BIG DATA 2024/2025

Development Language Models for Fraud Detection

Vitaliia Stepashkina

Motivation Fraud Detection with Transformers

1) Rising Threat:

- Credit card fraud causes billions in losses annually.

2) Limitations of Traditional Models

3) Why transformers?

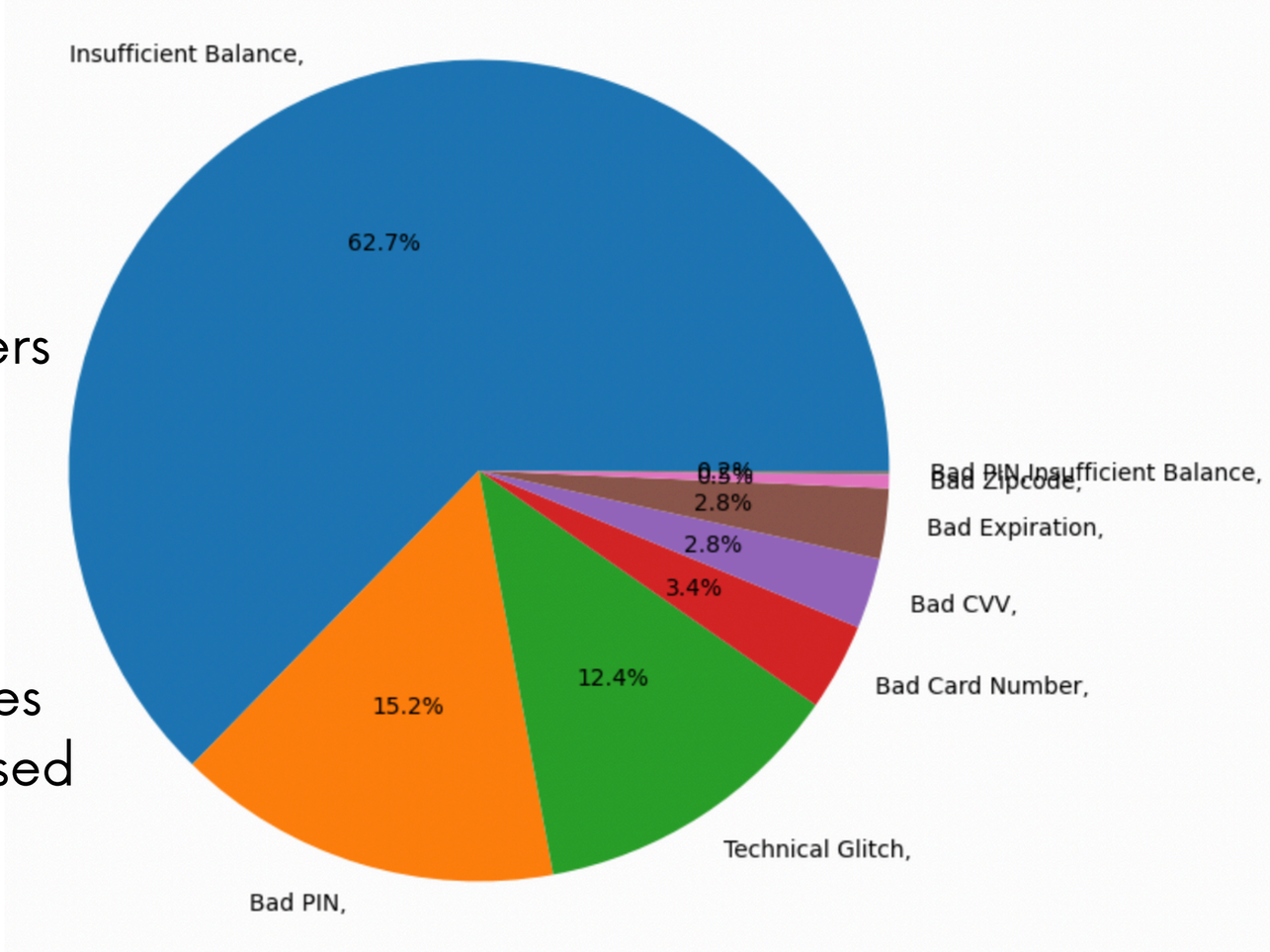
- Enable real-time, high-accuracy fraud detection

Labels/Errors scenario

Dataset Choice:

We use a synthetic transaction dataset generated via stochastic sampling.

- 24 million transactions from 20,000 users
- 12 fields per transaction
- Includes key attributes like merchant details, transaction amount, and fraud labels
- Unlike many public datasets, it preserves real data distributions without PCA-based distortions



Data Preprocessing

Step 1

Remove columns with missing values

Step 2

Apply Label Encoding for categorical data and Log Transformation for transaction amounts

Step 3

Convert numerical attributes (e.g., transaction amount, timestamp) into discrete bins

Step 4

Group transactions per user into sequential windows

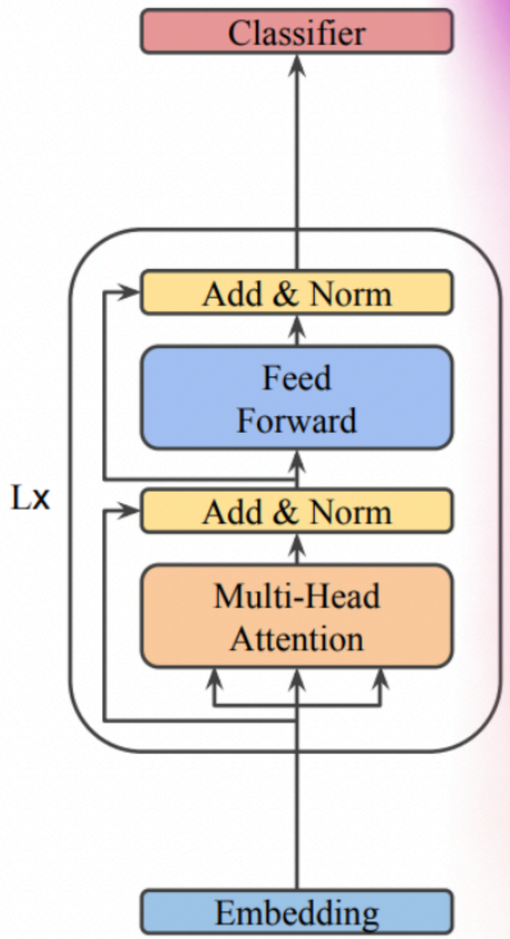
- 5 transaction per window
- stride 3

Neural networks approaches

Introducing to solutions

Preparing Data for BERT

- **Each field is assigned a local finite vocabulary**
- **Tokenization & Vocabulary Mapping**
  - We define a custom vocabulary.
  - Each categorical or discretized numerical feature gets a unique token ID within its respective field.
- **Batch Collation for Training**
  - we mask 15% of a sample's fields, replacing them with the [MASK] token

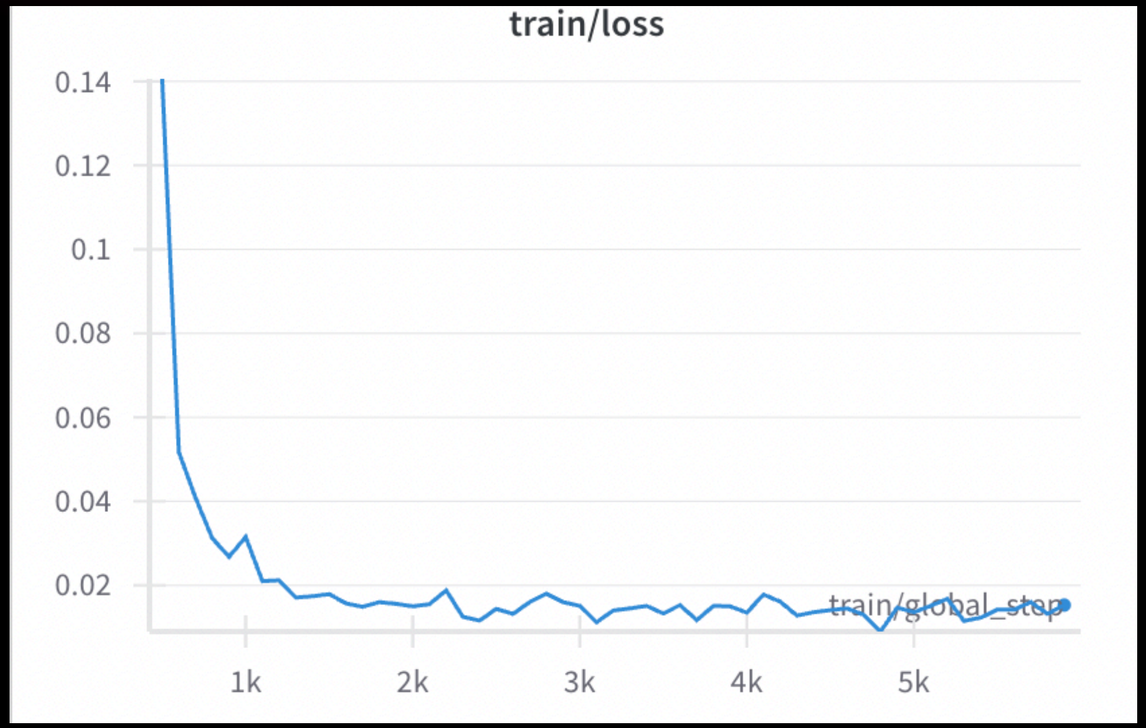


Training BERT

Custom Architecture:

- Built upon a customized configuration TabFormerBertConfig
- Trained to undesrtand local na dglobal dependencies in features/window
- Based on BertForMaskedLM from the Transformers library

Optimizer: AdamW
Epochs: 6
Batch size: 128
Lr: 0.00005 (adaptive)



BERT as a Feature Extractor

Uses BERT as a fixed feature extractor

Structure:

- Only classification head trained
- Simple FFN with one hidden layer
- Uses [CLS] token representation for classification
- ReLU activation

BERT with Partial Fine-Tuning

Fine-tunes the last encoder layer, allowing some adaptation of high-level features

Structure:

- Finetuning for downstream task
- Same simple classification head

Experiments with different heads

Finetuning Bert for classification

BERT + GRU

- **GRU Layer:** Bidirectional GRU on top of BERT outputs
- **Classifier:** Simple linear layer that maps GRU output (using [CLS] token) to the final class labels

BERT + LSTM + Attention layer

- **LSTM Layer:** Bidirectional LSTM on top of BERT outputs
- **Attention Mechanism:** Weights the LSTM outputs, highlighting important tokens
- **Classifier**: Linear layer that maps context vector to class labels

Results

BERT + Classification Head

Models	ROC_AUC
Baseline CatboostClassifier	0.9053
Bert-extractor + simple head	0.7146
Finetuned Bert + simple head	0.9127
Finetuned Bert + GRU head	0.9715
Finetuned Bert + (LSTM + Attention) head	0.9629

Do you have any questions?