Module 5 - Critical Thinking

Miles Exner

Colorado State University Global

Course Code: CS580-1

Instructor: Bingdong Li

Due Date: 4/20/2025

# Option #1: Improving the Accuracy of a Neural Network

In this follow-up to the initial Tox21 classification model, several key improvements were made to enhance performance, robustness, and flexibility. First, the fixed model architecture from the previous version was replaced with a modular, hyperparameter-tuned architecture that explores different combinations of hidden layer sizes, dropout rates, and learning rates. The model now supports multiple hidden layers and includes dropout as a tunable regularization parameter. Additionally, the training process was expanded to include grid search across these hyperparameters, with each configuration repeated multiple times using different random seeds to reduce variance in results. Threshold optimization was also significantly improved: instead of using a fixed sweep, an adaptive zooming approach was implemented to find the most effective threshold for classification based on the F1 score. The evaluation pipeline was refactored for clarity and modularity, and key metrics such as precision, recall, and AUC are now reported for both default and optimal thresholds. The original model used a fixed 100-neuron layer and basic thresholding. However, this updated approach provides a systematic and data-driven method for identifying high-performing models within a broader configuration space, even when running on CPU due to current limitations with 50-series GPU support in TensorFlow (currently using a 5070 Ti).

## Tuning Setup

A fully connected feedforward neural network was trained using TensorFlow. The following hyperparameters were varied:

- Number of Hidden Units: [50, 100]  
- Number of Layers: [1, 2]  
- Dropout Rate: [0.2, 0.3, 0.5]  
- Learning Rate: [0.001, 0.0005]  
- Class Weight (positive class): [3.0, 5.0, 8.0]  
- Batch Size: [64]  
- Epochs: 10 (with early stopping)  
- Repetitions per configuration: 5  
- Adaptive thresholding was applied after model training to identify the best probability cutoff for maximizing F1 score.

In total, 216 configurations were evaluated (108 hyperparameter sets × 2 seeds), with adaptive zoom-based threshold optimization applied to each.

## Key Observations from Tuning Results

### Hidden Units

Both 50 and 100 hidden units produced strong models, but 50 units showed slightly better optimized F1 scores and more consistent performance across runs.

### Dropout Rate

A higher dropout rate of 0.5 consistently performed better, likely helping reduce overfitting due to small dataset size.

### Class Weight

A class weight of 3.0 produced the best balance between recall and precision. Larger values (5.0 and 8.0) increased recall but significantly reduced precision, leading to lower F1 scores.

### Learning Rate

The default 0.001 learning rate was more effective than 0.0005. The latter may have slowed convergence too much for 10-epoch early-stopping windows.

### Best Model Configuration

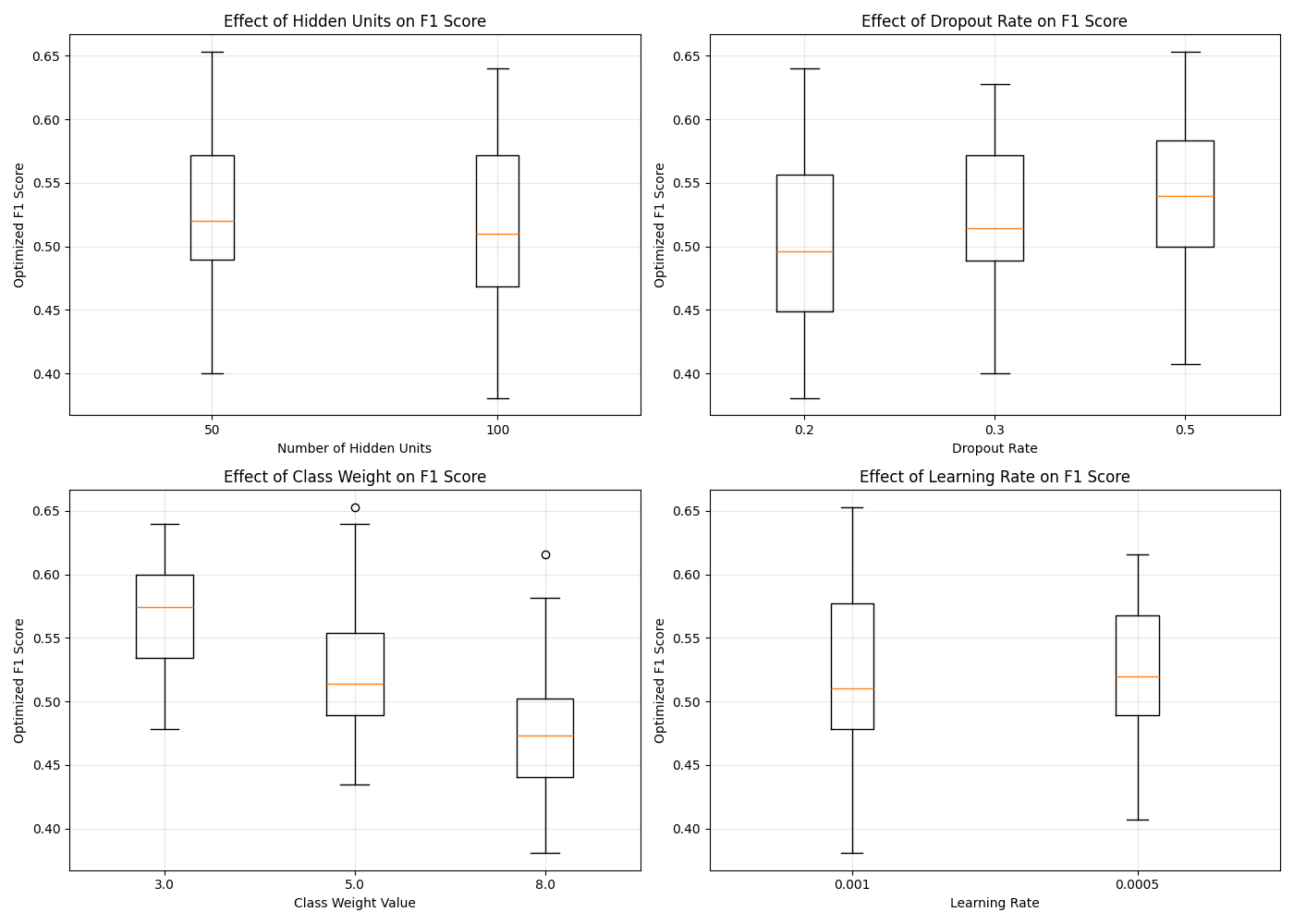
The top-performing model achieved an optimized F1 score of 0.6316 and had the following configuration:

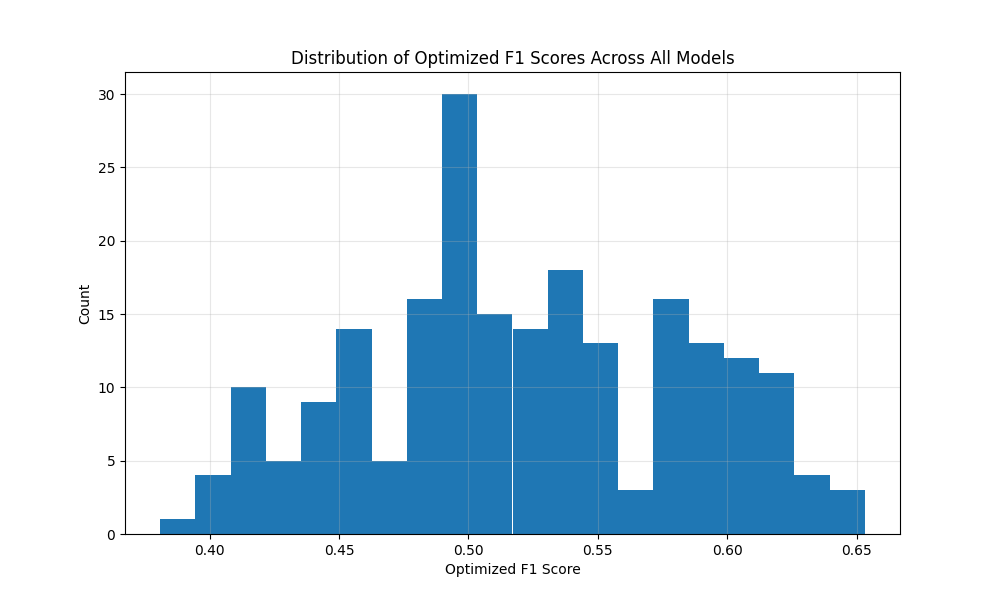
| Hyperparameter | Value |
| --- | --- |
| Hidden Units | 50 |
| Layers | 2 |
| Learning Rate | 0.001 |
| Dropout Rate | 0.5 |
| Class Weight | 3.0 |
| Batch Size | 64 |
| Epochs | 10 |
| Early Stopping | Patience = 2 |
| Best Threshold | 0.34 |

- Precision: 0.814  
- Recall: 0.516  
- AUC: 0.792  
- Actual Epochs (avg): ~6

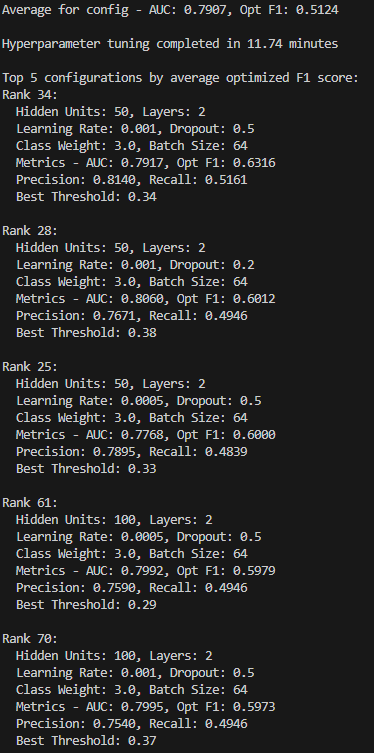
### Visualizations

The following plots support the findings:  
- Class Weight 3.0 yielded the highest median F1.  
- Dropout 0.5 had tighter and higher score distributions.  
- Histogram of F1 scores shows a fairly normal distribution with a positive skew — indicating some configurations performed substantially better than others.

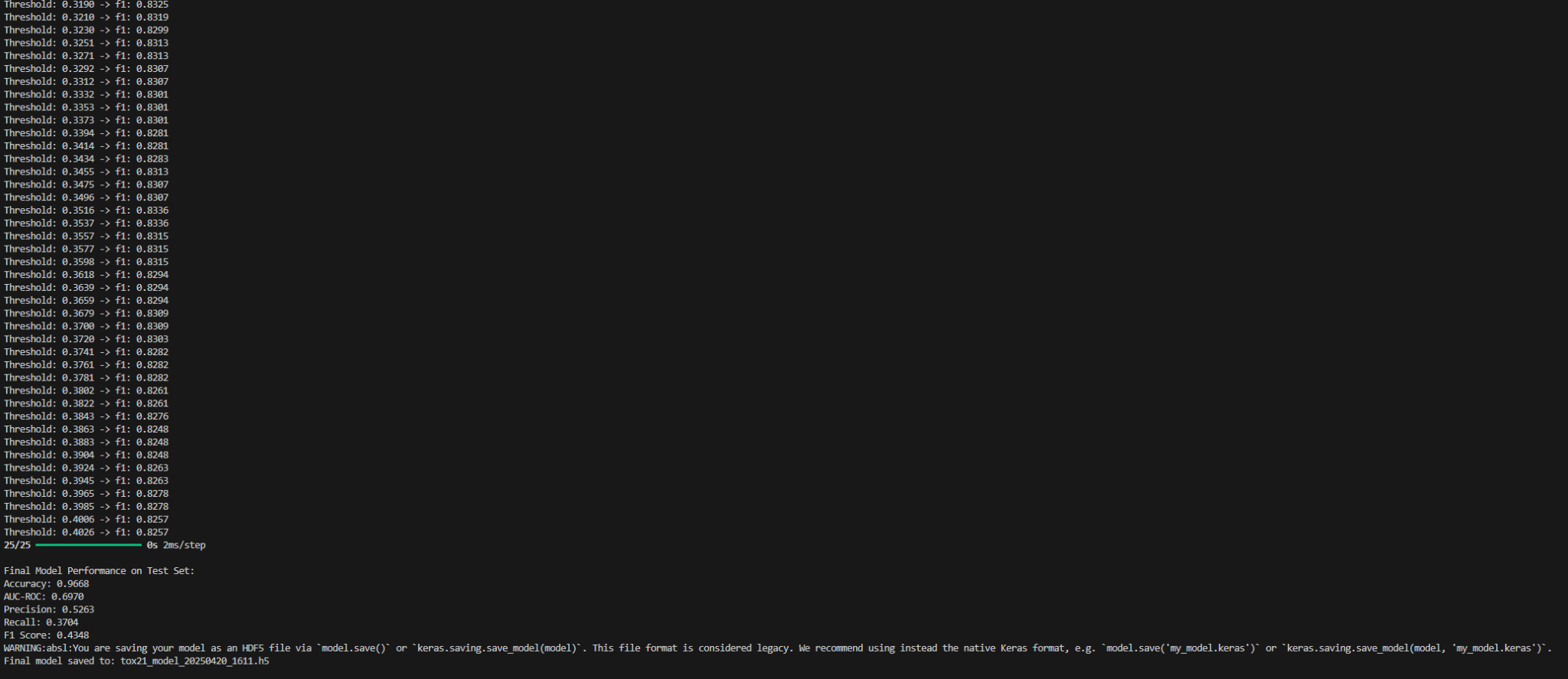




### CLI Output







### Final Notes

All evaluations used adaptive thresholding to tune the probability cutoff instead of default 0.5 — crucial for maximizing F1 on imbalanced datasets like Tox21.  
Early stopping significantly reduced training time while improving generalization.  
Although training was done on CPU (due to lack of current TensorFlow support for RTX 50-series GPUs), performance was solid thanks to efficient tuning and reduced grid size.