UNIVERSITY OF GRONINGEN

MASTER'S THESIS ARTIFICIAL INTELLIGENCE

Combining Visual and Contextual Information for Fraudulent Online Store Classification

Author:
WOUTER MOSTARD *(s2579006)*

Supervisor:
dr. M.A WIERING (Bernoulli institute)
External supervisor:
MSc. B. ZIJLEMA (Dataprovider.com)

Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence
University of Groningen, The Netherlands

Dataprovider.com, The Netherlands

rijksuniversiteit
groningen

January 13, 2019

Abstract

Following the rise of e-commerce there has been a dramatic increase in online criminal activity targeting online shoppers. This online oriented crime ranges from selling fake products, to not delivering the ordered products, and even stealing credit card information during check-out. Since the number of online stores has risen dramatically, manually checking these stores has become intractable. An automated process is therefore required. We will approach this problem by applying machine learning techniques to extract and detect instances of fraudulent online stores.

To determine the legitimacy of an online store two sources of information are used. First, a baseline model is proposed based on meta-features, such as whether an SSL certificate is present. Supervised learning algorithms are applied to achieve the best possible baseline results. Second, visual information, like the presence of logos of payment methods, are subsequently added to improve the baseline model. Numerous segmentation methods, pre-trained networks, and learning algorithms are compared to achieve the best possible performance regarding visual feature extraction of logos on online stores.

The *random forest* learning algorithm proved to be the best performing baseline learning algorithm. *Auto canny* and *selective search* show similar recall scores for extracting relevant image patches from the online stores, the former being the fastest method. The ResNet50 network proved to be both the fastest and best performing feature vector extractor for the image patches. Using the ResNet50 feature vectors, the *support vector machine* showed to be the best performing learning algorithm for logo classification. Combining both sources of information demonstrated a positive result, in particular when detecting fraudulent class cases.

This research shows that applying various information sources in fraudulent online store classification has a significant positive effect. Furthermore, it shows that reasonable results in logo detection on online stores can be achieved using simple binarization methods and pre-trained convolutional neural networks. Interesting future research include: expanding the logo detection algorithm to other visual information, applying multimodal learning instead of simple feature concatenation, and building a scalable solution that is production worthy. Implementing the given results into a scalable solution could help for example law enforcement to quickly find potentially fraudulent online stores before too much harm is done.

Acknowledgements

I would like to thank everyone who helped me during this project. Especially, I want to thank my supervisor Dr. M.A. Wiering for sharing his expertise with me throughout the project. Not only did he support me with his knowledge about machine learning but also with his help and support in doing a long research project, bringing it to a satisfactory end. Furthermore, I would like to thank my external supervisor Bastiaan for all the help at Dataprovider, especially with all the technicalities that come with bringing machine learning models into production. This also counts for the rest of my colleagues at Dataprovider, which gave me a good introduction into the world of big data and data science, making the writing of a thesis the best possible experience, with all the games of pool included. Last but not least I would like to thank my girlfriend Marjolein for all the support. Throughout the research she has always been a great source of support when finishing my thesis seemed so far away.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the past decade there has been a rapid increase in e-commerce.[1] As a result, large companies are transitioning from their physical shop to an online equivalent. This increase in e-commerce also paved the way for novel criminal activity. This includes fraudulent online stores not delivering ordered products or stealing credit card information given by the consumer during check-out. This type of criminality has a negative influence on online trade. Online sales decrease due to reduced consumer trust and a worse brand reputation. How these fraudulent online stores can be detected will be the main focus point of this research.

This research will be conducted at dataprovider.com (dataprovider). Dataprovider is a company founded in 2008 that scrapes and processes information from approximately 300 million domains. This information is subsequently provided to customers through various web applications and programming interfaces. Apart from storing the unprocessed HTML pages, meta-features such as whether an SSL certificate is present, are also processed. During this research it will be determined which of these meta-features are useful in the classification of fraudulent online stores.

Determining whether an online store is legitimate or fraudulent using training examples and input features is a case of supervised discrimination, or classification. Representing a website as a feature vector has yielded positive results in website topic classification (Kriegel, 2004). Since websites contain more information than the surface text, close examination has to be conducted on the underlying structure of the page (Kaur, 2014). Contextual features such as the HTML tags, search engine optimization (SEO) keywords, hyperlinks and the Uniform Resource Locator (URL) make up an important part of the context of the website. However most research assumes text as the primary information source. Other contextual features, such as meta-data and visual queues, merely serve to improve baseline classification. Classifying a web page merely on the textual features is called the *Text Only* approach and traditionally serves as a baseline. When the web pages are represented by the text as well as the contextual features it is referred to as the *Hypertext* approach (Sun, Peng, & Keong, 2002). Research conducted by Sun has shown that using contextual features can significantly improve performance (Sun et al., 2002). Online stores have the disadvantage that text is quite scarce, implying that a text only approach would not return satisfactory results. For this reason research will focus on meta and visual features, which are abundant in online stores. Visual features such as which payment methods or brand logos are present are covered by visual feature engineering. Logo detection on websites is extensively discussed in (Zhang, Zhu, Wang, & Feng, 2014).

Various terms will be widely used throughout this thesis. *Fraudulent*, in the context of this research, is any online store that pretends to sell any sort of item without actually selling it. Another common method of

[1]https://www.census.gov/retail/mrts/www/data/pdf/ec_current.pdf

fraudulent behaviour is the stealing of personal belongings such as credit card information. Other terms such as *hyperparameter* and *classification* are explained in their corresponding sections.

Currently the process of counteracting this type of fraud requires companies to manually prepare a list with online stores that are suspected of being fraudulent. Law enforcement will afterwards manually check all the given stores and sends a request to the service provider to take down an online store if the store is indeed determined as fraudulent. There are two main disadvantages for using this approach: 1) fraudulent stores are difficult to find without any prior knowledge, and 2) manually sustaining a list with possibly thousands of records requires a great deal of manual labor and is prone to making errors.

To address these difficulties an automated approach is needed for extracting possible fraudulent stores and a learning algorithm for detecting new instances. With an automatic classification and storage model for fraudulent online stores: 1) a considerable amount of manual labor for finding these online stores will be alleviated since no list has to be retained and 2) this approach compensates for the lack of human prior knowledge for finding new instances by applying a learning algorithm on existing data.

Presently, no constructive research has been done into combining visual features with meta-features in determining the legitimacy of an online store. Nevertheless, this is an important research topic, since humans often determine the legitimacy of an online store by its visual representation. A possible combined feature could be that the visual system detects a payment method whilst there is no URL found forwarding to the specific payment server. Furthermore, it could be of interest to find out whether an online store pretends to have a social media profile by placing the corresponding images whilst there is actually no redirection to any of the social media platforms.

Finding fraudulent stores in the wealth of information that is inside the internet is a daunting challenge. Especially given that online criminals tempt to mimic legitimate online stores poses a problem. The aim of this research is thus extracting discriminant features for the classification of these stores. Furthermore, a logo detection architecture will be developed in order to retrieve the relevant logos. Finally, the goal is to combine the most descriminative features or engineer new ones for a final classification system which outputs a probability score of a store being legitimate.

Given the description of the research problems in the previous paragraphs, the following main research question is posed:

Can visual and contextual features be effectively combined with machine learning techniques for predicting fraudulent online stores?

The research aims required to answer the research question are also given in the previous paragraph. These can be summed up with the following sub questions.

- Which meta-features yield the most discriminative power?
- What is the best method for extracting visual information from online stores?
- What combination of visual and contextual features improves baseline performance the most?

First, the theoretical background is presented in Chapter 2 and a basic introduction into the internal workings for various classifiers is given. The corresponding hyperparameters and their influence on performance are described. Next, object detection is explained and various steps and methods that make up an end-to-end image classification system are explained. In Chapter 4 the conducted research and subsequent implementation into a production worthy system are presented. In section 5 the classifier and research results are given. This part starts with showing the results for the baseline, visual, and combined features. Next classification results are discussed and pointers for possible future work are given. Lastly, conclusions are derived from the research.

Chapter 2

Classification

Classification is the task of determining which target variable y belongs to given input vector x using a learning algorithm, or *classifier*. Although there are numerous ways to learn, also called train, this research will focus on *supervised learning*. With supervised learning a classifier is trained using a data set D containing n (x,y) tuples, where x_i is the independent input vector and y_i the dependent output variable. This means that the classifier is trying to learn the unknown target function underlying the data. The approximated underlying function is called the *hypothesis* and should closely resemble the underlying target function. The class C is drawn from a hypothesis class $\mathcal{H}$. The classifier then uses the training data to find a hypothesis $h \in \mathcal{H}$ that resembles C as close as possible by fitting the parameters of the hypothesis class C (Alpaydin, 2010). In machine learning the independent variables are called *features*, and the dependent variables *labels*. In order to train a learning algorithm, D is split into a training d_{train} and test d_{test} set where $d_{train} \cap d_{test} = \emptyset$. The tuples in d_{train} are analyzed using a *machine learning* algorithm which outputs hypothesis $\mathcal{H}$ that represents the training data as well as possible, while retaining its ability to *generalize*. Generalization refers to how well the hypothesis can classify examples that are not in the train set (Alpaydin, 2010). In this chapter various machine learning algorithms for finding the best fitting hypothesis will be described, including Support Vector Machines (Cortes & Vapnik, 1995), Logistic Regression (Walker, 1967), Neural Networks, and Random Forests (Ho, 1995). The chapter concludes with a discussion regarding usage of pre-trained classifiers, called *transfer learning*.

2.1 Support Vector Machine

The Support Vector Machine (SVM) is a supervised learning algorithm devised by Vapnik in 1963. Classification is performed by finding the optimal class decision boundary, and subsequently classify the examples by using the side of the decision boundary they occur. It does this by maximizing the separability, or *margin*, of instances subject to a smoothness constraint on the solution (Alpaydin, 2010). Separation of the data is done by a hyperplane, which constitutes of finding a weight vector such that:

$$r^t(w^T x^t + w_0) \geq +1 \tag{2.1}$$

Where x^t is training example t, w^T is the weight vector, w_0 is the bias term, and $r^t = +1$ if $x^t \in C_1$ or $r^t = -1$ if $x^t \in C_2$. If the data is linearly separable, meaning that the classes can be separated using a single linear decision boundary, it would be intuitive to separate the data using a hyperplane. One of the complications with such separable data is that there will be infinitely many candidate lines that split the classes. This is solved by selecting the hyperplane which has the largest possible distance, or margin, between the training examples and the separating hyperplane. This is called the *maximal margin hyperplane*, which will always hold the optimal solution for the linearly separable case:

$$\min \quad \frac{1}{2}||w||^2 \quad \text{subject to} \quad r^t(w^T\mathbf{x}^t + w_0) \geq +1, \quad \forall t \tag{2.2}$$

or written as an unconstrained problem using Lagrange multipliers α:

$$L_p = \frac{1}{2}||w||^2 - \sum_t \alpha^t r^t(w^T x^t + w_0) + \sum_t \alpha^t \tag{2.3}$$

where we try to minimize L_p with respect to W, and maximized with respect to $\alpha^t \geq 0$ (Alpaydin, 2010). The margin is determined by the distance from the maximal margin hyperplane and the nearest training examples from each class. See Figure 2.1 for an example. These training examples are called the *support vectors* and are the only training examples that determine the position of the hyperplane. The case where the data is linearly separable is called *hard-margin*. In most real life scenarios however, there exists no linear separating hyperplane.

One solution would be to loosen the hard-margin and only state that *most* of the training examples should be on the correct side of the hyperplane. In general this results in greater robustness to individual data points and better classification for the largest portion of the data since one data point cannot have a large influence on the hyperplane. This is accomplished using a *soft-margin*, which adds slack variables ξ_i and a non-negative turning parameter C to equation 2.3 (Alpaydin, 2010):

$$L_p = \frac{1}{2}||w||^2 + C\sum_t \xi^t - \sum_t \alpha^t[r^t(W^T x^t + w_0) - 1 + \xi^t] - \sum_t \mu^t \xi^t \tag{2.4}$$

The *Penalty parameter C* acts as a sort of budget on how harsh it should penalize misclassification in the training set. As C increases the penalty for misclassification increases and separating the classes becomes increasingly more important than finding the maximum margin (Hastie, Tibshirani, & Friedman, 2001). Selecting a too large value for C will make the classifier prone to *overfitting*, i.e. a too complex model that learned too much from the data. Conversely, selecting a value too low results in *underfitting*, i.e. the model did not learn enough from the training data. Thus C is an important parameter in the bias-variance trade-off for the SVM.

Most of the data sets will yield poor performance using a linear decision boundary due to their highly non-linear nature. Learning a function from these data sets is made possible for the SVM by using non-linear *basis functions*. These include polynomials, radial-basis, and sigmoidal basis functions. This allows the SVM to transfer the input vector to a feature space with one dimension for each training example.

The radial basis function will be applied in this research and is given by the following equation

$$K(x_i, x_j) = e^{-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2} \tag{2.5}$$

One important parameter for the kernel functions is γ, which determines the relative importance of data examples near the decision boundary with respect to the other examples.

There is no formal method for finding the optimal values for C and γ. One often used approach is *grid search* which is an exhaustive search through a user specified parameter space. The search is guided by some specified performance metric, usually cross-validation (Hsu, Chang, & Lin, 2010).

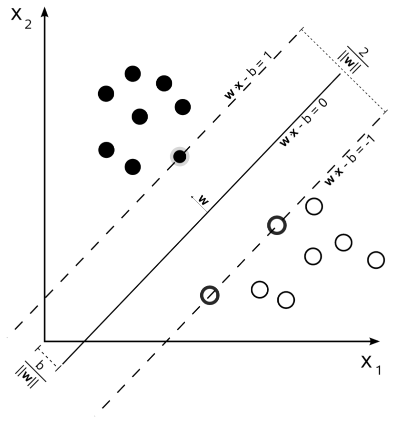


Figure 2.1: Example of a linear hyperplane separating classes. From: Wikipedia Support Vector Machine

2.2 *k*-Nearest Neighbor Classifiers

k-Nearest Neighbor (KNN) is a *memory-based* classification algorithm (Hastie et al., 2001). This means that the training data is represented as a set of points in feature space and no learning is required. Given a new training example x_0 the classifier selects $x_{(r)}, r = 1,..,k$ closest examples and performs classification using the majority vote among these k neighbors. Ties in voting are broken at random. *Closest* between two vectors x and j is determined using a distance metric, commonly Euclidean distance:

$$d(x,j) = ||x-j|| = \sqrt{(x_1-j_1)^2+(x_2-j_2)^2\ \dots\ (x_n-j_n)^2} \tag{2.6}$$

Classification is based on the idea that points that are located closely to each other in feature space will more likely share the same class than points that are further apart. Usually the features are standardized with a zero mean and a variance of one, allowing for features that are measured in different units. The main advantage of KNN is that no learning is required and only two hyperparameters need to be determined, the value k and a distance metric. The optimal number of neighbors is usually found using an exhaustive search from $k = 1$ to n, where n is some reasonable number. A low value of k potentially makes the algorithm susceptible to outliers and noise, i.e. overfitting since classification is based on too few examples. A large k can result in a too smooth decision boundary, underfitting.

2.3 Random Forests

Decision trees lie at the heart of the random forest classifier (Ho, 1995). Decision trees are a tree-based graph structure which partition the samples using the available features into a number of branches, terminating into leaf nodes when either the features are exhausted or a pre-set maximum depth is reached. A common method for choosing the best feature for branching uses *entropy* and *information gain*. Entropy, or impurity I, is used to calculate the homogeneity of the sample.

$$I_m = -\sum_{i=1}^{K} p_m^i \log_2(p_m^i) \tag{2.7}$$

Where K is the set of all classes, and p_m^i is the probability of class i at node m. If a node m is not pure more instances should be split in order to decrease impurity. In order to keep the tree as small as possible the feature that decreases the impurity the most promising feature should be selected, in our case this is the feature that decreases the impurity the most. The impurity after the split for node m is calculated as follows:

$$I'_m = -\sum_{j=1}^{n} \frac{N_{mj}}{N_m} \sum_{i=1}^{K} p^i_{mj} \, log_2 \, p^i_{mj} \tag{2.8}$$

Where N_{mj} are all the training examples will end up in branch j, p^i_{mj} is the probability of class j at node m. Lastly, information gain determines the improvement of the chosen split

$$\text{information gain} = I_m - I'_m \tag{2.9}$$

The feature with the highest information gain is chosen for splitting in the next step. After all the features have been used or a maximum depth has been reached the leave is assigned to the class with the highest class probability. The Random Forest algorithm uses bootstrapped aggregation, or *bagging*, for generating multiple trees from a single training set (Hastie et al., 2001). Bagging refers to selecting N training examples from the entire data set, with replacement. Next, instead of using all the available features the decision tree selects a subset K of the candidate features for splitting. Choosing the value K can drastically influence performance and is a hyperparameter to be determined using cross validation. Bagging in combination with feature subset selection at each split causes the final decision trees in the random forest to be different, thus likely making different errors. This is called to have a *decorrelated* effect (Hastie et al., 2001). In classification the class is selected by a vote from each tree, weighted by their probability estimates p_t given some input vector $\mathbf{x}$ with the highest probability.

$$c^* = \max_c \, p(c|\mathbf{x}) = \frac{1}{T} \sum_{t}^{T} p_t(c|\mathbf{x}) \tag{2.10}$$

The loss of its interpretability is a clear disadvantage of using a set of decision trees, as the final prediction is based on a weighted vote from each of the trees.

2.4 Neural Networks

Neural networks encompass a large class of models and learning methods. In this research the focus will be on *feed forward* neural networks, meaning that there are no closed directed cycles between nodes in the network. First, multilayer perceptrons (MLPs) are described and how training is accomplished. Then a model exploiting the local connectivity called the convolutional neural network is explained. The discussion finishes with methods for skipping much of the heavyweight training known as transfer learning.

2.4.1 Multilayer Perceptron

The multilayer perceptron is a class of feed forward artificial neural networks. The algorithm consists of at least three layers, the *input* layer, one or more *hidden* layers, and one *output* layer. An example network is shown in Figure 2.2. The input layer is analogues to the input vector described before and is used to feed the information into the network. Subsequently the information flows through n hidden layers with m nodes. Each of these nodes is a simple building block called the *perceptron*. The perceptron is a simple classifier that learns a N dimensional weight vector which represents a hyperplane in $\mathbb{R}^N$ dimensions using some non-linear activation function (Rosenblatt, 1958). In other words, the neural network applies the basic nonlinear basis functions $\psi_j(x)$ by making them depend on parameters and these parameters are adjusted during training (Bishop, 2006). First M linear combinations of the input features are calculated:

$$a_j = \sum_{i=1}^{D} w_{ji}^{n} x_i + w_{j0}^{n} \tag{2.11}$$

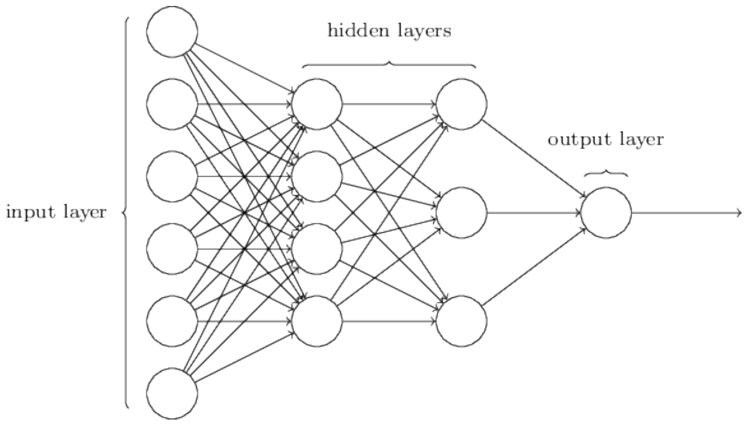


Figure 2.2: Example of simple MLP. From https://codeburst.io/

Where w_{ji}^{n} are the *weights* and w_{j0}^{n} the *bias* of layer n. Each activation is subsequently transformed using a nonlinear, differentiable *activation function*

$$z_j = h(a_j) \tag{2.12}$$

These hidden units are subsequently used to calculate the activation for the next layers:

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)} \tag{2.13}$$

This non-linear activation function is necessary since otherwise a network would only be able to learn linear functions. This occurs since a composition of successive linear transformations is itself a linear transformation (Bishop, 2006). The choice of activation function is a hyperparameter. The output of each of these perceptrons in layer n is given to layer $n+1$. The output layer does the actual classification based on the input given from the previous hidden layer. One common activation function for the output layer is the *softmax* activation.

$$\sigma(a) = \frac{e^{a_k}}{\sum_j^K e^{a_j}} \tag{2.14}$$

The main advantage for using this activation function in the output layer is that it will result in a probability density function over all the classes. The various layers of the network are grouped into a single equation used for moving through the network, i.e. *forward propagation*

$$y_k(x,w) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{2}\, h\left(\sum_{i=1}^{D} w_{ji}^{1} x_i + w_{j0}^{1} \right) + w_{k0}^{2} \right) \tag{2.15}$$

Thus one output node represents the probability of a given input vector belonging to that class. Learning is achieved by iteratively updating the weights in the network using *back-propagation*. Using a given loss function the difference between the actual and expected result is used to update the weight vector

$$W^{\tau+1} = W^{\tau} - \eta \Delta E(W^{\tau}) \tag{2.16}$$

Where $\eta > 0$ is the learning rate. One main advantage of using an MLP, or any type of neural network for that matter, is that no feature engineering is required, the network learns its own representation using the weights in the network.

One of the main disadvantages of neural networks is the number of hyperparameters to select. Some of the hyperparameters to be set are: the number of hidden layers, number of nodes in each layer, the optimizer, learning rate, activation functions, and loss function. Furthermore, specific settings such as early stopping or momentum also have influence on the algorithm's performance.

2.4.2 Convolutional Neural Networks

Convolutional Neural Networks, or CNNs, are a special type of feed forward neural network devised for processing data that are represented in a grid like structure (Lecun, Bottou, Bengio, & Haffner, 1998). This makes a CNN especially functional for the classification of images since they come naturally in a two dimensional grid of numbers. The main building block of CNNs are *convolutions*. These convolutions are applied in place of general matrix multiplication (Goodfellow, Bengio, & Courville, 2016). This idea is related to the convolution of two signals:

$$f[x,y] * g[x,y] = \sum_{n_1} \sum_{n_2} f[n_1,n_2] \cdot g[x-n_1, y-n_2] \tag{2.17}$$

Where $f[n_1,n_2] \cdot g[x-n_1,y-n_2]$ is the element wise multiplication and sum of a filter and signal, i.e. taking a filter and sliding it over an input image while computing the dot product at every location.

The image that is being convolved over is called the *input*, the convolving filter a *kernel*, and the resulting output the *activation map*. These steps make up a convolutional layer. Often, a non-linear activation function and *pooling* after each convolutional layer are applied. Pooling is a discretization process applied to reduce the spatial size of the representation in order to reduce the amount of parameters and computation required in the network. A frequently applied pooling method is *max-pooling*, which takes for example a max over four number (2×2 region) and is used to prevent overfitting. Furthermore, max-pooling introduces *translational invariance*, by making the network robust for small translation of the input (Goodfellow et al., 2016). Translational invariance means that the system can detect an object, even if it occurs at different places in an image.

CNNs have three major improvements over classical MLPs, *sparse (local) interactions*, *parameter sharing*, and *sparse weights*. These ideas greatly improve calculation speed and accuracy. As described in section 2.4.1 a traditional fully connected network will use each of the input units to generate the output for each output unit. Especially using images, which can consist of millions of pixels, this poses a problem. If there are m input units and n output units, a total of $m \times n$ parameters are required on each pass. CNNs solve this by using a kernel that is smaller than the input image. Using a kernel with size k only $k \times n$ parameters are required, greatly reducing runtime considering k will be much lower than m (Goodfellow et al., 2016). Parameter sharing refers to using the same weight vector during a convolution. Where a dense network only uses each calculation between the input and output unit once. The convolutions "re-use" the same convolution matrix for the input image, greatly reducing storage requirements. Furthermore, parameter sharing causes the network to have the *equivariance* property. For a layer to be equivariant means that when the input changes, the output changes in a similar way, i.e. $f(g(x)) = g(f(x))$.

CNNs revolve around the idea that these convolutions learn features from the previous layer, also called an *activation map*. In this process a kernel is moved across the input volume, where for every location in the input volume, called a receptive field, the dot product between the entries of the filter and the receptive field. Lastly, a non-linear activation for each node is calculated. This results in a feature map of the input volume for that specific filter, indicating the extent to which the filter matches its receptive fields. Since the convolutional layers learn features from the previous layer more complex patterns can be learned. See Figure 2.3 for an example. With these receptive fields the goal of the CNN is to learn the filters that extract the most relevant features for the final classification task.

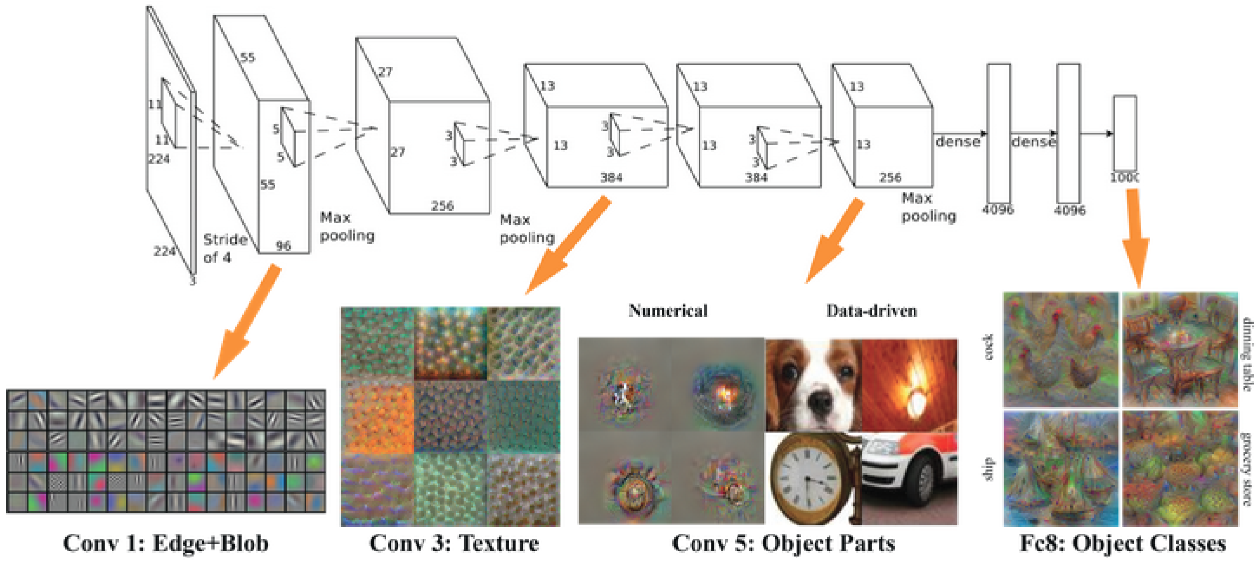


Figure 2.3: Features extracted by CNN. From vision03.csail.mit.edu

The final layers of a CNN are fully connected layers, which use the last feature maps to perform the actual classification. These last fully connected layers are analogous to the multilayer perceptrons described in section 2.4.1. The last layer in the network uses a *softmax* function to derive a probability function for the classes. This function is used to calculate the loss and subsequently update the weights in the network. Learning is performed using the backpropagation algorithm as explained in section 2.4.1.

2.4.3 Transfer Learning

Traditionally CNNs require large training sets in order to learn to discriminate between the various target classes. Not only is a large set of training examples required, also enough different logo examples need to be present in order to learn proper local features, i.e. feature maps, of the class. For this reason a large pre-trained CNN usually serves as feature generator, as proposed earlier in (Donahue et al., 2014), a process called *transfer learning*. There are three basic scenarios in transfer learning, *fixed feature extraction*, *fine-tuning*, and *pretrained*. In fixed feature extraction, all but the fully connected layers of the network are retained. The convolutional layers are then used for retrieving features. As shown in Figure 2.3 each layer adds more complex features, starting with simple features such as *Gabor filters* or color blobs, ending with complex features such as faces (Yosinski, Clune, Bengio, & Lipson, 2014). The generated features are then fed into a classical learning algorithm, as proposed in (Kim, Kavuri, & Lee, 2013). Although this loses the advantage of having an end-to-end classification, it does greatly reduce the need for computation (Bengio, 2012).

Fine tuning is based on the idea that although the network has learned generic features in the earlier layers the features become more specific to the original data set it has been trained on deeper into the network. (Yosinski et al., 2014). The first layers are kept fixed and the training data is used to fine tune the deeper layers of the network. When the data set is small it should be noted that fine tuning tends to overfit fast (Yosinski et al., 2014)

The pretrained method is used for custom networks. In this case an individual network is defined but instead of training the weights of the new network from scratch the weights from one of the large pre-trained networks are used. Although this approach seems similar to the fine tuning there is one significant difference, no layers are fine tuned with the training data available.

Chapter 3

Object Detection

In this section the theoretical background for logo detection will be explained. Logo detection is the process of finding and classifying one or more objects in an image. It is an application of a well studied problem in the field of computer vision and image processing, called *object detection*. Object detection can be split into two parts: localizing the relevant *Regions of Interest* (RoI) in an image, and secondly classifying these RoIs into a given number of target classes. First, in section 3.1, a short history of various object detection algorithms is given. Section 3.2 will describe three methods for localization: binarization, edge detection and selective search. Lastly, section 3.3 describes the combination of the previously described methods with CNNs (section 2.4.2) into a modern approach named R-CNN (Girshick, Donahue, Darrell, & Malik, 2013).

3.1 History of Object Detection

Historically, object detection is accomplished by hand crafting features and subsequently feeding these into a classification algorithm (Viola & Jones, 2001). One of the main disadvantages of such an approach is that different orientations or partial occlusion of the object potentially results in poor performance. Research conducted by (Dalal & Triggs, 2005) showed a more robust feature descriptor based on a *Histogram of Oriented Gradients*, or HoG descriptor. The primary concept of the algorithm is for each pixel to look at its neighbouring pixels and determining the gradient for the given pixel. Secondly, an arrow representing the magnitude is placed pointing to the steepest ascent. This process is repeated for every pixel, resulting in a matrix of gradient arrows representing magnitude. Then, the image is split into a regular grid and in each square the number of gradients in every major direction is counted and generated into a histogram of *n* bins. See Figure 3.1 for an example. This gradient matrix, or descriptor, will serve as a feature map for a learning algorithm. For example an SVM (Dalal & Triggs, 2005)

The *Scale-invariant feature transform*, or SIFT, is another method for object detection (Lowe, 1999). SIFT generates features by computing the gradient of each pixel in the vicinity of detected key points (Szeliski, 2010). These key points are then stored into a database. During classification, a distance metric such as Euclidean distance is used to find the target class with the most similar key point features. One main advantage of using key points instead of the entire object, is that the feature descriptors do not change when the image changes scale. Hence the name scale-invariant.

In 2012, the idea that object recognition should be done with well-crafted features fundamentally changed with the seminal paper of Krizhevsky et al. Using a large set of images, the convolutional layers in the network could learn the feature descriptors directly from the input and adapt to new information during training.

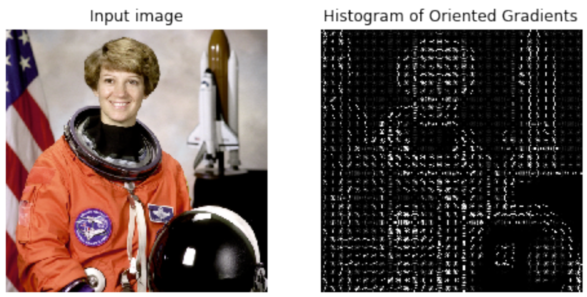


Figure 3.1: Example Histogram of Gradients. From http://scikit-image.org

During the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) CNNs scored significantly higher than any handcrafted method[1]. Together with increasing GPU power and large annotated data sets, CNNs have become state-of-the-art for image classification.

3.2 Localization of Objects

Classification of relevant objects in an image comprises more than just classification. One of the main difficulties is detecting *where* the relevant objects are before trying to classify them. One simple approach for finding the relevant objects in an image would be to move a sliding window over the image and perform a separate classification for each image patch generated by the sliding window. There are a numerous problems with this approach:

1. *Various object sizes*: the object to be classified can have various sizes or aspect ratios.
2. *Overlapping*: the possibility that an object could be partially obscured or have some other complex shape.
3. *Multiple responses*: a large number of neighbor patches could give the same classification.

It is possible to solve most of these problems with different implementations of a sliding window. Finding the object whilst the size of the object varies per image could be solved by using various sizes of bounding boxes and move them all over the image. One clear disadvantage of this brute force method that it vastly increases *computational complexity* due to the numerous sizes that will have to be tried. A more efficient approach would be to use the information in the image to propose regions where target objects might occur.

3.2.1 Binarization

Binarization is the simplest method for finding potential objects in an image. It is an image processing method which converts an RGB pixel image into a two valued image, with the possible values $[0,1]$ (Jyotsna, Chauhan, Sharma, & Doegar, 2016). The first step in the binarization process is turning the three channel RGB image into a one channel grey-scale image. This is accomplished by taking a weighted average of the three color channels, $\lambda_r R + \lambda_g G + \lambda_b B$, where $\sum_x \lambda = 1$ and are defined by some grey scale operator[2]. Consequently,

[1]www.image-net.org/challenges/LSVRC/2012/results.html
[2]https://www.w3.org/Graphics/Color/sRGB

each pixel in the image will be in the range $[0,255]$ given 8-bit quantization. The main advantage of using grey-scale images is the reduction in model complexity, since only one value will have to be considered for each pixel. The last step is to threshold each grey-scale value into a $[0,1]$ binary value using either a *global* or a *local* threshold function. Thresholding can critically effect the performance of successive steps, e.g. the classification of the proposed RoIs(Abak, Baris, & Sankur, 1997). Global threshold functions apply a single threshold for each pixel in a target image. Let T be the threshold applied to pixel g. The background of the binarized image consists of all pixels where $0 \leq g \leq T$ and the foreground is given by all the pixels where $T+1 \leq g$. Determining the optimal threshold T can be a tedious process, and thus various methods have been proposed in order to automate this (Szeliski, 2010). Local thresholding refers to selecting a collection of pixels in the vicinity of the given pixel in order to determine the threshold value. Thresholding methods are grouped into the following six groups according to the information being exploited (Mehmet Sezgin, 2004).

1. *Histogram-based*: where for example the peaks or valleys of smoothed histograms are analyzed.
2. *Clustering-based*: clustered in two classes, foreground and background, or as a mixture of two Gaussians.
3. *Entropy-based*: exploiting the entropy between the foreground and background pixels or the cross-entropy between the original and binarized image.
4. *Object attribute-based*: measuring similarity between the various objects.
5. *Spatial*: methods using higher order probability distributions and correlations between pixels.
6. *Local*: adapting the threshold value of each pixel with respects to its surrounding pixels.

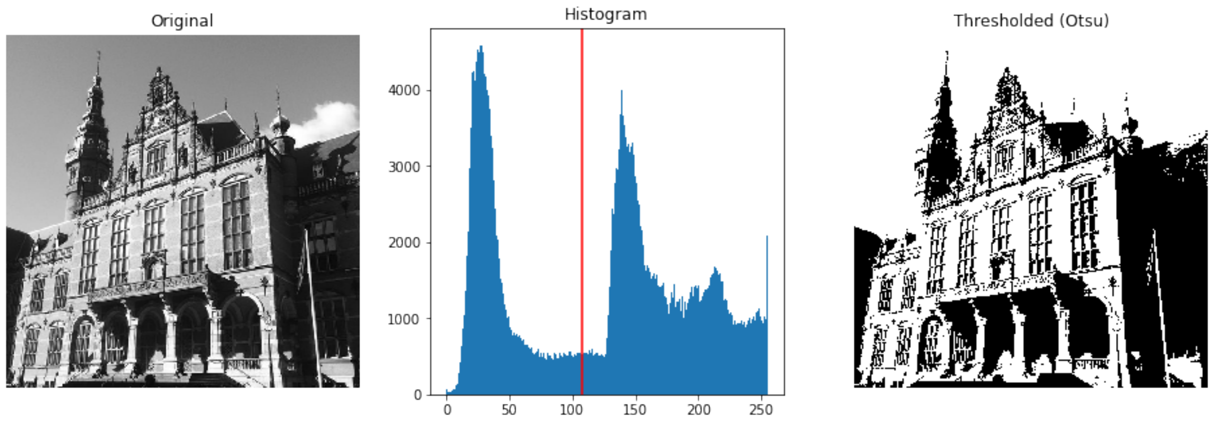


Figure 3.2: Example binarization using Global Otsu method

One popular method is the clustering based Otsu Method (Mehmet Sezgin, 2004). Given its popularity this method will be described in more detail. Otsu's thresholding comprises over iteratively moving through all the possible threshold values and calculating a measure of spread of the pixels on each side of the threshold, i.e. measuring how different each value is next to each other. The aim of the method is to find a threshold where the foreground and background spreads are at its minimum, or conversely, maximization of the between-class variance. The weighted within-class variance is given by the following equation

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t) \tag{3.1}$$

Where $q_1(t) = \sum_{i=1}^{t} p(i)$ and $q_2(t) = \sum_{i=t+1}^{T} p(i)$ are the class probabilities separated by threshold T, and $\sigma_x^2(t)$ is the variance of class x. Figure 3.2 shows a binarization of a grey-scale image of the academy building at the university of Groningen. The vertical line in the histogram depicts the optimal thresholding value T found by the global Otsu method.

One important assumption of global methods is that the entire image can be thresholded using one histogram. Although this greatly increases speed since only one histogram has to be calculated for the entire image, it greatly decreases performance for images with a shaded background. These include images with for example gradients (Puneet & Garg, 2013).

3.2.2 Edge Detection

Another commonly used method for finding Regions of Interest for classification is *edge detection*. Naturally, objects will be enclosed by edges. In a digital image an edge can be described as a significant local change in the image intensity, i.e. rapid intensity variation. Using this reasoning, an image can be seen as a height field of intensity variation. With the edges occurring at places of steep slopes. The slope of this height map is given by the gradient of any point **x** on the surface of the two-dimensional height field I:

$$J(\mathbf{x}) = \Delta I(\mathbf{x}) = (\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})(\mathbf{x}) \tag{3.2}$$

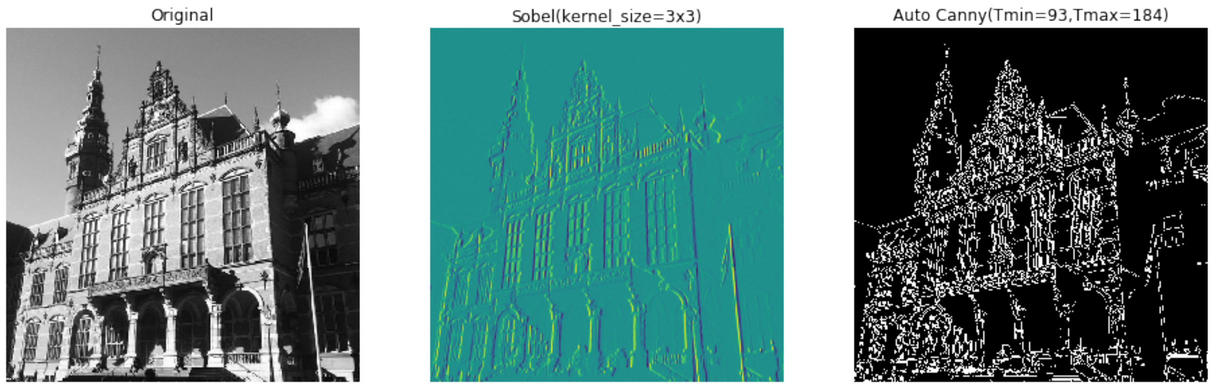


Figure 3.3: Edge detection using Sobel and Auto Canny

Exploiting this knowledge gives way to various methods of detecting edges in an image, including the Sobel (Sobel, 2014) and Canny (Canny, 1986) operator. The Sobel operator uses a 3×3 filter to find approximations of horizontal and vertical lines in the original image. Canny edge detection consists of four fundamental steps: preprocessing, calculating gradients, non-maximum suppression, and thresholding with *hysterisis*. In the preprocessing step an image is made optimal for finding edges. Usually this consists of converting the image into grey-scale and applying a Gaussian filter. Next, the gradients for each point in the image are calculated. The gradient in this context refers to how much the image is changing with respect to its color. A high gradient implies that there is a large color change, in turn implying that there is possibly an edge. Next, non max suppression is applied to retain only the most probable gradient. The last step is to use these gradients for thresholding using hysterisis. These edges can then be used to find objects in the image (Lai, Lin, Chen, Kang, & Yeh, 2013) (Zitnick & Dollár, 2014). An example edge detection can be found in Figure 3.3.

3.2.3 Selective Search

The above mentioned methods lack one important feature, namely that images are intrinsically *hierarchical*. This means that objects tend to occur on top or in front of each other. *Selective search* tries to implement this feature by applying a hierarchical partitioning of the image. Partitions are based on various sampling techniques, e.g. color, texture, enclosure. Due to the nature of objects occurring inside each other, most contemporary methods apply localization through the identification of an object (Uijlings, van de Sande, Gevers, & Smeulders, 2013). However, applying a regular grid for identification would quickly become computationally intractable. For this reason, selective search has been implemented to find regions of interest through diversifying sampling on a bottom up segmentation of a target image.

The hierarchical grouping algorithm forms the basis for selective search. Since bottom up grouping is applied, regions of all scales can be found. This is an important improvement for finding the regions of interest at various scales. For generating the starting regions, while preventing that they span multiple objects, the Felzenszwalb-Huttenlocher algorithm is used. From these starting regions, the most similar neighbouring region is selected and merged into a new region. This process is repeated until the whole image constitutes a single region. As described above, the similarity is based on various sampling strategies.

One strategy is that of *complementary color spaces*. To account for different lightning various color spaces such as RGB, grey-scale, and HSV are applied. In total, four similarity measures are used to determine the similarity s between region r_i and r_j:

$$s(r_i, r_j) = s_{colour}(r_i, r_j) + s_{texture}(r_i, r_j) + s_{size}(r_i, r_j) + s_{fill}(r_i, r_j) \tag{3.3}$$

Where $s_{colour}(r_i, r_j)$ is the color similarity, $s_{texture}(r_i, r_j)$ is the texture similarity, $s_{size}(r_i, r_j)$ is similarity in size, and $s_{fill}(r_i, r_j)$ is the measure of enclosure between r_i and r_j. The resulting 2000 bounding boxes serve as the regions of interest for classification, see Figure 3.4 for an example.

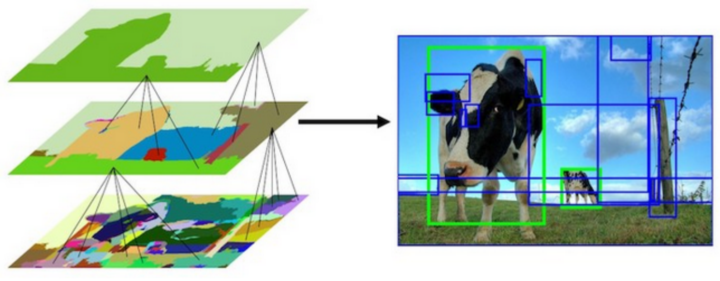


Figure 3.4: Hierarchical grouping of regions using selective search

3.3 Regional Based Convolutional Neural Networks

Regional Based Convolutional Neural Networks (R-CNNs) are devised to solve the entire object detection pipeline, from the initial region of interest proposals to the final classification of relevant image patches (Girshick et al., 2013). The first step of the R-CNN pipeline is to propose a list of RoIs to decrease the number of classifications that otherwise would have been necessary using a sliding window approach. Usually a traditional image processing technique is applied to quickly find these regions. These include methods such as *selective search*, as described in section 3.2.3. Another method which, exploits the edges found in the target image, is given by (Zitnick & Dollár, 2014).

The image patches extracted from the RoIs can occur in various sizes. Since a CNN requires a fixed size input these image patches are warped into a fixed square size which has the right input dimensions for the CNN (Girshick et al., 2013). Subsequently, each of these image patches are run through the network to generate a feature vector, and classified using a classifier such as an SVM. Moreover, a method called bounding box regression is applied to correct for any imperfect box retrievals (Girshick et al., 2013). One of the main problems with this approach is that for each of the bounding boxes a feature vector needs to be generated using the CNN and a classification needs to be done.

To improve performance, fast R-CNN has been developed (Girshick, 2015). In this method the whole image is run through a network, generating a convolutional filter map for the entire image. Then, using the region proposal algorithm as before the relevant regions in the image are located. But now, the regions are mapped onto the extracted convolutional filter. This allows for sharing the convolutional computations between various RoIs and thus speeding up the process. Since still some fixed size is required for the fully connected layers, a method called *RoI pooling* is applied to warp the RoIs from the feature map into the desirable size.

Due to this sharing of the convolutional filter, classification speed increases significantly. The region proposal method, however, now becomes the bottleneck in the pipeline. To overcome this bottleneck, a method called Faster R-CNN has been proposed (Ren, He, Girshick, & Sun, 2015). Instead of using a separate region proposal algorithm, the regions are done making the CNN itself do the region proposals, creating a unified model. This method has been extended by applying a per pixel level image segmentation (He, Gkioxari, Dollár, & Girshick, 2017).

Chapter 4

Data and Experimental Setup

In this chapter the methodology for answering the research questions mentioned in the introductory section is described. Section 4.1 is dedicated to gathering training data and annotation. Section 4.2 explains the proposed method of feature selection for the baseline classifier. Furthermore, the applied preprocessing and hyperparameter optimization results are described. Lastly, section 4.3 describes the methods for logo detection.

4.1 Training Data

All meta-data that is used for this research is acquired from the dataprovider database. The database contains 172 meta-features consisting mostly of public data from approximately 300 million domains. This data is gathered by various web spiders that scrape the known domains on a monthly basis. These raw pages are subsequently parsed into meta-features and together with the HTML stored into a database. These 172 meta-features serve as the data for the meta-features classifier.

For training the meta-features model a set of positive, i.e. online stores that are fraudulent, and negative examples are required. In May 2018 the consumers association, *consumentenbond*, published an article regarding fraudulent online stores[1]. This article enclosed a list of domain names of approximately 2000 fraudulent online stores made publicly available. The Enrichment API[2] was used to retrieve the relevant meta-features form each of the givens domains. This resulted in a total of 2022 indexed online stores, i.e. all the domain names that were actually indexed by dataprovider. These serve as positive examples. *Thuiswaarborg* is a Dutch national e-commerce association which provides a trust mark for legitimate online stores in the Netherlands. Since all the online stores that are listed on the website are manually checked for legitimacy, this list can confidently serve as the set of negative examples. A list of 1791 registered domain names has been scraped from the website and subsequently parsed using the Enrichment API. All the domains that did not return a `200 OK` were removed from the data set. This to make sure that only online stores that contained proper meta-features were retrieved. The final dataset consists of 3195 training examples with a total of 172 aggregated meta-features. To improve robustness, another 1920 examples have been manually annotated at dataprovider.

For retrieving image data the internal screenshot server of dataprovider is used. After running all the necessary JavaScript and CSS, a high resolution (`1920x1080`) screenshot is generated of the target domain.

[1]https://www.consumentenbond.nl/nieuws/2018/consumentenbond-laat-850-foute-webwinkels-offline-halen
[2]https://www.dataprovider.com/products/enrichment/

This is important since quite a large portion of each website is generated using Javascript and CSS. Making a screenshot of only the front page should be enough, since this research focuses on retrieving relevant social media and payment logos. These logos are, in most cases, visible on the front page. Subsequently, the relevant logos on the front page are annotated using Sloth[3]. The `x`, `y`, `width`, and `height` coordinates of each annotation are stored in JSON-format. In total 243 front-pages have been annotated, resulting in a total of 715 logos.

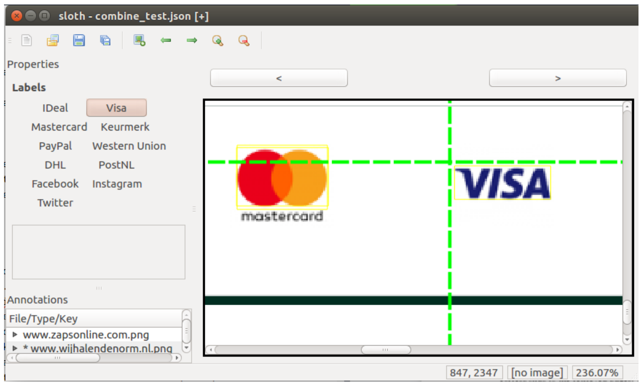


Figure 4.1: Example logo annotation using Sloth

4.2 Baseline Experiments

4.2.1 Feature Selection

Feature selection is applied for two distinct reasons: selecting the most discriminative features, and increasing interpretability for researchers (James, Witten, Hastie, & Tibshirani, 2014). In total, there are 172 meta-features stored in the dataprovider database. `Matching Fields`, `Score`, `DB`, and `Response` have been dropped since these fields are only used for internal matching and searching in the database. One of the main problems with the meta-features available in the dataprovider database is missing observations. Variables with 100 percent missing data are automatically deleted since these features have zero variance, and thus do not hold any discriminative power between the classes. Remaining features with more than 80 percent missing observations are looked into case-by-case. For 43 features it was impossible to determine whether the missing data was missing at random or missing completely at random. These features have been deleted to avoid excessive bias in the training data (Barnard & Meng, 1999). In total 82 features have been deleted given the preceding arguments, leaving 90 features for further analysis. Using domain knowledge, various calculated features were found. Since these features are merely a linear combination of other variables these have been removed. Lastly, purely textual features have been removed since they cannot easily be represented numerically. These include features such as the page header or keywords.

A total of 34 features remain. After preprocessing, discussed in section 4.2.2, the most predictive features are retrieved using *recursive feature elimination* (RFE). Given an external estimator that assigns weights to each feature, e.g. the coefficients of a linear model, RFE selects features by recursively considering smaller and smaller sets of features. First, the classifier is trained on the complete initial set of features and importance of each feature is obtained. Then, the least important features are pruned from the current set of features.

[3]https://github.com/cvhciKIT/sloth

The procedure is repeated recursively on the pruned set, until the least number of features is selected while retaining n percentage of the variance. Scikit-learn provides RFE in combination with cross-validation in RFECV[4]. 5-fold cross validation with a random forest of 500 trees is applied to improve selection robustness. The *F1-score* is used as performance metric, considering it gives a single scoring metric for both precision and recall (Genuer, Poggi, & Tuleau-Malot, 2010):

$$F1 = 2 * \frac{precision * recall}{precision + recall} \tag{4.1}$$

Figure 4.2 shows the relative features importance in decreasing order. Whether a social media platform is present is clearly the most important feature. In total the top N discriminative features, such that 90 percent of the variance is explained, are selected. This results in a subset of 20 features, which are described in Table 4.1.

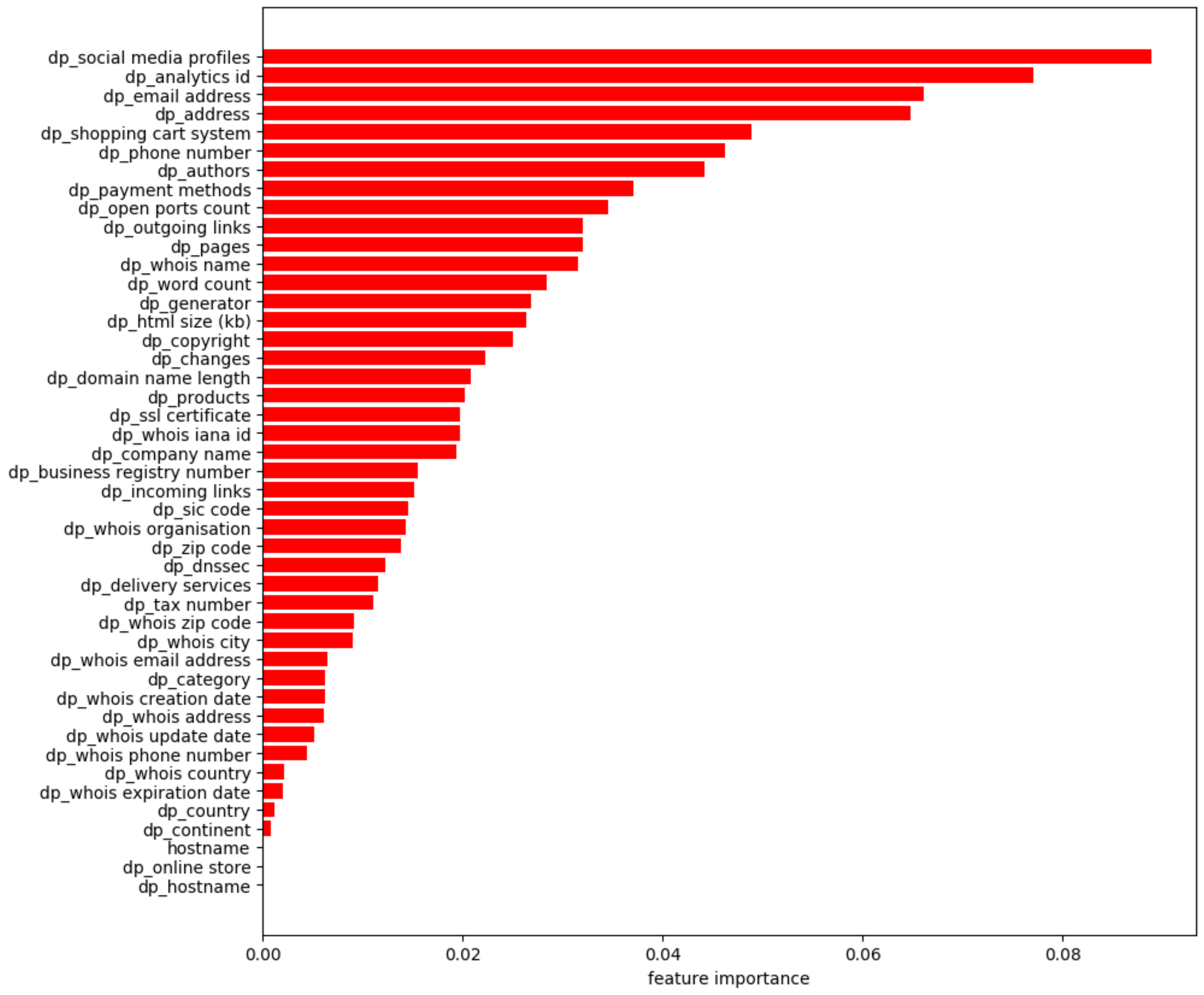


Figure 4.2: Importance of meta-features using Random Forest and Cross Validation

[4]http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFECV.html

column name	description	data type
dp_word count	calculated number of words on homepage	numerical
dp_pages	calculated number of pages	numerical
dp_HTML size (kb)	calculated size of homepage	numerical
dp_products	estimated number of products	numerical
dp_changes	number of changes since last indexation	numerical
dp_incoming links	number of hyperlinks directed to website	numerical
dp_outgoing links	number of hyperlinks on website	numerical
dp_open ports count	number of ports that is open for a website	numerical
dp_zip code quality	whether zipcode is found on website	boolean
dp_company name	whether company name is found	boolean
dp_business registry number	whether BRN is found	boolean
dp_tax number	whether tax number is found	boolean
dp_phone number	whether phone number is found	boolean
dp_email address	whether email address is found	boolean
dp_payment methods	whether payment methods are found	boolean
dp_copyright	whether the homepage has a copyright	boolean
dp_analytics ID	whether an analytics id is found	boolean
dp_address	whether an address is found	boolean
dp_social media profiles	whether social media is found	boolean
dp_shopping cart system	whether a shopping cart system is found	boolean

Table 4.1: Description selected meta-features in dataprovider.com database

4.2.2 Pre-processing

Data imputation is required for two reasons: 1) it can greatly improve performance, and 2) usually learning algorithms cannot handle missing values. *Multiple Imputation by Chained Equations* (MICE) has emerged as the principle method of dealing with missing data (Azur, Stuart, Frangakis, & Leaf, 2011). One of the main advantages of using MICE is its flexibility and allowance for various types of data, e.g. coninuous and binary. One of the main problems with single imputation methods, e.g. as *mean imputation*, is that imputed values are regarded as true values, rather than imputed (Graham, 2009). The multiple imputation method involves creating multiple *complete* data sets by filling in the missing values numerous times. Since there are multiple predictions for each missing value, the uncertainty of the imputations are taken into account (Azur et al., 2011). One important assumption of MICE is that the missing values are missing at random. The first step is to perform a simple imputation method, such as mean imputation. Next, the missing values are placed back for one random variable. This variable serves as the dependent variable in a regression model. All the other features are the independent features. Subsequently, the missing values in the dependent variable are set with the predictions from the regression model. This process is repeated until each variable has served as the dependent variable once. This is counted as one cycle. This procedure is repeated a number of times, generally ten times (Lee & Mitra, 2016).

For distance based classifiers, e.g. KNN, it is important to scale data. This prevents that more weight is given to features with larger numerical ranges (Aksoy & Haralick, 2001). Normalization is important for a gradient based learning algorithm, since it greatly improves speed of convergence (Aksoy & Haralick, 2001).

Gradient based learning algorithms include for example the multilayer perceptron. For all numerical features *standardization* has been applied. Standardization places the mean of each variable at zero and the variance at one, and is given by the following equation:

$$x' = \frac{x - \bar{x}}{\sigma} \tag{4.2}$$

Where x is a given feature, $\bar{x}$ is the feature's mean, and σ the corresponding standard deviation.

4.2.3 Baseline Hyperparameter Optimization

The classifiers described in Section 2 are applied to determine baseline performance. A separate test set is held back to give a fair test performance on truly unseen data. To generate robust results, while testing for overfitting, 10-fold cross validation is applied to all classifiers (Kohavi, 1995). The optimal parameters, as specified in chapter 2, are determined for the various classifiers.

For finding the optimal parameters for the SVM, an exhaustive grid search is applied. This applies a brute force method on all the possible combinations on the given parameter values, and stores them in a hyperparameter space grid. See Figure A.1 for the grid search space results for the SVM. What immediately becomes apparent is a diagonal line found in the best values, caused by a larger value for C requiring a smaller value for γ. The best performing parameters are $C = 2$ and $\gamma = 0.03125$, and these were selected for producing the baseline results.

No grid search is required for the KNN algorithm. For optimization, an exhaustive search is run from $K = 1$ to $K = 30$. An odd number of neighbors is selected to prevent the case where each class is equally likely. As seen in Figure A.1, the optimal number of neighbors is 5. For baseline classification $K = 5$ is selected.

For the Random Forest classifier, two parameters are selected for optimization: the minimum number of sample leaves, and the maximum percentage of features. See Figure A.1 for the grid search results. What becomes apparent is that a small number examples are needed in order for the branch to turn into a leave node. This means that quite some branches are required before the classifier can generalize well. The maximum percentage of features does not seem to have a real influence on performance. The best performing parameters, $max\_features = 0.40$ and $min\_samples\_leaf = 1$, are selected for baseline classification.

The best performing classifiers with their corresponding accuracy, precision, recall and F1-score are given in Table 4.2. It shows the random forest classifier is the best performing classifier, both in average metric score and in standard deviation. The standard deviation is calculated for each run to determine the variance of the K-fold. A lower variance implies that the learning algorithm is less prone to overfit and will usually perform better (Kohavi, 1995). The results on the unseen test set are described in section 5.1.

	Accuracy		Precision		Recall		F1-score	
	score	std.	score	std.	score	std.	score	std.
KNN	0.90	0.10	0.71	0.01	0.78	0.05	0.71	0.09
SVM	0.90	0.09	0.69	0.01	0.82	0.02	0.75	0.09
Random Forest	**0.95**	0.06	**0.77**	0.07	**0.88**	0.04	**0.79**	0.05
MLP	0.77	0.08	0.74	0.01	**0.88**	0.05	0.75	0.08

Table 4.2: Classification result using meta-features after optimization on 1920 online stores

4.3 Visual Features

This section describes the methods applied for detecting and classifying visual features. First, in section 4.3.1 three methods for extracting *Regions of Interest* (RoIs) in images are described. Next, in section 4.3.2 the pipeline for classifying the RoIs is described. Section 4.3.3 describes the performance metric that is applied for scoring localization and classification. In section 4.3.4, combining localization and classification of objects is described.

4.3.1 Localization

The first step in any object detection algorithm is finding Regions of Interest. The RoIs are found using two different segmentation methods, region and edge based. For region based segmentation using binarization, global and local Otsu are applied. After binarization, there exist n connected components that could be part of one of the target classes. Subsequently, the minimal up-right bounding box is placed around the connected component. Image patches are retrieved from these bounding boxes, and then used for classification.

Another method is based on finding the edges of relevant objects in the image. Calculating the gradient is easier using a one-channel image, thus the image is first converted into grey-scale. A 5×5 Gaussian filter is applied to reduce large gradients, enhancing edge detection performance (Szeliski, 2010). A zero-parameter canny edge detector is applied to find edges in the pre-processed image[5]. The main advantage of using a parameter free method is that it alleviates the burdon of manually finding the optimal min and max thresholding value for each image.

Regardless of the segmentation method, some *RoI selection* is applied on the bounding boxes to increase performance. This selecting primarily focuses on selecting the bounding boxes that approximately share the same aspect ratio as the annotated classes, and are of some reasonable size. Since logos should be large enough to be visible to the human eye, a minimum area for the bounding box can be specified. RoI selection can greatly improve classification speed, since it reduces the number of classifications the system needs to perform on each image. The optimal parameters for the min and max size of the bounding boxes are determined using hyperparameter optimization.

Using the region based binarization approach, the area of each connected component can directly be derived from the component since a region has a surface area. For the edge based method, the edges corresponding to a single object need to be enclosed. Quite often, however, it will occur that lines are not completely attached to each other. For this reason, a 5×5 dilatation filter is applied to connect the relevant edges. Next, the enclosed areas are filled using a morphological operation. This allows the area of each connected component to be calculated. Lastly, selective search as described in section 3.2.3 is also applied. Results comparing edge, region based binarization, and selective search are described in section 5.2.

4.3.2 Classification

For classification, a method based on R-CNN (section 3.3) is applied. The training examples are generated from extracting bounding boxes from the annotated JSON file, as described in section 4.1. Given the small size of the data set, training the weights for the CNN from scratch would be unfeasible. As described in (Yosinski et al., 2014), using a classical learning method for classification could yield satisfactory results. The images are loaded using `OpenCV` and re-sized into 224x224 or 229x229 images using the `PIL` library. This

[5]https://www.pyimagesearch.com/2015/04/06/zero-parameter-automatic-canny-edge-detection-with-python-and-opencv/

corresponds with the image size where many of the `Keras` pre-trained networks are trained on[6]. Multiple pre-trained networks will be tested to see which works best. Using the best performing pre-trained network, an image descriptor for each annotation is generated. A t-SNE visualization is given to show how well target classes are separated (Donahue et al., 2014). Using these input representations, the best performing learning algorithm is chosen. One of the main disadvantages of using this method is that it loses the end-to-end learning aspect of a convolutional neural network. However, considering the classification time of the fully connected layers and the sparsity of data training, an external classifier on top of the learned features will possibly yield better performance. The architecture is depicted in Figure 4.3.

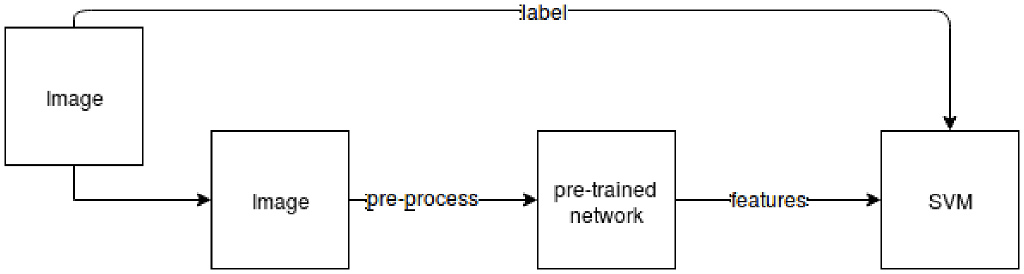


Figure 4.3: Classification architecture for logo detection

4.3.3 Performance Metric

In object detection, evaluation is a non-trivial undertaking comprising of two tasks: determining whether an object exists, and subsequently classify the relevant image patches. It has to be taken into account that, although region of interest selection is applied to decrease the number of classifications, an average image will still hold 500 to 1000 potentially relevant bounding boxes. Since both the position as the number of occurence matter in object detection, different performance metrics are required. In this section, important performance metrics for object detection are described: *Intersection over Union*, and *mean Average Precision*.

Bounding boxes have been placed around the target classes, as described in section 4.1. Intersection over Union (IoU) is the ratio between the intersection and the union of the predicted bounding box (b_p) and the annotated ground truth bounding box (b_t). When the IoU succeeds a specified threshold θ, the bounding box is regarded as a positive detection.

$$IoU = \frac{B_p \cap B_t}{B_p \cup B_t} \tag{4.3}$$

This means that the IoU is calculated by the sum of all pixels that occur in both the predicted and the target bounding box, divided by all the pixels given by the predicted and target bounding box. This definition of IoU can subsequently be used to determine the precision and recall for the logo detection system.

$$precision = \frac{TP}{TP+FP} = \frac{TP}{\text{all detections}} \qquad recall = \frac{TP}{TP+FN} = \frac{TP}{\text{all ground truths}} \tag{4.4}$$

True Positives (TP) are the bounding boxes where the IoU with the annotation is above 0.5, i.e. $IoU \geq 0.5$ and target class is correct. False Positives (FP) refer to the bounding boxes where the learning algorithm classified a bounding box as a target class while it is actually background, i.e. $IoU < 0.5$ and the predicted

[6]https://keras.io/applications/#inceptionv3

target class is wrong. True Negatives (TN) are non-existent in object detection, considering only bounding boxes, i.e. predictions, are taken into account for detection. False Negatives (FN) are the class objects that the model missed completely for any bounding box with respect to the ground truth annotation. In this research, we will only take into account the predictions that are classified as background if they overlap with the ground truth boxes above the threshold θ given for the IoU. The reason is most bounding boxes will be background, and thus classified as such. This would give a skewed impression of classification performance.

One problem with object detection is that performance is determined by two parts, recall and classification. This makes it hard to determine what which part influences performance. To address this issues, a metric called the Average Precision (AP) has been introduced (Everingham et al., 2015). *mean Average Precision* (mAP) calculates the maximum precision at different recall thresholds. Classifications are ordered by classification confidence. Selecting more examples, i.e. increasing recall, will result in selecting classifications with lower confidence and is thus prone to making errors. When multiple bounding boxes detect the same object, *non-maximum suppression* is applied to greedily select the most promising bounding box. This process is plotted into a precision x recall curve. Calculating the area under this graph with, for example, an 11-point interpolated precision gives the final mean Average Precision score.

4.3.4 Test Pipeline Performance

The previously mentioned methods for retrieval and classification all require hyperparameter optimization. One problem that arises is that the parameters are being overfit to the test data. For this reason, a separate test set with screenshots is used. This is a completely separated set, i.e. it is not used for any hyperparameter optimization in the recall step and no annotated patches are used for training the classifier. This is done to make sure that the test performance is representative for any unseen image. In total, another 82 stores have been annotated using the same approach as described in section 4.1. Performance is measured using the performance metric described in 4.3.3. This requires that for each bounding box: the predicted label, IoU, classifier certainty, and closest annotated image patch need to be stored in order to determine performance.

4.4 Combining Meta and Visual Features

In this research combining meta and visual features is done by simply concatenating the features. Although more complex methods exist, such as multimodal learning, these methods are out of scope. However, focus will be on finding descriminative features by combining the meta and visual information. Concatenating the features allows for a simple and easy to interpret model, where the linear correlation between the various features can easily be analyzed using the Pearson correlation coefficient. Given two random features X and Y and their corresponding means $\overline{X}$ and $\overline{Y}$, the correlation coefficient is calculated as follows:

$$p_{XY} = \frac{\sum(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum(X_i - X)^2 \sum(Y_i - Y)^2}} \tag{4.5}$$

Splitting the features is achieved by defining a binary feature for each of the payment methods and social media platforms found in the dataprovider database. Next, the presence of any payment or social media logo in the cropped image patches are added as a binary variable as well. Lastly, the correlation coefficient between all the features is calculated and plotted for manual investigation.

Chapter 5

Results

The following section elaborates on the results of the experiments laid out in section 4. First, in section 5.1, baseline results are shown for classification of online stores using the meta-features described in section 4.2. Next, section 5.2 shows performance for recall and classification in the logo detection pipeline. Lastly, performance for combining meta- and visual features is discussed in section 5.3.

5.1 Baseline

The baseline results for online store classification are based on 1340 manually annotated online stores. During annotation it was noticed that quite some online stores were either closed, i.e. the domain was not in use, or did not contain any information. These have been removed, resulting in a total of 1072 relevant online stores for classification. All the classifiers are trained with their optimal hyperparameter settings, as described in section 4.2.3. The baseline results are shown in table 5.1. What stands out, is that precision and recall scores are significantly lower than the accuracy score in all the classifiers. Partially, this can be assigned to the skewed nature of the target classes. With *82.2 percent* of the examples being legitimate. This skewness gives the high accuracy score a misrepresentation of classification performance.

Furthermore, it has to be taken into account that performance can slightly be altered by changing information on websites. Which in turn would yield to different classification results on each run. To make sure that each classifier uses the same data, the results in Table 5.1 are based on an export of the meta-features and not on the live application data. Moreover, these test results have been run only once to prevent optimizing on the test set. The *Multilayer Perceptron* has the highest score for both accuracy and precision. The *Random Forest* classifier has the highest recall score and the lowest cross-entropy. Overall, the Random Forest is the best performing classifier with an F1-score of 0.75. Noticeably the KNN classifier scores significantly worse than the other classifiers. Scoring lowest in precision, recall, and F1-score. Further discussion regarding the importance of precision and recall in this research can be found in section 6.1.

Table 5.2 shows class prediction performance for the legitimate and fraudulent class using the best performing Random Forest classifier. The table shows that the classifier is very much able to classify the legitimate examples, scoring an F1-score of 0.97. Classification of the fraudulent cases is more difficult, scoring 30 percent lower precision and 20 percent lower recall.

As discussed in section 4.2, the final classifier will yield a score between 0 and 100 instead of a binary prediction. Using domain knowledge it has been determined that there will be five *scoring baskets* with a semantic meaning, specified in Table 5.3. The main advantage of using such scoring baskets, is that it makes

	Accuracy	Precision	Recall	F1-score	Cross-Entropy
KNN	0.91	0.49	0.63	0.55	1.60
SVM	0.85	0.68	0.63	0.65	0.60
Random Forest	0.95	0.71	**0.80**	**0.75**	**0.32**
MLP	**0.96**	**0.74**	0.74	0.74	0.84

Table 5.1: Baseline results using meta-features on 1072 online stores

class	precision	recall	f1-score
Legitimate	0.98	0.97	0.97
Fraudulent	0.63	0.76	0.69
avg / total	0.81	0.87	0.83

Table 5.2: Best scoring meta-feature classifier on 1072 online stores

the scoring less fine grained and thus more robust for minor scoring errors. For example, an online store with a score of 0.43 would be classified as fraudulent, while an online store with 0.51 would be legitimate. In the context of trustworthiness, however, these online stores are equivalent. The best performing Random Forest classifier is used to generate the distribution shown in figure 5.1. The figure shows that most of the online stores, approximately 80 percent, are classified as legitimate. This distribution matches with previous research conducted by dataprovider.

Scoring basket	Description
0-20	Very low, almost certainly fraudulent
20-40	Low, high chance of being fraudulent
40-60	Medium, bad sites but could be legitimate
60-80	Good, most likely legitimate
80-100	Very good, very small chance of being fraudulent

Table 5.3: Semantic meaningful scoring baskets for online stores

Figure 5.1 shows the distribution of the classification scores over the various bins. The dotted vertical lines represent boundaries for the scoring bins. In the first bin, all but one of the online stores are classified as fraudulent. This is expected, since a score of ≤ 20 would presume an online store to be almost certainly fraudulent. In subsequent bins, the distribution shows a continuously increasing number of online stores being classified as legitimate. This is again a logical consequence, considering most online stores will be trustworthy. With only a small portion being legitimate but sketchy. As can be seen from the distribution, there are some fraudulent online stores classified in the highest scoring basket. Which is obviously wrong. On close inspection it was determined that three of the five shops were actually wrongly annotated, and thus should not have be considered as false negatives. Two others were indeed fraudulent and had a score of 100. These seem to be outliers and would require further investigation. Overall, the distribution of the meta-feature classifier is as expected. For visualizing the meta-features in lower dimensional space, a dimensionality reduction algorithm called t-SNE is applied. At the top of the point cloud it shows that most of the fraudulent examples occur there. As mentioned above, there were some outliers with a score of 100 percent while being fraudulent. These examples can be shown in the tail in the bottom in the point cloud.

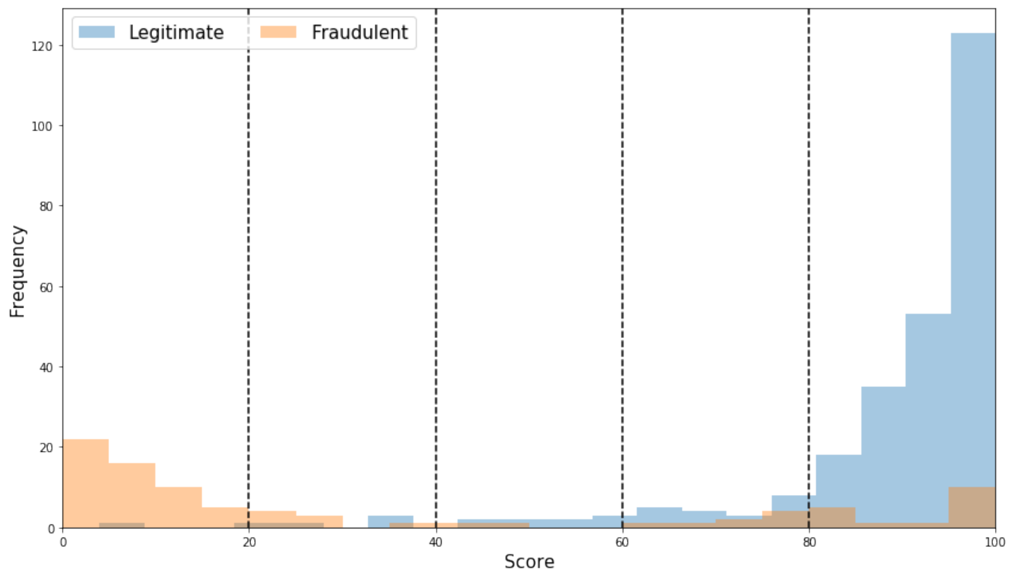


Figure 5.1: Scoring distribution of the SVM based meta-features classifier

5.2 Visual Features

For training the logo detection classifier, a total of 238 online stores have been annotated. This resulted in 1088 annotations. The frequency of each logo class is depicted in Figure 5.3. Another 750 random bounding boxes have been added for learning the background class. To make sure that no parts of any annotation are being used for the background class, only patches with $IoU = 0$ with any of the annotated image patches are selected. First, the performance for each step in the object detection pipeline is presented separately. This shows an isolated impression of the parts that constitute the pipeline, and prevents learning from the test set during hyperparameter optimization. In section 5.2.3, the mAP for the entire pipeline is given on an unseen data set.

5.2.1 Recall

For determining the *recall* score of the visual features classifier, the IoU of the bounding boxes around the segmented connected components and the bounding boxes of the annotations are used. For each annotation in an image, the segmented connected component with the largest IoU is selected as the candidate bounding box. To prevent the system of generating too many bounding boxes for classification, a min and max size of the connected components are determined. This is done using a grid search approach, where the max size is iteratively lowered and the min size is iteratively increased until the optimal values are determined. Optimal, in this context, means the highest possible recall score while having the least bounding boxes for classification. During grid search it was noticed that the maximum size did not have a large impact on classification performance, so only the minimum area of a bounding box is taken into account. As a result, the min area of retrieved bounding boxes is set at 522 pixels. The corresponding recall scores for the segmentation methods are given in Table 5.4.

Selective search[1] is the best performing segmentation method with a recall score of 0.92. Notably, the

[1]https://github.com/AlpacaDB/selectivesearch

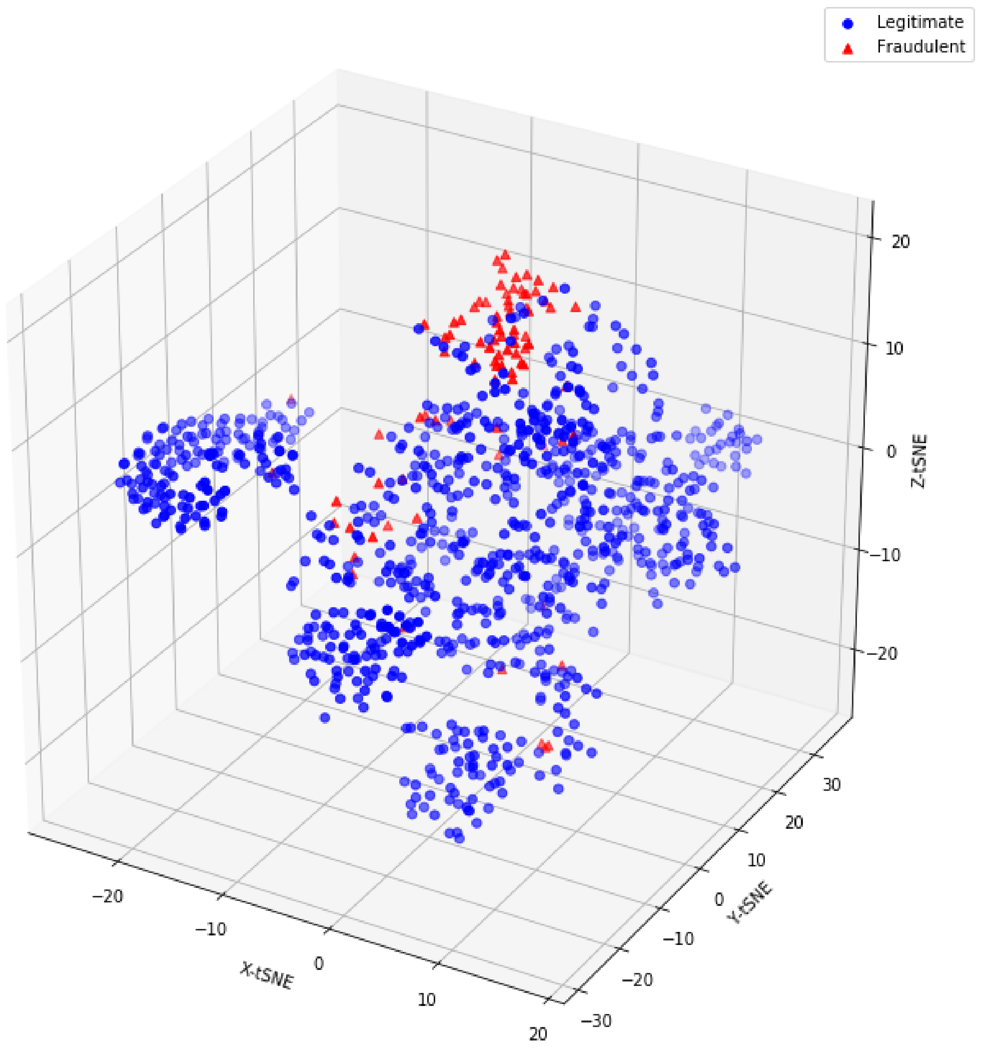


Figure 5.2: t-SNE plot for meta-features

Method	avg #BBs	$IoU \geq 0.5$	$IoU < 0.5$	$IoU = 0.0$	Recall	Time (sec)
Globat Otsu	679	403	147	29	0.69	**0.10**
Local Otsu ($\varnothing = 15$)	547	428	146	15	0.73	2.2
Selective Search ($\sigma = 0.9$)	783	**533**	46	0	**0.92**	3.2
Auto Canny (5×5 dilatation)	**441**	522	48	9	0.90	**0.10**

Table 5.4: Recall score after selecting optimal minimum area for bounding box on 243 online stores

selective search algorithm did not miss any of the annotations completely. However, this high recall score does come at a cost. On average, the selective search algorithms has a segmentation time 30 times longer than global Otsu or auto canny. The auto canny method performed second best. Scoring a recall score of only 0.02 lower than selective search, while generating far fewer bounding boxes and taking only a fraction of the

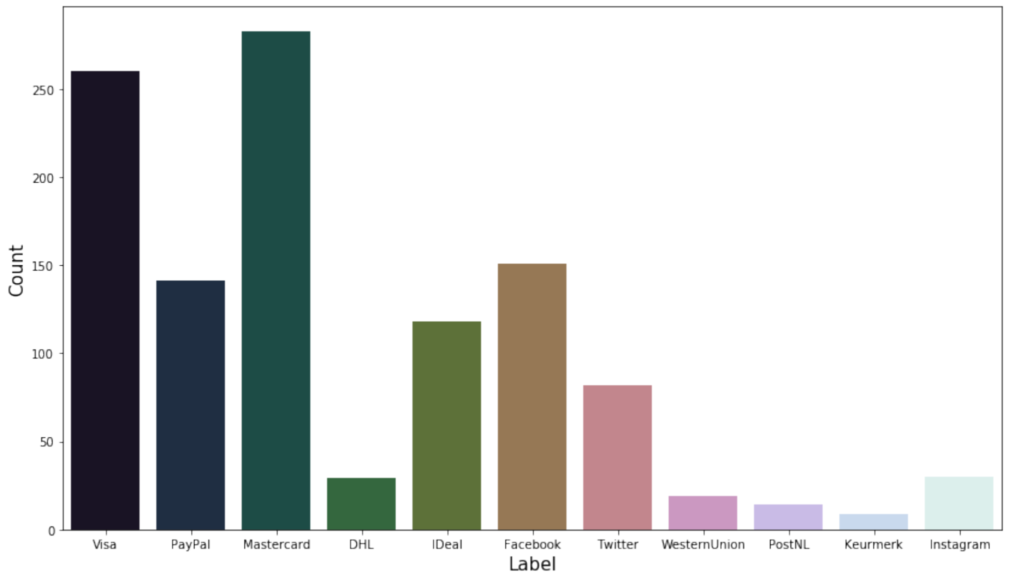


Figure 5.3: Distribution of logo occurence in 1088 annotations

time. Having a low number of bounding boxes is of importance, since classification of each bounding box is computationally expensive. Local Otsu performed worse than selective search. Another disadvantage, is the need to set the radius. Which can have a different optimal value for each image. Lastly, the global otsu method performed worse. Showing that having a single threshold is not enough for complex images such as websites. Auto canny has some distinct advantages over the other methods: it is parameter free, has a high recall score, and generates the lowest average number of bounding boxes. For this reason, the auto canny method will be applied for recall in the object detection pipeline.

5.2.2 Classification

Applying bounding box selection, as described in subsection 5.2.1, decreases the number of bounding boxes significantly. Still, approximately 500 bounding boxes remain to be classified for each image. In the final system, 1.9 million images will need to be parsed with approximately 500 bounding boxes per page. This makes speed, i.e. the time needed to run the image through the network and generating a list of predicted logos, of prime importance. Thankfully, Keras supports a number of pre-trained networks, called *applications*[2]. All the pre-trained networks have been trained on the *ImageNet* data set, only varying in input dimension they require. Classification performance and running speed are shown in Table 5.5. The *ResNet50* scores best on all given metrics, while also having the fastest running time per bounding box. More complex networks, such as DenseNet, have comparable performance, but considerably longer run time. For this reason, the ResNet50 pre-trained network is used for all subsequent image classification results.

For selecting the best performing classifier, the learning algorithms described in section 2 are applied. The results for classifying the annotated image patches are shown in Figure 5.6. The SVM classifier is the best performing classifier with an F1-score of 0.97. The Random Forest classifier is the worst performing classifier with an F1-score of 0.82. KNN and MLP show similar results.

[2]https://keras.io/applications/

network	precision	recall	F1-score	runtime (sec per box)
ResNet50	**0.90**	**0.91**	**0.90**	**0.14**
InceptionV3	0.84	0.85	0.84	0.22
DenseNet121	0.90	0.90	0.89	0.29
DenseNet169	**0.90**	0.90	**0.90**	0.35
DenseNet201	0.79	0.82	0.80	0.43
VGG19	0.88	0.88	0.88	0.23

Table 5.5: Performance for several pre-trained networks in Keras on 978 annotated logos

	Accuracy	Precision	Recall	F1-score
KNN	0.86	0.86	0.85	0.85
SVM	**0.97**	**0.97**	**0.97**	**0.97**
Random Forest	0.88	0.82	0.82	0.82
MLP	0.92	0.90	0.92	0.90

Table 5.6: Results applying classifiers on pretrained CNN filters using 978 annotated logos

The best performing feature generating network, ResNet50, together with the SVM classifier are used to determine per class classification performance on the annotated image patches. Table 5.7 shows the class prediction scores on a total of 715 annotated image patches stored in the development set. Although background is not one of the target classes, it is important to take into account how well the classifier can determine whether a bounding box is background or a logo. All logos, with the exception of *Keurmerk*, show satisfactory results with $F1 \geq 0.85$. The low support for Keurmerk suggests that more data should be gathered in order to get representative results for this class. The classes with more training examples showed high scores in both precision and recall. Of the dominant classes it shows that PayPal has by far the lowest recall score. It should be noted that classification results in Table 5.7 are an upper bound of actual, considering performance is based solely on perfectly extracted image patches.

It does, however, give an indication regarding the discriminative ability of the generated features. To see whether the pretrained network learned a good enough representation of the logos, t-SNE has been applied. The dimensionality has been reduced to a 2-dimensional vector since this represents the data better than 3 dimensions. Furthermore, the background class has been removed, considering it would yield too much noise in the scatter plot. As seen from Figure 5.5, the classes are separated quite well. Especially the classes MasterCard, IDeal, and Instagram. The classes PayPal and Visa show a larger variance. Representative end-to-end performance is presented in section 5.2.3.

From Figure B.1 it is shown that most of the logos are indeed classified correctly, with some outliers. One interesting fact is that a relatively large portion, around 20 percent, of the PayPay logos are being misclassified as background. This is possibly due to the fact with how the background images are sampled. In order to select background, a random sample has been selected from the bounding boxes that had $IoU = 0$ with any of the annotations. A quick inspection shows that most of these bounding boxes are pieces of text. The PayPal logo consists of the letters PayPal written out. This means that the representation of background will be closer to PayPal than to any other target class. Figure 5.4 shows an excerpt from a classified online store. It shows that multiple bounding boxes for the same object are classified, e.g. the letters around the visa logo are bounded separate from the button surrounding the logo. Using non-max suppression, only the classification with the highest confidence is selected.

logo	precision	recall	f1-score	support
DHL	0.88	0.88	0.88	8
Facebook	0.94	1.00	0.97	30
IDeal	1.00	1.00	1.00	15
Instagram	1.00	0.80	0.89	10
Keurmerk	0.50	1.00	0.67	1
Mastercard	0.99	0.98	0.99	106
Background	0.97	0.97	0.97	388
PayPal	0.94	0.87	0.90	53
PostNL	1.00	1.00	1.00	4
Twitter	0.88	1.00	0.93	14
Visa	0.97	0.97	0.97	77
WesternUnion	0.89	0.89	0.89	9
avg / total	0.96	0.96	0.96	715

Table 5.7: Per class classification report on 715 logo annotations in 243 online stores

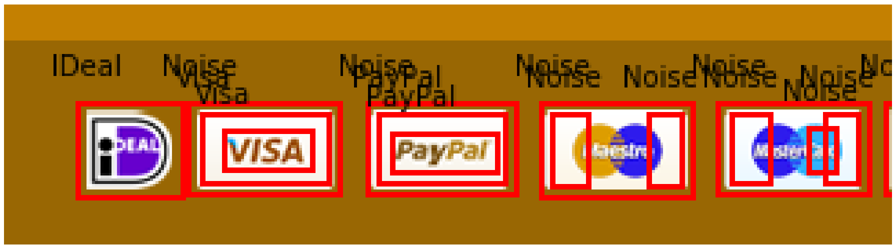


Figure 5.4: Example classification using SVM and ResNet50

Although the logo detection will only serve for detection of logos, it is interesting to look how well a vectorized representation of logos occurence on a page can determine whether an online store is fraudulent or not. For generating the input vector, the list of social media profiles and payment methods are each split into separate features. Each representing one logo class. This resulted in a n-by-11 training data set. Where n is the number of training examples and 11 is the number of logos that are possibly found.

class	precision	recall	f1-score
Legitimate	0.85	0.87	0.86
Fraudulent	0.50	0.46	0.48
avg / total	0.77	0.78	0.78

Table 5.8: Classification using visual features and SVM

Table 5.8 shows the classification result using merely the visual features. It shows that indeed most of the stores that are legitimate are classified as legitimate. The classifier does have quite a harder time classifying fraudulent online stores, with a recall and precision of 0.46 and 0.50 respectively.

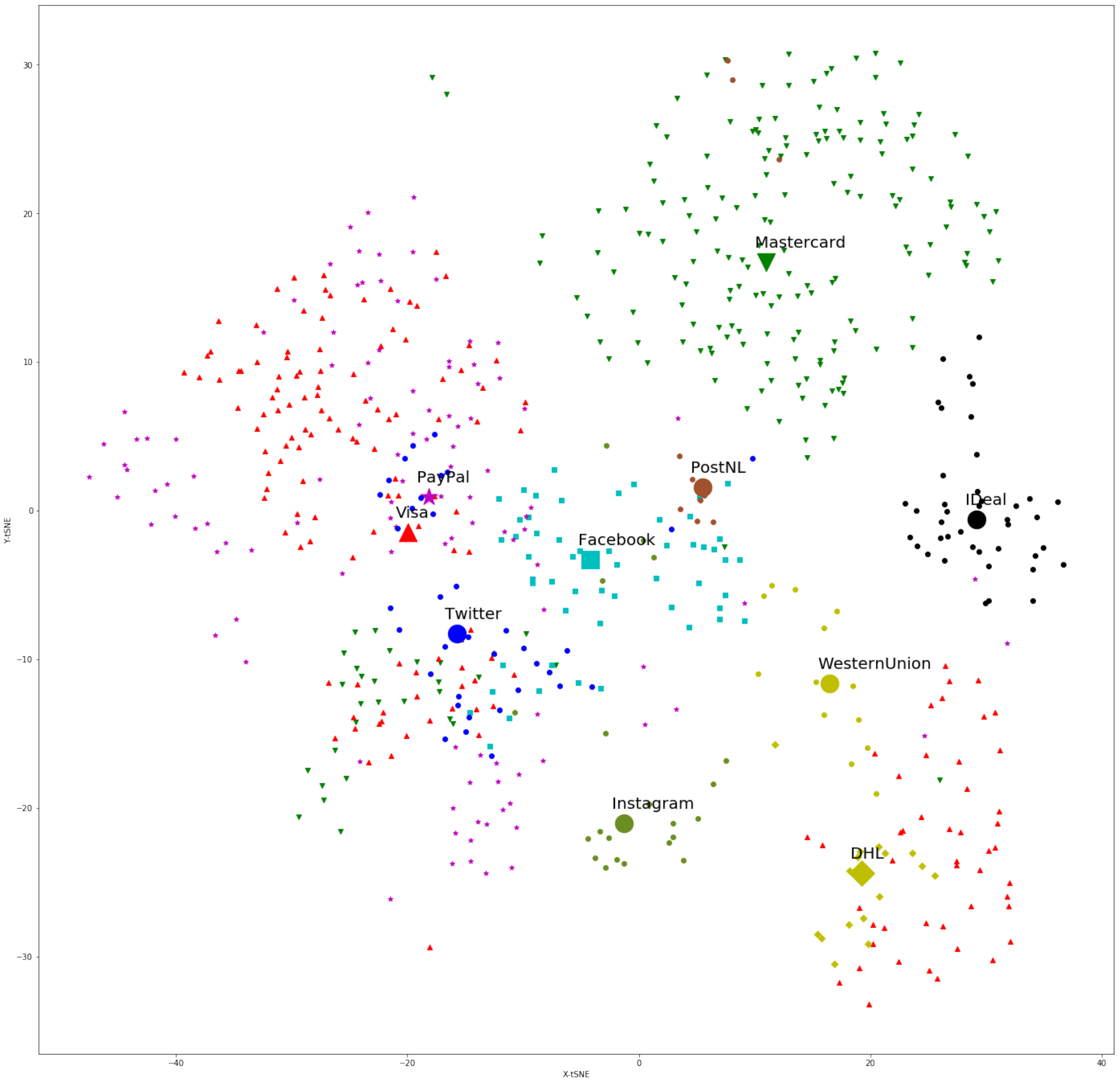


Figure 5.5: Two-dimensional t-SNE of logo representations

5.2.3 Pipeline

In total, 1136 bounding boxes have been classified different than background using canny edge detection. As described in section 4.3.3, performance for the entire pipeline is determined by calculating the mAP of each class using an 11-point interpolation of the precision recall plot. Figure C.1 depicts the precision when recall increases. Precision remains above 0.8 until recall reaches around 0.50. Then, precision quickly drops. Average precision is shown in Table 5.9. The Mastercard logo has the highest precision. Somehow, all the PostNL logos are misclassified. Further investigation is required, but potentially this is caused by the lack of training data. The mean average precision IoU thresholded at 0.50 is 0.36.

logo	precision	# of annotations
DHL	0.38	29
Visa	0.38	260
MasterCard	0.46	283
PayPal	0.44	141
Facebook	0.29	151
Twitter	0.37	82
Instagram	0.40	30
IDeal	0.38	118
Keurmerk	0.39	9
PostNL	0.00	14
WesternUnion	0.47	19
$mAP_{0.50}$	0.36	1136

Table 5.9: mean Average Precision for logo classifcation using Auto Canny & ResNet50

5.3 Combined

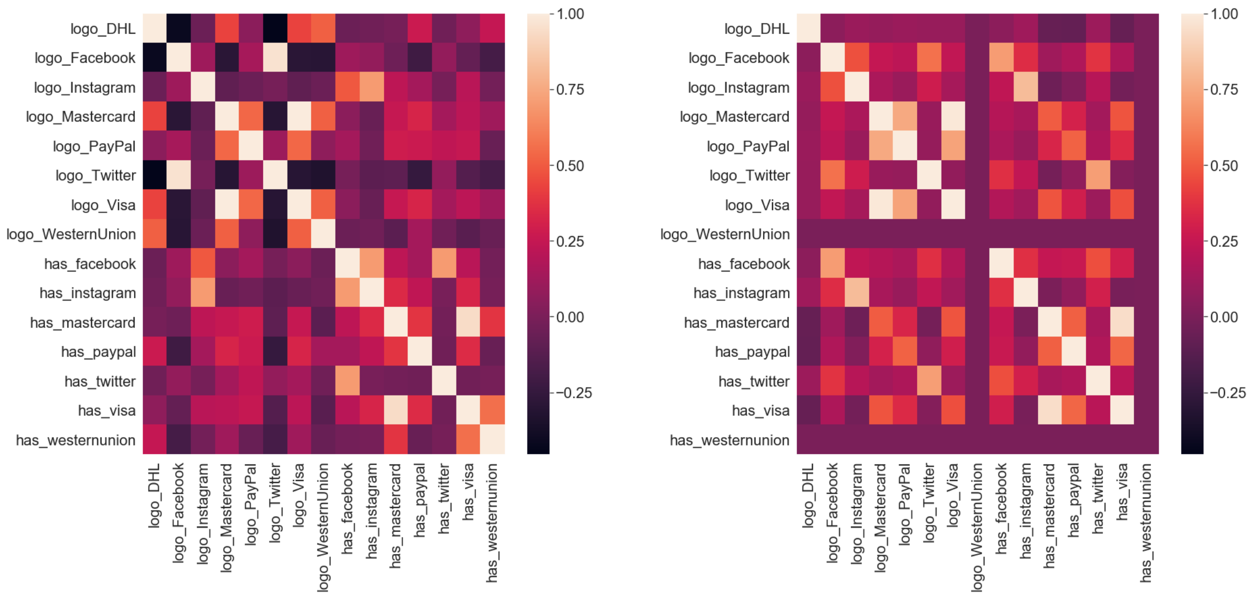
Befor combining the meta and visual features, some pre-processing is required. The social media profiles found in the meta-data are split into a separate feature for each of the given profiles, i.e. the binary variable *has_social_media_profile* is processed into K states. The same process is applied for the payment method. This allows for determining correlation between visual queues, i.e. the logos, and meta-information that is stored in the dataprovider database.

Table 5.10 shows the performance after combining the meta and visual features as described in the previous paragraph. Note, these are based on the annotations and not the actual object detection pipeline. This shows that combining the visual and meta-features indeed has an emphatic positive effect on performance. Especially notable, is the improvement in precision and recall for fraudulent online stores. The scores for the legitimate class remained around the same. Splitting social media platforms and payment methods greatly improve the classifier's ability to discriminate fraudulent cases. Possibly due to mismatching between visual queues and the meta information.

class	precision	recall	f1-score
Legitimate	0.95	0.95	0.95
Fraudulent	0.93	0.93	0.93
avg / total	0.94	0.94	0.94

Table 5.10: Performance after combining meta features and cropped logo annotation

Figure 5.6 shows the Pearson correlation matrix between the logos found on the website (*has_x*) using the cropped annotations, and whether they were found in the dataprovider database. Noticeably, there is no correlation between WesternUnion and any of the features in the legitimate data set. This is caused by the fact that WesternUnion is not found on any of the legitimate online stores. Furthermore, there is a strong correlation between the presence of both the Facebook and Twitter logo in the fraudulent set. Lastly, comparing the two



(a) Fraudulent (b) Legitimate

Figure 5.6: Correlation matrix of the payment methods and social media platforms found on the online stores and dataprovider database

correlation matrices shows that the legitimate set has a larger positive correlation between finding the logo on the page and information on dataprovider.

Chapter 6

Discussion and Conclusion

In the following chapter, the experiments performed are discussed with respect to the problems given in the introduction. Furthermore, there will be a discussion regarding the research questions and some indicators are given for future research.

6.1 Discussion

The results show, that for baseline classification the Random Forest classifier is the best performing classifier overall. A noticeable fact is that the KNN classifier performed significantly worse than the other classifiers, even with optimal hyperparameters and normalized features. This could be due to the method that has been applied for normalizing the data (Mullin & Sukthankar, 2000). Especially the fraudulent class caused an enormous decrease in precision and recall. Plotting the scores in the various baskets showed that the classifier produces a distribution similar to the distribution found using manual annotation. Using domain knowledge, it has been determined that the meta-features classifier works well enough to be pushed into production without logo detection. Trials have been run on 1.9 million online stores and the expected distribution arises. Some outliers were found with a score of 100, while the online stores were clearly fraudulent. Manual inspection and visualization using t-SNE showed that these cases were indeed outliers. One possible explanation is that the website content changed in the time window between data extraction and annotation. Criminals tend to buy unused domain names that, historically, had frequent traffic. Likely, feature generation is based on these old pages, causing the annotation and the retrieved data to be mismatched. In order to prevent this mismatch in the future, time between annotation and data extraction should be minimized. The lower fraudulent class classification performance for the meta-features classifier remains a drawback. This is problematic, since the main focus of the classifier is to determine whether an online store is fraudulent.

The classifier trained on only the visual features, i.e. logo presence on the page, scored a bit less than the meta-features classifier. But overall showed a similar pattern. The legitimate online stores were satisfactory classified, while classifying the fraudulent class remained problematic. The high performance for the legitimate case can probably be accounted to the fact that payment methods like PayPal and IDeal are mostly used by legitimate stores. This is due to the fact that these payment methods, other than for example WesternUnion, can be traced. Conversely, WesternUnion occurred more frequent on fraudulent online stores. Making it a discriminative feature of the fraudulent class.

During classification, the PayPal bounding boxes were often classified as background. A likely cause is sampling for the background class. For generating the background class, a random set without overlap on any of the annotations is selected. This would frequently result in pieces of text being selected as background. Since the PayPal logo is made up with the letters PayPal, the representation of pieces of background could be quite similar to the PayPal logo. Furthermore, there are two distinct clusters for the Visa logo representation. Close inspection from both clusters, show that they prototype a different type of the Visa logo. One logo only has the letters Visa written out, while the other also shows a large blue bar at the top and a large yellow bar at the bottom. Logo classification merely using the annotations gave satisfactory results with all classes except *Keurmerk* having $F1 \geq 0.90$. The dominant class, MasterCard, scored an F1-score of 0.99. This is important, considering in practical applications the classification of MasterCard is of prime importance. The low performance of the infrequent classes can potentially be alleviated by gathering more data.

The average precision and recall of using only visual features is a bit lower than with the meta-features, scoring 0.77 and 0.78 respectively. This results are surprisingly high considering only the presence of logos is taken into account. The t-SNE for the logo classes show that using the ResNet50 pre-trained network yield discriminative feature representations for each of the classes. In the plot two distinct point clouds are found for the PayPal class. During error analysis, we found out that for a large part this is due to the fact that all the PayPal logos have been annotated as one class. There are however two distinct PayPal logos: one with a double P, and one with PayPal written in letters. This is probably one of the main reasons the PayPal class performed worse compared to the other classes. It did not learn a representation from one logo, but rather a representation from two distinct logos. In the future this should be solved by training a separate representation for each type of a logo.

The classifier using the entire pipeline and unseen test data scored significantly lower. This has two reasons: automatic extraction of RoIs yield sub-optimal bounding boxes, and the annotation development set comes from the same set of websites as the train set. The last argument shows that overfitting can occur even if the data is split into train and development. This happens because the parameters are being optimized on the development results, implicitly turning it into training data. To account for this, a separate test set with annotations should be developed. Which is only used during a final test. Having more training data would potentially help prevent overfitting, but annotation of images is laborious work and in the context of this research unfeasible. In theory, we would also like to have a set of online stores that is representative for all the online stores. In practice this would be unfeasible given the unstructured nature of the internet. The decrease in performance caused by the sub-optimal bounding boxes should be investigated in future research. As seen in Figure C.1, the precision quickly decreases when recall increases. Carefully picking the optimal parameters for an algorithm such as selective search greatly increases the recall, but also increases computation time. In future research, an improvement to deal with large sets of high resolution images should be investigated. This includes sharing convolutions such as in Faster R-CNN, or applying some attention mechanism.

Investigating the correlations between the meta and visual features yielded satisfactory results. The performance for the legitimate class remained the same, while greatly improving the precision and recall of the fraudulent case. This is a desired effect, considering the classifier will be used mainly for finding fraudulent online stores. The correlation matrices showed some distinct results. For example, there is a larger positive correlation between finding a social media link on the website and a corresponding button in the legitimate class. This effect is probably caused due to the fact that it would take quite some effort to implement a fake social media profile for each online store. Especially since most fraudulent online stores are short lived. Different usage of payment methods also exists. Where legitimate online stores allow for traceable payment methods such as PayPal or IDeal, fraudulent stores mostly allow untraceable methods such as MasterCard and WesternUnion. Especially the presence of WesternUnion is a good predictor, since it is non-existent in the legitimate examples. In conclusion, the merger of features, potentially, increases the chances of finding fraudulent online stores. Even using simple feature concatenation. Further research into combining the features should be conducted.

My main research question reads: *Can visual and contextual features be effectively combined with machine learning techniques for predicting fraudulent online stores*? Considering the results and the discussion above this does seem to be the case. Overall, the detection rate for the fraudulent class is well over 90 percent. Which would be sufficient to make a production worthy system. *Which meta-features yield the most discriminative power?* During analysis of the meta-features we noticed that presence of social media and basic company information, like phone number and email address, yielded satisfactory discriminative power. *What is the best method for extracting visual information from online stores?* Although more complex segmentation methods are available, this research showed that simple edge detectors such as Auto Canny can show satisfactory results. Using neural networks as feature generators gave discriminative representations that could be classified using a traditional learning algorithm. Although the object detection does not give production worthy results yet this research has shown that meta and visual queues can be combined into a robust fraudulent online store detector. *What combination of visual and contextual features improves baseline performance the most?*. Using the Pearson correlation coefficient some distinct patterns have been shown, e.g. the absence of WesternUnion in the legitimate online stores, a strong correlation between social media profiles and fraudulent online stores, and a stronger positive correlation between payment methods on legitimate stores than on fraudulent stores.

6.2 Conclusion

In this research various methods have been applied to determine whether it is possible to discriminate between fraudulent and legitimate online stores. Research has been conducted regarding finding visual features in large scale images, and how these features can efficiently be applied to the meta-features available. At the start of this research it was expected that it would be quite complicated finding or engineering efficient features for discriminating between stores. Even though it is quite simple for a human to discriminate between legitimate and fraudulent online stores, by looking its general design for example, this poses quite a challenge for the computer. Furthermore, it was expected that finding a large enough set of fraudulent online stores would potentially pose a problem since they are infrequently visited and make up only a small portion of all the online stores. Lastly, detecting the relevant logos from the high resolution screenshots would pose a problem, since the logos make up a small portion of the website. Typically between one and three percent.

One of the primary findings is the classifier's ability to discriminate online stores using only the meta-features available. Features that yield discriminative power include: the presence of a phone number, social media platform, or an address. Using these features already gave an accuracy of 95.8 percent. However, it has to be taken into account that precision and recall scored significantly lower with 20 and 15 percent respectively. This is mainly because most of the online stores are legitimate anyway. During error analysis, it was found that quite some discrepancy exists between the social media profiles and payment images on the front-page, and links found in the dataprovider meta-features. To exploit this, a logo detection system has been devised based on R-CNN methods. Classifying the smallest logos still proves to be a problem. However, the system still had a recall score of 90 percent on the annotations. For a production worthy classifier, this recall score should be higher. Considering the recall score gives and upper bound on how well a classifier can perform. Exploiting the many pre-trained networks alleviated the problem of having a small data-set of logos. Using a different classification layer than the fully connected layers allowed for quick retraining of the model, which allows for fast learning of new logos in the future. There are two main problems with this approach: it loses the end-to-end solution, and a classification for each bounding box is required. The actual pipeline scored over 40 percent lower than the individual parts. This suggests that the algorithm is overfitted on the development set.

Future research should be conducted on using state of the art algorithms that perform end-to-end learning. The Faster R-CNN algorithm has shown promising results (Ren et al., 2015). Due to the scope of his project, this could not be investigated. Rfesearch in classifying small objects while still retaining the primary benefits of using the shared convolutions of Faster R-CNN could be of interest. Since the logos lack variance, methods such as *Single Shot Detectors* should be looked into (Liu et al., 2015). This would alleviate the need for retraining the learning algorithm and could potentially turn the system back into an end-to-end system.

As already pointed out in the introduction, there are quite some applications for the conducted research. First, to our knowledge, there has not been developed a true end to end system for finding fraudulent online stores. Given the enormous rise in e-commerce, this should be of prime importance to, for example, local law enforcement. This would allow for brand protection and quickly target fraudulent online stores. With the application of a basic R-CNN method, it has been shown that in images with little background noise, a simple global binarization method can work quite well for finding Regions of Interest. Furthermore, it has been shown that only looking at the text of a web page, which has been the dominant approach in the past years, is not always the best solution. For web pages that are lacking texts, such as online stores, researchers can greatly benefit by looking at the meta and visual features available.

Given the scope of the research, there are quite some recommendations for future research. One important aspect would be to research whether fraudulent online stores in other demographic areas would yield different scores using the same features. Currently, the meta-features classifier is run and further research will be conducted at dataprovider. Although the online stores lack real text pages, the contact page is an exception. In would be interesting to see whether detecting *translationism* could improve classification performance. Translationism refers to detecting whether a piece of text is machine translated. It was noticed that quite some fraudulent online stores were clearly automatically translated using multilingual machine translation services, such as Google Translate. For retrieving the visual features future research is required. First, although the logos look quite the same since companies want their logo to stand out, a lot of web pages still use various versions of the same logo. Possibly, a more generic representation of the logos would yield better results. Another problem is the time that is required to perform the feature generation for each of the bounding boxes. In future work it is definitely a good idea to look at some of the more advanced R-CNN architectures, even though they perform worse on small object detection (Chen, Liu, Tuzel, & Xiao, 2017). Another problem using the easier selection algorithms is that in some instances the logos would be part of a larger object, resulting in placing a bounding box around the entire object. This results in a misclassification, since there is too much background. Learning a better representation for each logo on a varied background would potentially show more robust results.

Concluding, this research delivered a good proof of concept system for detecting fraudulent online stores. Although the object detection algorithm does not yet yield production worthy results, the meta-features classifier has been taken into production. In the future this system will gradually be improved using the pointers given above, catching thieves along the way.

References

Abak, A. T., Baris, U., & Sankur, B. (1997, Aug). The performance evaluation of thresholding algorithms for optical character recognition. In *Proceedings of the fourth international conference on document analysis and recognition* (Vol. 2, p. 697-700 vol.2).

Aksoy, S., & Haralick, R. M. (2001). Feature normalization and likelihood-based similarity measures for image retrieval. *Pattern Recognition Letters*, *22*, 563–582.

Alpaydin, E. (2010). *Introduction to machine learning* (2nd ed.). The MIT Press.

Azur, M., Stuart, E., Frangakis, C., & Leaf, P. (2011). Multiple imputation by chained equations: What is it and how does it work? *International Journal of Methods in Psychiatric Research*, *20*(1), 40–49. doi: 10.1002/mpr.329

Baltrusaitis, T., Ahuja, C., & Morency, L. (2017). Multimodal machine learning: A survey and taxonomy. *CoRR*, *abs/1705.09406*.

Barnard, J., & Meng, X.-L. (1999). Applications of multiple imputation in medical studies: from aids to nhanes. *Statistical Methods in Medical Research*, *8*(1), 17-36. doi: 10.1177/096228029900800103

Beleites, C., Neugebauer, U., Bocklitz, T., Krafft, C., & Popp, J. (2013). Sample size planning for classification models. *Analytica Chimica Acta*, *760*, 25 - 33. doi: https://doi.org/10.1016/j.aca.2012.11.007

Bengio, Y. (2012, 02 Jul). Deep learning of representations for unsupervised and transfer learning. In I. Guyon, G. Dror, V. Lemaire, G. Taylor, & D. Silver (Eds.), *Proceedings of ICML workshop on unsupervised and transfer learning* (Vol. 27, pp. 17–36). Bellevue, Washington, USA: PMLR.

Bishop, C. M. (2006). *Pattern recognition and machine learning (information science and statistics)*. Berlin, Heidelberg: Springer-Verlag.

Bristow, H., & Lucey, S. (2014). Why do linear SVMs trained on HOG features perform so well? *CoRR*, *abs/1406.2419*.

Canny, J. F. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-8*, 679-698.

Chen, C., Liu, M.-Y., Tuzel, O., & Xiao, J. (2017). R-CNN for small object detection. In S.-H. Lai, V. Lepetit, K. Nishino, & Y. Sato (Eds.), *Computer vision – ACCV 2016* (pp. 214–230). Cham: Springer International Publishing.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)* (Vol. 1, p. 886-893).

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., & Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In E. P. Xing & T. Jebara (Eds.), *Proceedings of the 31st international conference on machine learning* (Vol. 32, pp. 647–655). Bejing, China: PMLR.

Doquire, G., & Verleysen, M. (2012). Feature selection with missing data using mutual information estimators. *Neurocomputing*, *90*, 3 - 11.

Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2015). The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, *111*(1), 98–136.

Genuer, R., Poggi, J.-M., & Tuleau-Malot, C. (2010). Variable selection using random forests. *Pattern Recognition Letters*, *31*(14), 2225 - 2236.

Girshick, R. B. (2015). Fast R-CNN. *CoRR*, *abs/1504.08083*.

Girshick, R. B., Donahue, J., Darrell, T., & Malik, J. (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, *abs/1311.2524*.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Graham, J. W. (2009). Missing data analysis: Making it work in the real world. *Annual Review of Psychology*, *60*(1), 549-576.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning*. New York, NY, USA: Springer New York Inc.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. B. (2017). Mask R-CNN. *CoRR*, *abs/1703.06870*.

Ho, T. (1995). Random decision forests. In *Proceedings of the third international conference on document analysis and recognition (volume 1) - volume 1* (p. 278-282). Washington, DC, USA: IEEE Computer Society.

Hoi, S. C. H., Wu, X., Liu, H., Wu, Y., Wang, H., Xue, H., & Zheng, Q. (2015). Logo-net: Large-scale deep logo detection and brand recognition with deep region-based convolutional networks. *CoRR*, *abs/1511.02462*.

Hosang, J. H., Benenson, R., & Schiele, B. (2017). Learning non-maximum suppression. *CoRR*, *abs/1705.02950*.

Hsu, Chang, & Lin. (2010). *A practical guide to support vector classification.*

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2014). *An introduction to statistical learning: With applications in R*. Springer Publishing Company, Incorporated.

Jyotsna, Chauhan, S., Sharma, E., & Doegar, A. (2016, Sept). Binarization techniques for degraded document images — a review. In *2016 5th international conference on reliability, infocom technologies and optimization (trends and future directions) (ICRITO)* (p. 163-166). doi: 10.1109/ICRITO.2016.7784945

Kaur, P. (2014). Web content classification: A survey. *CoRR*, *abs/1405.0580*.

Kiela, D., Grave, E., Joulin, A., & Mikolov, T. (2018). Efficient large-scale multi-modal classification. *CoRR*, *abs/1802.02892*.

Kim, S., Kavuri, S., & Lee, M. (2013). Deep network with support vector machines. In M. Lee, A. Hirose, Z.-G. Hou, & R. M. Kil (Eds.), *Neural information processing* (pp. 458–465). Springer Berlin Heidelberg.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th international joint conference on artificial intelligence - volume 2* (pp. 1137–1143). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Kriegel, H. (2004). Classification of websites as sets of feature vectors. In *Proceedings of the IASTED DBA international conference* (pp. 127–132).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th international conference on neural information processing systems - volume 1* (pp. 1097–1105). USA: Curran Associates Inc.

Lai, H.-E., Lin, C.-Y., Chen, M.-K., Kang, L.-W., & Yeh, C.-H. (2013). Moving objects detection based on hysteresis thresholding. In J.-S. Pan, C.-N. Yang, & C.-C. Lin (Eds.), *Advances in intelligent systems* *and applications - volume 2* (pp. 289–298). Springer Berlin Heidelberg.

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324. doi: 10.1109/5.726791

Lee, M. C., & Mitra, R. (2016, March). Multiply imputing missing values in data sets with mixed measurement scales using a sequence of generalised linear models. *Comput. Stat. Data Anal.*, *95*(C), 24–38. doi: 10.1016/j.csda.2015.08.004

Liu, Anguelov, Erhan, Szegedy, Reed, Fu, & Berg. (2015). SSD: single shot multibox detector. *CoRR*, *abs/1512.02325*.

Liu, Y., & Gopalakrishnan, V. (2017). An overview and evaluation of recent machine learning imputation methods using cardiac imaging data. *Data*, *2*(1). doi: 10.3390/data2010008

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Proceedings of the seventh ieee international conference on computer vision* (Vol. 2, p. 1150-1157 vol.2). doi: 10.1109/ICCV.1999.790410

Mehmet Sezgin, B. S. (2004). Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, *13*, -13-20. doi: 10.1117/1.1631315

Mullin, M., & Sukthankar, R. (2000). *Complete cross-validation for nearest neighbor classifiers*.

Puneet, & Garg, N. (2013). Binarization techniques used for grey scale images. *International Journal of Computer Applications*, *71*(1), 8-11.

Redmon, J., Divvala, S. K., Girshick, R. B., & Farhadi, A. (2015). You only look once: Unified, real-time object detection. *CoRR*, *abs/1506.02640*.

Ren, He, Girshick, & Sun. (2015). Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, *abs/1506.01497*.

Ren, X., & Ramanan, D. (2013). Histograms of sparse codes for object detection. In *2013 IEEE conference on computer vision and pattern recognition* (p. 3246-3253).

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65–386.

Rumelhart, D. E., & McClelland, J. L. (1987). Learning internal representations by error propagation. In *Parallel distributed processing: Explorations in the microstructure of cognition: Foundations*.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, *abs/1312.6229*.

Sobel, I. (2014). An isotropic 3x3 image gradient operator. *Presentation at Stanford A.I. Project 1968*.

Sun, A., Peng, L., & Keong, N. (2002). Web classification using support vector machine. In *Proceedings of the 4th international workshop on web information and data management* (pp. 96–99).

Szeliski, R. (2010). *Computer vision: Algorithms and applications* (1st ed.). Berlin, Heidelberg: Springer-Verlag.

Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T., & Smeulders, A. W. M. (2013, 01). Selective search for object recognition. *International Journal of Computer Vision*, *104*(2), 154–171.

Viola, P., & Jones, M. (2001, Dec). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001* (Vol. 1, p. I-I).

Walker, D. (1967). Estimation of the probability of an event as a function of several independent variables. *Biometrika*, *54*(1/2), 167–179.

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? *CoRR*, *abs/1411.1792*.

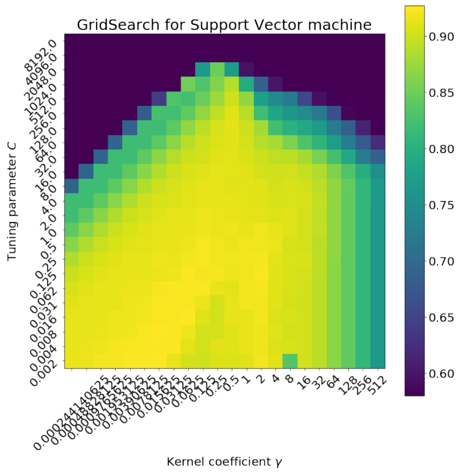
Zhang, Y., Zhu, M., Wang, D., & Feng, S. (2014). Logo detection and recognition based on classification. In F. Li, G. Li, S.-w. Hwang, B. Yao, & Z. Zhang (Eds.), *Web-age information management* (pp. 805–816). Cham: Springer International Publishing.

Zitnick, C. L., & Dollár, P. (2014). Edge boxes: Locating object proposals from edges. In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer vision – ECCV 2014* (pp. 391–405). Cham: Springer International Publishing.

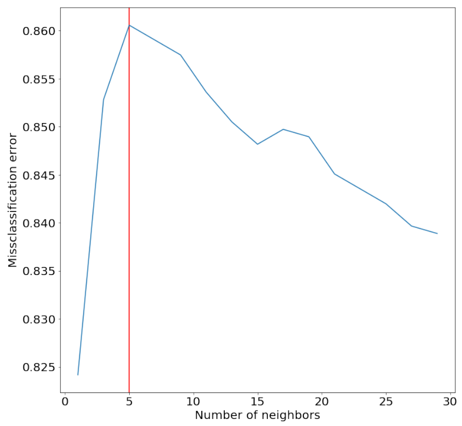
Appendices

Appendix A

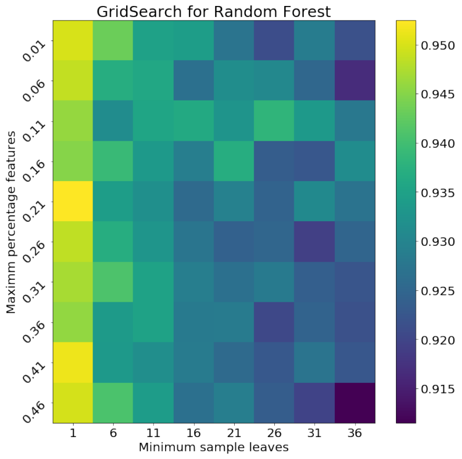
Grid search results



(a) Grid search for SVM



(b) The optimal number of neighbors for KNN classifier

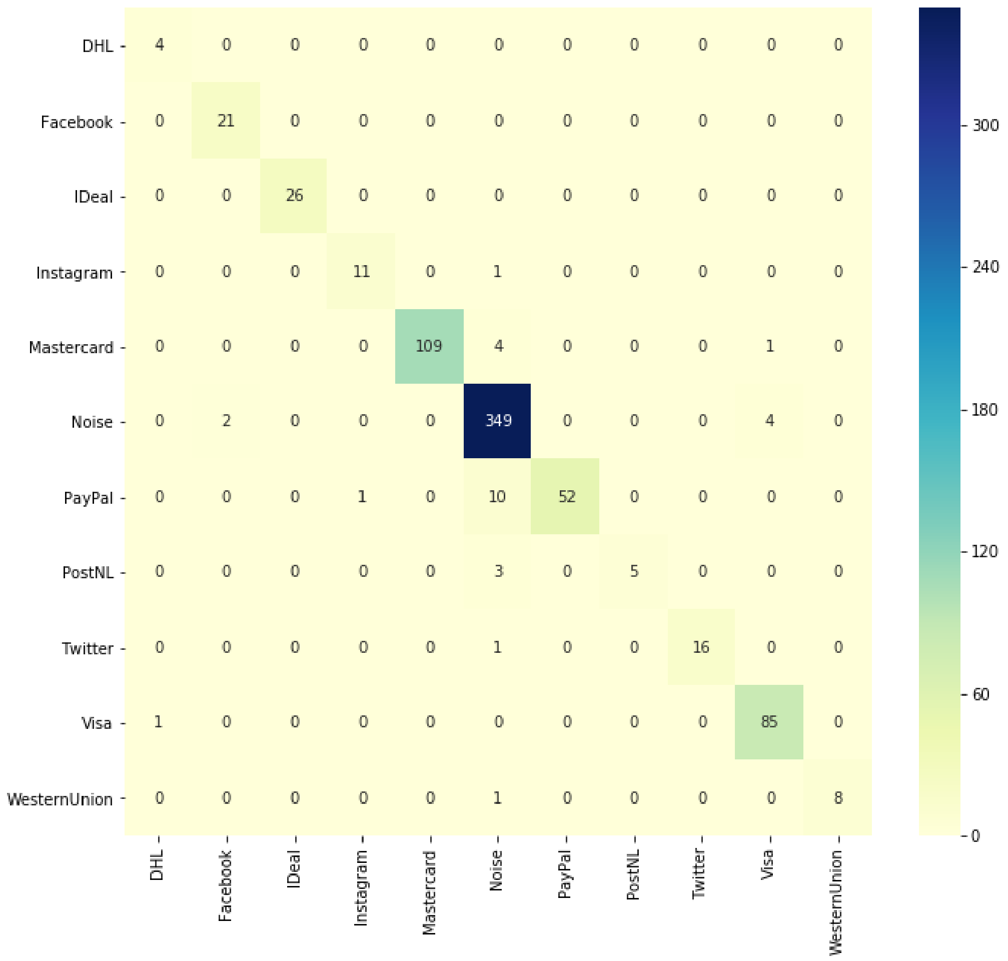


(c) Grid search for Random Forest

Figure A.1: Hyperparameter search for the meta-features classifiers using accuracy metric

Appendix B

Confusion matrix for logo classification



	DHL	Facebook	IDeal	Instagram	Mastercard	Noise	PayPal	PostNL	Twitter	Visa	WesternUnion
DHL	4	0	0	0	0	0	0	0	0	0	0
Facebook	0	21	0	0	0	0	0	0	0	0	0
IDeal	0	0	26	0	0	0	0	0	0	0	0
Instagram	0	0	0	11	0	1	0	0	0	0	0
Mastercard	0	0	0	0	109	4	0	0	0	1	0
Noise	0	2	0	0	0	349	0	0	0	4	0
PayPal	0	0	0	1	0	10	52	0	0	0	0
PostNL	0	0	0	0	0	3	0	5	0	0	0
Twitter	0	0	0	0	0	1	0	0	16	0	0
Visa	1	0	0	0	0	0	0	0	0	85	0
WesternUnion	0	0	0	0	0	1	0	0	0	0	8

Figure B.1: Confusion matrix classification logos on test set

Appendix C

Precision x Recall curve for logo classification

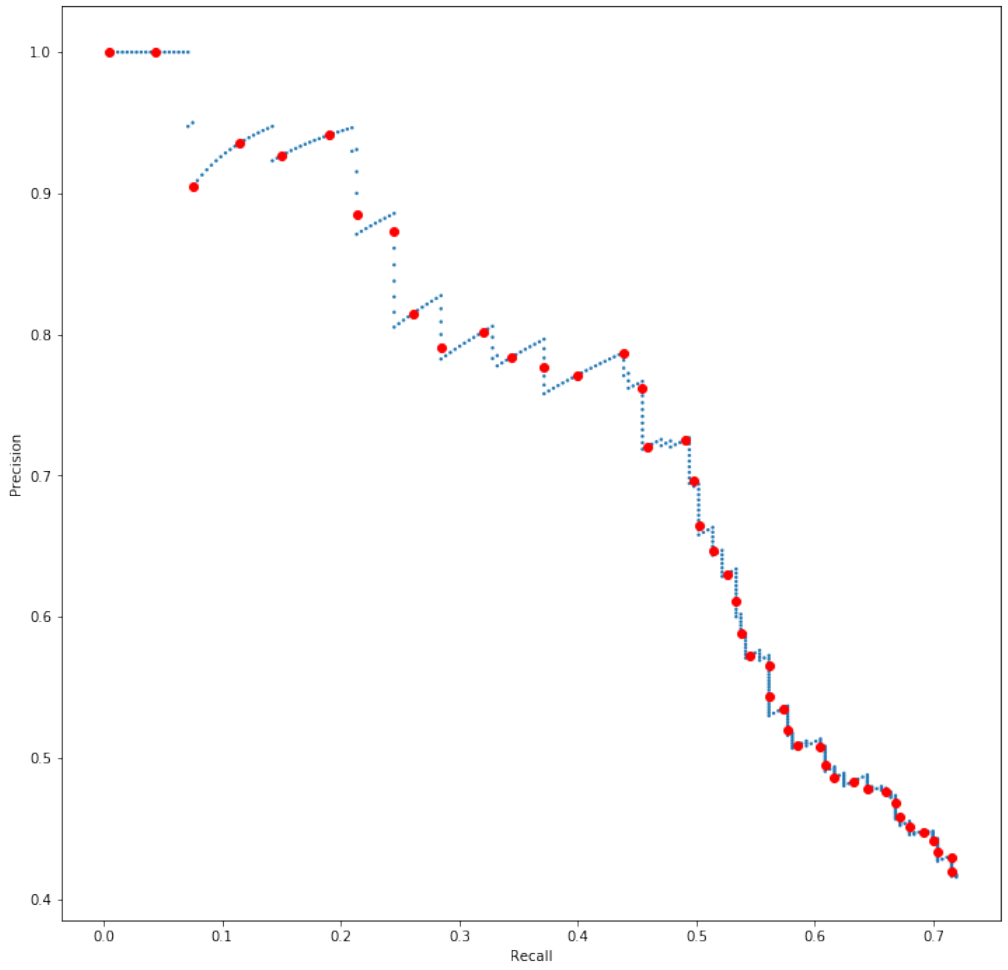


Figure C.1: Precision x Recall curve for logo classification