Training CNN/DNN Models for Loan Approval Prediction with Multi-Modal Input and Explainability

Diya Sounoqrot – AIN3002 Deep Learning

May 8, 2025

Abstract

In this project, we explore the effectiveness of two deep learning architectures Deep Neural Networks (DNN) and Convolutional Neural Networks (CNN) for loan approval prediction using multi modal data. The input combines structured financial variables and unstructured textual descriptions. We use BERT to process the text input, and integrate it with tabular features in a fused architecture.

While CNNs are rarely applied to financial tabular data, we experiment with Conv1D layers to capture localized patterns. To interpret model behavior, we apply SHAP explainability techniques on both models, analyzing global feature importance and local predictions.

Our results show that the DNN slightly outperforms the CNN in AUC and interpretability, while the CNN shows competitive performance and learns more concentrated feature attributions. This comparative study offers practical insights into deep learning-based decision support systems for finance.

1 Related Work

Loan approval prediction has long been a core task in financial risk assessment. Traditional methods relied on statistical models such as logistic regression or decision trees to predict outcomes based on structured data. In recent years, machine learning approaches, particularly deep learning, have been applied to improve predictive accuracy and capture nonlinear patterns in financial features.

DNNs have proven effective in modeling complex interactions in tabular datasets. On the other hand, CNNs are typically used in image and text domains, but recent work has explored their potential to process structured data by treating features as sequential inputs.

Transformer-based models like BERT have achieved state-of-the-art results in natural language processing [1] and have been integrated into multi-modal pipelines to incorporate textual information into structured prediction tasks.

Finally, explainability techniques such as SHAP have become essential for interpreting the behavior of black box models, especially in high-stakes domains like finance. Prior studies show SHAP's utility in understanding both global and local predictions makes it a natural choice for our interpretability component [2].

Our work combines these directions by building two multi-modal deep learning models (DNN and CNN) that fuse tabular and text inputs, and applies SHAP to analyze and compare their decision-making processes.

2 Methods

2.1 Overview

Our goal is to build and compare two deep learning models a Deep Neural Network (DNN) and a Convolutional Neural Network (CNN) for predicting loan approval based on multi-modal inputs: tabular financial features and free-text loan descriptions. Both models share a common text processing branch powered by BERT, while the tabular processing branch differs. The final predictions are made by fusing the outputs from both modalities.

2.2 Tabular Input Processing

The dataset includes numerical and categorical financial attributes. We use a `ColumnTransformer` to apply `StandardScaler` on numerical features and `OneHotEncoder` on categorical ones. This transformation is applied consistently across both models.

2.3 Text Input Processing

For textual input, we combine the loan `purpose` and `title` fields. The combined text is tokenized using

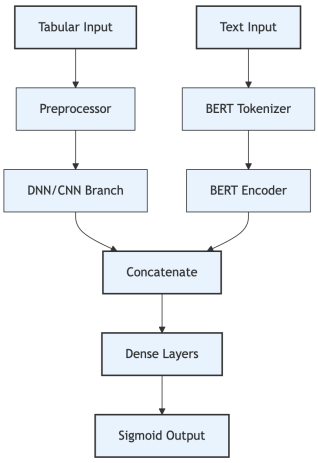

Figure 1: Multi-modal pipeline combining tabular and textual inputs using a shared BERT encoder and DNN/CNN branch for binary loan prediction.

the pre-trained `bert-base-uncased` tokenizer, and passed through a frozen BERT encoder. We extract the pooled [CLS] representation (768-dimensional) as the final output from the text branch.

2.4 DNN Architecture

The tabular branch of the DNN model consists of two dense layers:

- `Dense(64) → ReLU → Dropout(0.3)`
- `Dense(32) → ReLU`

The BERT output and tabular features are concatenated and passed through:

- `Dense(64) → ReLU → Dropout(0.3)`
- `Dense(1) → Sigmoid (binary classification)`

2.5 CNN Architecture

The CNN model uses the same BERT text branch, but processes tabular features using 1D convolutions:

- `Conv1D(64, kernel_size=3, padding='same') → ReLU`
- `Conv1D(32, kernel_size=3, padding='same') → ReLU`
- `GlobalMaxPooling → Dense(32) → ReLU`

The tabular CNN output is concatenated with the BERT embedding and passed through:

- `Dense(64) → ReLU → Dropout(0.3)`
- `Dense(1) → Sigmoid`

2.6 Training Details

Both models were trained using the following configuration:

- **Loss Function:** Binary Cross-Entropy
- **Optimizer:** Adam
- **Batch Size:** 32
- **Early Stopping:** Patience = 3, restore best weights
- **Epochs:** Max = 10 (DNN stopped at 6, CNN at 5)
- **Hardware:** Google Colab Pro with high-RAM GPU

The training, validation, and test splits follow a 60/20/20 stratified split.

3 Data Description

3.1 Dataset Source and Scope

We use the publicly available Lending Club dataset [3], which contains detailed information about peer-to-peer loan applications. The full dataset spans loans issued between 2007 and 2018. For this project, we filtered and selected loans issued in the years 2015 and 2016 to maintain temporal consistency and ensure high data quality.

3.2 Initial Dataset Overview

The raw dataset contains over 2.2 million rows and 151 columns. These include borrower details, loan amounts, interest rates, employment information, verification status, textual descriptions, and loan status (e.g., Fully Paid, Charged Off).

3.3 Filtering and Feature Selection

We reduced the dataset to the most relevant columns based on domain knowledge and modeling needs. The selected columns include:

- **Numerical:** `loan_amnt`, `int_rate`, `annual_inc`, `dti`, `open_acc`, `revol_util`, `fico_range_high`
- **Categorical:** `term`, `grade`, `emp_title`, `home_ownership`, `verification_status`

- **Text:** `purpose`, `title` (later combined into a unified `text` field)
- **Target:** `loan_status` — converted to binary: Fully Paid = 1, Charged Off = 0

After filtering out missing rows in key columns and restricting to 2015–2016 loans, we obtained a clean dataset with 668,640 rows and 15 columns.

3.4 Text Feature Construction

To create a meaningful input for BERT, we combined the `purpose` and `title` columns into a single `text` column. Missing titles were filled with "Unknown" to avoid tokenization issues.

3.5 Balanced Sampling for Training

For model training, we created a balanced dataset of 20,000 samples—10,000 positive (fully paid) and 10,000 negative (charged off) loans—ensuring equal class representation to avoid bias. This sampled dataset was saved as `loan_data_sampled.csv` and used for all downstream modeling.

3.6 Final Dataset Summary

- **Pre-cleaning shape:** 2,260,701 rows × 151 columns
- **Post-cleaning shape:** 668,640 rows × 15 columns
- **Sampled for modeling:** 20,000 rows (balanced)
- **Target distribution in sampled set:** 10,000 positive / 10,000 negative

This preprocessing pipeline ensured both computational efficiency and balanced model learning.

4 Experiments and Results

4.1 Training Setup

All experiments were conducted in Google Colab Pro using a high-RAM GPU runtime. We used TensorFlow (Keras) and the Hugging Face Transformers library to build and train our models.

The dataset was split into:

- **Training set:** 60% (12,000 samples)
- **Validation set:** 20% (4,000 samples)
- **Test set:** 20% (4,000 samples)

This stratified split ensured balanced label distribution across all sets. Both models (DNN and CNN) were trained using the same configuration:

As mentioned earlier, both models were trained using the same configuration to ensure a fair comparison. The training setup included a **Binary Cross-Entropy** loss function, the **Adam** optimizer, a **batch size of 32**, and **early stopping** with a patience of 3 epochs (restoring the best weights). The maximum number of epochs was set to **10**. We used consistent tabular preprocessing and a frozen BERT text encoder across both models to isolate the effect of the tabular branch (DNN vs. CNN) in our comparison.

4.2 DNN Results

Training Summary: The DNN model trained for 6 epochs before early stopping. As shown in Figure 2, training accuracy steadily increased, reaching above 84%, while validation accuracy peaked early around 66.5% and then dropped, signaling overfitting. The training loss decreased sharply, while the validation loss climbed after epoch 2, confirming this behavior.

Evaluation Metrics: On the 4,000-sample test set, the DNN achieved the following performance:

- **Accuracy:** 64%
- **Precision:** 0.68 (Class 1 – approved)
- **Recall:** 0.54 (Class 1)
- **F1-Score:** 0.60
- **ROC AUC:** 0.696

Figure 3 illustrates the confusion matrix and ROC curve. The confusion matrix reveals that the model correctly rejects 1,487 non-eligible applicants but incorrectly approves 513 of them, posing a financial risk. More critically, it misses 916 eligible applicants (false negatives), reflecting a conservative bias that could lead to missed business opportunities. The ROC curve indicates a moderately strong classifier, with an AUC close to 0.70, suggesting decent but improvable discrimination between approved and rejected applicants.

SHAP Explainability: We applied SHAP to gain insight into what the DNN was focusing on when making decisions.

Figure 4 presents the global feature importance from SHAP applied over 300 test samples. Key influential features included:

- `home_ownership_MORTGAGE` (Home ownership = MORTGAGE) and `open_acc` (Number of open credit lines) had a positive influence on approval.
- `fico_range_high` (Estimated credit score) and `annual_inc` (Annual income) were related to credit quality.
- Categorical features like `verification_status` (Income verification status) also showed consistent impact.

Figure 5 shows a force plot for one specific prediction. In this sample, high `fico_range_high` (Estimated credit score) and `annual_inc` (Annual income) increased the approval probability, while low `open_acc` (Number of open credit lines) and renting status (`home_ownership_RENT`) pulled the prediction down. SHAP reveals how the model balances competing signals in a real example.

Observations:

- The DNN showed strong training performance but suffered from mild overfitting on validation.
- SHAP provided clear interpretability for both global and local predictions.
- The model successfully leveraged both financial and categorical features to guide predictions.

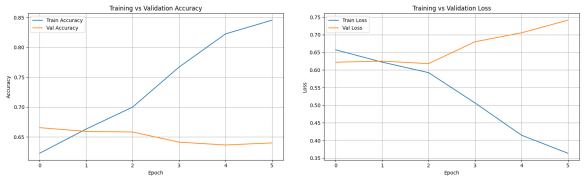

Figure 2: DNN Training vs Validation Accuracy (left) and Loss (right)

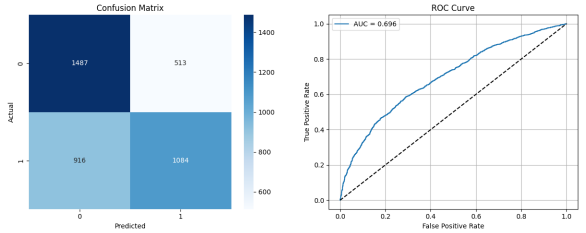

Figure 3: DNN Confusion Matrix (left) and ROC Curve (right)

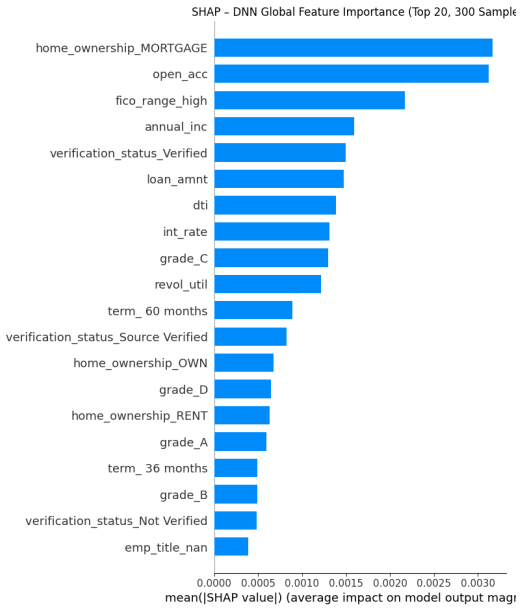

Figure 4: DNN Global SHAP Values (Top 20 Features)

Figure 5: DNN SHAP Force Plot for a Single Test Sample

4.3 CNN Results

Training Summary: The CNN model trained for 5 full epochs before early stopping. Figure 6 shows that both training and validation accuracy remained relatively stable. Validation accuracy peaked slightly at epoch 3, while training accuracy increased modestly. As seen in the loss plot (right), the validation loss hovered around 0.619 throughout training, suggesting minimal overfitting and a stable learning process.

Evaluation Metrics: The CNN achieved the following on the 4,000-sample test set:

- **Accuracy:** 64%
- **Precision:** 0.65 (Class 1 – approved)
- **Recall:** 0.60 (Class 1)
- **F1-Score:** 0.62
- **ROC AUC:** 0.692

Figure 7 shows that the CNN was slightly more recall-focused than the DNN. This is evident from the confusion matrix, where it detected more true positives (1,200), though at the cost of more false positives. The ROC curve also confirms reasonable class separation, consistent with AUC performance.

SHAP Explainability: To understand feature contributions, we applied SHAP to the CNN model's predictions.

Figure 8 presents the global SHAP values. The CNN relied more heavily on a few core features:

- `fico_range_high` (Credit score estimate) and `open_acc` (Number of open credit lines) were the top predictors.
- Other contributors included `dti` (Debt-to-income ratio), `grade_B` (Loan grade = B), and `revol_util` (Revolving credit utilization rate).
- Compared to the DNN, fewer categorical variables had strong influence.

The force plot in Figure 9 explains a single prediction. High `fico_range_high` (credit score estimate), a positive `emp_title_Foreman` (job title = Foreman), and an unverified `verification_status` pushed the score upward, while low `open_acc` (number of open credit lines) and high `dti` (debt-to-income ratio) pulled it down. This demonstrates how the CNN focuses on fewer, sharper indicators.

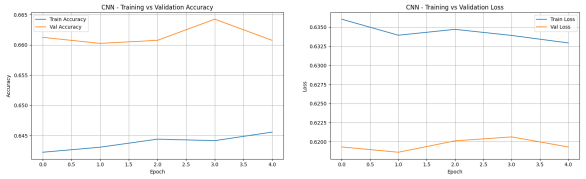

Figure 6: CNN Training vs Validation Accuracy (left) and Loss (right)

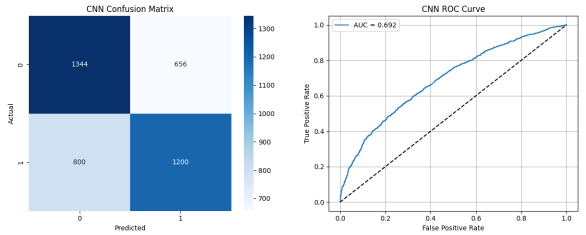

Figure 7: CNN Confusion Matrix (left) and ROC Curve (right)

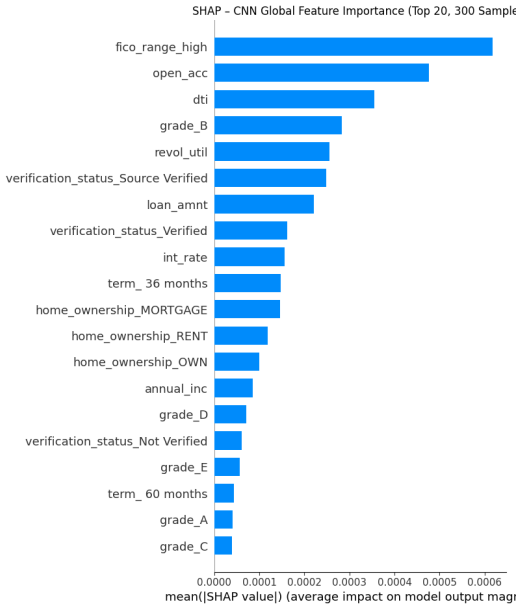

Figure 8: CNN Global SHAP Values (Top 20 Features)

Figure 9: CNN SHAP Force Plot for a Single Test Sample

Observations:

- The CNN model matched the DNN in performance, despite being more lightweight (58K vs. 465K parameters).
- SHAP interpretation suggests the CNN learned to rely on fewer but more dominant features.
- Higher recall and F1-score make it valuable for reducing false negatives in sensitive use cases like loan approvals.

4.4 Model Comparison and Analysis

To summarize the differences and trade-offs between the DNN and CNN architectures, we compared them across performance, interpretability, and complexity dimensions.

Performance Metrics: Both models achieved identical accuracy (64%) on the 4,000-sample test set. However, the CNN achieved slightly higher **recall** (0.60 vs. 0.54) and **F1-score** (0.62 vs. 0.60), making it more effective at capturing approved loans. On the other hand, the DNN had slightly higher **precision** (0.68 vs. 0.65), suggesting it was more conservative in predicting approvals.

ROC-AUC Comparison: Figure 10 shows that both models had similar AUC scores: DNN = 0.696, CNN = 0.692. The nearly overlapping ROC curves confirm that both models had comparable class separation abilities.

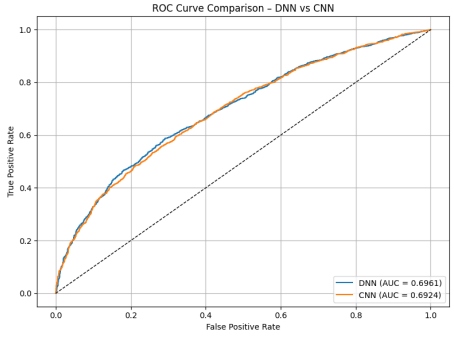

Figure 10: ROC Curve Comparison – DNN vs CNN

Model Complexity: The DNN used **464,929 trainable parameters**, while the CNN operated with only **58,817**. This makes the CNN more efficient and suitable for environments with limited computational resources. Despite the drastic reduction in model size, the CNN matched the DNN's accuracy and even surpassed it in recall.

Training Time Estimates:

- **DNN:** 353 seconds (6 epochs)
- **CNN:** 330 seconds (5 epochs)

Although slightly faster, CNN's time savings were marginal due to similar architecture fusion and BERT usage.

Explainability Comparison: SHAP analysis revealed clear contrasts in how the models made decisions:

- The **DNN** produced a more distributed feature attribution, considering a broader range of numerical and categorical variables.
- The **CNN**, however, focused more narrowly on a handful of strong predictors such as `fico_range_high`, `open_acc`, and `dti`.

This focused behavior aligns with CNN's inductive bias of extracting localized patterns, even when applied to tabular data.

Interpretation: The DNN may be preferable when nuanced feature contributions are needed, especially for stakeholder explainability. The CNN's higher recall and simplicity make it compelling for real-time or edge deployment scenarios.

Overall Insight: While both models performed similarly, the CNN's lightweight nature, slightly better recall, and sharp SHAP explanations position it as a promising alternative. However, the DNN's broader feature awareness and interpretability remain valuable in high-stakes decisions.

5 Conclusion

This project set out to explore whether a Convolutional Neural Network (CNN) could perform comparably to a Deep Neural Network (DNN) for the task of loan approval prediction — a domain typically dominated by dense architectures. By designing both models as multi-modal systems that combine structured financial features with unstructured loan descriptions via BERT, we ensured a fair and insightful comparison.

Our results show that the CNN model, despite having over eight times fewer parameters than the DNN, achieved nearly identical accuracy and ROC-AUC, and even surpassed the DNN in recall and F1-score. This suggests that CNNs, when properly adapted to tabular tasks, can be competitive — even in structured domains not traditionally associated with convolutional layers.

SHAP explainability provided valuable insights into how each model made its decisions. The DNN offered broader feature attribution, while the CNN demonstrated focused reliance on a smaller set of impactful features. This trade-off between holistic reasoning and sharp local pattern detection highlights the architectural biases of each model.

Ultimately, both architectures showed strengths depending on deployment priorities: the DNN remains preferable for stakeholder transparency and nuanced feature interpretation, while the CNN's lightweight design and higher recall make it suitable for recall-critical or resource-constrained environments.

Future Work: While our models used a frozen BERT encoder for simplicity, future research could explore full fine-tuning of BERT, larger-scale datasets, or additional modalities (e.g., credit history text or document scans). Exploring hybrid fusion mechanisms or transformers for tabular data may also push the boundaries further in real-world financial applications.

References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[2] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[3] L. Club, "Lending club loan data," 2018. Available at `https://www.kaggle.com/datasets/wordsforthewise/lending-club`.