Itsy Bitsy SpiderNet:
Fully Connected Residual Network for Fraud Detection

Sergey Afanasiev[1,2] svafanasev@edu.hse.ru
Anastasiya Smirnova[1,2] aasmirnova_24@edu.hse.ru
Diana Kotereva[1,2] dmkotereva@edu.hse.ru

[1]*Faculty of Computer Science, National Research University Higher School of Economics, Russia*
[2]*Department of Statistics and Research, Renaissance Credit Bank, Moscow, Russia*

Abstract

With the development of high technology, the scope of fraud is increasing, resulting in annual losses of billions of dollars worldwide. The preventive protection measures become obsolete and vulnerable over time, so effective detective tools are needed. In this paper, we propose a convolutional neural network architecture SpiderNet designed to solve fraud detection problems. We noticed that the principles of pooling and convolutional layers in neural networks are very similar to the way antifraud analysts work when conducting investigations. Moreover, the skip-connections used in neural networks make the usage of features of various power in antifraud models possible. Our experiments have shown that SpiderNet provides better quality compared to Random Forest and adapted for antifraud modeling problems 1D-CNN, 1D-DenseNet, F-DenseNet neural networks. We also propose new approaches for fraud feature engineering called B-tests and W-tests, which generalize the concepts of Benford's Law for fraud anomalies detection. Our results showed that B-tests and W-tests give a significant increase to the quality of our antifraud models. The SpiderNet code is available at https://github.com/aasmirnova24/SpiderNet

1. Introduction

The development of high technologies contributes not only to the growth of corporations and world economies, but also to the development of fraud, which leads to losses of billions of dollars every year around the world.

In 2018 year, eight Indian banks incurred $1.3 billion in losses in a fraud case involving Kingfisher Airlines founder Vijay Mallya[1]. In another case, the Agricultural Bank of China faced losses of $497 million after being defrauded by employees of billionaire Guo Wengui[2].

Hacker attacks are another global problem. In 2019, the FBI issued an official announcement that global losses from fraudulent Business Email Compromise (BEC) reached $26 billion during the period from June 2016 to July 2019[3].

Another growing threat is social engineering, which has hit Russian bank customers seriously. According to the official data of the Bank of Russia, losses of Russian banks' clients from card fraud reached $130 million in 2020, which is 10 times higher than similar losses in 2017[4].

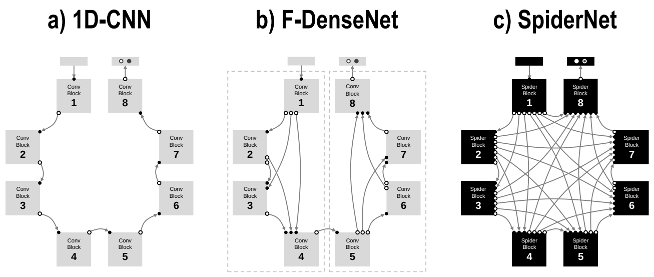

Figure 1: Convolutional neural network architectures designed for fraud detection: a) 1D-CNN, b) F-DenseNet, c) SpiderNet

Anti-fraud tools can be roughly divided into directive, preventive and detective. Directive tools such as instructions and warnings work like a scarecrow and only affect untrained fraudsters. Preventive tools help prevent fraud, but over time, fraudsters adapt and find ways to get around them. Detective tools are essential to detect fraud and minimize losses if fraud has not been prevented. Statistical approaches and machine learning methods are used to develop detective tools. However, there are a number of unresolved problems in this area, such as instability and low generalizing ability of antifraud models, as well as high privacy of domain expertise [6].

On the other hand, in recent years, we have seen outstanding advances in deep learning and successful application of neural networks to practical tasks such as computer vision [28, 16, 17, 57] and natural language processing [15, 52, 44, 42, 53]. This gives us hope that innovative ideas proposed in deep learning will help to remove some of the issues in fraud detection modeling.

In this paper, we propose a convolutional neural network architecture SpiderNet designed to solve fraud detection problems. We noticed that convolutional and pooling layers principles are very similar to the methods of manual processing of information by anti-fraud analysts during investigations. In addition, skip-connections used in convolutional networks [16] make it possible to use features of various power, including fraud scores from external providers.

Our proposed technique allows us to increase the generalizing ability of anti-fraud models and is an important advantage of SpiderNet over classical machine learning

[1] https://www.theguardian.com/world/2020/apr/20/kingfisher-airlines-tycoon-vijay-mallya-loses-appeal-extradition-india
[2] https://www.reuters.com/article/us-china-corruption-tycoon-idUSKBN1900DL
[3] https://www.ic3.gov/Media/Y2019/PSA190910
[4] https://cbr.ru/analytics/ib/fincert/#a_119487

methods and popular neural network architectures. We show that SpiderNet provides better quality compared to Random Forest and convolutional 1D-CNN and F-DenseNet network architectures adapted for fraud modeling (Figure 1). Moreover, comparing the SpiderNet results with the classic 1D-DenseNet architecture, we show that the feature locality property is lost while working with tabular data but remains crucial for images. Therefore, the technique of transferring CNN architectures from the CV domain, which is popular in applied tasks, does not give the best result. This confirms the thesis that the application of neural networks to applied tasks requires a deep understanding of the domain area, and it is important to adapt neural network architecture to the specifics of the task.

In addition to the SpiderNet architecture, in this paper we propose new approaches for developing anti-fraud rules called B-tests and W-tests. Our feature engineering method is based on the ideas of identifying manipulation in financial statements using Benford's law [4]. Our proposed approaches can be generalized to any data type, which allows developing B-tests and W-tests for any fraudulent schemes, and not just for identifying accounting manipulations, where Benford's law is applicable [37, 40, 38]. Our results showed that B-tests and W-tests give a significant increase to the quality of our fraud detection models.

To assess the quality of fraud detection models, we use the AUC ROC and Average Precision (AUC PR) metrics, as well as developed by our team business metric PL (Prevented Losses) that allows estimating the funds saved from internal fraud. Our proposed PL metric allows us to solve an important for the industry issue of assessing the economic efficiency of models developed for industrial using.

We train, evaluate and compare our SpiderNet with other algorithms on two datasets – private and public. The private dataset contains data on loan applications and fraud by POS partners of a large Russian bank, one of the top 50 Russian banks in terms of assets. The public dataset was obtained from an online payment fraud detection competition organized by Ant Financial Services Group[5] [6].

Testing SpiderNet on two datasets with different types of fraud (internal and transactional) gives us reason to believe that our proposed methods will work well for other types of fraud, because SpiderNet is based on the general concepts of fraudulent behavior formulated by Edwin Sutherland and Donald Cressey in criminology [49] and Gary Becker in behavioral economics [3].

The rest of the paper is structured as follows:

- In section 2, we describe a general theoretical model of fraudulent behavior, provide a classification of anti-fraud tools, describe the principles of developing fraud detection models, and outline current problems in this area;
- In section 3, we provide review of the current state of neural networks for image classification and fraud detection problems;
- In section 4, we describe the general intuition of the SpiderNet architecture and present the schema of the Spider-Block;
- In Section 5, we describe methods for the automated development of B-tests and W-tests antifraud rules;
- In Section 6, we describe our experiment: provide characteristics of samples, feature engineering methods, data preprocessing methods, tricks for training models, hyperparametres of algorithms and metrics for models' quality assessment;
- In section 7, we demonstrate the results of the experiments;
- And in section 8, we summarize, draw general conclusions and outline open questions for future research.

2. Fraud Detection

The foundations of modern criminology and the theory of white-collar fraud were laid nearly 100 years ago by Edwin Sutherland and his student Donald Cressey [49], who are considered some of the most influential criminologists of the 20th century. Sutherland and Cressy proposed to consider the crime not from the position of criminal law, as was customary, but from the position of sociology, applying basic sociological concepts in criminology.

Four decades later, Nobel laureate Gerry Becker, using the principle of economics imperialism, described the economic model of crime [3], according to which crime can be viewed as an activity that some people choose rationally, comparing the expected benefits and expected costs:

$$(1-\pi) * U(W_C) - \pi * S > U(W_L) \tag{1}$$

where π – the probability of being caught (assessed by the criminal, i.e. subjectively);
$U(\cdot)$ – individual utility function;
S – penalties incurred in the event of capture (for example, a fine or criminal punishment);
W_C – proceeds of crime;
W_L – income from legal activities.

The left side of inequality (1) characterizes crime-related elements; the right side is the utility from legal earnings. Logically, when inequality is satisfied, the individual, other things being equal, will prefer to break the law.

To minimize losses from fraudulent activities, companies need to develop and implement anti-fraud tools that affect the components of inequality (1). In particular, the total losses from fraud can be reduced by increasing the probability π and decreasing the W_C component by increasing fraudsters' costs for bypassing anti-fraud protection.

Anti-fraud actions can be divided into 3 types[7]:

- *Directive* – instructions, regulations, training materials, contracts, etc.;
- *Preventive* – tools that are aimed at preventing fraudulent activities (locks, safes, passwords, etc.);
- *Detective* – tools used to detect fraud (anti-fraud rules and models, investigation techniques, etc.).

[5] https://dc.cloud.alipay.com/index#/topic/data?id=4
[6] https://www.kaggle.com/gmhost/atec-anti-fraud/version/2

[7] The Information Security Standards (CISSP) use more granular typing: preventative, deterrent, detective, corrective, recovery, compensation, directive, administrative, logical/technical, physical.

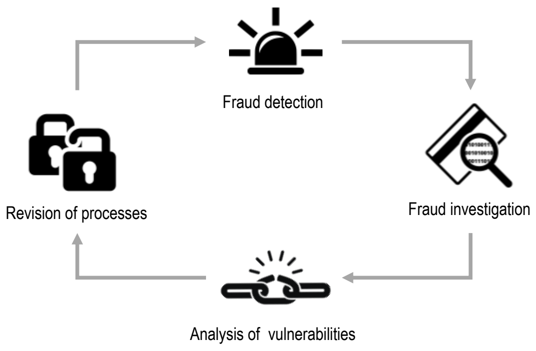

Figure 2: Arrangement of anti-fraud processes in the bank according to the principles of the scientific method.

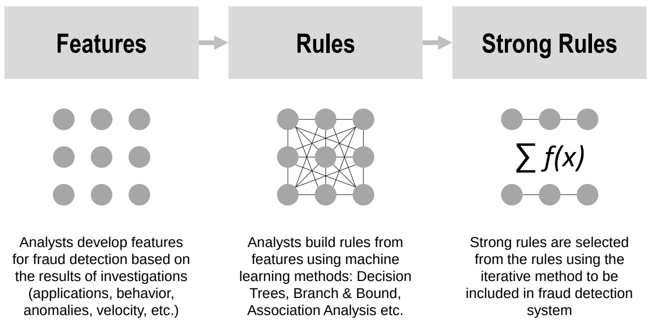
Features
Rules
Strong Rules
$\sum f(x)$
Analysts develop features for fraud detection based on the results of investigations (applications, behavior, anomalies, velocity, etc.)
Analysts build rules from features using machine learning methods: Decision Trees, Branch & Bound, Association Analysis etc.
Strong rules are selected from the rules using the iterative method to be included in fraud detection system

Figure 3: Scheme of developing strong rules for fraud detection.

Preventive tools are the most effective, but fraud is highly adaptive, which means that fraudsters constantly come up with new ways how to get round the company security. That is why the fight against fraud is compared to "confrontation of armor and projectiles". In order to face this constant arms race, the company anti-fraud sector requires an integrated, systems approach. One of the main principles of the system approach is the arrangement of anti-fraud processes in the form of a cyclic scheme according to the principles of the scientific method.

The scheme in Figure 2 shows how anti-fraud technologies are developed and improved. One of the main ideas of the scheme is that fraud and counteraction to fraud (anti-fraud) constantly influence each other [54]. It means that, according to the scheme, the company should regularly review its anti-fraud processes, modifying and improving them. Therefore, detective tools that allow detecting new fraudulent schemes and vulnerabilities in processes are an important part of the anti-fraud system in a company.

The detective tools are based on strong rules developed by experts and antifraud models built on these strong rules. Strong rules development consists of three key stages (Figure 3):

1) At the first stage, simple features and rules are developed by experts;
2) At the second stage, complex rules are combined using arithmetic and logical operations over simple features and rules;
3) At the third stage, strong rules with a high predictive power for detecting fraud are selected from complex rules.

The general principle of developing strong rules is similar to the method of collecting evidence in forensic science, when all kinds of evidence are first collected, and then, based on the combinations of the collected evidence, a crime is proved. A similar principle is at the heart of convolutional neural networks, in which sequential convolution operations form complex features, and pooling operations filter out noise, i.e. select strong features that are good at predicting the target variable. This is why convolutional networks can be considered a powerful and intuitive algorithm for developing fraud detection models.

3. Related work

3.1. CNN Architectures

Over the past decade, convolutional neural networks (CNNs) have made a breakthrough in computer vision problems solution [43]. The basic principles of CNN's work were borrowed from the works of Hubel and Wiesel, who studied the visual cortex in the 1950s and 1960s [24, 25]. The first CNN architecture was proposed by Yann LeCun in the late 1980s [32], and in the late 1990s LeCun's research group developed the LeNet-5 architecture [31], which consisted of convolutional and pooling layers that perform the function of implicit regularization of the neural network.

A breakthrough in computer vision came in 2012, when the deep convolutional neural network AlexNet won the ILSVRC computer vision competition in image classification problems [28]. The authors noticed that increasing CNN depth has an implicit regularization effect, which later became mainstream and was exploited in such architectures as VGG [48], NiN [35], Inception and GoogleNet [50], ResNet [16] and others.

In 2014, Google team showed excellent results at ILSVRC, proposing the GoogLeNet architecture [50], a feature of which were Inception blocks with battlenecks and 1x1 convolutions, which allowed to increase the number of convolutional channels, i.e. the width of the neural network. This technique also became popular and found its application in the architectures WRN [61], Xception [9], ResNeXt [60], MobileNets [21], NASNet [63, 64], etc.

In 2015, Microsoft Research team proposed another breakthrough architecture – ResNet [16] that used skip connections, which create an ensemble effect. This technique proved to be very effective and was later used in WRN [61], Xception [9], ResNeXt [60], FractalNet [30], DenseNet [22], etc.

In 2019, Google Research team published an EfficientNet approach [51] to scale deep CNNs. Authors noticed that the key characteristics of convolutional architectures, such as depth, width and resolution, depend on each other, therefore, for scaling, it is necessary to select the optimal combination of these parameters. Later, the EfficientNet approach allowed the participants to take the top places in the Deepfake Detection Challenge on Kaggle, which was organized by Facebook in early 2020.

In addition to the developed architectures, a number of other effective tricks have been proposed for training CNN:

- Regularization method Weight Decay (L2 regularization) [29, 26, 18];
- The Dropout technique [20], which allows to accidentally disconnect links in fully connected layers;
- Stochastic Depth technique [23] used in ResNet networks to random disable blocks;
- Augmentation methods: Cutout [11], Mixup [62], CutMix [46], Valid/Diverse Noise, Flipping, etc. [59];
- Learning-rate reduction strategies that improve the convergence of gradient descent methods [41, 36, 13, 34].

These and other proposed ideas allowed to achieve high results in the Computer Vision.

3.2. Fraud Prediction Models

Statistical methods and machine learning have been used for fraud detection tools development for many years. In 2002, Bolton and Hand [6] highlighted the key problems of using statistical methods in fraud detection modeling. Here are some of these issues that remain relevant today:

1) The scope of fraud is increasing with the development of high technology;
2) Fraudsters bypass preventive technologies over time, so detective tools are needed;
3) On the other hand, detective algorithms also degrade over time, so they need to be regularly updated;
4) Databases and anti-fraud methods are closed from the scientific community. This makes it difficult to research and develop this area;
5) To develop antifraud models, unsupervised (search for anomalies) and supervised (search for known fraudulent patterns) algorithms are used. Unsupervised models make a lot of mistakes, because anomalies may be caused by operational errors, marketing promotions, etc. Supervised models are trained on historical data, so they are poor at catching new fraudulent schemes;
6) There is a global problem of class imbalance - there are much fewer fraudulent observations than non-fraudulent ones. This leads to a high false positive rate, which is why many false positives are sent for investigation and the use of models in anti-fraud processes becomes expensive.

In the discussion of this paper, Provost and Breiman noted other important issues:

7) Models are customized for a specific fraudulent scheme, which makes them difficult to scale to other types of fraud;
8) Big data is needed to develop effective anti-fraud models, so this remains the lot of large companies;
9) Machine learning algorithms do not solve the antifraud problem, expert knowledge and understanding of fraudulent schemes are needed.

The development of antifraud models is most often solved as a binary classification problem (fraud/non fraud), where classical algorithms are used, such as: SVM, Logistic Regression, Random Forest and others [5].

With the onset of the deep learning boom, neural networks began to be used in fraud detection modeling and almost completely replaced classical machine learning methods in research papers [27].

Wiese and Omlin [58] proposed using a recurrent LSTM network to detect fraudulent credit card transactions. Authors suggested that since recurrent networks were designed to process sequences, they should be good at detecting fraudulent patterns in card transaction sequences. Authors were able to demonstrate the advantage of the LSTM over the SVM algorithm, but the LSTM network failed to beat the simple fully connected neural network FFNN due to the insufficient number of fraudulent transactions in the sample.

Fu et al. [14] solved a similar problem in detecting card fraud by training the architecture of the LeNet-5 convolutional neural network developed by Yann LeCun for image processing. To train LeNet-5 on the card fraud detection problem, the authors presented transactions in the form of rectangular matrices of features, which were fed to the CNN input as two-dimensional pictures. The results obtained showed the superiority of LeNet-5 over the classical algorithms SVM, ANN and Random Forest.

Heryadi and Warnars [19] continued to develop these ideas and tried to combine CNN with a recurrent LSTM network. The authors' hypothesis was the following: CNNs, due to convolutions, should detect short-term fraudulent patterns, and LSTM network, due to long short-term memory, should work well with long sequences of fraudulent transactions. Authors have developed a hybrid CNN-LSTM architecture, but experiments have shown that simple CNN detects fraud better than CNN-LSTM. Authors made an important conclusion from their results: long-term fraudulent schemes that cannot be detected for a long time are extremely rare, in contrast to short-term fast fraudulent schemes. This is confirmed by Becker's economic model of crime [3].

Attempts to apply the popular neural network architectures CNN and RNN for fraud detection tasks continued in subsequent research efforts.

Li el al. [33] used the DenseNet architecture to detect electricity theft in China.

Chen and Liu [7] refined the DenseNet architecture by adding an Inception module to the beginning of the network and additional skip connections between the Inception layer and Dense blocks. Their new CNN architectures named LI and DI have improved the results of standard CNN and DenseNet for the task of transaction fraud detection.

Cheng et al. [8] proposed a spatio-temporal neural network STAN based on attention. The STAN architecture includes an Attention module and a simple CNN. For the task of detecting transaction fraud, STAN showed better quality compared to gradient boosting, CNN, LSTM, etc.

Li and Liu [65] proposed to use a special loss function for transaction fraud detection, which solved the problem of intraclass variability. Their loss function FCL (full center loss), constructed as a combination of DCL (distance center loss) and ACL (angle center loss), worked like batch normalization.

For the tasks of organized fraud on Internet sites (fake reviews, bots, spam, etc.) detection, graph neural networks are gaining popularity today, allowing feature extraction for interconnected objects. [47, 55, 12, 56].

4. SpiderNet

4.1. Problem Formulation

One of the main problems of using neural networks in fraud detection tasks is that many of the proposed architectures were migrated from other domains (mainly from popular CV and NLP) without a deep understanding of why these architectures should work on fraud detection tasks.

When designing a neural network architecture for fraud detection, our intuition is that anti-fraud rules developed by experts are a kind of digital evidence. At the same time, anti-fraud rules have different power like forensic evidence. Moreover, anti-fraud rules can work in conjunction with each other, strengthening the evidence base. This intuition tells us that CNN's convolution and pooling operations are the most appropriate tools for combining anti-fraud rules and selecting strong combinations of them.

On the other hand, if the combination of anti-fraud rules obtained on the hidden layers has a strong predictive ability, then we want to use this combination without additional processing, forwarding it to the output layer of the neural network using skip connections.

This intuition well represents the best practices of anti-fraud investigations and can be implemented using a fully connected residual network, which we call SpiderNet (Figure 1).

4.2. Spider Block

SpiderNet consists of blocks that are connected to each other using skip connections. Thus, each block receives features from all previous blocks, and these features are processed using convolutional and pooling layers and are forwarded to all subsequent blocks. Cause we want to have only the strongest features at the output, the blocks that are farthest from the network output contain several pooling layers, filtering out weak and medium features. Conversely, the closer the block is to the network output, the fewer pooling layers it contains, since the features that have reached this block have already passed several convolutional and pooling layers. The general architecture of the SpiderNet's block is shown in Figure 4.

SpiderNet is a convolutional feedforward network with skip connections between blocks. Formally, the *k-th* Spider-block is defined by the recursive formula:

$$y_k = \mathcal{F}_k(y_{k-1} \oplus \ldots \oplus y_1) = \mathcal{F}_k\left(\textstyle\sum_{i=1}^{k-1} \oplus\, y_i\right) \qquad (2)$$

where y_k is vector of *n-k* outputs for the *k-th* block;
$\mathcal{F}_k(\cdot)$ is the *k-th* block operator, which combines the functions dropout, convolution, batch normalization, ReLU and Max-pooling;
$\oplus$ is the concatenation operator for incoming vectors;
$\sum\oplus$ is a short notation for vector concatenation;
y_i is the output vector of the *i-th* block, which is fed to the input of the *k-th* block ($1 \le i < k$).

In the last *n-th* Spider-block, after convolution and BatchNorm+ReLU and MaxPooling layer, there is also global average pooling which is applied to reduce the dimension of the channels. After these operations, the vector y_n is fed to two fully connected layers with dropout and SoftMax output for binary classification.

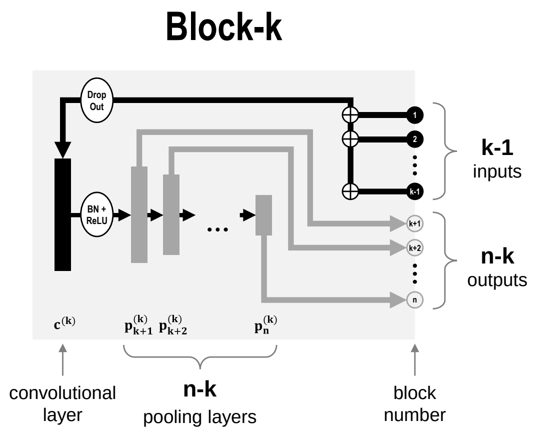

Figure 4: Scheme of the *k-th* Spider-block with convolutional layer and *n-k* pooling layers (*n* is the total number of Spider-blocks).

Expression (2) demonstrates the mathematical intuition of the SpiderNet architecture. The output vector of the *k-th* block is obtained by transforming the concatenated outputs from the previous blocks, which are smaller neural networks, which receive concatenated vectors of the previous blocks as input, etc. Thus, SpiderNet works as an ensemble of neural networks, which allows to improve the convergence and generalization of the entire neural network.

5. B-tests and W-tests

The peculiarity of fraud modeling is that most of the fraud rules are developed manually by experts. This is due to the fact that fraudulent schemes are diverse, especially for internal fraud schemes that are designed for specific vulnerabilities of the organization.

We offer B-tests and W-tests approaches that automate the manual process and generalize feature engineering for various types of internal fraud.

5.1. B-tests

B-tests are based on the Benford's law, discovered at the end of the 19th century by Simon Newcomb [39], who noticed that the values of some numerical data start with one more often than two, two more often than three, and so on. Later, Frank Benford [4] empirically confirmed this law for various social and physical phenomena, and the first-digit law was named after him.

In the 90s of the twentieth century, Mark Nigrini used Benford's law to audit financial statements and managed to identify managerial embezzlement of $2 million in the office of the Arizona State Treasurer [40].

Summarizing these ideas for all types of data (numerical and categorical), we propose a B-tests technique, which consists of comparing the distributions of data characterizing an object with the general population of all objects (for example, the activities of employees or partners of the company).

The general idea is that internal fraud is rare, i.e. it does not greatly affect the distribution of the general population, while the data on the activities of fraudsters are very different from the average (Figure 5).

B-tests can be calculated using various metrics reflecting the divergence between two distributions, for example, Chi-squared test, K–S test, Anderson–Darling test, etc.

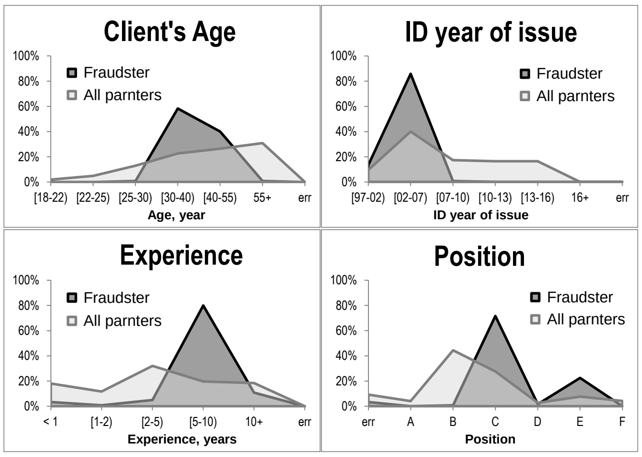

Figure 5: Examples of B-tests for POS-partner fraud detection.

	Private Data	Public Data
Source	Russian Bank	Ant Financial Services Group
Type of data	POS credits	Payments
Type of fraud	Internal	Transaction
Time period	03.2014-10.2019	09.2017-11.2017
All samples, #	1 880 499	990 006
Fraud samples, #	5 327	12 122
Fraud rate, %	0.28	1.22
All features, #	509	297
Selected features, #	163	128

Table 1: Characteristics of private and public datasets.

For our B-tests, we calculate the area difference for two discrete distributions using the following formula:

$$S = \frac{1}{2}\sum_{i=1}^{n}|a_i - b_i| \qquad (3)$$

where a_i, and b_i are the compared distributions;
n is the number of quantiles in the distribution.

The number of quantiles in the distribution n and the threshold for S depend on the number of objects in the samples and are tunable hyperparameters [1].

5.1. W-tests

W-tests solve the same problem as B-tests using Wasserstein metric, which is defined as:

$$W_p(\mu, \nu) = inf_{\gamma \in \Gamma(\mu,\nu)} E_{(x,y)\sim\gamma}[||x - y||] \qquad (4)$$

where $E[Z]$ denotes the expected value of a random variable Z and the infimum is taken over all joint distributions of the random variables X and Y with marginals μ and ν respectively.

Wasserstein metric solves the problem of insufficient number of objects in samples but it can be calculated only for numeric data types, so W-tests cannot completely replace B-tests.

6. Experiment Setup

6.1. Datasets

We trained and compared SpiderNet with other algorithms on two datasets: private data and public data. General characteristics of datasets are presented in Table 1.

Private dataset. Our private dataset contains data on the credit activity of POS partners of a large Russian bank, one of the top 50 Russian banks in terms of assets. The records in the private dataset are presented as a vector of features for each POS partner as of the weekly slice date. Therefore, a single POS partner can be included in the sample several times with different slice dates. The depth of calculation of the features ranges from 7 to 60 days ago from the slice date. A flag, indicating fraud event, was set to each record as a target variable. The overall task comes down to predicting the internal fraud of the POS partner based on the data of its historical activities.

Public dataset. The public dataset comes from an online payment fraud detection competition hosted by Ant Financial Services Group. The dataset contains the values of features for payment transactions and a binary target variable that reflects whether the payment is fraudulent.

6.2. Feature Selection

To form the final samples, we did data preprocessing. Firstly, we checked the features for validity and removed features with a low fill rate. Secondly, we selected features using a cross-correlation matrix. Of a pair of highly correlated features, we left the one that had a stronger correlation with the target variable.

As a result of data preprocessing, 163 features remained in the private dataset, and 128 features remained in the public dataset.

6.3. Model Architectures and Tricks

To assess the generalization ability and demonstrate the benefits of SpiderNet, we compared the results of several machine learning algorithms:

1) *Random Forest* is a baseline model that has been chosen as a strong algorithm and industry standard for modeling on tabular data. We used 5-fold cross-validation and the Optuna library to tune the Random Forest hyperparameters [2].
2) *1D-CNN* is a one-dimensional convolutional network with classical alternation of convolutional and pooling layers and two fully connected layers and a SoftMax layer. We trained CNN with 3, 6 and 8 convolutional layers;
3) *1D-DenseNet* is a classic DenseNet architecture [22] with an implementation for one-dimensional vectors. We trained two block architectures with 3 and 4 convolutional layers in each block;
4) *F-DenseNet* is a DenseNet architecture adapted for fraud prediction with two fully connected convolutional blocks containing convolutional and pooling layers. We trained architectures with 3 and 4 convolutional layers in each block;
5) *SpiderNet* is our fully connected residual convolutional network with convolutional-pooling blocks. We trained 6 and 8 block architectures (Figure 6).

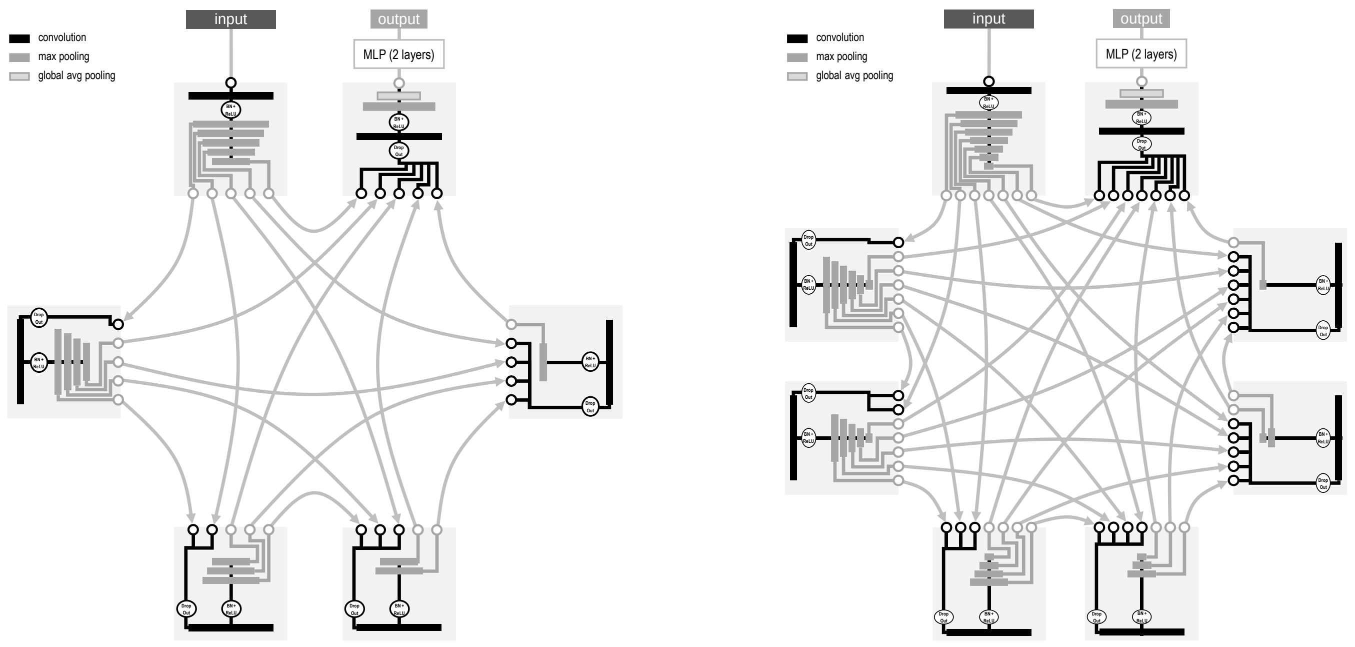

Figure 6: SpiderNet-6 (left) and SpiderNet-8 (right) neural network architectures for fraud detection tasks.

We used weight decay (L2-regularization) in BatchNorm and fully connected layers, and dropout on fully connected layers as regularizers for all neural networks. In the 1D-CNN, F-DenseNet and SpiderNet architectures we applied the BatchNorm and ReLU transformations after each convolutional layer. In the F-DenseNet and SpiderNet blocks, as an additional regularization for the skip connections, we used dropout after concatenating the input vectors for each block (Figure 4). We also used the fraud-rate leveling technique in batches to solve the problem of the lack of fraud samples in batches in case of over class imbalance.

To train and assess the quality of the models, we performed stratified split of private and public datasets into train (80%), validation (10%), and test (10%) samples. We tuned the neural networks to the validation sample using Grid-Search and early stopping.

6.4. Performance Measures

We used AUC ROC and AUC PR metrics to evaluate the quality of models. The AUC ROC metric and the linearly related Gini coefficient are industry standards in banking modeling. However, as shown by Saito and Rehmsmeier [45], the AUC ROC accepts unreasonably high values and becomes uninformative on over unbalanced samples, in contrast to the AUC PR, which adequately estimates the quality of models on unbalanced samples. Therefore, we tuned the hyperparameters of the models using the AUC PR.

To evaluate the quality of fraud detection models on the private dataset, we have developed the special metric PL (Prevented Loss), which shows how much loss from internal fraud the model prevents. To calculate the PL, we estimate prevented loss for each partner:

$$PL^{(i)} = P_{(T_l - T_a)} \cdot \frac{DR - DR_0}{1 - DR_0} \tag{5}$$

where T_l is the whole considered period of loss;
T_a is the period on which the model works;
$(T_l - T_a)$ is the period after the model is triggered (for our sample, it is 90 days – the empirical period for which the bank's Security detects fraud without using the model);
DR is the Default-Rate for loans issued by the partner for the period $(T_l - T_a)$;
DR_0 is a "zero target" for Default-Rate in which the loan portfolio has zero profit;
$P_{(T_l - T_a)}$ is the partner's loan portfolio for period $(T_l - T_a)$.

Total PL is calculated based on the company's total investigative resources, i.e. on the assumption that security costs do not increase. Total Prevented Loss shows the net profit from the model due to the earlier detection of fraud than it had been before the model was applied:

$$PL = \sum_{i=1}^{k} PL^{(i)} \cdot b_i \tag{6}$$

where $PL^{(i)}$ is prevented loss for the *i-th* fraud partner;
k is the number of the first k partners with the highest model probability of fraud (for our bank $k = 40$);
b_i is a binary variable showing the event for the *i-th* partner: 1 – fraud, 0 – no fraud.

7. Experiment

7.1. Model Optimization and Tuning

We tuned models by AUC PR and selected the best hyperparameters for the models.

Random Forest:
Private data: cl_weight=0.0167, max_dep=7, n_estim=129;
Public data: cl_weight=0.0744, max_dep=7, n_estim=90.

CNN:
Private data: layers=8, l2_batch=0.0001, kernel_size=3, n_filters=10, weight_decay=0.0002, learning_rate=0.003, d_hidden=100, dropout=0.25;

#	Model	Private data (test sample)			Public data (test sample)	
		AUC PR	AUC ROC	PL	AUC PR	AUC ROC
1	Random Forest	0.0650 (±0.0011)	0.9371 (±0.0093)	$ 2 079 527	0.4881 (±0.0031)	0.9709 (±0.0036)
2	CNN-3	0.0527 (±0.0010)	0.9339 (±0.0130)	$ 2 235 707	0.4462 (±0.0031)	0.9670 (±0.0047)
3	CNN-6	0.0644 (±0.0011)	0.9385 (±0.0114)	**$ 2 753 821**	0.4908 (±0.0031)	0.9711 (±0.0048)
4	CNN-8	0.0708 (±0.0012)	0.9288 (±0.0096)	$ 2 337 297	0.5099 (±0.0031)	0.9718 (±0.0045)
5	DenseNet-6 [3; 3]	0.0646 (±0.0011)	0.9315 (±0.0091)	$ 2 324 181	0.4757 (±0.0031)	0.9669 (±0.0049)
6	DenseNet-8 [4; 4]	0.0691 (±0.0011)	0.9310 (±0.0105)	$ 2 433 914	0.4854 (±0.0031)	0.9686 (±0.0047)
7	F-DenseNet-6 [3; 3]	0.0732 (±0.0012)	0.9263 (±0.0145)	$ 2 297 848	0.5092 (±0.0031)	0.9708 (±0.0051)
8	F-DenseNet-8 [4; 4]	0.0575 (±0.0011)	0.9186 (±0.0158)	$ 2 402 470	0.4968 (±0.0031)	0.9704 (±0.0048)
9	SpiderNet-6	**0.0948** (±0.0013)	**0.9484** (±0.0080)	$ 2 570 014	**0.5375** (±0.0031)	**0.9721** (±0.0048)
10	SpiderNet-8	0.0680 (±0.0011)	0.9277 (±0.0096)	$ 2 379 977	0.5160 (±0.0031)	0.9684 (±0.0047)

Table 2: Quality of models for internal (private dataset) and transactional (public dataset) fraud detection: the best results are highlighted in bold; 95% confidence intervals are highlighted in gray.

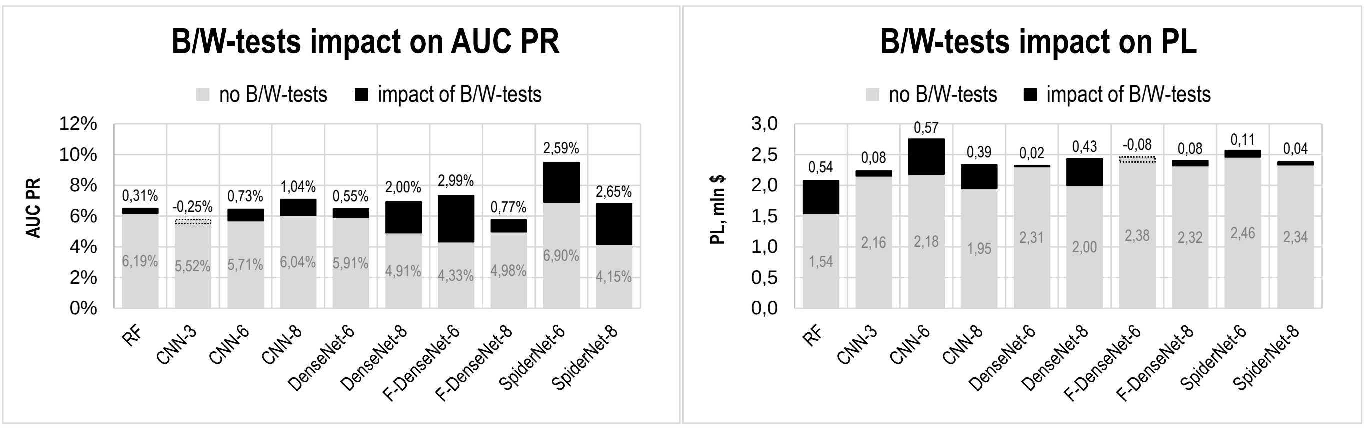

Figure 7: Influence of B-tests and W-tests on model quality (negative impact means a decrease in the model quality when adding B/W-tests).

Public data: layers=8, l2_batch=0.0001, kernel_size=5, n_filters=10, weight_decay=0.0002, learning_rate=0.003, d_hidden=30, dropout=0.25.

DenseNet:
Private data: block_sizes = [4, 4], init_filters=5, init_str=1, k=5, conv_kernel_width=3, bottleneck_size=2, theta=0.5, tr_p_str=1, init_c_width=5, init_p_width=2, init_p_str=2;
Public data: block_sizes = [4, 4], init_filters=5, init_str=1, k=10, conv_kernel_width=5, bottleneck_size=2, theta=0.5, tr_p_str=1, init_c_width=5, init_p_width=2, init_p_str=2.

F-DenseNet:
Private data: blocks=2, conv=3+3, l2_batch =0.0002, kernel_size=7, n_filters=15, d_hidden=60, weight_decay=0.0002, learning_rate=0.003, dropout=0.25;
Public data: blocks=2, conv=3+3, l2_batch =0.0002, kernel_size=5, n_filters=15, d_hidden=100, weight_decay=0.0002, learning_rate=0.003, dropout=0.25.

SpiderNet:
Private data: blocks=6, l2_batch=0.0001, kernel_size=3, n_filters=10, d_hid=100, weight_decay=0, dropout=0.25, learn_rate=0.005, dropout_block_n=0.001*(n^4), n={4, 5};
Public data: blocks=6, l2_batch=0.0002, kernel_size=7, n_fil=15, d_hid=30, weight_decay=0.0002, dropout=0.25, learn_rate=0.005, dropout_block_n=0.001*(n^4), n={4, 5}.

For all neural networks we used the Adam optimizer.

7.2. Results

The results of the trained models showed that the SpiderNet-6 architecture had the best quality on both datasets for both AUC PR and AUC ROC (Table 2). We see clear dynamics that increase in the number of skip-connections results in increase in quality for fraud detection models: {CNN-8: 0 skip connections} → {F-DenseNet-6: 2 skip connections} → {SpiderNet-6: 10 skip connections}. This confirms our hypothesis that strong features can pass well from different layers of neural network immediately to its output due to skip-connections. On the other hand, we see that the best model of the classic DenseNet-8 performs worse than the best CNN-8, which has no skip-connections. Apparently, this is due to the bottleneck between the blocks, on which strong features are lost. The F-DenseNet architecture, which is adapted for fraud detection tasks, does not contain a bottleneck between blocks, so the quality of the best F-DenseNet-6 surpasses the quality of CNN-8 and DenseNet-8. This result demonstrates the importance of developing a neural network architecture for the specifics of the task.

We also confirm Saito's and Rehmsmeier's results [45] that AUC ROC metric is not informative for problems with unbalanced data and shows unreasonably high values. For our datasets, AUC PR metric is preferred. On the other hand, on a private dataset, the AUC PR shows low values, so we evaluate the effectiveness of the models using our PL metric, which shows a positive economic effect for all trained models.

We also see that for the private dataset, SpiderNet-6 ranked second according to PL metric, losing to the simpler CNN-6. This may be because internal fraud is very adaptive, so the models for internal fraud detection are unstable (this can be seen from the values of the AUC PR metric, according to which SpiderNet-6 greatly outperforms CNN-6). This may also be due to the fact that the hyperparameters in the Grid-Search selected according to the AUC PR.

We tested the effectiveness of B-tests and W-tests by training the models on two private samples:

1) Full sample containing 24 B-tests, 15 W-tests and 124 expert rules;
2) Truncated sample containing 124 expert rules.

Our results showed high efficiency of B-tests and W-tests by AUC PR and PL (Figure 8). Adding of B-tests and W-tests gave us AUC PR increase of 19.4% on average and PL increase of 9.1% on average.

We also see that SpiderNet-6 trained on expert rules without B-tests shows the best quality by AUC PR and PL, which once again proves the stability of the results obtained.

All program codes and detailed results are available at: https://github.com/aasmirnova24/SpiderNet

8. Conclusions

In this paper, we investigated deep learning methods for fraud prediction and proposed a novel architecture of neural network for fraud detection problems. Taking inspiration from the skip connection concepts of Resnet [16], FractalNet [30], Adanet [10] and DenseNet [22] convolutional networks for imaging, we developed SpiderNet architecture for fraud detection, guided by antifraud expert knowledge. Using convolutional layers, our SpiderNet creates hierarchical combinations of anti-fraud rules, and a fully-connected structure of skip connections between blocks allows strong rules to be forwarded immediately to the network output. Also, our network can select strong rules early on through the use of a multi-layered structure of pooling layers in Spider-blocks. Moreover, the down-dropout technique we use between Spider-blocks is an additional regularizer against the excessive complexity of the network.

In our opinion, SpiderNet should work well for heterogeneous input data, when there are clear leaders among the rules supplied to the network input and they must be forwarded to the output without additional transformation (for example, scores of other models or strong rules).

It can also be noted that our results confirm the hypothesis of ResNet authors, which they formulated in their paper [16]: "The residual learning principle is generic, and we expect that it is applicable in other vision and non-vision problems".

In this paper, we also proposed new methods for developing antifraud rules – B-tests and W-tests, which significantly affect to the quality of the models. The quality metric Prevented Losses proposed by us allows to solve an important issue for the industry: how to evaluate the economic efficiency of antifraud models developed for industrial use.

We understand that our SpiderNet does not solve all the anti-fraud modeling problems identified by Bolton, Hand, Provost and Breiman [6]. In particular, we still use expert rules designed for specific fraud types to train models. Our B-tests and W-tests partially solve this problem, but there are other strong methods for fraud feature engineering, such as graph methods [47, 55, 12, 56], entropy changing methods [14] and variance anomaly detection methods (V-tests, similar to B-tests). We plan to work on these topics in our future research.

An important component of SpiderNet is skip connection, which helps to forward strong features directly to the output layers of the network, partially solving the problem of locality in convolutions, when the order of features in the input vector is important, and their rearrangement leads to a change in the quality of the model. However, the current implementation of SpiderNet does not completely solve the locality problem. Our future work will focus on this problem.

We also assume that SpiderNet might work well not only for fraud detection, but also for other types of modeling tasks that use tabular data. Testing this hypothesis is also a topic for our future research.

Acknowledgements

We are grateful to Dmitry Efimov for valuable advice and comments.

References

1. Afanasiev S. and Smirnova A. Predictive fraud analytics: B-tests. (2018). DOI: 10.21314/JOP.2018.213
2. Akiba T., Sano S., Yanase T., Ohta T., Koyama M. Optuna: A Next-generation Hyperparameter Optimization Framework (2019). arxiv.org/abs/1907.10902
3. Becker G. (March 1968). "Crime and punishment: an economic approach". Journal of Political Economy. 76 (2): 169–217. doi:10.1086/259394
4. Benford F. (March 1938). "The law of anomalous numbers". Proc. Am. Philos. Soc. 78 (4): 551–572. doi: 10.2307/984802
5. Bhattacharyya S., Jha S., Tharakunnel K., J. Westland J. C. Data mining for credit card fraud: A comparative study. (2011). DOI:10.1016/j.dss.2010.08.008
6. Bolton R. J., Hand D. J. Statistical Fraud Detection: A Review. (2002) Statistical Science, 17(3): 235–255, 1999, doi: 10.1214/ss/1042727940
7. Chen Z., Liu G. DenseNet+Inception and Its Application for Electronic Transaction Fraud Detection. (2019). DOI:10.1109/HPCC/SmartCity/DSS.2019.00357
8. Cheng D., Xiang S., Shang C., Zhang Y., Yang F., Zhang L. Spatio-Temporal Attention-Based Neural Network for Credit Card Fraud Detection. (2020). DOI:10.1609/aaai.v34i01.5371
9. Chollet F. Xception: Deep Learning with Depthwise Separable Convolutions (2017). arxiv.org/abs/1610.02357
10. Cortes C., Gonzalvo X., Kuznetsov V., Mohri M., Yang S. AdaNet: Adaptive Structural Learning of Artificial Neural Networks. (2017). Conference: Efficient Methods for Deep Neural Networks (EMDNN). arxiv.org/abs/1607.01097
11. DeVries T. and Taylor G. W. Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552, 2017.
12. Dou Y., Liu Z., Sun L., Deng Y., Peng H., Yu P. S. Enhancing Graph Neural Network-based Fraud Detectors against Camouflaged Fraudsters. (2020). DOI:10.1145/3340531.3411903
13. Fort S., Jastrzebski S. Large Scale Structure of Neural Network Loss Landscapes. (2019). arxiv.org/abs/1906.04724
14. Fu K., Cheng D., Tu Y., Zhang L. Credit Card Fraud Detection Using Convolutional Neural Networks. (2016). In: (eds) Neural Information Processing. ICONIP 2016. Lecture Notes in Computer Science, vol 9949. Springer, Cham. DOI: 10.1007/978-3-319-46675-0_53
15. Graves A., Abdel-rahman Mohamed, and Hinton G. Speech recognition with deep recurrent neural networks. In 2013 IEEE international conference on acoustics, speech and signal processing, pages 6645–6649. IEEE, 2013. arxiv.org/abs/1303.5778
16. He K., Zhang X., Ren S., Sun J. (2015). Deep Residual Learning for Image Recognition. Microsoft Research. https://arxiv.org/pdf/1512.03385v1.pdf
17. He K., Gkioxari G., Dollár P., Girshick R. Mask r-cnn. In Proceedings of the IEEE international conference on computer vision, pages 2961–2969, 2017. arxiv.org/abs/1703.06870
18. He T., Zhang Z., Zhang H., Zhang Z., Xie J., Li M. Bag of Tricks for Image Classification with Convolutional Neural Networks. (2018). arxiv.org/abs/1812.01187
19. Heryadi Y., Harco Leslie Hendric Spits Warnars. Learning temporal representation of transaction amount for fraudulent transaction recognition using CNN, Stacked LSTM, and CNN-LSTM. (2017). IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom). DOI: 10.1109/CYBERNETICSCOM.2017.8311689
20. Hinton G. E., Srivastava N., Krizhevsky A., Sutskever I., Salakhutdinov R. R. Improving neural networks by preventing co-adaptation of feature detectors. (2012). arxiv.org/abs/1207.0580
21. Howard A. G., M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. (2017). arxiv.org/abs/1704.04861
22. Huang G., Liu Z., Laurens van der Maaten, Kilian Q. Weinberger. Densely Connected Convolutional Networks. (2018). DOI: 10.1109/CVPR.2017.243 arxiv.org/abs/1608.06993
23. Huang G., Yu Sun, Zhuang Liu, Daniel Sedra, Kilian Weinberger. Deep Networks with Stochastic Depth. (2016). arxiv.org/abs/1603.09382
24. Hubel, D. H., Wiesel, T. N. (1959). Receptive Fields of Single Neurones in the Cat's Striate Cortex. Journal of Physiology, 148, 574-591.
25. Hubel, D. H., Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex". J. Physiol. 160: 106–154. doi:10.1113/jphysiol.1962.sp006837.
26. Jia X., S. Song, W. He, Y. Wang, H. Rong, F. Zhou, L. Xie, Z. Guo, Y. Yang, L. Yu, et al. Highly scalable deep learning training system with mixed-precision: Training imagenet in four minutes. (2018). arXiv:1807.11205
27. Kanika, Dr Jimmy Singla. A Survey of Deep Learning based Online Transactions Fraud Detection Systems. (2020). DOI:10.1109/ICIEM48762.2020.9160200
28. Krizhevsky A., Sutskever I., Hinton G.E. (2012) ImageNet Classification with Deep Convolutional Neural Networks. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
29. Krogh A., John A Hertz. A simple weight decay can improve generalization. (1992). In Advances in Neural Information Processing Systems 4, J. E. Moody, S. J. Hanson, and R. P. Lippmann (Eds.). Morgan-Kaufmann, 950–957. http://papers.nips.cc/paper/563-a-simple-weight-decay-can-improve-generalization.pdf
30. Larsson G., Michael Maire, Gregory Shakhnarovich. FractalNet: Ultra-Deep Neural Networks without Residuals. (2016). arxiv.org/abs/1605.07648
31. LeCun Y., Bottou L., Bengio Y., Haffner P. (1998). "Gradient-based learning applied to document recognition" (PDF). Proceedings of the IEEE. 86 (11): 2278–2324. doi:10.1109/5.726791
32. LeCun Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. Neural computation, 1989. DOI: 10.1162/neco.1989.1.4.541
33. Li B., Xu K., Xiaoyan Cui, Yiheng Wang, Xinbo Ai, Yanbo Wang. Multi-Scale DenseNet-Based Electricity Theft Detection. (2018). researchgate.net/publication/344779530_Multi-scale_DenseNet-Based_Electricity_Theft_Detection
34. Li Y., C. Wei, T. Ma. Towards Explaining the Regularization Effect of Initial Large Learning Rate in Training Neural Networks. 2020. arxiv.org/abs/1907.04595
35. Lin M., Qiang Chen, Shuicheng Yan. Network In Network. (2014). arxiv.org/abs/1312.4400

36. Loshchilov I., Frank Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts. ICLR 2017. arxiv.org/abs/1608.03983
37. Lu. F., Boritz J.E., Covvey D. "Adaptive Fraud Detection Using Benford's Law" in Luc Lamontagne and Mario Marchand (eds.) Advances in Artificial Intelligence: Proceedings of the 19 th Conference of the Canadian Society for Computational Studies of Intelligence, Canadian AI 2006, Quebec City, Canada, June 2006, Springer 2006, pp. 347-358. doi: 10.1007/11766247_30
38. Lu F., Boritz J. E. Detecting Fraud in Health Insurance Data: Learning to Model Incomplete Benford's Law Distributions. In 16th European Conference on Machine Learning, pages 633–640, Porto, Portugal, 2005. Springer. doi: 10.1007/11564096_63
39. Newcomb S. Note on the frequency of use of the different digits in natural numbers. (1881). American Journal of Mathematics 4 (1): 39–40.
40. Nigrini M. J. I've Got Your Number: How a mathematical phenomenon can help CPAs uncover fraud and other irregulaities. (1999). Journal of Accountancy.
41. O'Donoghue B., Emmanuel Candes. Adaptive Restart for Accelerated Gradient Schemes. (2012). arxiv.org/abs/1204.3982v1 DOI: 10.1007/s10208-013-9150-3
42. Paulus R., Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. arXiv preprint arXiv:1705.04304, 2017.
43. Raghu M., E. Schmidt. A Survey of Deep Learning for Scientific Discovery. (2020). arxiv.org/abs/2003.11755
44. Rajpurkar P., Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. arXiv preprint arXiv:1606.05250, 2016.
45. Saito T., Rehmsmeier M. The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. (2015) PLoS ONE 10(3):e0118432, March 2015. DOI: 10.1371/journal.pone.0118432.
46. Yun S., Han D., Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, Youngjoon Yoo. CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features. (2019). arxiv.org/abs/1905.04899
47. Shuhan Y., Wu X., Li J., Lu A. (2017). Spectrum-based deep neural networks for fraud detection. https://arxiv.org/abs/1706.00891
48. Simonyan S., Zisserman A. (2015) Very deep convolutional networks for large-scale image recognition. Visual Geometry Group, Department of Engineering Science, University of Oxford. https://arxiv.org/pdf/1409.1556.pdf
49. Sutherland E., Cressey D. Principles of Criminology. 11th ed. Lanham, Md.: AltaMira Press, 1992. ISBN 0-930390-69-5
50. Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke V., Rabinovich A. (2014). Going deeper with convolutions. Google Inc. https://arxiv.org/pdf/1409.4842v1.pdf
51. Tan M., QuocV.Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. (2019). arxiv.org/abs/1905.11946
52. Van den Oord A., Dieleman S., Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. arXiv preprint arXiv:1609.03499, 2016.
53. Vaswani A., Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in neural information processing systems, pages 5998–6008, 2017. arxiv.org/abs/1706.03762
54. Wallace W. The Logic of Science and Sociology. Chicago: Aldine-Atherton, 1971. doi: 10.4324/9781315132976
55. Wang D., Lin J., Peng Cui, Quanhui Jia, Zhen Wang, Yanming Fang, Quan Yu, Jun Zhou, Shuang Yang, Yuan Qi. A Semi-supervised Graph Attentive Network for Financial Fraud Detection. (2019). DOI:10.1109/ICDM.2019.00070
56. Wang H., Li Z., Jiaming Huang, Pengrui Hui, Weiwei Liu, Tianlei Hu and Gang Chen. Collaboration Based Multi-Label Propagation for Fraud Detection. (2020). https://doi.org/10.24963/ijcai.2020/343
57. Wang X., Girshick R., Abhinav Gupta, and Kaiming He. Non-local neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 7794–7803, 2018. arxiv.org/abs/1711.07971
58. Wiese B.J., Omlin C. (2007). Credit Card Transactions, Fraud Detection, and Machine Learning: Modelling Time with LSTM Recurrent Neural Networks. Innovations in Neural Information Paradigms and Applications. P. 231-268. Springer. DOI: https://doi.org/10.1007/978-3-642-04003-0
59. Xie Q., Dai Z., Eduard Hovy, Minh-Thang Luong, and Quoc V Le. Unsupervised data augmentation. arXiv preprint arXiv:1904.12848, 2019.
60. Xie S., Girshick R., Piotr Dollár, Zhuowen Tu, Kaiming He. Aggregated Residual Transformations for Deep Neural Networks. (2017). arxiv.org/abs/1611.05431 DOI: 10.1109/CVPR.2017.634
61. Zagoruyko S., Komodakis N. Wide Residual Networks. Conference: British Machine Vision Conference 2016. DOI: 10.5244/C.30.87. (2017) arxiv.org/abs/1605.07146
62. Zhang H., Cisse M., Yann N. Dauphin, David Lopez-Paz. mixup: Beyond Empirical Risk Minimization. (2017). arxiv.org/abs/1710.09412
63. Zoph B., Vasudevan V., Jonathon Shlens, Quoc V. Le. Learning Transferable Architectures for Scalable Image Recognition. (2018). arxiv.org/abs/1707.07012
64. Zoph B., Quoc V. Le. Neural Architecture Search with Reinforcement Learning. (2017). arxiv.org/abs/1611.01578
65. Zhenchuan Li , Guanjun Liu. Deep Representation Learning with Full Center Loss for Credit Card Fraud Detection - IEEE Transactions on Computational Social Systems. (2020). DOI:10.1109/TCSS.2020.2970805