Forest Cover Prediction: Deep Learning vs. Ensemble Methods

(Deep Learning Module Project)

Submitted By: **Muhammad Furqan Rauf**

Submitted to: **Umm E Salma**

Date: **02-01-2026**

Executive Summary

In this project, I developed a Deep Learning solution to classify forest cover types using the UCI Forest CoverType dataset. The objective was to engineer a Neural Network (MLP) capable of competing with traditional ensemble methods on tabular data. By implementing advanced regularization, architectural tuning, and training management strategies, I achieved a 91.6% accuracy with the Neural Network. I further benchmarked this against a Random Forest Classifier (95.3% accuracy), demonstrating the trade-offs between deep learning and decision-tree-based approaches for structured datasets.

Data Pipeline & Preprocessing

The dataset consists of cartographic variables (elevation, slope, soil type, etc.) to predict one of 7 forest cover types.

- **Ingestion:** I implemented a robust manual ingestion pipeline using Pandas to bypass API restrictions (HTTP 403 errors).
- **Class Imbalance:** The dataset is heavily imbalanced (Type 1 & 2 dominate). I utilized **Stratified Splitting** to ensure the test set distribution mirrored the training set.
- **Feature Scaling:** Unlike tree models, Neural Networks are sensitive to feature magnitude. I applied **StandardScaler** to normalize continuous variables (like Elevation) to mean=0 and variance=1, ensuring stable gradient descent.

Neural Network Architecture

I designed a "Funnel" architecture (Wide $\to$ Narrow) to distill high-dimensional features into class probabilities.

- **Structure:** Input **->** Dense(256) **->** Dense(128) **->** Dense(64) **->** Softmax(7).
- **Activation:** Used **ReLU** for all hidden layers to mitigate the vanishing gradient problem.
- **Stabilization:** I integrated **Batch Normalization** after every dense layer. This allowed for higher learning rates and faster convergence by normalizing layer inputs during training.

Regularization & Optimization Strategy

To prevent overfitting (where training accuracy exceeds validation accuracy), I engineered a multi-tiered regularization strategy:

- **Dropout:** implemented progressively (0.3 **->** 0.2 **->** 0.1) to force the network to learn redundant feature representations.
- **L2 Regularization:** Added a penalty (1e-4) to weights to discourage overly complex decision boundaries.

- **Optimization:** I compared **Adam** vs. **RMSprop**. Adam proved superior for this sparse/tabular data, showing faster convergence.
- **Scheduling:** Implemented ReduceLROnPlateau to halve the learning rate when validation loss stagnated, helping the model settle into a deeper global minimum.

Comparative Analysis: MLP vs. Random Forest

Deep Learning Model: ~91.6% Accuracy.

Random Forest: ~95.3% Accuracy.

Analysis: The Random Forest outperformed the MLP. This confirms the industry understanding that tree-based models excel at tabular data, as they naturally handle "jagged" decision boundaries (e.g., orthogonal splits like Elevation > 3000). Neural Networks, which approximate functions via smooth manifolds, require significantly more complexity to model these sharp, rule-based thresholds.

Conclusion

This project demonstrated that while Deep Learning is powerful, it requires significant engineering (scaling, regularization, batch norm) to perform well on tabular data. I successfully built a production-grade training pipeline that handled overfitting and optimization dynamics effectively, resulting in a Distinction-level accuracy of >91%.