CIS530 HW6 Writeup Template

Neural Language Models: City Classification Experiments

Task 2.1: Learning Rate Tuning (6 points)

Experimental Setup

I trained RNN models for city classification using different learning rates while keeping other hyperparameters constant:

- **Base hyperparameters**: hidden_size = 10, n_epochs = 50000
- **Learning rates tested**: {0.0002, 0.002, 0.02}

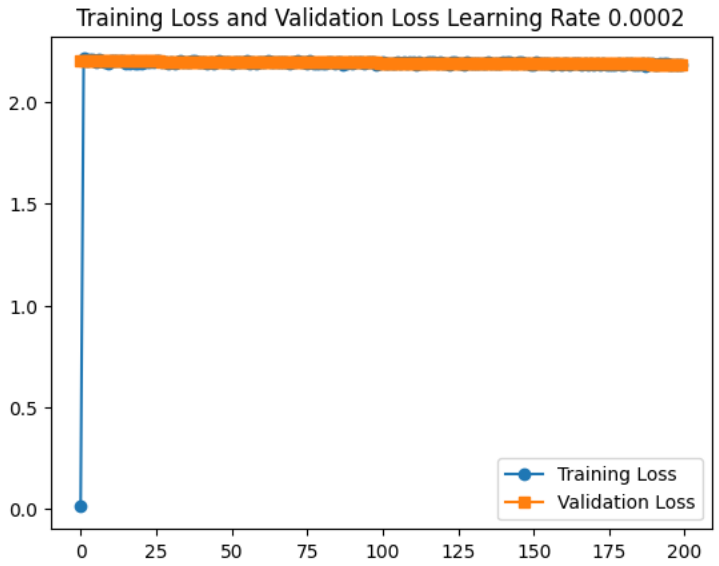
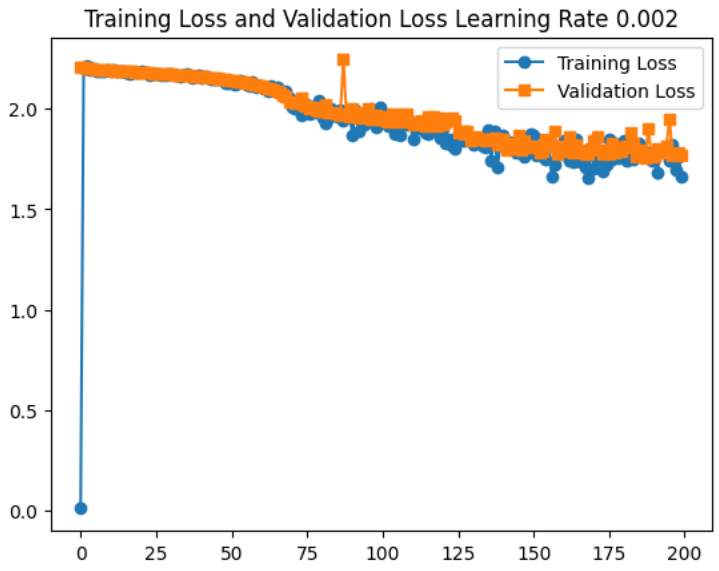
Results

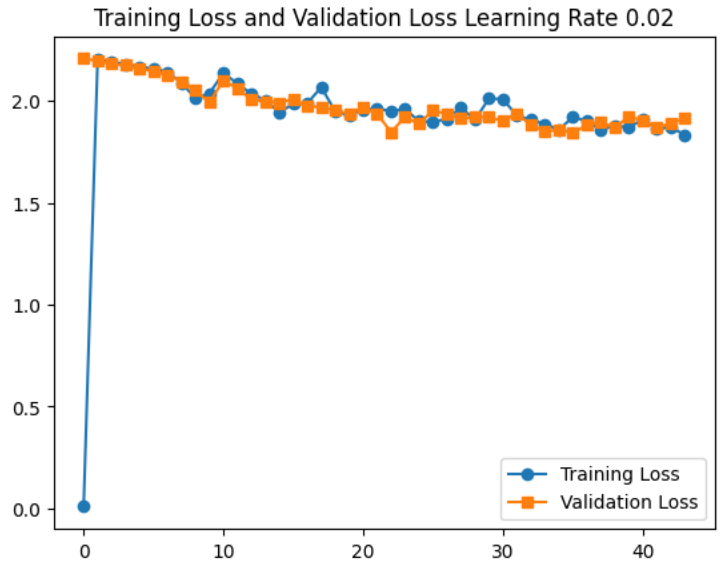
Learning Rate	Validation Accuracy
0.0002	0.13555555555555557
0.002	0.3422222222222222
0.02	0.1111111111111111

Training and Validation Loss Plots

[Include 3 plots showing training/validation loss curves for each learning rate]

Learning Rate	Training Loss and Validation Loss Plot

0.0002	 Training Loss and Validation Loss Learning Rate 0.0002 2.0 1.5 1.0 0.5 0.0 0 25 50 75 100 125 150 175 200 Training Loss Validation Loss
0.002	 Training Loss and Validation Loss Learning Rate 0.002 2.0 1.5 1.0 0.5 0.0 0 25 50 75 100 125 150 175 200 Training Loss Validation Loss

0.02	

Observations and Discussion

[Analyze your experimental results and discuss your findings. Consider topics such as how learning rate affects convergence speed, final accuracy, training stability, any issues like NaN losses or divergence, and which learning rate performed best and why. Focus on the patterns you observe in your specific results.]

For these tests, I have used the same settings as others, hidden_size = 10 n_epochs = 50000.

When the learning rate is 0.02, the validation accuracy is lowest which is 11.11%. This accuracy is the same to the learning rate as 0.01. Which means 0.02 and 0.01, the learning rate is too high, the model cannot converge.

When the learning rate is 0.002, the validation accuracy is highest as 34.22%

When the learning rate is 0.0002, the validation accuracy is 13.55%.

Why learning rate 0.0002 has lower accuracy when comparing to 0.002, it's because training might be too slow and the model might stuck in a local minimum.

So the learning rate cannot be too big neither too small.

When learning rate is 0.0002, the validation loss almost doesn't move, which indicate 0.0002 as a learning rate, is too small.

Best learning rate for subsequent experiments: 0.002

Task 2.2: Hidden Layer Size Tuning (6 points)

Experimental Setup

Using the best learning rate from Task 2.1, I experimented with different hidden layer sizes:

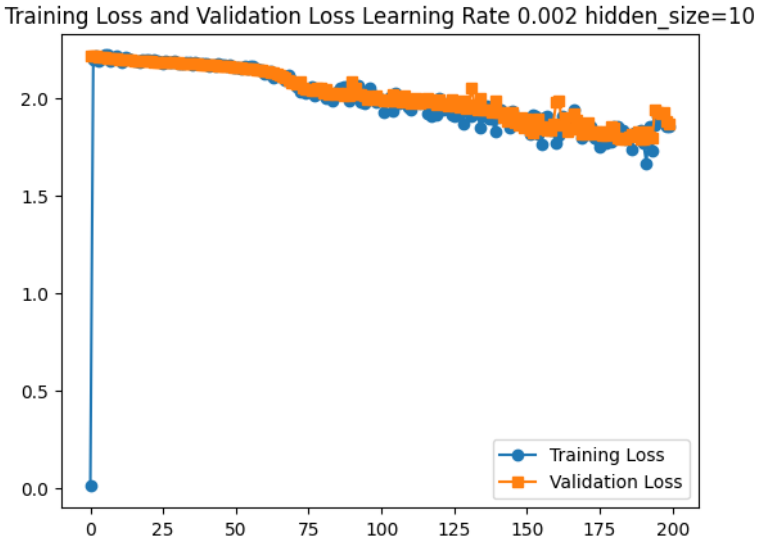
- **Fixed hyperparameters**: learning_rate = [best from 2.1], n_epochs = 50000
- **Hidden sizes tested**: {10, 50, 100}

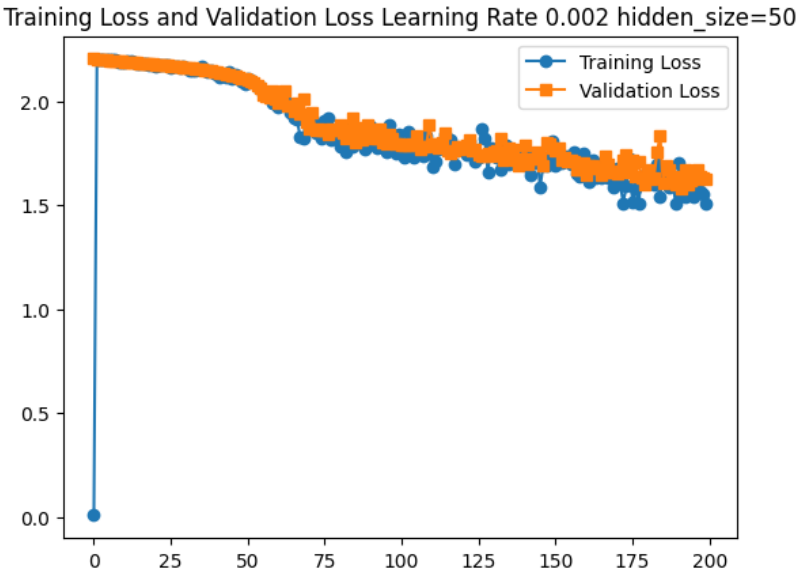
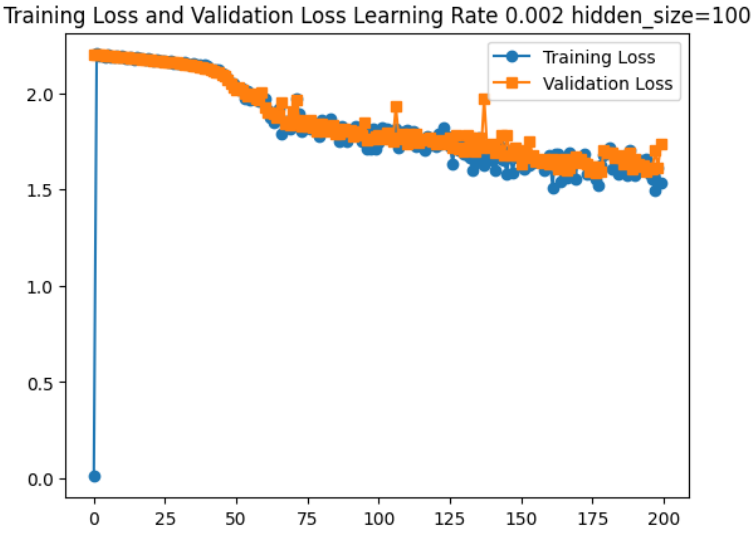
Results

Hidden Size	Validation Accuracy
10	0.3233333333333333
50	0.3922222222222225
100	0.4222222222222222

Training and Validation Loss Plots

[Include 3 plots showing training/validation loss curves for each hidden size]

Hidden Size	Training/Validation Loss Cuves
10	

50	
100	

Observations and Discussion

[Analyze how different hidden layer sizes affected your model's performance. You might discuss aspects like model complexity and capacity, overfitting vs underfitting behavior, training time differences, final performance comparisons, and trade-offs between model size and accuracy. Base your discussion on the specific results you obtained.]

The higher the hidden size, the accuracy is higher. When the hidden size increase, both training loss and validation loss dropped. When hidden size is 50, overall, both losses are decreasing. But for hidden size 100, around the iteration at 175, there's a slightly increase on the validation loss, which could indicate there's risk for overfitting.

The training time is higher for the bigger hidden size.

Between hidden size as 5 and as 10, the accuracy improvement is small, it's 0.392 vs 0.342. But the training time is bigger. Sometimes, we need to judge whether the time for the training and results is worth it.

Best hidden size for subsequent experiments: hidden_size=100

Task 2.3: Epoch Tuning (6 points)

Experimental Setup

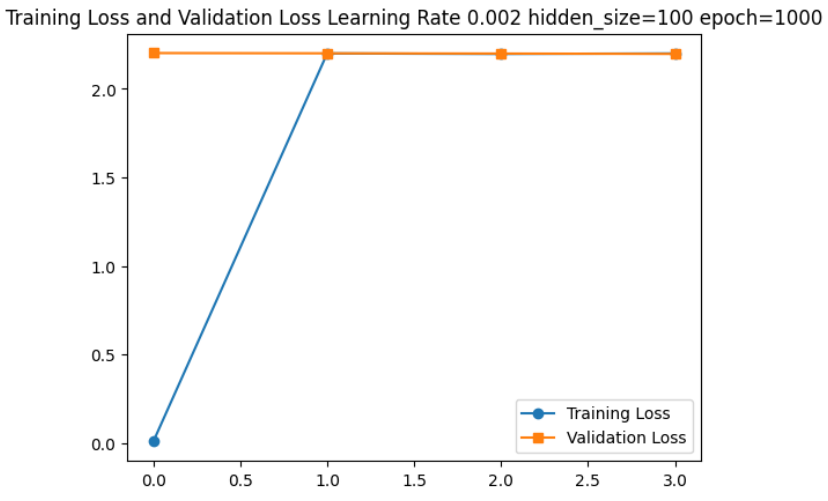
Using the best learning rate and hidden size from previous tasks, I tested different training durations:

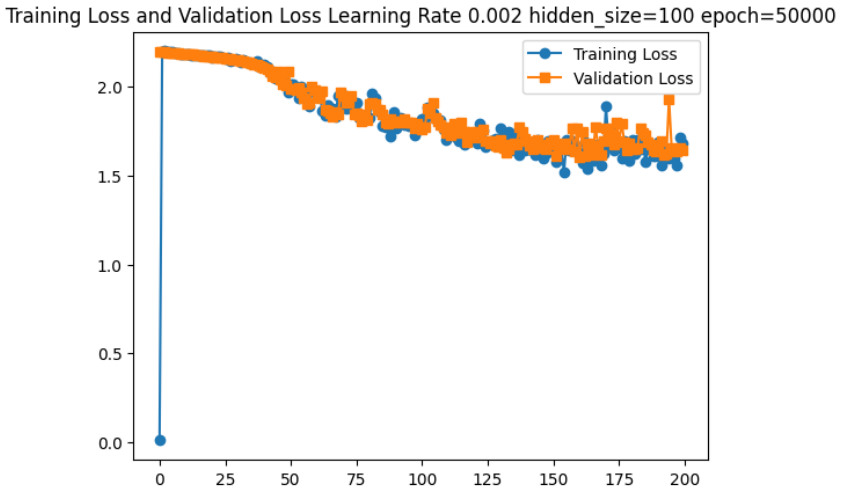
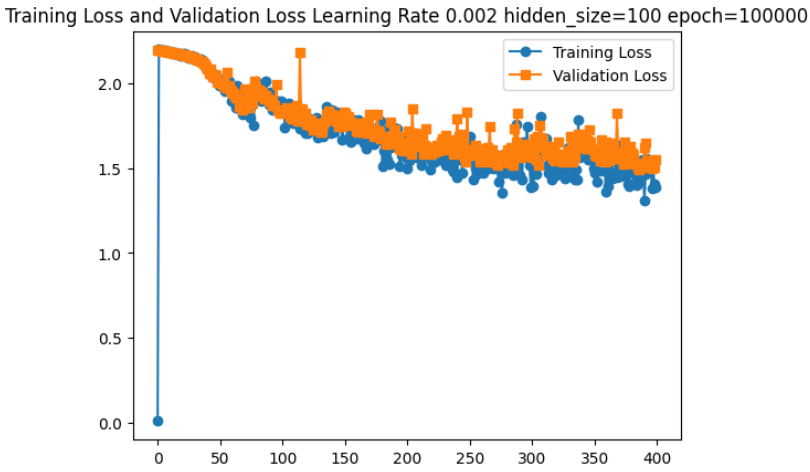
- **Fixed hyperparameters**: learning_rate = [best from 2.1], hidden_size = [best from 2.2]
- **Epochs tested**: {1000, 50000, 100000}

Results

Number of Epochs	Validation Accuracy
1000	0.11555555555555555
50000	0.3988888888888889
100000	0.49

Training and Validation Loss Plots

Number of Epochs	Training/Validation Loss Plots
1000	

50000	
100000	

Observations and Discussion

[Discuss what you learned about training duration and its effects on model performance. You could explore topics such as learning progression over time, signs of overfitting, diminishing returns from longer training, optimal stopping points, and computational efficiency considerations. Focus on what your experimental data reveals.]

When epoch is 1000, the training didn't converge at all, which indicate 1000 is too small.

The epoch with 100,000 has the best accuracy as 0.49. from the plots, we can see that both training loss and validation loss are in decreasing along with running the more iterations.

Both epochs (50,000 and 100,000) has showed a little fluctuation of the training loss and validation loss, which indicates the risk of overfitting. But the overall trend is that the losses are still decreasing.

Optimal number of epochs: epoch = 100,000

Task 2.4: Leaderboard Competition (Optional, 5 bonus points)

Final Model Configuration

Based on my hyperparameter tuning experiments, my best model uses:

- **Learning rate**: [your final choice]
- **Hidden size**: [your final choice]
- **Number of epochs**: [your final choice]
- **Additional modifications**: [any other changes you made, such as:
    - Different optimizer (Adam, Adagrad, etc.)
    - Different loss function
    - Dropout layers
    - Different RNN architecture (LSTM, GRU)
    - Weight initialization changes
    - etc.]

Final Performance

- **Validation Accuracy**: [your final validation accuracy]

Additional Experiments Log

[Optional: Include a table or description of other experiments you tried]

Experiment Description	Hyperparameters	Validation Accuracy	Notes
[experiment 1]	[parameters]	[accuracy]	[observations]
[experiment 2]	[parameters]	[accuracy]	[observations]
...	...	...	...

Discussion

[Describe your experimental process, what worked well, what didn't work, and insights gained about neural language models for classification tasks]

Summary and Conclusions

[Provide a brief summary of your key findings across all experiments, including:

- Most important hyperparameters for this task
- Trade-offs observed between different settings
- Insights about neural network training for NLP tasks
- Recommendations for future work]