AN APPLICATION OF SEQUENCE MODELS FOR CREDIT CARD FRAUD DETECTION

AHMED AHMED [AHMEDAHM@SEAS], ADITYA RATHI [ADITYARA@SEAS], ARJUN [ARJ67022@SEAS],

ABSTRACT. In this paper we experiment with the applicability of sequence models for credit card fraud detection. Recent research has indicated that tree based models seem to systemically outperform deep learning models on tabular data [1]. We introduce 1) A benchmark of tree based models on tabular data 2) A predictive analysis of sequence models and 3) novel applications of Temporal Convolutional Networks (TCN) and Attention to this task. In this work, we do not require feature engineering for our deep learning models and still show that we can outperform highly tuned tree based models using Temporal Convolutional Networks with an Attention Mechanism. In particular, we find that our TCN achieves $\approx 1.5\%$ better accuracy, and $\approx 1\%$ better f1 score on our test set.

1. INTRODUCTION

The growing popularity of electronic payment methods has also led to a rise in financial fraud. Credit card fraud results in billions of dollars in losses for financial institutions every year which has caused these companies to turn to machine learning models for fraud detection. The current models used in industry and research for anomaly detection are tree-based ensemble methods such as gradient boosted decision trees, random forest models, and XGBoost due to their excellent performance when modeling tabular datasets. Recent literature has explored improvement in deep learning methods for tabular data through data pre-processing and architecture variations. Basic recurrent architectures have shown promising results for anomaly detection to the point where they may be able to outperform traditional fraud detection models. Our objective in this work was to extend on this literature by implementing more advanced recurrent architectures for fraud detection and comparing their performance to traditionally used tree-based models.

2. BACKGROUND/RELATED WORK

Deep learning has become the gold-standard for modeling of homogeneous data (images, text, audio) due to the excellent performance of neural network models. However, these models have struggled to perform as well on heterogeneous (tabular) data. This form of data is found in many applications spanning fields such as medicine, finance, and cybersecurity.

Data quality tends to be a major issue for tabular datasets. These datasets usually have missing values, outliers, and class-imbalances due to imperfections in the data collection process [2]. Decision tree models easily handle these characteristics of the dataset but deep learning models struggle to overcome them. One benefit of homogeneous datasets is that deep learning models can take advantage of spatial correlations within the data [3]. In addition, deep learning models are able to learn features that minimize the amount of preprocessing that is required for the dataset [4]. However, tabular datasets often do not have spatial correlations and the preprocessing can have a major effect on model results. Finally, slight changes to a single feature in a tabular data point can completely change the label of the data point. However, dramatically changing a data point in a homogeneous dataset requires changing many features (ex. pixels in an image) [5].

Due to the issues previously stated, deep learning research has focused on data pre-processing to improve model performance, advanced architectures, and regularization methods to allow models to generalize better [2].

Credit card data presents itself as a tabular data set and detecting credit card fraud is a significant goal for financial companies. Branco et. al proposed a method to treat credit card payments as an interleaved sequences where the card history is an unbounded, irregular sub-sequence [6]. This allowed the researchers to use recurrent architectures to demonstrate that comparable and sometimes superior results can be obtained with deep learning architectures compared to random forest models for fraud detection. This paper placed an emphasis on machine learning operations and ensuring that recurrent architectures can be used in a production environment rather than using more complex architectures. We hope to build on this work by exploring more complex architectures, namely transformers with a self-attention mechanism and temporal convolutional networks (TCN).

3. Approach

3.1. **Dataset.** Credit Card Fraud Detection: The dataset contains transactions made by credit cards in September 2013 by European cardholders. The target variable is Class, which is either 1 (for yes) or 0 (for no). The dataset contains 30k examples where each example contains metadata of each transaction at a certain point in time.

3.2. **Preprocessing.** Although the dataset is structured, we followed some preprocessing strategies from [6]. First, we normalize the numerical columns with outlier clipping. Empirically, the referenced paper set T_0, the number of standard deviations from the mean above which to consider a value an outlier, to 3 and so did we. Next, we did percentile bucketing, with the number of buckets set to 10. Finally, we did categorical featurization by mapping each possible value to an integer based on the number of occurrences it has in the training set:

$$x'_{cj=l-1}$$

where for a given categorical feature, x_{c_j}, the l^{th} most frequent value is mapped to the value $l-1$.

Moreover, in order to emphasize the value of deep learning models over tree based models, we added manually engineered features that were only available to the tree based models and not the deep learning models. The first added feature indicates whether someone was a client at time t. The second added feature indicates the average expenses up until time t. And the last added feature indicates how far the current expenses are from the maximum allowed for that client.

3.3. **Metrics.** Given the highly imbalanced nature of an anomaly detection task such as predicting credit card fraud, we need to prioritize both the precision and recall of our model. In credit card fraud, a higher precision means that we are saving money by rejecting truly fraudulent transactions and higher recall means that we are not rejecting truly non-fraudulent transactions. We cannot simply prioritize precision because it will lead to customers becoming very unhappy with always having their transactions blocked, and therefore we use the f1 metric as our primary measure of how well our model does.

3.4. **Sequence Data.** We define our input data as a follows: Let $X_i \in \mathbb{R}^{D \times S_0}$ be the input feature vector of length S_0 for time step t for $0 < t \leq T$ where T is the number of transactions customer i has had, and D is the number of features recorded for each transaction. For our dataset specifically, T = 6.

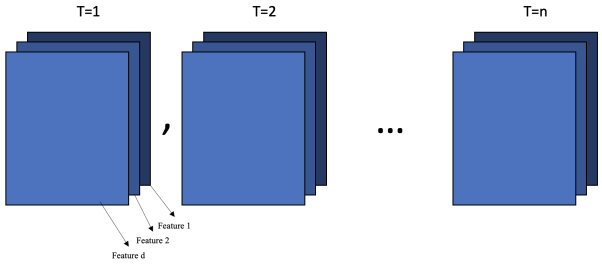


Figure 1. Sequential organization of our dataset

3.5. **Tree Model Benchmarks.** For tree-based models, we choose 3 state-of-the-art models used by practitioners: Scikit Learn's RandomForest, GradientBoostingTrees (GBTs), and XGBoost [7]. In prior research benchmarking tree based models, random forest, xgboost, and gradient boosted decision trees were, in most cases, the best performing models on 55 datasets [8], so we use these models as our tree baselines. Each model was highly finetuned to maximum results by performing grid search.

3.6. **Recurrent Model.** Before going to more advanced architectures, we implemented both a gated recurrent unit (GRU) and an Long-Short Term Memory (LSTM) model. Compared to a traditional RNN, both a GRU and an LSTM feature additional parameters that allow for added control to the output of each hidden state. The GRU features a reset and zero variable on top of the usual parameters of an RNN. In an LSTM, the output gate replaces the reset variable the the input and forget gates replace the zero variable. The greater complexity of LSTMs when compared to GRUs may allow them to perform better than GRUs. However, GRUs have a lower memory requirement and would be an easier

model to operate in production. For both the GRU and LSTM, two stacked recurrent layers were with a hidden state size of 150 were used. The final layer was a linear layer with two logits. To improve on these more basic recurrent architectures, we turned to a transformer that incorporates an attention mechanism and a temporal convolutional network which combines aspects of both a CNN and a RNN and should lead to improved performance.

3.7. **Transformer Model.** While transformer models are usually developed for text processing tasks such as translation [9] and for next-word prediction, they are also commonly used for sequential tasks such as time series and classification as well. It is particularly useful for tasks that involve long sequences, as it is able to handle dependencies between elements that are far apart in the sequence. Due to the self-attention mechanism in transformers, we are able to model sequential data and use it for classification tasks. In our case, we use the features in our dataset to develop temporal relationships to classify whether the current transaction is fraudulent or not given a user's history.

We created the ClassificationTransformer class for performing classification tasks using the Transformer architecture. The inputs we take include the input and output dimensions, the number of layers and heads in the Transformer, a dropout rate to apply during training, and the maximum length of the input sequences.

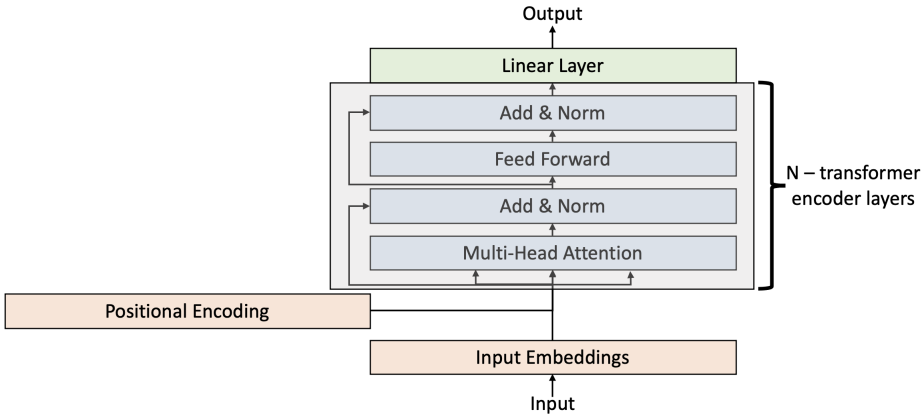


FIGURE 2. Visual example of Transformer architecture

Describing the architecture, we first have an embedding layer, which converts the input data into a dense representation that can be processed by the Transformer. Next, the Transformer itself consists of a series of encoder layers, which apply attention and transformation operations to the input data. The Transformer also uses a positional encoding to incorporate information about the relative position of the elements in the input sequence. Finally, the output of the Transformer is passed through a fully-connected layer and returned as the final prediction.

During the forward pass of the model, the input data is first passed through the embedding layer and then transformed using the Transformer encoder. The output of the encoder is then passed through the fully-connected layer to produce the final prediction.

An important part of transformer architectures is positional encoding:

$$P[i, 2j] = sin(i/10000^{2j/d})$$
$$P[i, 2j+1] = cos(1/10000^{2j/d})$$

Where P is the positional encoding matrix with dimensions (sequence length, embedding dimension), i is the position in the sequence, j is the index of the embedding dimension, and d is the total number of embedding dimensions. We use sine and cosine functions of different frequencies to create this positional encoding.

3.8. **Temporal Convolutional Model.** Recent developments in deep learning by Lea et al [10] resulted in Temporal Convolutional Neural Networks which is a variation of Convolutional Neural Networks for sequence modelling tasks by combining aspects of RNN and CNN architectures. The proposed paradigm for credit card fraud detection is composed of three steps: First, for each transaction in a user's history, we compute the low-level features using something line a CNN which encodes spatiotemporal information locally. Secondly, we input the captured sequential low level features into a model that captures high-level temporal relationships with some form of sequence model. Lastly, we use the

computed high-level temporal relationships to classify whether the current transaction is fraudulent or not given a user's history. Rather than defining three separate models for this task, we use TCNs to decouple this task and reduce the individual complexities required of specifying separate models and allows for the capture of more nuanced long-range spatio-temporal relationships as features in the fraud classification task.

By using a TCN, we can hierarchically capture low, medium, and high level temporal information more effectively through three primary mechanisms. A TCN is based on the principles that the network produces an output of the same length of the input, and that there can be no leakage from the future to the past. The former principle is achieved by using 1D fully convolutional filters and the second is achieved by using casual convolutions where the output at time t is only convolved with features from before time t given by:

$$F(s) = (x *_d f)(s) = \sum_{i=0}^{k-1} f(i) \cdot x_{s-d*i}$$

where $x \in \mathbb{R}^n$, d is the dilation factor, k is the filter size, and f is a filter.

Each layer in the TCN consists of a dilated convolution and a residual connection that combines the layer's input and the convolution signal. The dilation factor d increases between consecutive layers where $d_l = 2^l$

Results after the dilated convolutions are applied over two timestamps t and t-d and can be calculated by the following expression:

$$\tilde{Z}_t^{j,l} = BN(f(W_0 \tilde{Z}_{t-d}^{(j,l-1)} + W_1 \tilde{Z}_{t-d}^{(j,l-1)}))$$

Where W_0 and W_1 are the parameterized weight matrices where $W_i \in \mathbb{R}^{r \times r}$ where r is the number of filters.

and BN is the batch normalization instead of traditional weight normalization as per recent literature [11] which suggests that batch normalization may result in better test accuracy.

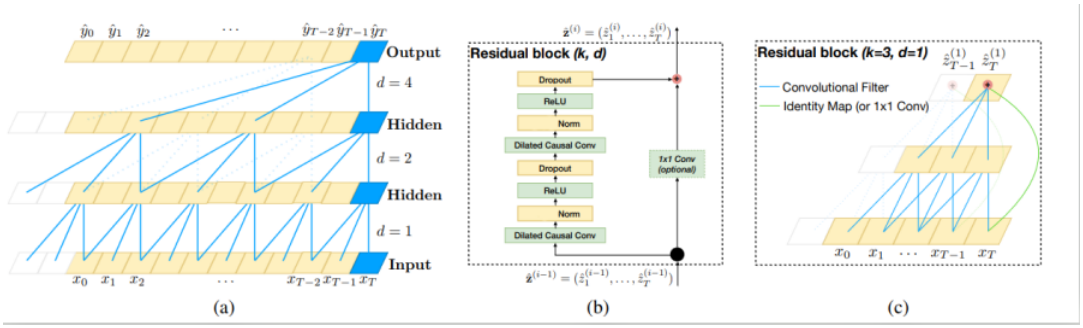
$$\text{BatchNorm}(\mathbf{x}) = \frac{\mathbf{x} - \mu}{\sqrt{\sigma^2 + \epsilon}} \cdot \gamma + \beta$$

where γ and β are learnable scale and shift parameters

The result after adding the residual connection at timestamp t can be calculated by:

$$Z_t^{(j,l)} = Z_t^{(j,l-1)} + V \tilde{Z}_t^{j,l} + B$$

Where $V \in \mathbb{R}^{r \times r}$ is the weight matrix and $b \in \mathbb{R}^r$ is the bias vector for the residual block. [12]



[13]

FIGURE 3. Visual example of TCN architecture

After the input data has been propogated through the temporal network, we feed it through an attention block as a premilinary feature extractor to a fully connected linear layer that takes the outputs of the attention block and outputs two probabilities for fraud vs not fraud.

The final network consisted of a TCN with a kernel size of 3, and 6 temporal blocks. The blocks consisted of layers with dimensionality (64, 64), (64, 64), (64, 128), (128, 128), (128, 256), (256, 256) and dilation factors of 1, 2, 4, 8, 16, 32 respectively. The model was trained with the adam optimizer, a learning rate of 1e-3 (multiplied by .8 every epoch), weight decay of 1e-4 and cross entropy loss over 15 epochs.

4. Experimental Results

Model	Precision %	Recall %	F1 %	Accuracy %
GBDT	91.0	80.7	85.6	86.3
Random Forest	91.1	79.9	85.1	85.8
AdaBoost	91.1	78.4	84.2	85.3
XGBoost	91.6	79.2	81.0	85.9
LSTM	81.3	91.9	86.3	86.5
GRU	80.7	91.4	85.2	85.9
Transformer	79.5	90.6	84.2	84.6
TCN w/ attention	81.0	93.4	86.7	87.5

Table 1. Performance comparison of Deep Learning and Tree Based Models

5. Discussion

5.1. **Model Analysis.** With the above results, it is clear to see that deep learning models can be more applicable than tree based models for fraud detection. While the model accuracy and f1 scores are relatively similar, we observe our models have higher recall than the traditional tree-based models. Moreover, we observe that TCN w/ attention is the best-performing deep learning method, consistently outperforming the other deep learning models in recall, F1 score and accuracy, as well as being significantly better than tree-based models for recall.

While recall is a good metric to discuss here, given that models can obtain 100% recall by predicting everything as the positive class, it is more appropriate to discuss f1 scores which balances this out by also considering precision. With this in mind, while the GRU and Transformer models perform similarly to the tree-based models, the LSTM and TCN w/ attention successfully outperform all tree-based models.

Another noteable difference is that the deep learning models were generally able to achieve similar or better metrics despite the tree based models having access to manually engineered features. This suggests that deep learning models have the capability of learning these manually engineered features which would reduce the overhead cost of manual feature engineering in a production environment.

5.2. **Future Directions.** In the future, we would also like to explore the model performance on other datasets since concluding the efficacy of our solution soley based on its outperformance of tree-based models on this dataset is not sufficient to conclusively state that our models are always going to be applicable in this domain.

There is also some nuances regarding the sequence length of fraud detection tasks where certain research suggests that deep learning models are able to better predict fraud given a longer history in comparison to tree based models. Our dataset only had a sequence length of 6 which is a relatively short timeframe, so it would be interesting to see how the comparison of the two model types would change given a longer sequence length and perhaps more transactions in between.

REFERENCES

[1] Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *CoRR*, abs/2106.03253, 2021.

[2] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey, 2021.

[3] Ami Abutbul, Gal Elidan, Liran Katzir, and Ran El-Yaniv. Dnf-net: A neural architecture for tabular data, 2020.

[4] Yury Gorishniy, Ivan Rubachev, and Artem Babenko. On embeddings for numerical features in tabular deep learning, 2022.

[5] Ira Shavitt and Eran Segal. Regularization learning networks: Deep learning for tabular datasets, 2018.

[6] Bernardo Branco, Pedro Abreu, Ana Sofia Gomes, Mariana S. C. Almeida, João Tiago Ascensão, and Pedro Bizarro. Interleaved sequence RNNs for fraud detection. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery &amp Data Mining*. ACM, aug 2020.

[7] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016.

[8] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on tabular data?, 2022.

[9] Shumin Deng, Ningyu Zhang, Wen Zhang, Jiaoyan Chen, Jeff Pan, and Huajun Chen. Improving the transformer translation model with document-level context. pages 533–542, 2018.

[10] Colin Lea, René Vidal, Austin Reiter, and Gregory D. Hager. Temporal convolutional networks: A unified approach to action segmentation. *CoRR*, abs/1608.08242, 2016.

[11] Igor Gitman and Boris Ginsburg. Comparison of batch normalization and weight normalization algorithms for the large-scale image classification. *CoRR*, abs/1709.08145, 2017.

[12] Shumin Deng, Ningyu Zhang, Wen Zhang, Jiaoyan Chen, Jeff Pan, and Huajun Chen. Knowledge-driven stock trend prediction and explanation via temporal convolutional network. 03 2019.

[13] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *CoRR*, abs/1803.01271, 2018.