Politecnico di Milano

Dipartimento di Elettronica Informazione e Bioingegneria
Laurea Magistrale in Computer Science and Engineering

POLITECNICO
MILANO 1863

A Natural Language Processing Approach to Fraud Detection

Relatore:

Prof. Stefano Zanero

Correlatore:

Ph.D. Michele Carminati

Tesi di Laurea Magistrale di:

Javier Fernández
Matricola n. 916133

Anno Accademico 2019-2020

Contents

List of Figures

List of Tables

Listings

Sommario

A causa della proliferazione dell'online banking, le persone sono più esposte che mai agli attacchi. Inoltre, le frodi stanno diventando sempre più sofisticate, superando le misure di sicurezza messe in atto da parte delle istituzioni finanziarie.

Proponiamo un nuovo approccio all'individuazione delle frodi, basato sull'uso di modelli di Natural Language Processing. Questi modelli hanno mostrato una performance eccezionale nel loro campo, che condivide molte caratteristiche con il problema di individuazione delle frodi. La nostra soluzione si basa sulla profilazione degli utenti. L'obiettivo finale è quello di prevedere il comportamento degli utenti e rilevare le frodi come deviazioni da essi. Questo si ottiene grazie al meccanismo di attenzione, che permette al modello di sfruttare appieno la storia dell'utente.

Il nostro modello raggiunge un buon equilibrio tra Precisione e Recupero, superando i modelli state-of-the-art in diversi scenari. Il modello proposto offre inoltre prestazioni migliori rispetto a modelli simili basati su LSTM, dimostrando così la superiorità dei modelli basati sull'attenzione rispetto agli approcci classici. Infine, studiamo la robustezza del nostro approccio contro gli attacchi avversari.

Abstract

Due to the proliferation of online banking, people are more exposed than ever to attacks. Moreover, the frauds are becoming more sophisticated, bypassing the security measures put in place by the financial institutions.

We propose a novel approach to fraud detection, based on the use of Natural Language Processing models. These models have shown an outstanding performance in their field, which shares similarities with the problem of fraud detection. Our solution is based on user profiling. The final goal is to predict the users' behaviour and detect frauds as deviations from them. This is achieved thanks to the attention mechanism, which allows the model to fully exploit the user's history.

Our model achieves a good balance between precision and recall, outperforming state-of-the-art models in different scenarios. The proposed model also performs better than similar models based on LSTM, which shows the superiority of attention-based models over classic approaches. Finally, we study the robustness of our approach against adversarial attacks.

Chapter 1

Introduction

The past twelve years have seen an increment of online banking usage. For example, in Great Britain, the usage has stepped up from 30% in 2017 up to 73% in the last year [1]. People use their computers and mobiles to pay, perform transactions, and manage their finances. This has attracted the attention of fraudsters. According to Marchini et al. [2] 14.4 million people were affected by banking fraud in 2018.

Banking frauds are becoming more sophisticated as time goes along, bypassing the protection mechanisms put in place. Banks are demanding solutions for this pressing issue. The Fraud Detection and Prevention market is valued in 19.5 billion dollars and raising [3]. Fraud detection in the field of finance is difficult because of a variety of reasons: Lack of data, scarce frauds, concept drifts and the time dimension of the data renders the usual techniques useless.

Existing solutions can be categorised according to the input they use to perform the detection of frauds [4]: Local models, global models, and temporal models:

Local models build a representation of the user behaviour to detect frauds as a deviation from normal behaviour. This representation is built through metrics aggregated by user, as mean monthly expenditure or the total number of transactions carried out.

Global models instead focus on characterising legitime transactions without taking into account which user performed them. These models usually rely on clustering techniques to group similar transactions, flagging as a fraud the transactions that do not fall into the legitimate groups.

Temporal models are similar to local ones but taking into consideration the time dimension. They also build a representation of the user behaviour but instead of using aggregated metrics, temporal models use directly the transactions. In order to do so, temporal models arrange the users' transactions in form of time-ordered sequences and process them using LSTM or GRU units. As local models,

they later employ the user behaviour to detect frauds as a deviation from normal behaviour.

In this thesis, we present a novel approach based on the ideas introduced by the Transformer model [5], a state-of-the-art model in the Natural Language Processing field. This approach was successfully used on other fields [6] but never applied to banking frauds. The proposed model is trained to predict the next transactions of the users. By doing so, learns to model the user's spending behaviour. An anomaly score is then drawn by comparing the predicted transaction with the actual transaction. The score is used to flag the frauds. These models are based on attention mechanisms which are proved capable of remembering long sequences of data [5]. Moreover, they are unsupervised multitasks learners [7] capable of learning even how to compose music [8] or play chess [9].

We also explore other architectures based on the same ideas. The approach presented in the previous paragraph is generative because it is trained to generate the next transaction of the user. By modifying the architecture of the model, we can obtain a discriminative model. The discriminative model is directly trained to discriminate between legitimate and fraud transactions. Both types of models share the same basic blocks. They only differ in the training goal and the arrangement of those blocks. This approach has been recently proposed by Clack et al. [10] with good results in the NLP field.

We perform a benchmark against several synthetic scenarios to validate the proposed approach. The scenarios are organised according to the classification proposed by Baggio [11]. In his work, Baggio proposes three different categories, which depends on the amount of information about the system an attacker would possess:

- **Black box**: The attacker does not have prior knowledge of the system. The frauds present on these scenarios are generated randomly according to fixed parameters defined beforehand. By doing so, we try to replicate real-world scenarios as a session hijacking attack, a persistent attack or credentials stealing attack. The proposed approach outperforms state-of-the-art models in the most complex scenarios of this category.

- **Grey box**: The attacker has information about previous transactions. We simulate a scenario in which the attacker trains a model with the data he has, which is called an oracle. The attacker runs the frauds trough the oracle, discarding the ones that are flagged as frauds. The result is a challenging scenario, in which our approach outperforms state-of-the-art models.

- **White box**: The attacker knows every detail of the system and uses this knowledge to build a copy of the system. Therefore, he runs the frauds through his copy before performing them in the real system. The proposed approach fails to cope with this attack, the attacker inserts frauds without being detected.

To summarize, the proposed model presents the following improvements over the state of the art:

- Fully exploits the past behaviour of the users. Taking into account the time dimension enables better detection performances.
- Resistance against concept drifts. As proven by [12], attention-based mechanism applied to the past transactions of the user improves the robustness of the model against concept drift.
- Adaptable history length. The model can deal with users having as little as one past transactions to users with thousands.
- Less hand-crafted features. By extending the neuronal network part of the model as much as possible we allow the optimiser to select more parameters of the model. The result is fewer hand-crafted features and better representation of the input, which typically increases performance [13].

Chapter 2

Motivation

The objective of fraud detection is to found a separation between fraudulent transactions and legitimates ones. A fraudulent transaction is defined as a transaction not executed by the user, but by an attacker with the intention of stealing money.

Frauds are difficult to detect because there is not a set of characteristics that defines them. Although not entirely accurate because frauds are not necessarily outliers, most fraud detection models define a fraud as a deviation from normal behaviour. Therefore, the most important aspect of a fraud detection system is how to model normal behaviour.

Existing solutions such as [14], [15] and [16] built a user profile through aggregated metrics on the users' past transactions. Those metrics are, among others, the average amount spent, the number of transactions issued, or the accumulated amount spent in one month. These metrics are easily interpretable by humans, which adds explainability to the final result. Although useful, these metrics are not optimisable by and algorithm. Representations of the input generated by neuronal networks are usually more performant [12], [13], [17].

The objective of the proposed approach is to build a better model of the user. Thanks to the advances made in the NLP field, our model is able to fully exploit the past transactions of the user to build a rich, complex user profile.

2.1 Problem Statement

To build a model capable of creating a user profile out of past transactions, we need to define an analogous problem, which can later be optimisable by and algorithm. Usually, the problem of choice for modelling is prediction [7]. In our case, the model is trained to predict the next transaction given the past transactions of the user. More formally, the model will take the last 1024 $(t_0, \ldots, t_{1023})$

transactions and will output a representation of the next transaction t'. The input size is kept at 1024, large enough to accommodate years of transactions while remaining computability feasible for our training setup. This parameter can be adjusted as needed. However, it must be a power of two to allow efficient computation [5].

Once we have obtained the transaction expected by the model t', we compare it with the actual transaction. The dissimilarities between them yield a probability of fraud. If the probability is higher than the threshold p^{th}, the transaction is marked as fraud.

Given a model architecture and a mathematical expression for the problem, the optimiser will found the correct weights for the model and the threshold p^{th}. The architecture of the model can be found in Chapter 4. The mathematical representation of the stated problem can be found in Section 4.2.7

2.2 Related work

Along this section, we review the literature publications related to our work. The first part of this section analyses of the most relevant models in the banking fraud detection field. The last part is devoted to NLP models, as our proposed approach is based on the ideas introduced by them.

2.2.1 Fraud detection

Fraud detection is an unbalance classification problem in which the positive class, the frauds, are underrepresented. Because this unbalance, fully supervised approaches are usually discarded due to their low performance [4]. Therefore, this section focuses on semi-supervised and no supervised models.

Fraud detection techniques can be categorised upon the type of input they take:

- Local models: The model is user-centric, is feed with metrics aggregated by user.
- Global models: The model is system-centric, tries to model the behaviour of the system.

Local models

Banksealer [18] is a semi-supervised model which builds a local, global and temporal profile for each user using methods with well-known statistical meaning. The local profile is used to quantify the anomaly of the incoming transaction with respect to the user behaviour. The global profile helps to group similar users, which mitigates the effect of users with fewer transactions. Lastly, the

temporal profile is employed to assess the anomaly of the spending pattern of the user. Thanks to the use of these profiles and meaningful statistics, the result of Banksealer is highly understandable by humans. Our proposed solution is based entirely on neuronal networks, which hinders explainability but enables the model to better optimise the feature representation of the input.

Recent models rely on neuronal networks specifically designed to deal with sequences of data, creating thus a representation of the user behaviour. Among them, LSTM and GRU based solutions are the more popular options. These units allow the system to process sequences of data by storing information into the network. Thanks to this technique, the neuronal network can save and update information that can later use.

Papale [19] propose an ensemble of Random Forest, XGBoost and LSTM to detect frauds. His main contribution is the LSTM network, which takes into account the last transactions of the user. Papale's model can only process users with 50 transactions or more, which left out an important amount of them. Our solution can process users that have from 1 transaction up to 1024 thanks to the use of attention mechanism.

Fraudmemory [12] adds attention on top of the LSTM output. Attention is a mechanism that allows the model to search in the input, selecting the more relevant information. Fraudmemory outperforms state-of-the-art solutions in terms of Precision, Recall and AUC. Our solution employes also attention, but applied directly to the input. LSTM has been proven a bottleneck for the flow of information inside neuronal networks [5]

Global models

Global models detect frauds by comparing them with the majority of the transactions. Samples that deviate from global behaviour are flagged as frauds.

Semi-supervised models exploit the fact that training data belongs to one class. These techniques train an autoencoder or a CNN network to extract relevant information about the legitimate examples in the data. The output of this model is fed to a supervised algorithm that performs the classification. The main goal is to find transactions that differ from normal ones.

FraudsDigger [16] uses an autoencoder trained to reconstruct legitimates transactions. Because the autoencoder has never seen a fraud, it will reconstruct them poorly. The reconstruction error is fed into a Random Forest classifier, which decides if the transaction is a fraud. Although FraudsDigger uses metrics related to users, its main functioning is based on finding outliers at a global level. Therefore, it is considered a global model. One shortcoming of this kind of solutions is the impossibility of training the model as a whole. The autoencoder is optimised to reconstruct transactions, not to detect fraud. Hence, it fails to extract rich differential features to detect outliers [4]. Our proposed solution is trainable end-to-end, which means that all layers are optimised to

perform fraud detection.

Unsupervised models are based on the probability distribution of the data, which uses later to spot anomalies [16]. Usual techniques consist of clustering or outlier detection. Among the most known techniques are k-NN [20], [21]. Although simple, k-NN performs well in comparison with more complex approaches, according to Campos et al. [22].

OC-SVM [23] is another unsupervised technique used in Fraud Detection. It is an SVM modified to find a separation plane that englobes all the legitimate samples. Frauds will fall outside this plane and therefore, they will be detected. The kernel used by OC-SVM is of great importance [24]. Gaussian kernels are usually prefered. Therefore, we compare our results against an OC-SVM with Gaussian kernel.

Another promising unsupervised technique is OC-NN [25], [26], which partially solves the problem present in Autoencoders. OC-NN models are able to extract a rich representation of the input optimised for fraud detection. Its functioning is similar to OC-SVM, in which both of them tries to found a separation hyperplane that englobes the legitimate samples. However, OC-NN exploits neuronal networks to extract information from the input. Although these models show promising results [4], the training times grow exponentially with the input dimension. Thanks to the use of attention, our model does not suffer from this issue.

2.2.2 Natural Language Processing

Natural Language Processing studies the interactions between machines and human language. This field has seen a dramatic change in recent years thanks to models based on the Transformer [5]. Moreover, Transformer-based models have been proven to be universal approximators of any sequence-to-sequence functions [27]. As the proposed model is based on the Transformer, this section will review its architecture and relevant details.

Embeddings

Natural language models rely on an embedding layer to translate words to numbers, usually called Word2Vec. The objective is to map the one-hot encoded word, which is a sparse high dimensional space, to a reduced dense space. There are two methods to archive this, both of them based on neuronal networks and proposed by Mikolov et al. [28]:

- **CBOW**: Common bag of words. Takes the context surrounding a word and tries to predict the original word.
- **Skip-gram**: Takes a word and tries to predict the context.

The weights of either of the two models configure a transformation between the word space and the representational space. Words with similar meanings are mapped closer. For example, Paris, Madrid, and Rome will be mapped close because they share meaning; they are all capitals. Sum and subtraction on this space are also related to meaning:

$$Paris + country = France$$
$$Rome + country = Italy$$
$$Madrid + country = Spain$$

The embeddings allow the following layers to know if two terms are similar or not. Because the vectors of two similar words are similar, the dot product between them will be higher. This property is used in the attention mechanism explained in the next section

Attention mechanism

Attention mechanishm were introduced by Bahdanau [29] and Luong [30]. Equation (2.1) shows a slightly modified version called scaled dot-product attention. In transformer models $K = V$. Attention equation takes a query Q and a set of values V. Then, the dot product between the query Q and each value in V is calculated yielding a matrix with higher numbers where query and values are similar. $softmax(\cdot)$ is applied to this matrix resulting in a matrix with values between 1 and 0 called attention score matrix. It indicates where the relevant information is. By multiplying the attention score $softmax(QK^T)$ by the set of values V the most relevant information is returned.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) \tag{2.1}$$

Equation (2.1) also contains the regularisation factor $\sqrt{d_k}$ to counteract large values of QK^T that can have a negative impact in training.

This mechanism is the heart of the model. It allows the model to search for information in long sequences without much computation overhead.

Architecture

Figure 2.1 shows the architecture of the Transformer. The original transformer model is a sequence-to-sequence model. Its input and output are both sequences. The model consists of two blocks: the encoder, on the left; and the decoder, on

the right. The encoder builts a representation of the input and passes it to the decoder. The decoder takes the information from the encoder and the output generated so far and generates the next item in the output sequence. The inner workings of both of them are similar. The main differences are the number of layers and the input they take.

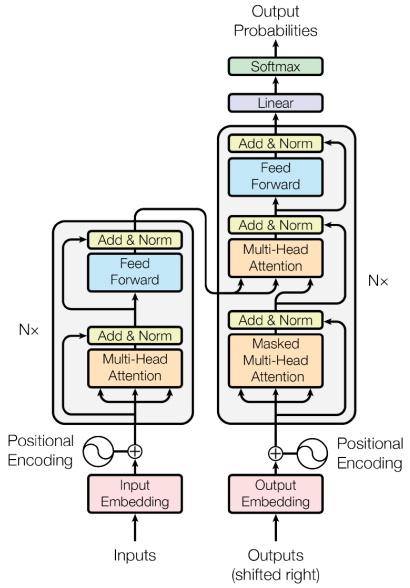

Figure 2.1: The Transformer architecture

The main blocks are the Input Embedding, Multihead Attention and the Feed Foward layer. The first one applies the concept of embedding explained in Section 2.2.2. The second one applies attention to the input, it is explained with greater detail in Section 4.2.2. Finally, the Feed Foward layer consists of two dense layers, more details can be found in Chapter 4.

The original transformer model excels in tasks like translation, where a sequence-to-sequence model is required. However, modern models based on the Transformer based their functioning on the encoder or the decoder part, not on both of them. This broadens the application of the model to other tasks as prediction or language modelling. For example, BERT [31] is based on the encoder, yielding state-of-the-art results for downstream tasks, and GPT-2 [7] is based on the decoder, with state-of-the-art results for text generation.

2.3 Goals and Challenges

The challenges are multiple, and some of them hard to solve:

- **Lack of data**: Financial datasets are hard to find due to privacy concerns. A large Italian bank has provided the dataset to develop this work. Unfortunately, many features are useless because of the anonymisation process.
- **Scarcity of frauds**: Frauds are rare by definition. In fact, the analysed dataset do not contain any fraud. The imbalance between classes seriously hurts the performance of supervised models.
- **Concept drift**: Users, as well as fraudsters, behave differently as time goes by.
- **Time dimension**: Time information is difficult to represent and understand by neuronal networks. Sequences of events are challenging to process, the firsts Recurrent Neuronal Networks suffered from the Vanishing Gradient Problem which hindered the ability of the network to learn [32].
- **Frauds are not outliers**: As discussed previously, frauds are not necessarily outliers, which hinders detection.

Most of these problems are solved by the architecture of the model itself. The transformed model has been engineered from the start to deal with sequences of data, which helps to deal with time-series data. Similar models to the one proposed in this thesis have proven robustness against concept drift thanks to the use of attention over the user past transactions [12]. Others are intrinsical to the problem at hand, and cannot be solved but somehow mitigated. For example, the lack of data is mitigated by using a smaller model.

The goal of this work is to provide a model able to detect the frauds by modelling user behaviour. By doing so, the model is more performant, resist concept drift and uses fewer hand-crafted features.

Chapter 3

Dataset

The dataset analysed contains the transactions of a large Italian bank. The transactions belong to two periods. One goes from December 2012 to September 2013 and the other goes from October 2014 to February 2015. This accounts for a total of 1,043,478, comprising 6195 different users. Each transaction has 31 features, most of them left blank or useless because of the anonymisation process. The dataset is the same analysed in [16] and [14]. The usual histograms, distribution plots and PCA analysis are already present on those works. Therefore, the following sections will centre on analysing the effect of time on data.

3.1 Exploratory analysis

There are 31 features, of which only 9 are useful. The amount is the main characteristic of a transaction. Figure 3.1 shows the amount distribution of all the transactions. The maximum amount allowed for a transaction is of 50,000, and the minimum is 0.01. The majority of the transactions have an amount between 0 and 2000. The average amount is of 1,783.39€. The graphic shows spikes in numbers like 1000, 1500, 2000 and so on. Which basically means that people prefer round numbers.

There are 1375 uniques ASNs. Figure 3.2 shows the distribution of the most frequent ASNs which accounts for 86% of the total. The most common one is 3269 which belongs to TIM, the major telecommunications operator in Italy. The seventh more common ASN is n./d. which represents missing data. This ASN will be treated as a valid ASN and will be fed to the model.

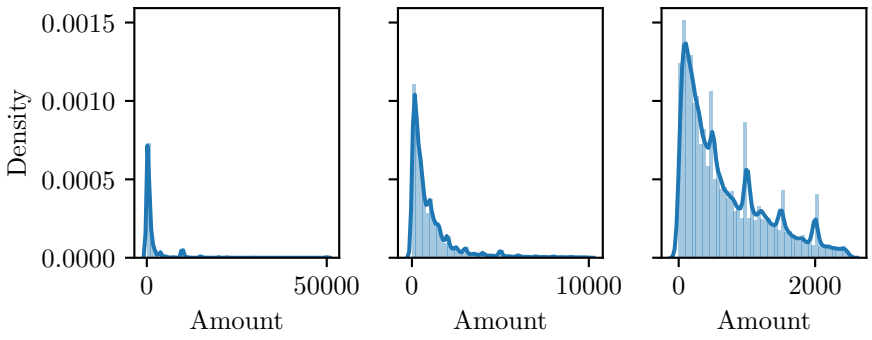

Figure 3.1: Amount distribution

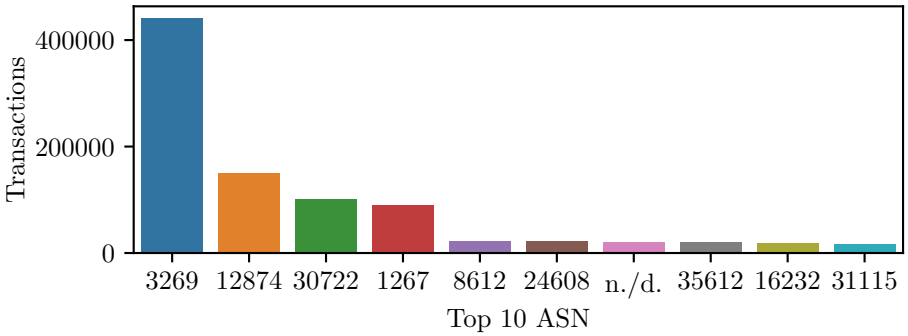

Figure 3.2: ASN distribution

3.2 Effect of the time variable

The proposed model takes into account sequences of transactions. The notion of time is important. All the transactions have a timestamp, which is composed of an hour, day, month and year. This section analyses which of these components are the most relevant ones.

Figure 3.3 shows the number of transactions and their amount during the day. Data is colourized by day of the week. The number of transactions issued during the day varies significantly depending on the hour. The majority of the transactions are issued during working hours. The behaviour of the users seems to be constant along the weekdays, following the same pattern in terms of amount and number of transactions. During weekends, the movement of money is less intense. The anomalous spike at 4:00 in Thursdays corresponds to a couple of transactions with a large amount. Due to very few transactions issued at night, outliers have a greater impact on the median and mean amount graphics.

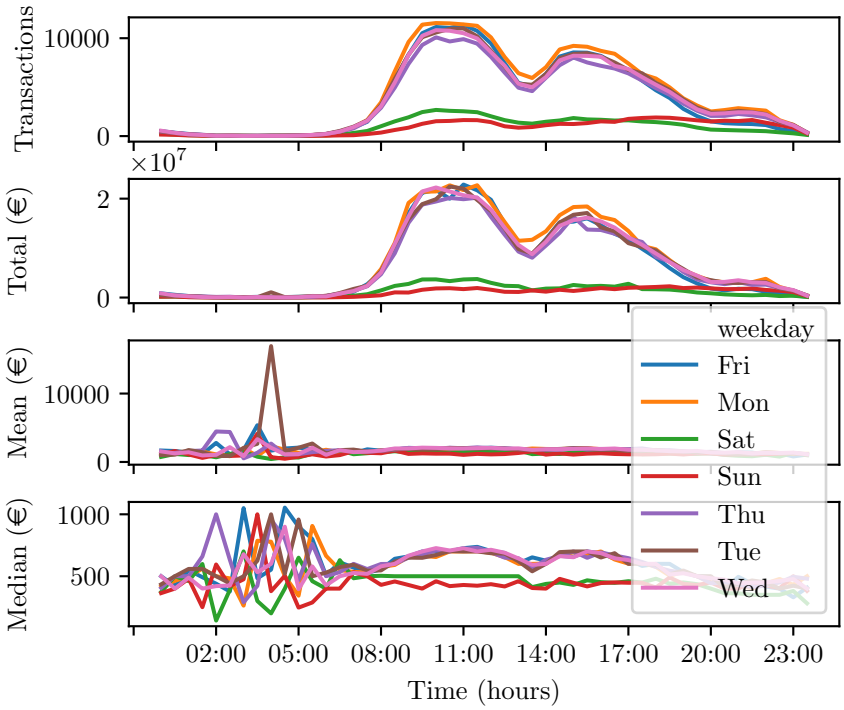

Figure 3.3: Distribution of amount over day

Figure 3.4 shows the number of transactions during the week. Data is colourized by month. During weekends users issue fewer transactions with less amount.

The behaviour during working days is more or less constant.

The behaviour is the same across all months. The differences in term of quantity of transactions are either due to holidays or incomplete data. The dataset does not have all the transactions issued during April, September and May. However, although fewer transactions, the mean amount of them is almost the same from month to month.

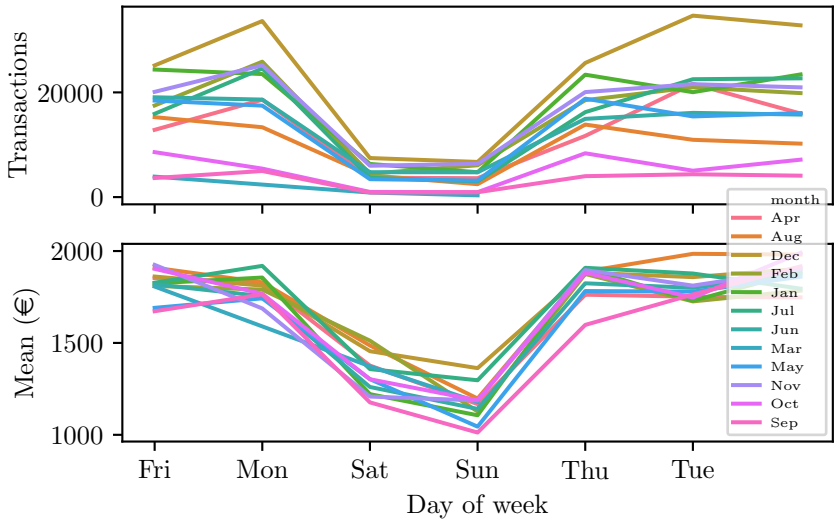

Figure 3.4: Distribution of amount over weekday

Figure 3.5 gives more details about the users' behaviour throughout the year. On the left part of the figure, it is shown the raw data without any modification. On the right, the day of the month has been adjusted to the day of the week. The 1st day of the month is always Monday. As can be seen, the behaviour of the users from month to month is similar and highly influenced by the day of the week. December seems to be the busiest month in terms of money movement while August is the calmer one.

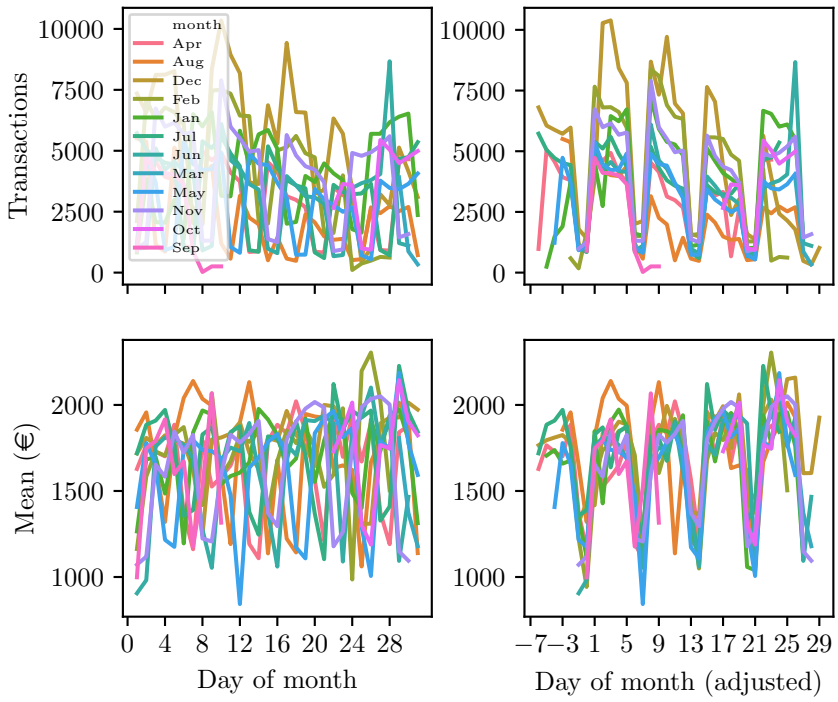

Figure 3.5: Distribution of amount over month

3.3 Spending and connection habits

Figure 3.6 represent the amount of the transactions issued by 50 users. As can be seen, most of the users issue transactions with similar amounts. Only a few users have more erratic behaviour, with amounts spanning all the range.

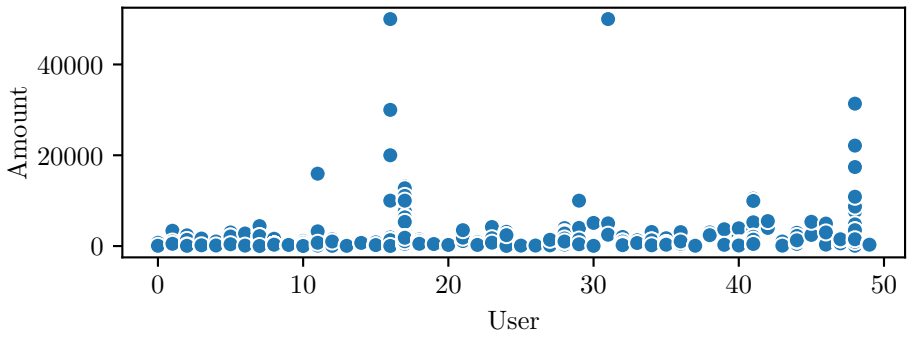

Figure 3.6: Amount of transactions of 50 users

Figure 3.7 represents the number of ASNs used by a user. Most of the users only use one ASN. But there is also a significant amount of them that uses two or three. This is probably to the use of the online bank account at home and work, in which the telecom operators may be different. The user can also have different operators on the mobile phone and at home. On the contrary, IP addresses vary much more due to dynamic addresses. Therefore ASN is a better parameter to define the connection of a user.

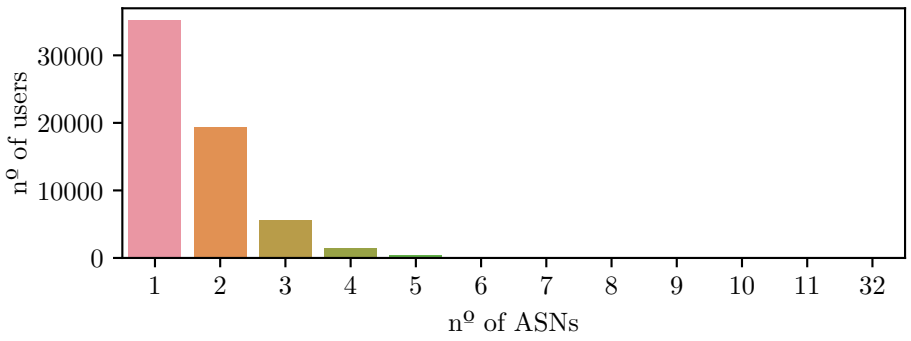

Figure 3.7: Number of ASNs used by the users

3.4 Feature selection

Bearing in mind the exploratory analysis, the most relevant features are:

- Time: Hour, day of the week, adjusted day of the month.
- Amount
- Connection related data: ASN

The month is not considered because of the lack of data. It is relevant to detect holidays but with only data from one year it is impossible to learn anything valuable. The amount is the most important feature, the behaviour of a user is determined by his/her spend profile. The ASN feature helps to include some information related to the Internet connection of the user.

Lastly, some *synthetic* features are added:

- International: 1 if the transaction destination is international.
- New IBAN: 1 if the transaction destination is a new IBAN for the user.

98% of the transactions are destinated to Italian IBANS. Including information about the destination country will add complexity to the model without bringing any benefit. To mitigate this issue a flag which indicates whenever the transaction is international is included.

If the fraudster pretends to steal some money, he needs to redirect the money to an IBAN controlled by him. Hence, all frauds are transactions directed to a new IBAN, unseen by the user. The new IBAN flag accounts for this fact.

3.5 Preprocessing

In order to feed the data to the model, some preprocessing is needed.

Time

The time components need special treatment. The hour and the day of the month are cyclic features. For instance, the last day of the month is close to the first day of the next month. Following the example set by Dignani [16], $sin(\cdot)$ and $cos(\cdot)$ are use to code each of the cyclic feature. Thusly, there are two dimensions for each feature. Equations 3.1 and 3.2 show the transformations applied to the hour.

$$h_1 = sin\left(\frac{2 * \pi * x}{23}\right) \tag{3.1}$$

$$h_2 = cos\left(\frac{2 * \pi * x}{23}\right) \tag{3.2}$$

Amount

The amount is a continuous variable that goes from 0.01 to 50,000. Transformer models only handle discretized variables. As seen on Figure 3.1, the amounts are highly imbalanced. Instead of using bins of equal width, a quantile strategy is used. Figure 3.8 shows the result of the discretization. Each bin has roughly the same number of points.

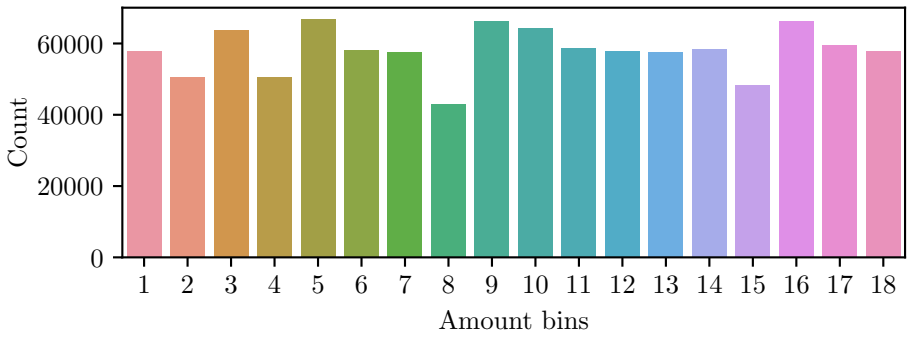

Figure 3.8: Discretization of amount feature

ASN

The ASN feature is already discrete. To facilitate the task of the embedding layer, the ASNs are reorganized and numbered from 1 to 1376.

Chapter 4

Approach

4.1 Approach Overview

The Transformer models [5] have demostrated an outstanding capacity of modelling human language [33] and they learn how to do so by predicting the next word given the past words. The problem stated in Section 2.1 is similar, replacing words with transactions.

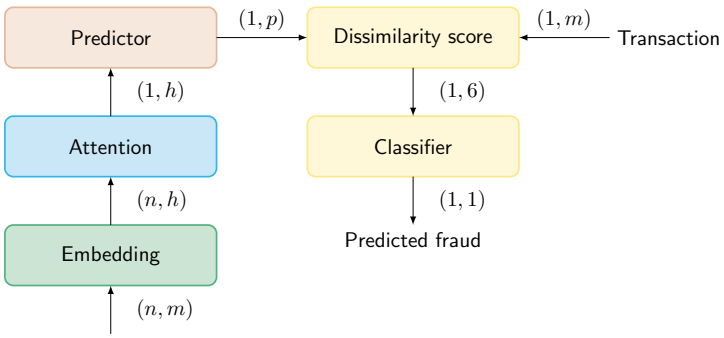

Figure 4.1: Overall architecture

Therefore, a Transformer-alike model is proposed. The architecture of the model is shown in Figure 4.1. Significant changes have been made to adjust the model to the particularities of the problem. Those changes together with the details of each block are discussed in Section 4.2.

4.1.1 Embeddings and Vocabulary

Oxford English Dictionary has 171,476 words [34]. Transformer models can deal with large vocabularies thanks to preprocessing and training. Preprocessing helps to reduce the vocabulary. There are several algorithms like BPE [35], WordPiece [36] or SentencePiece [36]. The idea is the same for all of them, break down words in smaller pieces. Training also helps. Transformer models are trained during days using huge text corpus. For instance, GPT-2 is trained using the text of 8 million web pages.

In the case of transactions, the question is how to build the vocabulary. There are several options:

- **Transaction**: Each transaction is a word
- **One vocabulary**: Each feature of the transaction is a word sharing the same vocabulary
- **Several vocabularies**: Each feature of the transaction is a word of a different vocabulary

The first approach is unfeasible due to the size of the vocabulary. The other two are similar to the preprocessing techniques used in Transformer models, split words into smaller pieces. The differences between them are the size of the vocabularies.

The second option would have a vocabulary of 173,124. A gigantic vocabulary taking into account the small dataset at hand.

The third one would result in three vocabularies of sizes 7, 1375 and 18. This is the only feasible approach and therefore, the chosen one.

4.1.2 Attention

The attention mechanism allows the model to search the input for useful information. Thanks to attention, the model can deal with long sequences of inputs. The limit has been set to 1024 transactions, which is only surpassed by very few users.

The implementation follows closely the one presented in [5], the original Transformer paper. However, the Transformer is a sequence-to-sequence model, conceived for language translation. Predicting the next transaction is a sequence-to-one problem. Hopefully, models like GPT-2 [7] are engineered also for prediction. Therefore, the proposed architecture is similar to the one used by GPT-2.

4.1.3 From Prediction to Fraud Detection

Some adjustments are needed to convert a predictive model to a generative one. The model outputs the probability vector for each feature. With that, it is easy to get the probability of a given transaction. Then the anomaly score is obtained by applying the $-log(\cdot)$ to the probability. Lastly, a meta-learner is trained to convert the anomaly scores of the features to the probability of fraud.

The techniques used to obtain the fraud probability from the prediction are well known and widely used in the literature. For instance, the use of $-log(\cdot)$ is applied in [6]. The use of a neuronal network as meta-learner is a popular ensemble method known as stacking [37].

4.2 Approach Details

The model consist on 47 layers arranged in different blocks depending on their function. Figure 4.2 shows a detailed diagram of the main blocks that compose the model. All the layers sum a total of 1,961,358 weights that will be optimised by the Adam optimiser.

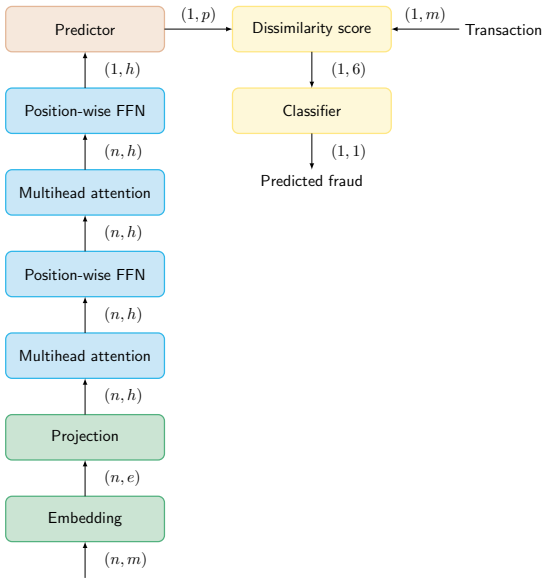

Figure 4.2: Detailed architecture

The following sections dissect the model. Each section contains the detailed architecture of each block alongside with an explanation of why that particular architecture has been chosen.

The main blocks of the architecture are the Multihead Attention and the Position-wise feed-forward network. Both of them are very similar to the original Transformer model [5]. These blocks are used several times in the architecture, but their inner working is the same.

The last section is devoted to the loss function. Choosing the right loss is of paramount importance to archive good results.

4.2.1 Embedding

The task of the embedding layer is to represent the input data more compactly. The input consist of one-hot encoded elements. Those elements are translated to a dense vector, with fewer dimensions. The output vectors have interesting

properties. Similar elements are mapped closer. This property helps the model to search and pay attention to the relevant elements. For instance, if the model is interested in transactions with amounts in the 5th bin, the search will return also transactions belonging to bins 4th and 6th as they are close in meaning. The layer learns the concept of similarity thanks to the context. Elements that usually appears with the same context are similar.

Figure 4.3 is an overview of the embedding architecture. There are different types of inputs. Therefore, a different embedding is used for each one. For the discrete inputs, a normal embedding layer is used. This layer works as described above. The continuous inputs like the hour and day are passed through.

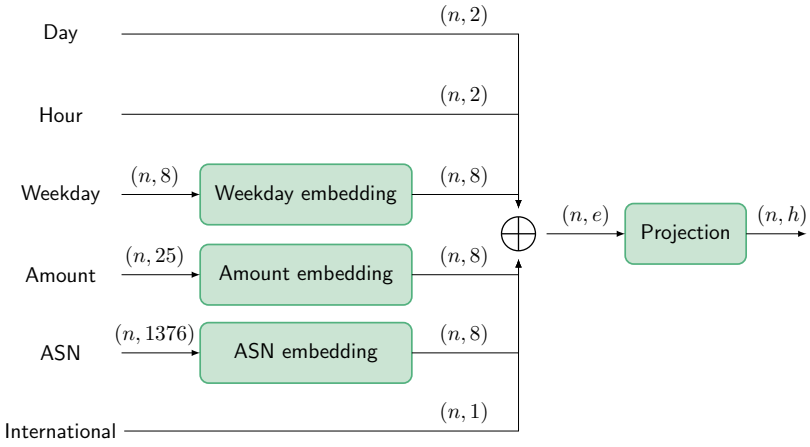

Figure 4.3: Embedding architecture

In a normal neuronal network, the inputs are fixed and each of them have weights than can be optimised. However, in models based on attention, the neuronal network is able to search in the input and retrieve the relevant data. Because of the search, the model does not longer know which was the position of the input he just found.

Hence, Transformer models use something called Positional Encoding. This mechanism encodes the position of the input in the input itself, adding a cosine and sine signals. Positional Encoding assumes that all the elements at the input are equidistant but transactions are not. Instead of using positions, timestamps are used. Similar to the Positional Encoding, cosines and sines are used to encode the information. Figure 4.4 shows the timestamp encoding for the 500 first transactions.

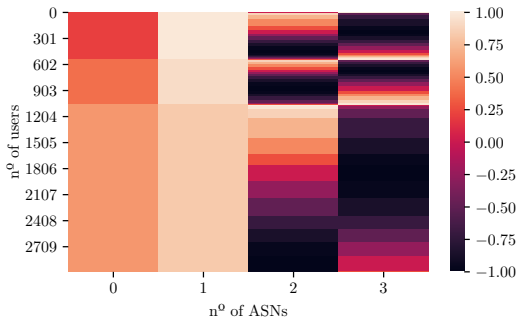

Figure 4.4: Time embeddings of the 500 first transactions

4.2.2 Multihead attention

Attention is the core of the model. The matrix calculated in Equation 4.1 indicates which positions of the input V are more relevant given the query Q. Attention is later multiplied by the value as shown in Equation 4.2. The result V' contains the input that is more relevant given V, K and Q. It worth noticing that V' will have the same dimensions of Q. In Transformer-alike models, $K = V$.

$$Attention(Q, K) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) \tag{4.1}$$

$$V' = Attention(Q, K) * V \tag{4.2}$$

There are different types of attention depending on how Q and V are calculated. In this case, all the blocks use self-attention except for the last one.

Self-attention is the case in which Q and V are obtained applying a linear transformation to the input. It is called self-attention because each input position pays attention to the other positions of the input. In this way, each input is enriched with the context around them. The output has the same dimensions as the input.

The last layer of the model consists of cross-attention. Instead of getting Q from all the input, only the last position is used. This forces V' to have the same dimension of Q, which is the dimension of one transaction. This idea is used in models like GTP-2 [7] with good results.

Figure 4.5 shows the attention layer. This layer is called Multihead attention. Instead of using only one *head*, several are used. Each *head* performs the Equations 4.1 and 4.2 on different parts of the input. Therefore, the model is able to pay attention to several positions at once with the same computational cost.

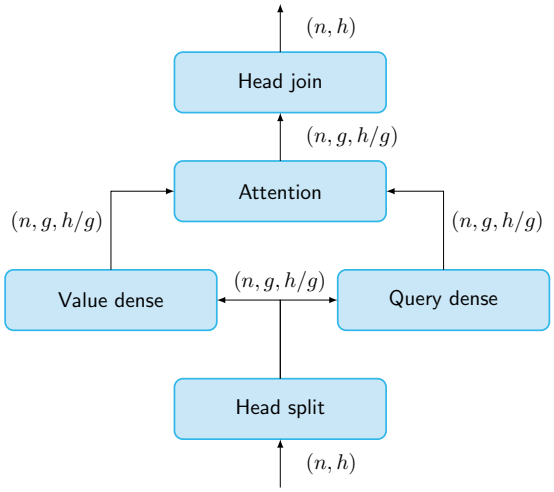

Figure 4.5: Multihead attention architecture

4.2.3 Position-wise feed-forward network

All the dense layers used in the model are linear except the ones used in this layer. The Point-wise feed-forward network adds non-linearities to the model.

Figure 4.6 shows its architecture. This operation can be seen as a non-linear transformation applied to each position of the input with the same parameters. Hence why it is called Position-wise.

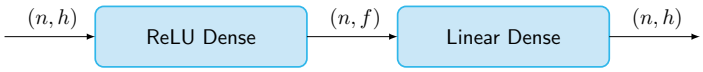

Figure 4.6: Position-wise FFN architecture

4.2.4 Predictor

The predictor consist on several dense layers. Each layer performs a linear transformation of the hidden state. Each of them is trained to predict the next transaction feature. In the case of the continuous variables, as day or month, MSE is used as the loss function. For the discrete ones instead, the layer is trained using Sparse Crossentropy. Figure 4.7 shows its architecture.

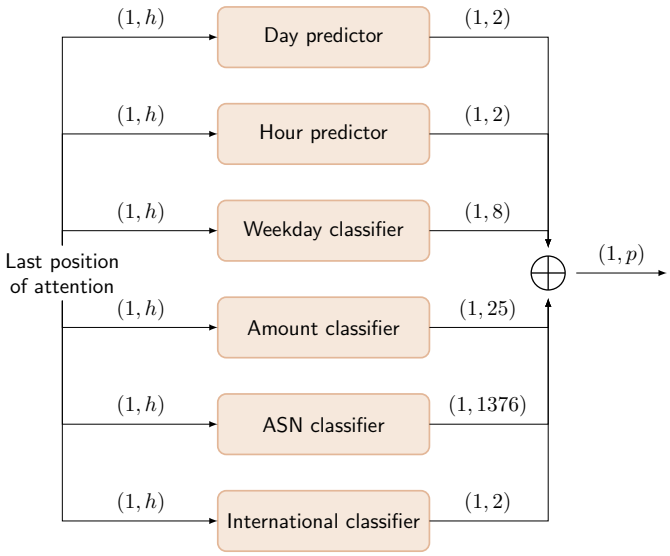

Figure 4.7: Predictor architecture

4.2.5 Dissimilarity score

This layer receives the prediction from the model and the current transaction. Outputs the probability of the current transaction given the prediction.

Figure 4.8 states the inner-working of the layer. The prediction given by the model consists of a vector of probabilities for each feature. This layer gets the probability that the current transaction has, feature by feature. Lastly, $-log(\cdot)$ is taken for each feature, yielding a dissimilarity score rather than a probability. This approach is similar to the one taken by [6].

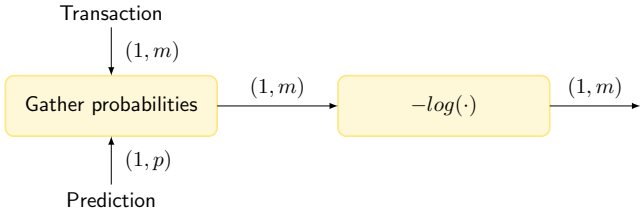

Figure 4.8: Dissimilarity score architecture

4.2.6 Classifier

The classifier can be seen as a meta-learner. Its function is to weight the dissimilarity scores of each feature to get a proper classification of frauds. Figure 4.9 shows the inner working of the layer. In practice, this is the definition of a dense layer with one neuron and so it is implemented.

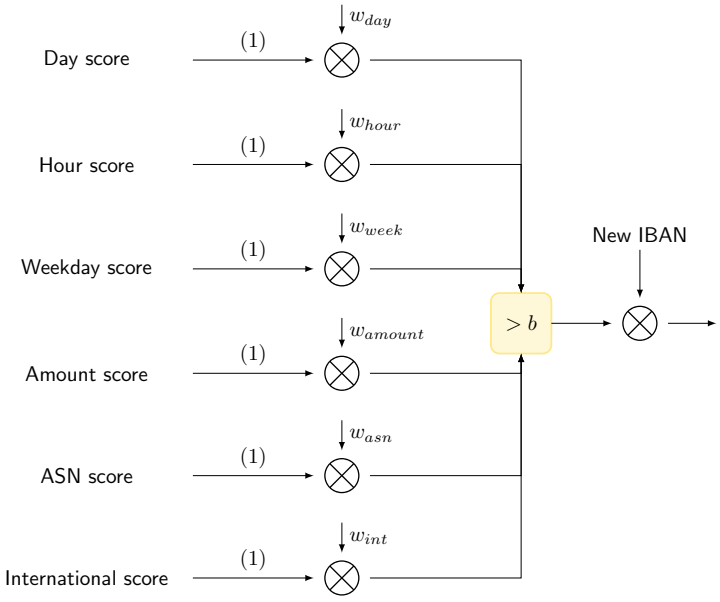

Figure 4.9: Classifier architecture

4.2.7 Loss function

Prediction

The first part of the model has the task of predicting the next transaction. Prediction is usually modelled as classification among all the possible options.

A transaction is composed of different features with different loss functions. Each feature contributes to the final loss function in equal manner:

- Day: Mean Square Error
- Hour: Mean Square Error
- Amount: Sparse Crossentropy
- Weekday: Sparse Crossentropy

- ASN: Sparse Crossentropy
- International: Binary Crossentropy

The different nature of the features induces the use of different loss functions. The day and hour features are continuous ones, and therefore MSE is used. The other features are categorical, which corresponds to a Crossentropy loss.

$$L_{total} = L_{day} + L_{hour} + L_{amount} + L_{week} + L_{asn} + L_{int} \tag{4.3}$$

The magnitudes of the losses are different as they account for different problems. For example, the ASN loss is larger than the day loss because it is a prediction between 1375 different options whilst the day MSE usually ranges between 0 and 1. This issue is mitigated by choosing Adam as optimiser, which scales the loss according to the learning rate [38]. Weighting the contribution of each loss has been tested with no good results. Therefore, the sum is left untouched.

Fraud detection

The second part of the model comprises the dissimilarity score and the classification layer. When it is in place, the output of the model changes and so should do the loss function.

The output of the model is the probability of fraud of the current transaction. It is a classification problem with two classes, thus Binary Crossentronpy is used.

Chapter 5

Implementation Details

5.1 System Architecture

From the implementation point of view, there are three different parts:

- Preprocessing
- Prediction
- Fraud detection

The preprocessing phase takes the original dataset and applies the transformations specified in Section 3.5 This process saves the preprocessed data in a specific format file, adequate to be consumed by the model.

The prediction is the most important part. It is arranged in the blocks described in Chapter 4. The most relevant layers are the embeddings, the attention and the prediction layers. This phase takes as inputs the data generated by the preprocessing data and outputs the probability of fraud for each incoming transaction.

The last part performs fraud detection comparing the transaction under analysis with the transaction predicted in the previous step.

5.2 System Details

This section starts by reviewing the setup needed to run the proposed approach. Details about the programming language, the technologies and the computation resources used are given.

Later, we discuss the specificities of the model implementation. This section provides an insight into the main parts that compose the proposed model: The preprocessing phase, the attention mechanism and the prediction layer.

5.2.1 Setup

The proposed solution is entirely written in python 3.6.9. The tools used to manipulate the data are written in plain python using well-known libraries as pandas or NumPy. The model is written in TensorFlow 2.3.0, a library for building neuronal networks.

The model is trained using a machine with 32GB, Intel(R) Xeon(R) CPU and a Tesla P100 16GB GPU. Training neuronal networks is a GPU-intensive task and therefore the used GPU is a measure of how complex the model is.

Lastly, Comet.ml is used to recollect and analyse the different metrics of the experiments. Comet.ml is an external service which facilitates the tracking and comparison of the different experiments. 626 experiments were executed and logged in this platform to arrive at the model proposed in this thesis.

5.3 Preprocessing

The preprocessing phase cleans, rearrange and discretize the original dataset. It is also capable of generating synthetic frauds for training or evaluation. It is implemented in a modular way, which allows choosing what information to include and how to insert the synthetic frauds. Listing 5.1 offers an overview of its functioning. First, function load at line 2 loads and clean the original dataset. Then, the data is split by user and the frauds are inserted at line 7. Lastly, the data is discretized at line 9 and written into a special TensorFlow format.

Listing 5.1: preprocessing.py

```
1  config = DataConfig()
2  data = load(config.source_files)
3  discretizer = Discretizer(config, data)
4
5  data = split_by_user(config, data)
6  data = add_is_new_iban(data)
7  data = insert_cross_frauds(config, data)
8  data = join(data)
9  data = discretizer.discretize(data)
10 data = split_by_user(config, data)
11
12 write_records(config.destination, data)
```

In the case of Listing 5.1, the preprocessing phase is configured to insert cross frauds, which are used to train the model. However, this part is configurable. There are seven different functions to generate and insert frauds, each of one inserting different types of frauds. Listing 5.2 shows an example of those functions: insert_random_frauds. This function is the simplest one, as only inserts frauds with random features. Lines from 5 to 15 perform the creation of the fraud.

Listing 5.2: frauds.py

```
1  def insert_random_frauds(config: DataConfig, contexts):
2      start_date = config.start_date
3      end_date = config.end_date
4
5      def _craft(item):
6          if _random_true(config.p):
7            i = item.index[-1]
8            item.at[i, 'amount'] = random.randrange(1, 50000)
9            item.at[i, 'Timestamp'] = _random_date(start_date, end_date)
10           item.at[i, 'ip_country'] = random.choice(config.countries)
11           item.at[i, 'iban_country'] = random.choice(config.countries)
12           item.at[i, 'asn'] = random.choice(config.asns)
13           item.at[i, 'is_new_iban'] = True
14           item.at[i, 'fraud'] = True
15       return item
16
17     context = map(_craft, contexts)
18     return tqdm(context, desc='Inserting random frauds')
```

Listing 5.3, on the contrary, shows a complex example. This function generates the adversarial attack analysed in the grey attack scenario. An oracle is created in line 4 and used at line 23 to assess the quality of the created fraud. If the new fraud is flagged by the oracle, it is discarded and another one is created.

Listing 5.3: frauds.py

```
1  def insert_grey_frauds(config: DataConfig, contexts, data):
2      start_date = config.start_date
3      end_date = config.end_date
4      oracle = Oracle(config, data)
5
6      def _craft(item):
7        if _random_true(config.p):
8        fraud = True
9
10       while fraud:
11          i = item.index[-1]
12          j = item.index[-2]
13          start_date = item.at[j, 'Timestamp']
```

```
14            asns = set(item.asn.values)
15
16            item.at[i, 'amount'] = random.randrange(500, 1000)
17            item.at[i, 'Timestamp'] = _random_date(start_date, end_date)
18            item.at[i, 'asn'] = random.choice(tuple(asns))
19            item.at[i, 'is_new_iban'] = True
20            item.at[i, 'iban_country'] = IT
21            item.at[i, 'fraud'] = True
22
23            fraud = oracle.predict(item)[0]
24
25        return item
26
27        contexts = map(_craft, contexts)
28        return tqdm(contexts, desc='Inserting grey frauds')
```

5.4 Prediction

Listing 5.4 shows the model main function, where the prediction takes place.

Listing 5.4: GeneratorWithMultiHeadAttention class

```
1         context, _ = inputs
2         days, hours, weekdays, amounts, asns, ibans_are_international =
              context
3
4         value_mask = self.value_mask_calculator(amounts)
5
6         # Embedding of values
7         ibans_are_international = tf.cast(ibans_are_international,
              tf.float32)
8         ibans_are_international =
              tf.expand_dims(ibans_are_international, axis=-1)
9
10        weekdays = self.weekday_embedding(weekdays)
11        amounts = self.amount_embedding(amounts)
12        asns = self.asn_embedding(asns)
13
14        context = tf.concat([days, hours, weekdays, amounts, asns,
              ibans_are_international], axis=-1)
15        context = self.projection_dense(context)
16
17        # Self attention
18        attention1 = self.self_attention([context, context],
              [value_mask, value_mask])
19        attention1 = self.normalize1(attention1 + context) # Skip
              connection
```

```
20          attention1_out = self.ffn1(attention1)
21          attention1_out = self.normalize2(attention1_out + attention1) #
                Skip connection
22
23          # Cross attention
24          query = attention1_out[:,-1] # The last item is the query
25          attention2 = self.cross_attention([query, attention1_out],
                [self.query_mask, value_mask])
26          attention2_out = self.ffn2(attention2)
27          attention2_out = self.normalize3(attention2_out + attention2)
28          attention2_out = tf.squeeze(attention2_out, axis=1)
29
30          # Classification
31          day = self.day_predictor(attention2_out)
32          hour = self.hour_predictor(attention2_out)
33          weekday = self.weekday_classifer(attention2_out)
34          amount = self.amount_classifier(attention2_out)
35          asn = self.asn_classifier(attention2_out)
36          iban_is_international = self.iban_classifier(attention2_out)
37
38          return day, hour, weekday, amount, asn, iban_is_international
```

The first line gets the context, which consists of the last 1024 transactions performed by the user. Lines from 6 to 15 applies the embeddings to the input, obtaining a representation with the properties discussed in Section 2.2.2.

Then self-attention is applied at lines 18-21. Recalling Section 4.2.2, attention takes a value V and a query Q and returns the values in V that are more similar to Q. The inputs of the function self_attention at line 18 are Q and V, in that order. In the case of self-attention, the query and the values are the same. This allows the model to enrich each position of the input with information from other positions.

Later, lines from 24 to 28 apply cross-attention. The mechanism is the same as in self-attention but this time $Q \neq V$. We take as query the most recent transaction of the user, as will be more relevant to predict the next transaction. This approach is similar to the one taken by GPT-2 [7] for text generation. Lastly, the prediction takes place taking into account all the information obtained thanks to the attention layers. Lines from 31 up to 36 are devoted to this task.

5.5 Fraud detection

Listing 5.5 shows how the prediction is used. Line 2 separates the context, the past transactions of the user, from last, the transaction under analysis. Line 3 executes the prediction model, which is shown in Listing 5.4. Then, the prediction and the transaction under analysis are compared, yielding a fraud score for each feature that is combined to get the final fraud probability at line

9.

Listing 5.5: Detection Header

```
        inputs = self.get_inputs()
        context, last = inputs
        probabilities = self.model(inputs)
        _, _, scores = self.improbability_score([probabilities, last])

        fraud = Dense(1, activation='sigmoid')(scores)
```

5.6 Attention implementation

Due to its importance to this work, we show in Listing 5.6 the implementation of the Multihead attention. Lines from 8 to 10 perform a linear transformation of the query, the key and the values. Later, they are split into different heads to finally apply the dot product at line 18. Lines from 21 to 28 join the result of all the heads and returns the result.

Listing 5.6: MultiHeadAttention class

```
      q, v = inputs
      k = v
      batch_size = tf.shape(q)[0]

      q_mask = masks[0]
      q_mask = tf.expand_dims(q_mask, axis=-1)

      q = self.wq(q) # (batch_size, seq_len, d_model)
      k = self.wk(k) # (batch_size, seq_len, d_model)
      v = self.wv(v) # (batch_size, seq_len, d_model)

      q = self.split_heads(q, batch_size) # (batch_size, num_heads,
          seq_len_q, depth)
      k = self.split_heads(k, batch_size) # (batch_size, num_heads,
          seq_len_k, depth)
      v = self.split_heads(v, batch_size) # (batch_size, num_heads,
          seq_len_v, depth)

      # scaled_attention.shape == (batch_size, num_heads, seq_len_q,
          depth)
      # attention_weights.shape == (batch_size, num_heads, seq_len_q,
          seq_len_k)
      scaled_attention, attention_weights =
          scaled_dot_product_attention(
     q, k, v, masks)

```

```
21        scaled_attention = tf.transpose(scaled_attention, perm=[0, 2, 1,
              3]) # (batch_size, seq_len_q, num_heads, depth)
22
23        concat_attention = tf.reshape(scaled_attention,
24                  (batch_size, -1, self.d_model)) # (batch_size,
                        seq_len_q, d_model)
25
26        output = self.dense(concat_attention) # (batch_size, seq_len_q,
              d_model)
27        output = output * q_mask
28        return output
```

Chapter 6

Experimental Validation

6.1 Goals

The goal of the proposed experiments is to demonstrate the effectiveness of the model in different scenarios. The experiments are classified depending on the knowledge the attacker has, as proposed in [11]. There are three categories, each of them with a different goal:

- **Black box**: The attacker has no knowledge of the system. This experiment validates the model in the most common scenario and also proves its resilience to concept drift.

- **Grey box**: The attacker has some knowledge of the system. This experiment proves the robustness of the model against adversarial attacks. It is not usually conducted and therefore, of great value.

- **White box**: The attacker knows everything about the system. This experiment estimates the performance of the model under the challenging scenario in which the attacker uses the proposed model to attack the system.

A comparison against state-of-the-art models is also carried. It is difficult to compare different state-of-the-art models due to differences in the training and evaluation sets. For example, a model trained in transactions of a Chinese bank [12] will considerably differ from our model trained on Italian users. Also, the features available could be different. However, there are works in the literature based on the same dataset as ours. We will compare our results against one of them.

6.2 Dataset

The dataset is first preprocessed according to what it is explained in Section 3.5 which involves cleaning, discretization and grouping by user. Then the dataset is split into three different sets: The training set used to train the model, the validation test used to assess the performance of the model at each set of the training, and the test set used to get the final results of the model.

The training and the validation sets are used to teach the model how to predict the next transaction of the user, and therefore no frauds are injected at this phase. Later, we teach the model to differentiate between transactions belonging to the user under analysis and other transactions. To do so, we reuse the same training and validation sets mixing the last transactions of the users.

The test set is only used to create the different scenarios. Synthetic frauds are injected using different rules and techniques. The details are explained in Section 6.3.1.

6.3 Experimental Setup

6.3.1 Training

During training, the model learns how to predict the next transaction of a user and to use the prediction to get a fraud probability of the transaction under analysis.

The first step is to teach the model to predict the next transaction. This process involves the first blocks of the model: the embedding, the attention and the predictor blocks shown in Figure 4.1. During this phase, these blocks learn how to model user behaviour by trying to predict the next transaction. We use the training set without any modification, as we only need legitimate transactions to do the training.

The second step is to teach the model to use the prediction to spot frauds. This process involves the last blocks: the dissimilarity score and the classifier blocks. These blocks account for a small part in the model. Therefore, they need fewer iterations to learn. In this phase, we reuse the same training dataset but we inject frauds into it. To avoid using the same frauds of the testing scenarios we use transactions from other users as frauds. In this way, the model learns to distinguish between a transaction performed by the user under analysis and a transaction that not follow the user's behaviour.

6.3.2 Metrics

Fraud Detection is a classification problem in which the classes are very unbalanced. Therefore, the usual classification metrics, such as accuracy, could lead to wrong conclusions. For example, a model predicting that all the transactions are legitimate in a dataset containing 1% of frauds have a 99% of accuracy but will not detect frauds. Hence, we propose the use of the following metrics, more appropriate to assess the quality of a model in an unbalance scenario as fraud detection

Precision and Recall

Precision measures the number of predicted frauds that are actually frauds. It is similar to the accuracy of the positive class.

$$Precision = \frac{tp}{tp + fp} \tag{6.1}$$

Recall measures how many frauds are detected from the total number of frauds. Provides an indication of missed frauds.

$$Recall = \frac{tp}{tp + fn} \tag{6.2}$$

Both metrics are related. Often, increases in one metric imply decreasing the other.

Curves

There are two curves:

- Receiver operating characteristic, or ROC, shows the behaviour of the system for different thresholds.
- Precision-Recall, or PR, shows the tradeoff between precision and recall.

ROC curve relates True Positive Rate with False Positive Rate. Given a desired FPR, the model is better as higher the TPR is. A common metric to measure the quality of the curves is the area under the curve or AUC. Higher values indicate better models.

Matthews correlation coefficient

Also called phi coefficient, Matthews correlation coefficient measures the correlation between the observed and predicted classification.

As shown in Equation 6.3.2, MCC takes into account all the metrics given by the confusion matrix. It is regarded as one of the most reliable statistics for imbalance problems [39].

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{6.3}$$

F-score

F-score is a statistical accuracy measure. It is calculated from the Precision and Recall, as shown in Equation 6.4.

$$F_\beta = (1 + \beta^2) * \frac{precision * recall}{(\beta^2 * precision) + recall} \tag{6.4}$$

β is a weighting parameter. In our case, we use the F_1 score, which is the same as the harmonic median between Precision and Recall.

False Positive Rate

The False Positive Rate, also known as fall-out, is the ratio between the number of misclassified negative samples and all negatives samples. Equation 6.5 shows its calculation, FP is the number of False Positive, and TN the number of True Negatives.

$$FPR = \frac{FP}{FP + TN} \tag{6.5}$$

6.3.3 Testing scenarios

There are five datasets for the testing scenarios. Four of them are black-box attacks and the last one is the grey box attack.

The first four are build following the ideas presented in FraudsDigger [16]:

- **Stealing**: Simulate a phising attack in which the user's credentials are stolen. The amount transferred is very high, the connection can be originated from a national or foreign IP.
- **Hijacking**: Simulate a Man-in-the-browser attack. The connection details are legitimate. The amount transfer is high, the transfer happens no later than ten minutes from a legitimate one.
- **Persistent**: Simulates the infection of a banking trojan. The frauds have a low amount and the connection details are legitimate.

- **Mix**: It consists of all the previous scenarios mixed together.

The last one is the grey box attack, which consists of an adversarial attack. In this scenario, the attacker posses half the data available. With this information, the fraudster trains a model that is called an oracle. Before inserting a fraud, the attacker runs it through the oracle and discards the ones detected.

6.4 Black box attacks

Table 6.1 compares the proposed model trained as usual against the baselines algorithms trained on the same dataset in which are tested. Be aware of the unfairness of the comparison, the baseline algorithms are specifically trained for each scenario while the proposed model is trained on a general dataset.

Scenario	Model	Precision	Recall	AUROC	AUPR	MCC	F1	FPR
Persistent	Proposed	**0.751**	0.499	**0.834**	**0.652**	0.559	**0.600**	**0.030**
	Random F.	0.206	**0.760**	0.603	0.477	**0.702**	0.403	0.037
	Isolation F.	0.256	0.251	0.560	0.159	0.123	0.253	0.092
	OC-SVM	0.019	0.019	0.503	0.013	0.019	0.019	0.107
	k-NN	0.174	0.178	0.514	0.087	0.028	0.176	0.099
Stealing	Proposed	0.869	0.986	**0.997**	**0.987**	0.910	0.924	**0.001**
	Random F.	**0.994**	**0.995**	0.996	0.845	**0.993**	**0.994**	**0.001**
	Isolation F.	0.839	0.832	0.902	0.699	0.807	0.835	0.0197
	OC-SVM	0.816	0.910	0.937	0.720	0.836	0.860	0.032
	k-NN	0.802	0.822	0.893	0.675	0.778	0.812	0.030
Hijacking	Proposed	0.864	0.968	0.978	**0.922**	0.897	0.913	0.007
	Random F.	**0.983**	**0.979**	**0.989**	0.873	**0.984**	**0.981**	**0.001**
	Isolation F.	0.644	0.694	0.547	0.611	0.672	0.668	0.024
	OC-SVM	0.243	0.250	0.555	0.152	0.109	0.247	0.097
	k-NN	0.362	0.364	0.624	0.259	0.248	0.363	0.080
Mixed	Proposed	**0.828**	0.833	**0.936**	**0.871**	0.801	0.830	**0.006**
	Random F.	0.790	**0.928**	0.889	0.731	**0.836**	**0.853**	0.010
	Isolation F.	0.644	0.694	0.815	0.547	0.611	0.668	0.023
	OC-SVM	0.410	0.454	0.672	0.328	0.331	0.431	0.059
	k-NN	0.486	0.518	0.713	0.393	0.415	0.501	0.052

Table 6.1: Metrics on the test scenarios, different training sets.

The proposed model works better in the most complex scenario, in which the past behaviour of the user is useful to distinguish between legitimate transactions and frauds.

In the case of hijacking and stealing the frauds are easily clustered together, a task in which Random Forest excels. However, the proposed model works better than the Random Forest in the mixed scenario. Baseline algorithms have problems in detecting frauds that not belong to the same class.

Table 6.2 is a fair comparison. All the models use the training set stated in Section 6.3.1. The proposed model is better in almost all scenarios.

The lower performance of baseline algorithms is due to concept drift. Traditional algorithms have a hard time detecting frauds that have not been seen before.

Scenario	Model	Precision	Recall	AUROC	AUPR	MCC	F1	FPR
Persistent	Proposed	**0.751**	**0.499**	**0.834**	**0.652**	**0.559**	**0.600**	0.030
	Random F.	0.737	0.387	0.657	0.507	0.525	0.507	**0.014**
	Isolation F.	0.234	0.135	0.528	0.099	0.123	0.171	0.032
	OC-SVM	0.169	0.103	0.506	0.052	0.015	0.128	0.041
	k-NN	0.181	0.103	0.510	0.099	0.072	0.033	0.033
Stealing	Proposed	0.869	**0.986**	**0.997**	**0.987**	0.910	0.924	**0.001**
	Random F.	**0.926**	0.978	0.982	0.804	**0.943**	**0.951**	0.014
	Isolation F.	0.695	0.788	0.961	0.152	0.801	0.739	0.031
	OC-SVM	0.667	0.809	0.956	0.166	0.779	0.731	0.041
	k-NN	0.684	0.801	0.959	0.158	0.793	0.738	0.034
Hijacking	Proposed	0.864	**0.968**	**0.978**	**0.922**	0.897	0.913	**0.007**
	Random F.	**0.927**	0.954	0.970	0.794	**0.929**	**0.940**	0.013
	Isolation F.	0.696	0.921	0.952	0.687	0.791	0.793	0.032
	OC-SVM	0.407	0.336	0.624	0.270	0.269	0.368	0.041
	k-NN	0.668	0.918	0.918	0.647	0.739	0.773	0.034
Mixed	Proposed	0.828	**0.833**	**0.936**	**0.871**	**0.801**	**0.830**	**0.006**
	Random F.	**0.902**	0.756	0.871	0.703	0.800	0.823	0.015
	Isolation F.	0.602	0.707	0.814	0.532	0.589	0.650	0.033
	OC-SVM	0.467	0.474	0.691	0.365	0.382	0.470	0.043
	k-NN	0.576	0.669	0.793	0.503	0.552	0.619	0.035

Table 6.2: Metrics on the test scenarios, same training set.

Figures 6.1, 6.3, 6.4 and 6.2 shows the ROC curves of each model. All the models have been trained in equal conditions, i.e., using the same dataset.

Regarding the persistent (Figure 6.1) and mix (Figure 6.2) scenarios, the proposed model is clearly better. The model behaves better for all the FPR values. The curve is similar in both cases due to the persistent frauds.

Because of the curved shape, there are probably two groups of frauds in these scenarios. The easy ones, which corresponds to the first ramp, and the hard ones which belong to the second ramp.

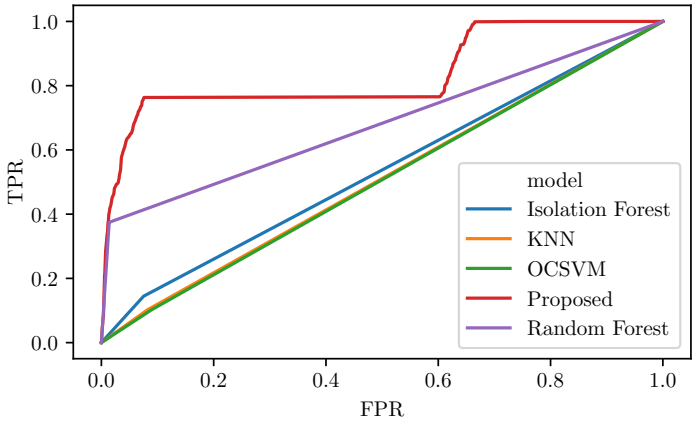

Figure 6.1: Models' ROC on the persistent scenario

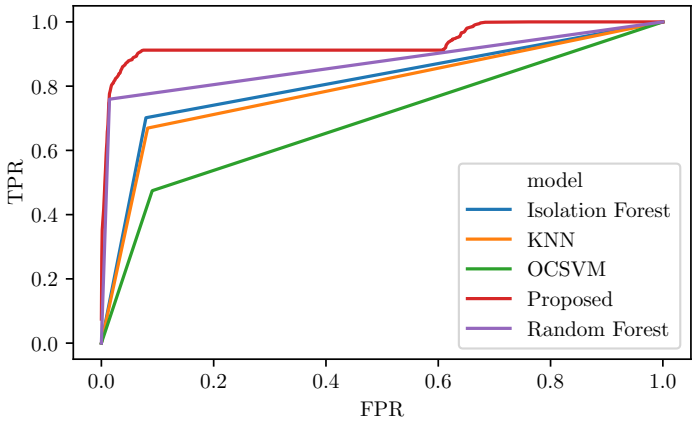

Figure 6.2: Models' ROC on the mix scenario

Figure 6.3 and Figure 6.4 shows the ROC curves for stealing and hijacking scenarios. In these scenarios, almost all the models perform similarly.

The proposed model works better than baseline algorithms for very low values of FPR. Because of that, the proposed model has higher AUROC.

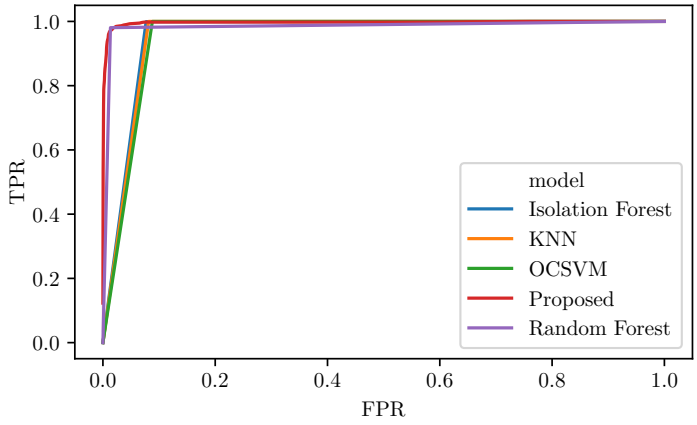

Figure 6.3: Models' ROC on the stealing scenario

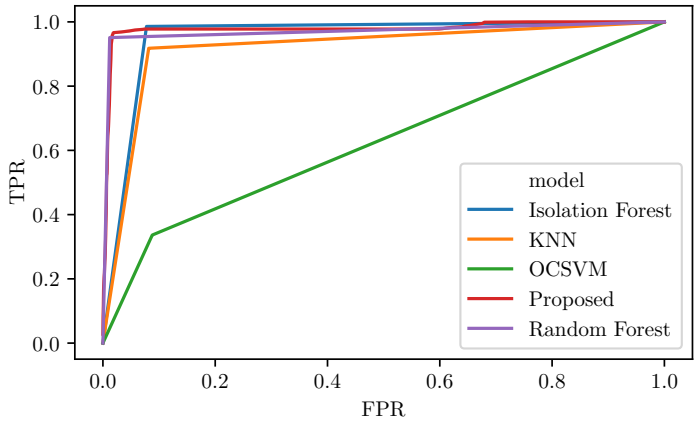

Figure 6.4: Models' ROC on the hijacking scenario

Worth noticing the poor performance of OC-SVM, surpassed even by k-NN.

6.5 Grey box attacks

Grey box attacks consist of attacks performed by an attacker with information about the target system.

All the transactions going from December 2012 to September 2013 are provided to the attacker which uses the data to train a Random Forest classifier. This

classifier will be treated as an oracle. The attacker will try the fraud in the oracle before issuing it to our model. If the oracle flags the fraud, the attacker will change the transaction and will try again It is reasonable to think that the attacker will choose a Random Forest to be the oracle, it is considered state of the art for fraud detection.

Table 6.3 summarises the result obtained for each model in the grey box test scenarios. The proposed model outperforms by far the baseline algorithms. The recall of the Random Forest is heavily hurt, which makes sense taking into account that the oracle is a Random Forest.

Model	Precision	Recall	AUROC	AUPR	MCC	F1	FPR
Proposed	**0.769**	**0.556**	**0.867**	**0.691**	**0.606**	**0.645**	**0.020**
Random F.	0.548	0.096	0.541	0.237	0.182	0.163	0.026
Isolation F.	0.094	0.044	0.483	0	-0.047	0.060	0.033
OC-SVM	0.146	0.084	0.497	0	-0.007	0.107	0.042
k-NN	0.141	0.075	0.496	0	-0.011	0.098	0.035

Table 6.3: Grey attack scenario

The results show how beneficial can be having different approaches to Fraud Detection. The proposed model suffers less from the grey box attack because is based on user modelling, while the oracle and baseline algorithms rely on finding a separation plane between frauds and legitimate transactions.

6.6 White box attacks

In a white box attack, the attacker knows everything about the system. Therefore, the attacker can reproduce the model he tries to defeat.

This kind of attacks is difficult to detect. The fraudster can try unlimited attacks in the local model before performing an attack in the real model. It is enough for the attacker to found one fraud that is not detected by the local model to succeed

The proposed model is a generative one, which renders the attack even easier. The fraudster can use the model to generate a fraud that will be identical to the one predicted by the system. Therefore, the fraud will not be detected.

6.7 Generative versus discriminative approach

The proposed model is a generative model. It generates the expected transaction probabilities. Those probabilities are later used to detect fraud.

There are other models which are discriminative. Those models are trained to discriminate between frauds and legitimate transactions. Random Forest or OC-SVM are examples of discriminative models.

Following the architecture of the generator, a discriminative model is proposed. The differences between the two models are in the last layers of the model and in the training. The discriminative model does not have the Predictor nor the Dissimilarity score shown in Figure 4.2. Regarding training, the generator is trained for prediction whilst the discriminator is trained for classification. Also, the discriminative model uses cross-attention instead of self-attention. It uses the incoming transaction as a query to search in the past transactions.

The results of both models are reported in Table 6.4. Generally speaking, the generative approach performs better than the discriminative one.

Scenario	Model	Precision	Recall	AUROC	AUPR	MCC	F1
Persistent	Generator	**0.751**	0.499	0.834	**0.652**	**0.559**	**0.600**
	Discriminator	0.549	**0.509**	**0.886**	0.578	0.446	0.528
Stealing	Generator	**0.869**	**0.986**	**0.997**	**0.987**	**0.910**	**0.924**
	Discriminator	0.674	0.869	0.972	0.896	0.799	0.759
Hijacking	Generator	**0.864**	**0.968**	**0.978**	**0.922**	**0.897**	**0.913**
	Discriminator	0.672	0.862	0.966	0.882	0.779	0.755
Mixed	Generator	**0.828**	**0.833**	0.936	**0.871**	**0.801**	**0.830**
	Discriminator	0.622	0.743	**0.938**	0.800	0.690	0.677

Table 6.4: Discriminator versus Generator, same training set.

6.8 State of the art comparison

Banking information is not public due to obvious reasons. Because of the lack of standard databases and benchmarks, is difficult to perform a fair comparison between different models.

The work of Michele Papale [19] uses the same dataset used in this work but with different benchmark scenarios. To overcome this issue, the performance of Random Forest is set as the baseline. The models will be compared in terms of improvement over the Random Forest results.

Table 6.5 summarises the results for the mix scenarios, where all type of frauds are taken into account.

Model	Precision	AUROC	AUPR	MCC	F1	FPR
Papale's model	0.422	0.975	0.0.872	0.588	0.583	0.145
Baseline	0.505	0.961	0.803	0.634	0.649	0.098
Improvement	-0.083	0.014	0.069	-0.046	- 0.066	0.046
Proposed model	0.828	0.936	0.871	0.801	0.830	0.006
Baseline	0.902	0.871	0.703	0.800	0.823	0.015
Improvement	**-0.074**	**0.065**	**0.168**	**0.001**	**0.007**	**-0.009**

Table 6.5: Comparison between LSTM essemble and the proposed model.

The improvements of both models over the baseline are close, even more taking into account the nature of the comparison. Worth mentioning the poor performance of both models in terms of precision, falling below the baseline performance.

The proposed model performs slightly better than the ensemble proposed in [19]. Transformer models are better than LSTM modelling sequences of data.

It is also important to notice that the proposed model is able to deal with users with very few transactions while the model proposed by Michele Papale needs at least 50 transactions per user.

Chapter 7

Conclusions and Future Works

7.1 Limitations and Future Work

7.1.1 Limitations

The model shares some limitations with the NLP models. The need for a large dataset and vocabulary are the most burdensome. Another drawback is the need for labelled data because the proposed model is not unsupervised.

Lack of data

Modern NPL models rely on enormous models. Figure 7.1 shows the evolution in terms of the size of the NLP models. The first models had 66 million weights, nowadays there are models with 8.3 billion parameters. The model proposed in this thesis has only 2 million parameters.

The leverage of these models is the transfer learning. The models are trained during days on gigantic datasets. Once trained, the model is published. Anyone can fine-tune the model for a specific task and archive state of the art results thanks to pretraining.

In this case, the size of the model is limited by the amount of data available. With the data at hand, a larger model will overfit quickly. The proposed architecture is able to scale to hundreds of millions of parameters. Modern datacenters can train these models in a matter of days. The problem is the lack of data.

Figure 7.1: Evolution in number of weights of NLP models

Vocabulary

The power behind an NLP model is its ability to model language. Thanks to the language, the model can reason about the problem and provide solutions.

A limit vocabulary restrains the performance of the model. For instance, the quantity of international transactions in the dataset is really low. It is impossible for the proposed model to have a good performance when dealing with them.

This problem is related to the previous one. The root is the lack of data. The techniques discussed in Section 4.1.1 help by reducing the vocabulary size. But to apply those techniques more data is needed.

Supervised training

Fraud detection does not play well with supervised techniques. Frauds are by definition rare, and supervised techniques usually requires a labelled and balanced dataset.

The generative part of the model is trained in an unsupervised way, the only data needed are legitimate transactions, which are common in banks' databases.

However, the last part of the model is discriminative. This part needs a labelled balanced dataset, which is difficult to obtain.

This problem could be overcome using only the generative part and selecting each day the transactions with larger dissimilarity score.

7.1.2 Future Works

Dealing with the limitations stated in Section 7.1.1 should be considered the principal line of work to follow. More data can greatly improve the performance of the model. Nevertheless, there are a couple of issues worth future investigation.

Esemble with Random Forest

This thesis presents a novel model for Fraud Detection, based on modelling user behaviour. As can be seen in Chapter 6, Random Forest has a very good performance. Therefore, an ensemble joining the proposed model and a Random Forest should outperform both of them.

This approach is common in literature [19] and usually works fine.

Applying the model to other fields

The target of this thesis is the detection of fraudulent transactions. NLP models have very good performance in downstream tasks [7]. Future work should further investigate how this model could be adapted to other tasks.

Fraud detection in credit card payments should be a good starting point. The inputs have a lot of similarities with the transactions. A model trained into transactions should not have any problem adapting itself to credit card data.

7.2 Conclusions

The purpose of this thesis was to present a novel approach to Banking Fraud Detection. The proposed model has been able to overcome the limitations of previous models. It was capable of dealing with concept drift, it was fully optimisable, and completely exploited the past behaviour of the users. The proposed approach not only fulfilled all the desiderata but also had better performance than baseline algorithms in complex scenarios and against adversarial attacks.

This work took ideas from Natural Language Processing and applied them to Fraud Detection. The attention mechanism allowed the model to deal with transactions going from very few to thousands, being very valuable for behaviour modelling. The embeddings and the choice of a suitable vocabulary was also key to archive a good performance. This work proved the applicability of NLP techniques to the modelling of user behaviour. Fraud detection is one of its numerous applications, but many others deserve study. User profiling or credit card fraud are examples of future lines of work.

The future of Banking Fraud Detection is standardisation and transfer learning. The field needs standard benchmarks like ImageNet [40] for image classification

or GLUE [41] for NLP models. It is a daunting task and will require open datasets, which are difficult to obtain because of the sensitivity of the data. Withal, it would bring enormous advantages. Pretrained models could be fine-tuned for specific tasks in a matter of days, allowing financial institutions and researchers to surpass new frontiers.

Appendix A

Ethical, Economical and Environmental Factors

A.1 Introduction

The work proposed in this thesis can have a large impact on different aspects of the society.

Everything related to financial institutions needs to be carefully analysed. Users and companies rely on the correct functioning of banks to develop their activities. Modern society is based on the constant exchange of money, and the proposed system affects the flow of money. Therefore, a depth analysis needs to be carried out.

A.2 Ethical concerns

The proposed system is an automated one, trained without human supervision. The system learns how to model user behaviour from the data at hand. The concept of *normal* or *fraud* is drawn from the dataset.

This kind of models usually raises ethical concerns. For example, Apple Card is under investigation for sex discrimination [42]. Apple uses an automated system to impose credit limits to their users. The system has learned from the data to assign a lower limit to women.

The proposed model is trained to model user behaviour. The transactions that not fit with the model of the user will be flagged as fraud. The model will be more influenced by the average users because there are more of them. Users with strange behaviours will have more probability to be flagged as fraud.

This problem is mitigated by human intervention. Every flagged transaction is reviewed by a human operator. Nonetheless, outsider users will suffer from delays.

A.3 Economical concerns

Because of the very nature of the problem, the proposed solution has an economical impact. However, the economical consequences of a fraud detection system are limited because it is limited to flags frauds.

But this would be temporal. The proposed model can also be used to predict user behaviour. Banks could rely on this fact to allocate fewer funds per user, enabling them to free resources for investing. Nevertheless, the current models are not accurate enough to allow this kind of operation.

A.4 Enviromental concerns

The proposed model is an Artificial Neuronal Network. These models are well-known because are compute-intensive, which harm the climate.

To have rough estimation, GPT-3 [7] which is a state-of-the-art model with 175 billion parameter needs $3.14 * 10^{23}$ FLOPS fo training. Taking into account the performance of Google's TPU which is 4.17 watts per TFLOP [43] gives as $1.31 * 10^{15}$ Watts.

The proposed model has millions of parameters instead of billions and has been trained for a short period. The expected environmental impact is much lesser. However, computation power brings environmental effects.

A.5 Conclusions

The automatisation of the process always brings challenges. Among them, and often disregarded, are social challenges.

In our case, the most relevant ones are related to ethics. Users have the right to access to their funds. Denying it by means of an automated tool have important connotations.

Machine Learning models are heavily influenced by data. These models often learn to replicate the same prejudices that we have. The only way to deal with the problem is by mean of human intervention, which defeats the very objective of automatisation.

Bibliography

[1] Statista. (2019). "Share of people using internet banking in great britain 2007-2019," [Online]. Available: https://www.statista.com/statistics/286273/internet-banking-penetration-in-great-britain/ (visited on 06/11/2020).

[2] A. P. Kyle Marchini, "2019 identity fraud study: Fraudsters seek new targets and victims bear the brunt," *Javelin Strategy*, 2019.

[3] "Market research report," *MarketsandMarkets*, 2018.

[4] R. Chalapathy and S. Chawla, "Deep Learning for Anomaly Detection: A Survey," *arXiv e-prints*, arXiv:1901.03407, arXiv:1901.03407, Jan. 2019. arXiv: 1901.03407 [cs.LG].

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *arXiv e-prints*, arXiv:1706.03762, arXiv:1706.03762, Jun. 2017. arXiv: 1706.03762 [cs.CL].

[6] A. Brown, A. Tuor, B. Hutchinson, and N. Nichols, "Recurrent Neural Network Attention Mechanisms for Interpretable System Log Anomaly Detection," *arXiv e-prints*, arXiv:1803.04967, arXiv:1803.04967, Mar. 2018. arXiv: 1803.04967 [cs.LG].

[7] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[8] G. Branwen. (2020). "Gpt-2 folk music," [Online]. Available: https://www.gwern.net/GPT-2-music (visited on 06/11/2020).

[9] S. Alexander. (2020). "Https://slatestarcodex.com/2020/01/06/a-very-unlikely-chess-game/," [Online]. Available: https://slatestarcodex.com/2020/01/06/a-very-unlikely-chess-game/ (visited on 06/11/2020).

[10] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators," *arXiv e-prints*, arXiv:2003.10555, arXiv:2003.10555, Mar. 2020. arXiv: 2003.10555 [cs.CL].

[11] L. Santini, "Evasion attacks against banking fraud detection systems," 2019.

[12] Y. Kunlin, "A memory-enhanced framework for financial fraud detection," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2018, pp. 871–874. DOI: 10.1109/ICMLA.2018.00140.

[13] L. Nanni, S. Ghidoni, and S. Brahnam, "Handcrafted vs. non-handcrafted features for computer vision classification," *Pattern Recognition*, vol. 71, pp. 158–172, 2017, ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2017.05.025. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320317302224.

[14] A. Baggio, "Fraudbuster: Time-based analysis of internet banking fraud," 2016.

[15] A. Biondani, "Fraudhunter: A supervised fraud detection tool for internet banking transactions," 2016.

[16] A. Dignani, "Fraudsdigger: An active learning tool for online banking fraud detection," 2018.

[17] L. Cai, J. Zhu, H. Zeng, J. Chen, and C. Cai, "Deep-learned and hand-crafted features fusion network for pedestrian gender recognition," in *Proceedings of ELM-2016*, J. Cao, E. Cambria, A. Lendasse, Y. Miche, and C. M. Vong, Eds., Cham: Springer International Publishing, 2018, pp. 207–215, ISBN: 978-3-319-57421-9.

[18] M. Carminati, R. Caron, F. Maggi, I. Epifani, and S. Zanero, "Banksealer: A decision support system for online banking fraud analysis and investigation," *Computers & Security*, 2015, ISSN: 0167-4048.

[19] M. Papale, "An ensemble approach for banking fraud detection," 2020.

[20] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," *SIGMOD Rec.*, vol. 29, no. 2, May 2000, ISSN: 0163-5808. DOI: 10.1145/335191.335437. [Online]. Available: https://doi.org/10.1145/335191.335437.

[21] F. Angiulli and C. Pizzuti, "Fast outlier detection in high dimensional spaces," *Proceedings of the Sixth European Conference on the Principles of Data Mining and Knowledge Discovery*, vol. 2431, pp. 15–26, Aug. 2002. DOI: 10.1007/3-540-45681-3_2.

[22] G. O. Campos, A. Zimek, J. Sander, R. J. G. B. Campello, B. Micenková, E. Schubert, I. Assent, and M. E. Houle, *On the evaluation of unsupervised outlier detection: Measures, datasets, and an empirical study*, Jul. 2016. DOI: 10.1007/s10618-015-0444-8. [Online]. Available: https://doi.org/10.1007/s10618-015-0444-8.

[23] B. Lamrini, A. Gjini, S. Daudin, F. Armando, P. Pratmarty, and L. Travé-Massuyès, "Anomaly detection using similarity-based one-class svm for network traffic characterization," Aug. 2018.

[24] A. Bounsiar and M. G. Madden, *Kernels for one-class support vector machines*, 2014.

[25] R. Chalapathy, A. K. Menon, and S. Chawla, *Anomaly detection using one-class neural networks*, 2019. arXiv: 1802.06360 [cs.LG].

[26] L. Ruff, R. Vandermeulen, N. Görnitz, L. Deecke, S. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," Jul. 2018.

[27] C. Yun, S. Bhojanapalli, A. Rawat, S. Reddi, and S. Kumar, "Are transformers universal approximators ofsequence-to-sequence functions?," 2019.

[28] T. Mikolov, K. Chen, G. Corrado, and J. Dean, *Efficient estimation of word representations in vector space*, 2013. arXiv: 1301.3781 [cs.CL].

[29] D. Bahdanau, K. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate*, 2014. arXiv: 1409.0473 [cs.CL].

[30] M.-T. Luong, H. Pham, and C. D. Manning, *Effective approaches to attention-based neural machine translation*, 2015. arXiv: 1508.04025 [cs.CL].

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

[32] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, pp. 107–116, Apr. 1998. DOI: 10.1142/S0218488598000094.

[33] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[34] O. U. Press. (). "How many words are there in the english language?" [Online]. Available: https://www.lexico.com/explore/how-many-words-are-there-in-the-english-language (visited on 06/11/2020).

[35] R. Sennrich, B. Haddow, and A. Birch, *Neural machine translation of rare words with subword units*, 2015. arXiv: 1508.07909 [cs.CL].

[36] M. Schuster and K. Nakajima, *Japanese and korean voice search*, 2012.

[37] J. Rocca, *Ensemble methods: Bagging, boosting and stacking*, 2019. [Online]. Available: https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205 (visited on 08/24/2020).

[38] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2014. arXiv: 1412.6980 [cs.LG].

[39] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, Dec. 2020. DOI: 10.1186/s12864-019-6413-7.

[40] S. V. Lab. (). "Imagenet," [Online]. Available: http://www.image-net.org/ (visited on 09/02/2020).

[41] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 353–355. DOI: 10.18653/v1/W18-5446. [Online]. Available: https://www.aclweb.org/anthology/W18-5446.

[42] N. Vigdor. (). "Apple card investigated after gender discrimination complaints," [Online]. Available: https://www.nytimes.com/2019/11/10/business/Apple-credit-card-investigation.html (visited on 10/03/2020).

[43] M. Tyson. (). "Google benchmarks its tensor processing unit (tpu) chips," [Online]. Available: https://hexus.net/tech/news/industry/104299-google-benchmarks-tensor-processing-unit-tpu-chips/ (visited on 10/03/2020).