MoneyLion Take-home Assessment

# Predicting Categories of Bank Transaction Data

Documentation

*Author: Chau Jia Yi*

*Date: 11th February 2025*

# Contents

# 1 Introduction

This documentation outlines the development of a multi-class classification model to predict categories of bank transactions data. The goal is to build a model that results in good classification performance while generalizing well to unseen transactions.

# 2 Dataset Description

The model is trained using two datasets: bank_transaction.csv and user_profile.csv.

1. **bank_transaction.csv** contains **258,779** transaction records with fields such as transaction amount, date, description, and category. Each transaction belongs to one of **33 categories**.
2. **user_profile.csv** contains **1,000** user profiles with binary indicators representing financial interests (e.g., investment, debt management).

The dataset consists of both numerical (e.g., transaction amount, date-based features) and textual (transaction descriptions) attributes. Textual descriptions require preprocessing and conversion into numerical representation. Some transaction categories (e.g., "Tax Refund") are underrepresented, while others (e.g., "Uncategorized", "Third Party", "Restaurants") dominate the dataset. This imbalance makes weighted loss functions important to avoid bios towards majority class.

Given the structured nature of the dataset, a **supervised classification approach** is appropriate. For example, models such as decision trees, gradient boosting, or neural networks can be effective in capturing complex patterns. The presence of continuous and high-dimensional text embeddings suggests that a deep learning-based approach (deep neural network) can also generalize well.

# 3 Model Architecture

## 3.1 Architecture Overview

The model is implemented as a **multi-layer artificial neural network (ANN)** designed for transaction classification. It consists of three fully connected layers with 256, 128, and output neurons corresponding 33 categories. Batch normalization and dropout (0.3) are applied to improve generalization. The activation function used is ReLU, with a final sigmoid layer for multi-label classification. The model is optimized using Adam with binary cross-entropy loss.

## 3.2 Justification for Model Choice

In my opinion, there is no absolute right or wrong in choosing a ML/DL method, any method has the potential to outperform another.

ANN was chosen due to the presence of non-linear relationships, high-dimensional text embeddings, and categorical data which traditional models like logistic regression may struggle to capture complex relationships in this mixed-data format.

Unlike CNNs, which are more suited for spatial data like images, and LSTMs, which performs well with sequential dependencies like language modelling, an ANN provides a balance between flexibility and efficiency for tabular classification tasks. Moreover, ANNs are relatively efficient in terms of computational cost, making them more scalable.

# 4 Model Training

## 4.1 Data Preparation

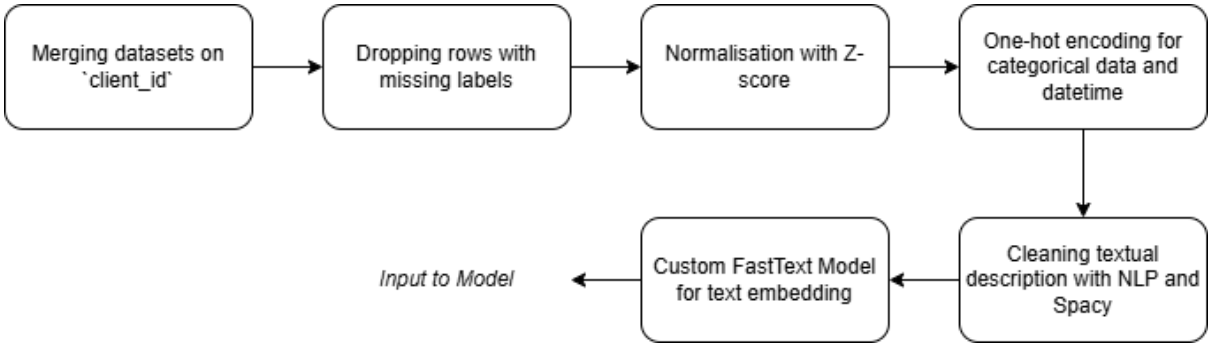


*Figure 1: Data Preparation Flowchart*

Data Preparation Steps are as follows:

1. **Handling Missing Data**: Rows with missing category labels were dropped and later used for model inference.
2. **Feature Normalisation**: Numerical features were normalized using z-score normalization to standardize their scales.
3. **Categorical Encoding**: Categorical data, including transaction categories, was one-hot encoded.
4. **Text Preprocessing**: Transaction descriptions were cleaned and normalized using SpaCy and rule-based text processing methods.
5. **Text Embedding**: A custom FastText model was trained to generate dense vector embeddings, capturing semantic relationships in transaction descriptions.

The final dataset consists of 11 structured features, 100 numerical embedding features, and 33 categorical class labels.

## 4.2 Avoiding Overfitting

Regularization techniques were applied to prevent overfitting, including dropout (0.3) in hidden layers and batch normalization to stabilize learning and improve generalization. Optionally, class weights were used to balance the dataset and reduce bias toward majority classes. Since the model is trained on a large dataset (over 250k instances), a single train-test split is usually sufficient. However, if overfitting were detected, k-fold cross-validation could be implemented to ensure the model generalizes well across different data splits.

## 4.3 Training Methodology

Model training was conducted using the PyTorch framework. The dataset was split into 70% training and 30% testing. Initial training was conducted using batch size = 32, learning rate = 0.001, and 50 epochs, optimized using Adam. However, these parameters can be adjusted during hyperparameter tuning to optimize performance.

Training parameters include:

1. Number of epochs
2. Learning rate
3. Number of layers
4. Dropout rate
5. Activation functions

Due to time constraints, extensive hyperparameter tuning was not conducted, but further optimization could improve model accuracy and generalization.

# 5 Model Evaluation & Performance

## 5.1 Results

The model was trained using a batch size of 32, a learning rate of 0.001, and 50 epochs, optimized using Adam. The model achieved an overall accuracy of 83%. However, looking at accuracy alone can be misleading when working with an imbalanced dataset.

| | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| **ATM** | 0.98 | 0.98 | 0.98 | 1680 |
| **Arts and Entertainment** | 0.89 | 0.84 | 0.86 | 123 |
| **Bank Fee** | 0.50 | 0.20 | 0.29 | 10 |
| **Check Deposit** | 0.96 | 0.66 | 0.78 | 67 |
| **Clothing and Accessories** | 0.70 | 0.44 | 0.54 | 942 |
| **Convenience Stores** | 0.50 | 0.96 | 0.66 | 5634 |
| **Gyms and Fitness Centers** | 0.00 | 0.00 | 0.00 | 22 |
| **Loans** | 0.96 | 0.90 | 0.93 | 5910 |
| **Internal Account Transfer** | 0.99 | 0.98 | 0.98 | 3517 |
| **…** | | | | |
| **accuracy** | | | 0.83 | 77536 |
| **macro avg** | 0.75 | 0.64 | 0.67 | 77536 |
| **weighted avg** | 0.84 | 0.83 | 0.82 | 77536 |

*Table 1: Classification report*
*(full report can be found under /results/ANN_50e_1e-3lr_bce_classifier)*

The model demonstrated strong performance in well-represented classes such as "Internal Account Transfer" (98% recall) and "Loans" (90% recall). However, performance drops for underrepresented classes like "Gyms and Fitness Centers" and "Tax Refund" with 0% recall, indicating the model struggles with rare categories. Precision-recall trade-offs are evident,

with "Convenience Stores" achieving 96% recall but only 50% precision, meaning false positives are relatively higher for some classes.

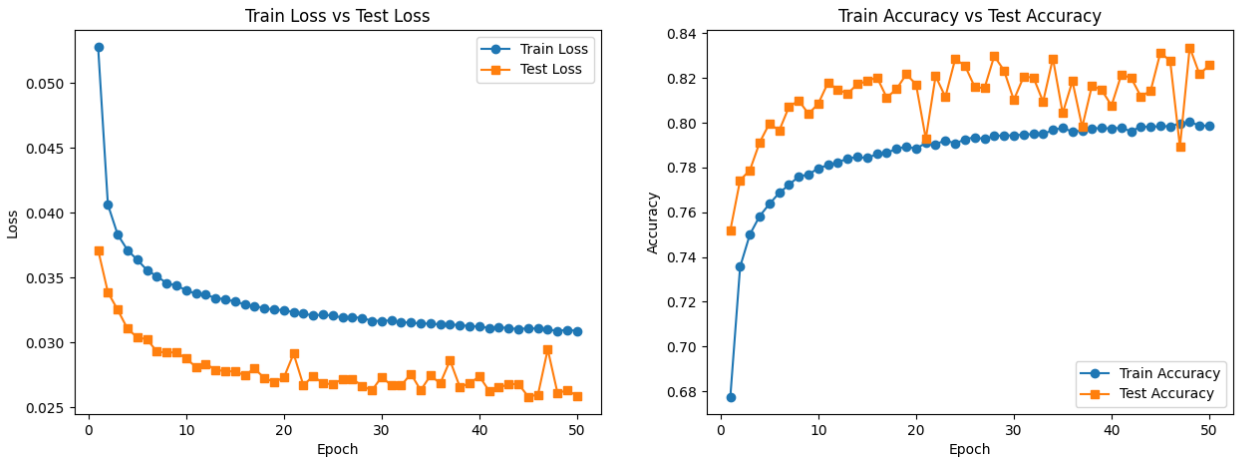


*Figure 2: Train Test Loss & Accuracy Plots*

Loss fluctuations in later epochs suggest potential instability in convergence. Overfitting does not appear to be a major issue as test loss closely follows train loss, but further hyperparameter tuning or data augmentation may be beneficial to reduce both train and test loss.

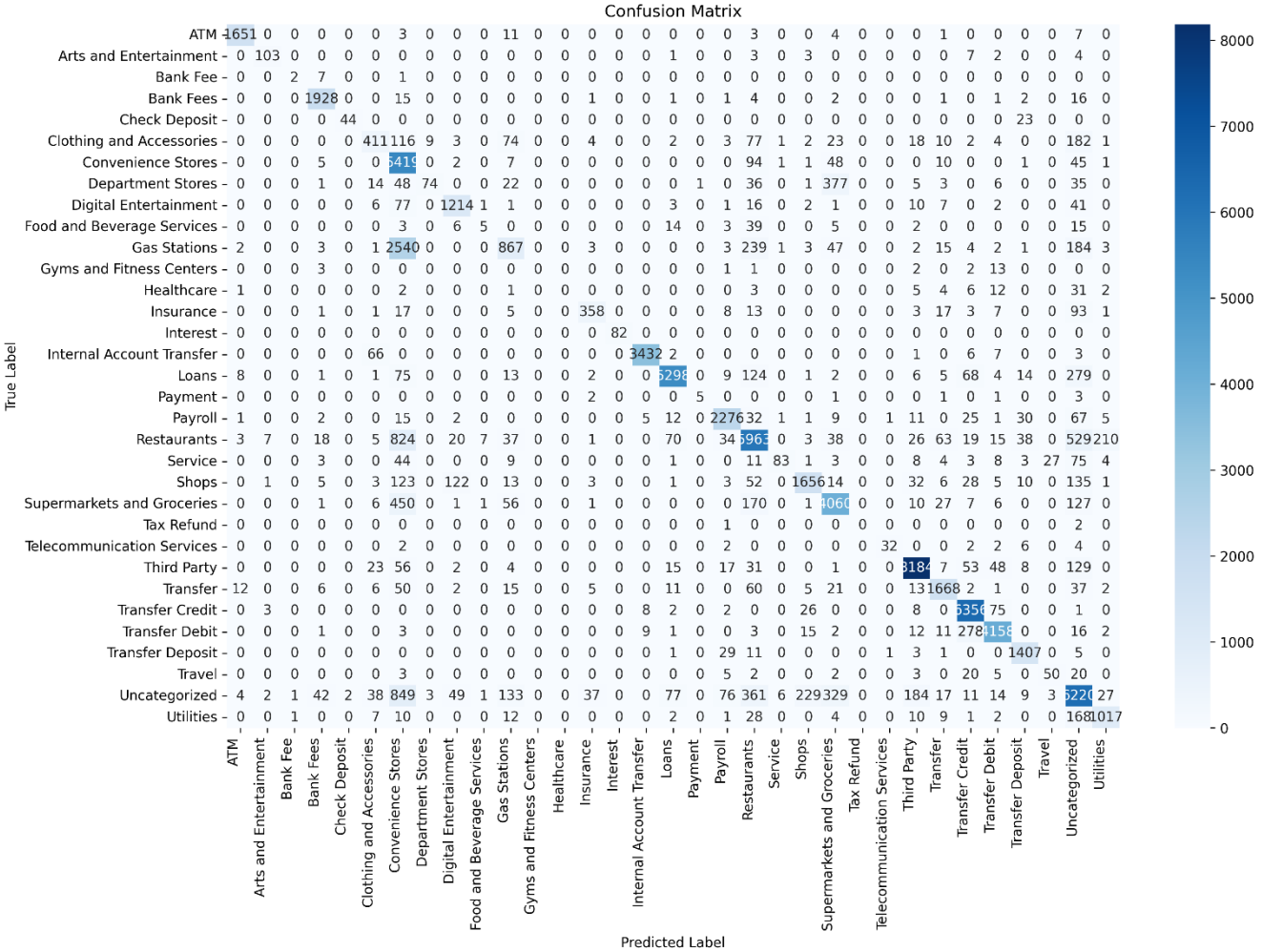


*Figure 3: Confusion Matrix Result*

## 5.2 Inference

The `model_inference.ipynb` section performs inference on unseen transactions that were dropped during preprocessing due to missing class labels. Since there is no ground truth available, direct accuracy evaluation is not possible. However, sample predictions indicate the model's ability to classify transactions based on learned patterns, such as:

| *Description* | *Predicted Category* | *Likely Correct?* |
|---|---|---|
| *Wendy's* | ATM | No |
| *Dandy Mini Mart* | Convenience Stores | Yes |
| *Dunkin Donuts* | Restaurants | Yes |
| *Transfer to Chime Savings Account* | Transfer Credit | Yes |

*Table 2: Inference Examples*

# 6 Future Development Plans

Future improvements could be to:

1. **Optimize ANN Architecture**: Conduct hyperparameter tuning (epochs, number of hidden layers, learning rate, activation functions) to enhance classification accuracy.
2. **Address Class Imbalance**: Augment minority class data or apply sampling techniques for better accuracy.
3. **Improve Text Embeddings**: Experiment with alternative embeddings like BERT or Word2Vec.
4. **Explore Unsupervised Learning**: Utilize deep clustering to classify new market entries without relying on manually annotated dataset.
5. **Deploy as an API**: Develop a backend inference server for real-time transaction classification and inference.

# 7 Appreciation

A huge thank you to the MoneyLion team for providing me the opportunity to try out on this task. Apologies for any oversights, and I truly appreciate your time and consideration in reviewing my work.