, yet the shape of the loss curves over BPTT steps remains similar. This indicates that while an extreme learning rate can lead to under- or overshooting, it does not fundamentally change how RNN or GRU benefit from increased lookback.

Overall, these results align with our hypothesis that learning rate and batch size influence convergence speed and stability, and while a poorly tuned model can obscure performance differences, it does not override the core architectural behaviours of RNN and GRU when varying BPTT depth.

1 Appendix

Batch Size	Best Loss for each Hdim				
	30	**40**	**50**	**60**	**70**
1	**4.69104**	4.71508	4.72454	4.69651	4.75257
5	**4.59784**	**4.59034**	**4.58357**	4.61982	4.61789
10	4.60293	4.59646	4.61088	**4.59234**	**4.58015**
25	4.65264	4.66751	4.66319	**4.64768**	4.65437
50	4.75839	4.78395	**4.71440**	**4.72342**	4.75162
100	4.83429	4.83374	**4.81340**	4.83882	4.83856

Table 3: Performance for 30-70 Hidden Dimensions and 1-100 Batch Sizes. Other Parameters: 0.5 Learning rate, 0 back steps, 20 epochs and anneal 0. (Hdim = hidden units)

(a) For Anneal 0 and 5

Anneal	Epochs	Batch size	Best Loss
0	20 40	1	4.724542
	20 40	5	4.583569
	20 40	10	4.610882
	20	25	4.663190
	20	50	4.742849
	20	100	4.811522
5	30	5	4.556446

(b) For Anneal 2

Epochs	Batch size	Best Loss
20 40	1	4.587614
20	5	4.560959
20	10	4.656687
20	25	4.775460
20	50	4.874701
30 40	**5**	**4.547022**
40	10	4.626713

Table 4: Performance for different anneal values (0, 2 and 5) across various batch sizes and epochs. Other Parameters: 50 hidden units, 0.5 learning rate, 0 back steps.

Hidden Units	Batch size	Epochs	Anneal	Best Loss
70	10	30	2	4.634257
50	**5**	30	2	**4.547022**

Table 5: Performance for 30 epochs and anneal 2, comparing (70,10) and (50,5) hidden dimensions and batch size pairs. Other Parameters: 0.5 learning rate and 0 back steps.

Lr. rate	Epochs	Batch size	Best Loss	Perplexity
0.8	20	5	4.550939	94.721
0.8	20	10	4.649094	104.490
0.8	**30** 40	**5**	**4.544536**	**94.117**
0.8	40	10	4.618916	101.384

Table 6: Performance for 0.8 learning rate, 20 and 40 epochs, and batch sizes 5 and 10. Other Parameters: 50 hidden units, 0 back steps, anneal 2.

Back steps	Best Loss	Perplexity
0	4.544536	94.117
1	4.537238	93.432
2	4.532099	92.954
3	**4.530544**	**92.809**
4	4.533967	93.127
5	4.536860	93.397
8	4.546472	94.299

Table 7: Performance for 2, 3, 4, 5, and 8 back steps. Other Parameters: 50 hidden units, 0.8 learning rate, 30 epochs, anneal 2, batch size 5.

Hidden units	Lr. rate	Back steps	Epochs	Anneal	Batch size	Best Loss	Perplexity
75	0.5	0	20	0	10	4.609320	100.416
75	0.5	0	20	0	25	4.646521	104.222
75	0.5	0	20	0	50	4.719158	112.074
75	0.5	0	20	2	25	4.778700	118.950
75	0.5	0	20	2	50	4.885484	132.354
75	0.5	0	20	7	10	4.587034	98.203
75	0.5	0	20	7	25	4.694509	109.345
75	0.5	0	20	7	50	4.791502	120.482
75	0.5	0	**30**	0	10	4.609320	100.416
75	0.5	0	30	0	25	4.607160	100.199
75	0.5	0	30	0	50	4.679813	107.750
75	0.5	0	30	2	10	4.628137	102.323
75	0.5	0	30	2	25	4.756103	116.292
75	0.5	0	30	2	50	4.861926	129.273
75	0.5	2	30	2	5	4.540281	93.717
50	0.8	3	30	3	5	4.534073	93.137
55	0.8	3	30	2	5	4.531990	92.943
50	0.5	0	20	0	200	4.945752	140.577
50	0.5	0	20	0	100	4.811522	122.919
50	0.5	0	20	0	50	4.742849	114.761
50	0.5	0	20	0	25	4.663190	105.974
50	0.5	0	20	0	10	4.610882	100.573
50	0.5	0	20	0	5	4.583569	97.863
50	0.5	0	20	0	1	4.724542	112.679
50	0.5	0	20	5	100	4.900470	134.353

Table 8: Performance (best cross-entropy loss, rounded) for our own parameter settings

Hidden units	Lr. rate	Back steps	Epochs	Anneal	Batch size	Best Loss	Perplexity
50	0.5	0	30	0	10	4.610882	100.573
50	0.5	0	30	0	25	4.604723	99.955
50	0.5	0	30	0	50	4.700068	109.955
50	0.5	0	**40**	0	1	4.724542	112.679
50	0.5	0	40	0	5	4.583569	97.863
50	0.5	0	40	0	10	4.610882	100.573
50	0.5	0	40	**2**	1	4.587614	98.260
50	0.5	0	40	**2**	5	4.547022	94.351
50	0.5	0	40	**2**	10	4.626713	102.178
50	0.5	0	20	2	10	4.656687	105.287
50	0.5	0	20	2	25	4.775460	118.565
50	0.5	0	20	2	50	4.874701	130.935
50	0.5	0	20	2	1	4.58761369	98.260
50	0.5	0	20	2	5	4.560959	95.675
50	0.5	0	20	5	50	4.800834	121.612
50	0.8	0	20	0	1	4.851276	127.904
50	0.8	0	20	0	5	4.617512	101.242
50	0.8	0	20	0	10	4.602601	99.743
50	0.8	0	20	2	1	4.657802	105.404
50	0.8	0	20	2	5	4.550939	94.721
50	0.8	0	20	2	10	4.649094	104.490
50	0.8	0	40	0	1	4.851276	127.904
50	0.8	0	40	0	5	4.617512	101.242
50	0.8	0	40	0	10	4.602601	99.743
50	0.8	0	40	2	1	4.657802	105.404
50	0.8	0	40	2	5	4.544536	94.117
50	0.8	0	40	2	10	4.618916	101.384
50	0.8	2	30	2	5	4.532099	92.954
50	0.8	**3**	30	2	5	**4.530544**	**92.809**
50	0.8	4	30	2	5	4.533967	93.127
50	0.8	5	30	2	5	4.536860	93.397
50	0.8	8	30	2	5	4.546472	94.299

Table 9: Performance (best cross-entropy loss, rounded) for our own parameter settings

Hdim	Epochs	**Batch size**	Best epoch	**Best loss**	Perplexity
30	20	1	6	4.72206	112.399
30	20	**5**	12	**4.55904**	95.492
30	20	10	18	4.58415	97.920
30	20	25	17	4.67101	106.806
30	20	50	19	4.75551	116.222
30	20	100	19	4.89249	133.285
40	20	1	5	4.69664	109.578
40	20	**5**	15	**4.57082**	96.623
40	20	10	17	4.59162	98.654
40	20	25	20	4.66344	106.0
40	20	50	14	4.78704	119.946
40	20	100	15	4.89007	132.963
50	20	1	7	4.71754	111.892
50	20	5	9	4.57501	97.029
50	20	**10**	18	**4.56059**	95.640
50	20	25	17	4.64378	103.937
50	20	50	17	4.73437	113.792
50	20	100	19	4.84034	126.512
60	20	1	3	4.73102	113.411
60	20	**5**	14	**4.57350**	96.883
60	20	10	20	4.57517	97.004
60	20	25	19	4.65290	104.888
60	20	50	15	4.77232	118.194
60	20	100	20	4.85918	128.919
70	20	1	6	4.73204	113.527
70	20	**5**	12	**4.56889**	96.437
70	20	10	19	4.56980	96.524
70	20	25	18	4.66454	106.117
70	20	50	16	4.95667	142.120
70	20	100	20	4.90351	134.762

Table 10: Performance for 20 epochs, anneal 0, and batch sizes between 1 and 200. Other Parameters: 50 hidden units, 0.5 Learning rate, 3 back steps, anneal 0.