CS5300 Final Project Report - Phase 3

Scott Sanchez

December 2, 2024

1 Introduction

Phase 3 focused on testing and refining neural network architectures to improve prediction accuracy for car accident severity. I tested performance across various configurations - both of the data, and of the neural network.

2 Dataset and Preparation

I again used the cleaned dataset from Phase 1 but did further preprocessing to improve these new experiments. I normalized numerical features and adjusted class weights to counteract the severe imbalance in the S̀everityv̀ariable.

This made the data much more usable for the neural networks. Without these adjustments, the model would fail to predict rare classes (i.e., Severity 1 and 4).

Class weights were initially over-tuned, causing instability in the training process. After several iterations, I found weights that were both fair and stable. The final set of weights was determined to improve stability and reduce overfitting, allowing the model to predict rarer classes without overfitting.

3 Baseline Logistic Regression Model

To establish a baseline for comparison, I implemented a logistic regression model before testing the neural network architectures. Logistic regression provided a simpler approach to multi-class classification, helping to set expectations for performance and highlight the benefits of more complex models.

The logistic regression model used a single layer with a softmax activation function to generate probabilities for each of the four severity classes. The model was trained using categorical cross-entropy loss and the Adam optimizer, with the class weights applied to address the imbalance in the dataset.

The validation accuracy plateaued at approximately 55%.

4 Further Tests on Model Architectures of Varying Complexity

I tested five architectures:

- **Small Network:** [32, 4]
- **Medium Network:** [64, 32, 4]
- **Large Network:** [128, 64, 32, 4]
- **XL Network:** [256, 128, 64, 32, 4]
- **XXL Network:** [512, 256, 128, 64, 32, 4]

The models used ReLU activations for hidden layers and softmax for the output layer. These configurations allowed the networks to handle non-linearity while producing probabilities for each class.

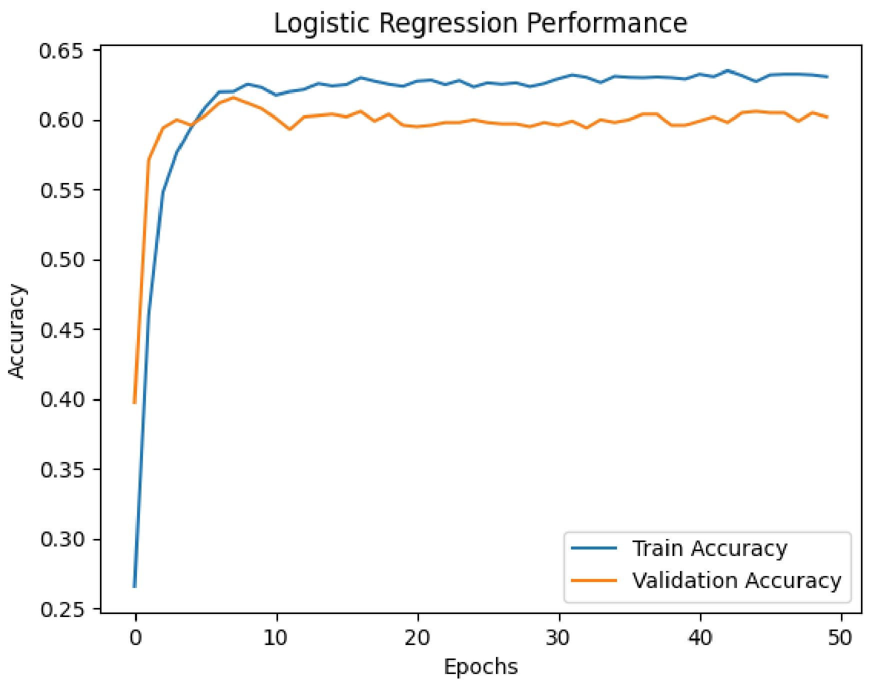

Figure 1: Learning Curve for the Logistic Regression Model. The performance plateaued early, indicating the limitations of this approach for the task.

5 Training and Evaluation

I trained each model using categorical cross-entropy loss and the Adam optimizer.

The training process spanned 500 epochs.

The results, shown in Table 1, show that the XL architecture found the best balance between training and validation accuracy, while the XXL network showed signs of overfitting, despite achieving a high training accuracy.

5.1 Comparison with Logistic Regression

The logistic regression model, despite its simplicity, achieved comparable performance to some of the smaller Neural Networks. This suggests that the dataset's features are effective for broad predictions and provide a solid foundation for modeling.

The performance of the Neural Networks may also stem from the dataset's severe imbalance and the challenges this posed - even after applying class weights. Neural networks offer greater potential for capturing non-linear relationships, but their success is highly dependent on the richness and diversity of the input features.

Architecture	**Train Accuracy**	**Validation Accuracy**	**Train Loss**	**Validation Loss**
32-4	0.5685	0.6056	0.7536	0.7620
64-32-4	0.5045	0.5319	0.7415	0.7493
128-64-32-4	0.6910	0.6653	0.6106	0.6737
256-128-64-32-4	0.7565	0.6823	0.4976	0.7363
512-256-128-64-32-4	0.7643	0.6703	0.4585	0.7317

Table 1: Training and Validation Results Across Architectures

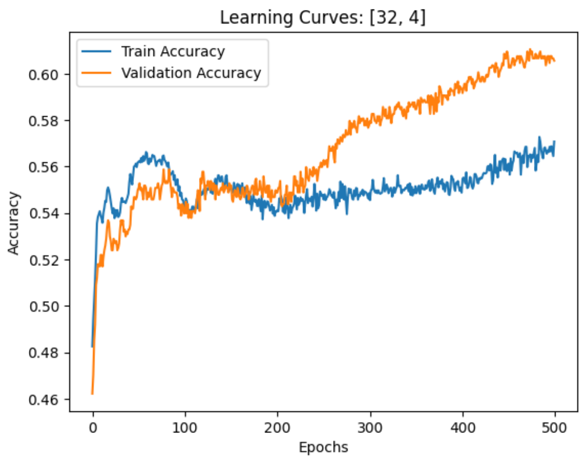

Figure 2: Learning Curve for the 32-4 Model

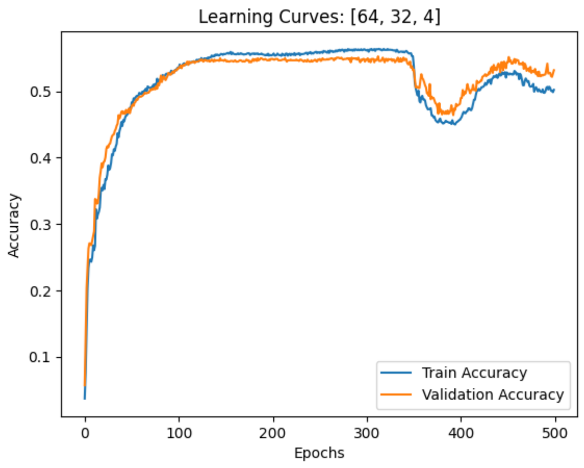

Figure 3: Learning Curve for the 64-32-4 Model

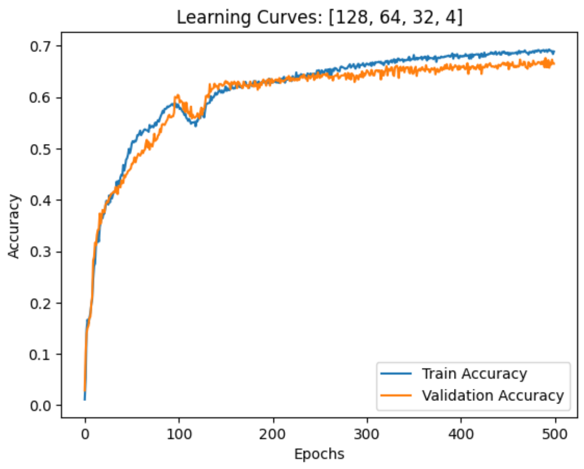

Figure 4: Learning Curve for the 128-64-32-4 Model

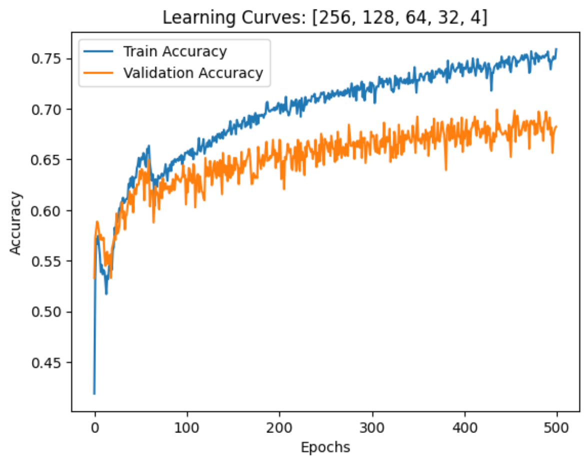

Figure 5: Learning Curve for the 256-128-64-32-4 Model

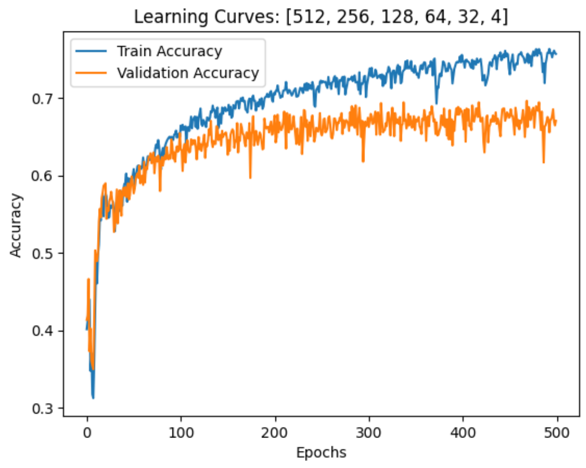

Figure 6: Learning Curve for the 512-256-128-64-32-4 Model

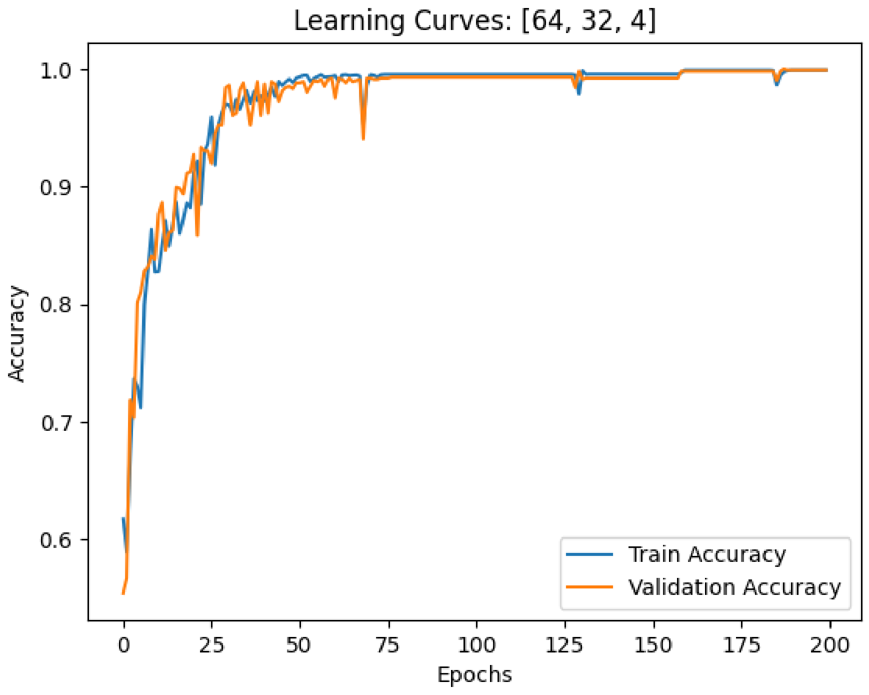

Figure 7: The learning curve for the 64-32-4 model, when given access to data with a feedback loop. All models in this situation approached 100% accuracy because they could, in effect, cheat by looking at the answer.

6 Feedback Loop Experimentation

I experimented with adding a feedback loop by including the S̀everityv̀ariable as an input feature. The goal was to evaluate whether providing this extra information could change the model's performance.

By giving the model direct access to the target variable, training and validation accuracy rose sharply, and all models approached near-perfect results. However, this improvement was superficial; rather than learning meaningful patterns, the model effectively "memorized" the target variable. This created an illusion of success while undermining the model's ability to actually predict the severity of the accidents.

This experiment was to teach us the importance of keeping inputs and targets separate. Including the feedback loop created an "information leakage" in the preprocessing pipeline. Recognizing these issues, I reverted to the original dataset for subsequent experiments.

7 Custom Prediction Function

As part of the graduate requirements, I implemented a custom prediction function to manually replicate the forward pass of the neural network. This function used weights extracted from the trained model. This custom function used these weights to manually compute predictions layer-by-layer. I validated the function by comparing its outputs to those of the model's built-in prediction method. The predictions matched exactly.

8 Conclusion

Phase 3 showed how model complexity and dataset characteristics can interact. The experiments illustrated the risk of overfitting with overly complex architectures and the challenges posed by imbalanced datasets.

The custom prediction function verified the integrity of the trained model and reinforced my understanding of its internal operations.

Future work will focus on enhancing performance for underrepresented classes and refining the model's generalization capabilities.