BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF TELECOMMUNICATIONS

ÚSTAV TELEKOMUNIKACÍ

PROTECTION OF SENSITIVE DATA CONTAINED IN IMAGES

OCHRANA CITLIVÝCH DAT OBSAŽENÝCH V OBRAZE

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR AUTOR PRÁCE	Anzhelika Mezina
SUPERVISOR VEDOUCÍ PRÁCE	doc. Ing. Radim Burget, Ph.D.

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Informační bezpečnost**
Ústav telekomunikací

Studentka: Anzhelika Mezina	***ID:*** 185934
Ročník: 3	***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Ochrana citlivých dat obsažených v obraze

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s problematikou konvolučních neuronových sítí, nástroji Tensorflow, Keras a algoritmem pro detekci objektů v obraze Single Shot Multibox Detection (SSD). Vytvořte trénovací množinu nejméně 100 trénovacích a 30 testovacích snímků, ve kterých se bude vyskytovat klíčové slovo (například PIN, heslo, sarin, či podobně) a pomocí dokumentu XML označte pozici klíčového slova v tomto obrázku (bližší informace poskytne školitel). Natrénujte detektor. Dosažené výsledky přesnosti vhodně prezentujte a výsledky komentujte.

DOPORUČENÁ LITERATURA:

[1] Tutorial, Deep learning for coders, [Online]http://course.fast.ai/

[2] Framework Single Shot Multibox Detection, [Online] https://goo.gl/VFh8Cb

Termín zadání: 5.2.2018 ***Termín odevzdání:*** 29.5.2018

Vedoucí práce: doc. Ing. Radim Burget, Ph.D.
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

UPOZORNĚNÍ:

Autor bakalářské práce nesmí při vytváření bakalářské práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ABSTRACT

This work is focused on application of deep learning in security problem of escape sensitive information, that is contained in images. The presented solution of this problem is using Single Shot Multibox Detector and Fully Connected Network (FCN). FCN is faster than other methods and can be applied in industry, where is a need to analyse input and output information very quickly, for example, in network traffic analysis. In the first part of this work, methods that can be used in keyword detection are described. The second part contains a description of experiment and achieved results for two models of neural network: Single Shot Multibox Detector and Fully Connected Network. The second one gave better results and can be used in practice.

KEYWORDS

convolutional neural network, deep learning, fully connected network, image segmentation, image processing, Keras, keyword detection, object detection, Python, sensitive information, Single Shot Multibox Detector, TensorFlow

ABSTRAKT

Tato bakalářská práce je zaměřena na využití hlubokého učení v bezpečnostním problému úniku citlivých informací ve formě obrazových dat. Pokusem o vyřešení tohoto problému bylo použití Single Shot Multibox Detectoru (SSD) a plně propojené sítě, poslední je mnohem rychlejší než jiné metody a může být použitá v praxi, kde je potřeba velmi rychlé analýzy příchozí a odchozí informace, například analýzy provozu sítě. V první části práce jsou popsané metody, které mohou být použité pro detekci klíčových slov. Druhá část obsahuje popis experimentu a dosažených výsledků pro dva modely neuronových sítí: Single Shot Multibox Detector a plně propojené sítě. Druhý model dosahuje uspokojivých vlastností jak z pohledu času zpracování tak i přesnosti a lze jej použít v praxi.

KLÍČOVÁ SLOVA

citlivá informace, detekce klíčových slov, detekce objektu, hluboké učení, Keras, konvoluční neuronová síť, plně propojená síť, Python, segmentace obrazu, Single Shot Multibox Detector, TensorFlow, zpracování obrazu

MEZINA, Anzhelika. *Protection of sensitive data contained in images*. Brno, 2018, 53 p. Bachelor's Thesis. Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Telecommunications. Advised by doc. Ing. Radim Burget, Ph.D.

Vysázeno pomocí balíčku thesis verze 2.63; http://latex.feec.vutbr.cz

DECLARATION

I declare that I have written the Bachelor's Thesis titled "Protection of sensitive data contained in images" independently, under the guidance of the advisor and using exclusively the technical references and other sources of information cited in the thesis and listed in the comprehensive bibliography at the end of the thesis.

As the author I furthermore declare that, with respect to the creation of this Bachelor's Thesis, I have not infringed any copyright or violated anyone's personal and/or ownership rights. In this context, I am fully aware of the consequences of breaking Regulation § 11 of the Copyright Act No. 121/2000 Coll. of the Czech Republic, as amended, and of any breach of rights related to intellectual property or introduced within amendments to relevant Acts such as the Intellectual Property Act or the Criminal Code, Act No. 40/2009 Coll., Section 2, Head VI, Part 4.

Brno . . . .

author's signature

ACKNOWLEDGEMENT

Ráda bych poděkovala vedoucímu bakalářské práce panu doc. Ing. Radimovi Burgetovi, Ph.D. za odborné vedení, konzultace, věnovaný čas, trpělivost a podnětné návrhy k práci.

Brno . . . .

author's signature

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic
http://www.six.feec.vutbr.cz

ACKNOWLEDGEMENT

Research described in this Bachelor's Thesis has been implemented in the laboratories supported by the SIX project; reg. no. CZ.1.05/2.1.00/03.0072, operational program Výzkum a vývoj pro inovace.

Brno . . . .
author's signature

MINISTERSTVO ŠKOLSTVÍ,
MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

2007–13
OP Výzkum a vývoj
pro inovace

Tato práce vznikla jako součást klíčové aktivity KA6 - Individuální výuka a zapojení studentů bakalářských a magisterských studijních programů do výzkumu v rámci projektu OP VVV Vytvoření double-degree doktorského studijního programu Elektronika a informační technologie a vytvoření doktorského studijního programu Informační bezpečnost, reg. č. CZ.02.2.69/0.0/0.0/16_018/0002575.

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ,
MLÁDEŽE A TĚLOVÝCHOVY

Projekt je spolufinancován Evropskou unií.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

In recent years, information technologies have been developed extremely fast and the most of information can be found in a digital form. Of course, for the big companies applying the new technologies, this is an advantage, because they can process information in very short time.

But on the another side, it's getting more difficult to control information in the digital form since people actively use cloud services and try to push everything into public network. That's why, for the security reasons, monitoring of documents' content can be critical in order to protect confidential information from being leaked and many companies need to invest into protection of their data.

This work focuses on detection of the keywords in the images, since people actively use scanned documents. Traditional way to find keywords in the image is to get the input image, using Optical Character Recognition (OCR), perform text detection and recognition, use spelling checker to correct mistakes and, finally, define if the document contains a keyword or not. But with this approach some problems can occur. First of all, described technique is relatively time-consuming, it isn't suitable for monitoring network of a company, because a lot of data goes through it. Second problem, in the case when the text contains images, tables, blocks (so called sparse text documents), detection of text line in OCR can fail, or will not be accurate, which will further lead to bad results of text recognition. Consequently, it will fail also to detect the monitored keywords.

In recent years there were a lot of attempts to improve text detection, but it can be summed up into 3 types of methods:

1. texture based methods (treat texts as a special type of texture and make use of their textural properties),
2. component based methods (first extract candidate components through a variety of ways, and then filter out non-text components using manually designed rules or automatically trained classifiers),
3. hybrid methods (a combination of texture based methods and component based methods, which make use of the advantages of these two types of methods) [1].

The thesis is divided into two main parts: theoretical and practical. In the theoretical part of this work, there are two chapters. The first chapter is focused on "deep learning". In this chapter, the term deep learning is defined, explained and possible applications are shown. In the last part of this chapter, convolutional neural network and its architecture is explained. The convolutional neural network has been used in my experiment, shown in chapters 3 and 4.

The topic of the second chapter is keyword detection. The chapter opens with a description of methods that can be used in keyword detection, such as Optical Character Recognition, Object Detection and Image Segmentation. Moreover, in the sections "Object Detection" and "Image Segmentation", methods that are built on neural networks are shown.

The second part of this work is practical, where I describe my experiment and its results. It consists of two chapters: "Experiment description and application design" and "Results and discussion".

In the third chapter, there's the description of experiment using two methods: Object Detection (model Single Shot Multibox Detector) and Image Segmentation (Fully Connected Network). For both methods, neural network is initially trained with training data (dataset of images), after that I performed several runs of neural networks on testing dataset.

The results of text detection of my neural networks are depicted and compared with traditional methods in the fourth chapter. According to the results, the main contribution of this thesis is a novel approach, which doesn't use OCR at all and utilizes Convolutional Neural Networks in order to detect keywords in the scanned documents. It significantly increases accuracy of detections in complex, low quality documents.

1 DEEP LEARNING

Recent years it's getting more difficult to process information, since volume of it becomes bigger in time. The normal person or classical algorithms can't process all data, because it's time and source consuming. That's why people need to find some new approach, that analyses part of existing dataset, is able to "learn" and automatically performs particular actions. For example predicts results and detects objects in images or words in speech.

Over the past few years, "deep learning" became a buzzword. Especially, when a topic of discussion is big data, artificial intelligence and analytic. Deep learning produces great results, when it's applied to development systems, that should be self-teaching and autonomous and can be used in many industries [2].

1.1 Description

Deep learning is a part of machine learning. Machine learning is a type of artificial intelligence that allows software applications to become more accurate in predicting outcomes without being explicitly programmed. In general, the aim of machine learning is to receive input data and, using statistical analysis, predict output value within an acceptable range [3]. Turning to the definition of deep learning, it's part of machine learning, it can discover latent structures within unlabeled, unstructured data, using non-linear transformations. The difference between machine learning and deep learning is that deep learning focuses more on tools and techniques and applies them to solve tasks where's necessary to think as a human.

Consequently, the main idea of deep learning is to develop networks and tools, which can effectively work with a huge dataset, for example, Google's image library or Twitter's database of tweets. That's why it requires large amount of labeled data to train and computing power to rapidly process data. Nowadays, it's easy and relatively cheap to get computer with required hardware, specifically performing graphics card with parallel architecture. Graphics card with parallel processing of threads is efficient for deep learning and on this type of hardware, the model can be trained relatively fast. In addition, deep neural network can be fed with any form of data: audio, video, speech, text, machine signals. The ability to process various types of data makes it more flexible and usable as conventional approach.

1.2 Application

Thanks to capability of deep neural network to take any kind of data and to work with a large dataset, it is applied in many spheres of industry and a lot of companies realize that they need this kind of system for processing their data.

Right now, deep learning is actively used in systems, that work with images and video, for example, classification of images, restoring colours in grayscale images and videos, real-time analysis of behaviour or face recognition. For sure, for human it will take more time to analyse images or videos and make any conclusions basing on their results. On the other hand, already trained network can do it faster and sometimes better than a human. Also, some of the most popular tasks for deep learning are recognition and generation of handwriting texts. But it requires a lot of handwriting texts for training a network before the network will be able to give reasonable results.

Self-driving cars is another interesting field of application of deep neural network. Self-driving cars have been significantly improved in last few years. Tesla, the most famous company in this area, is actively testing its cars. Obviously, it would be impossible to program a self-driving car with every possible scenario it might encounter. That's why this system requires something, that can predict and decide how to behave in each situation. Deep learning can deal with the most part of these problems, but it still needs some improvements. However, people are now close to resolve them [4].

Neural network takes also a part in cyber security. Since it's very difficult for normal person to analyse network traffic non-stop or go through endless files with logs, this task should be done by some kind of classification algorithm. The classification algorithm should be relative fast to prevent any kind of attack. For example, there is a neural network for malware classification [5] or phishing URLs classification [6]. Moreover, in recent time information security is the most actual problem, because methods of attacks are extremely fast improved, and the speed of distribution of malware is also accelerating. That's why conventional methods of protection relatively stay behind because it's impossible to react immediately on each new virus or method of attack. That's the reason, why people need system, that can be self-teaching and can detect new kind of danger.

1.3 Convolutional Neural Network

1.3.1 Description

One of the most used type of deep neural network is Convolutional Neural Network (CNN). Nowadays this kind of neural network is actively used in computer vision tasks, for example, image classification.

This term may sound like a weird combination of biology and mathematics. Turning to history of CNN, the idea of this network is really based on knowledge of neuroscience and biology. CNN takes the inspiration from the visual cortex – the part of a brain, that processes visual information. The visual cortex has small regions of cells, which are sensitive to specific regions of the visual field. The most important principle is, that neuronal cells are fired only in the presence of edges or certain orientation. For example, some regions are fired, when we see vertical edges or some of them are fired, when there are horizontal edges. Neurons are arranged into columns, which are producing visual perception. All of these principles are employed in CNNs.

Turning to the nature, for a human this task of recognition is obviously one of the first skills learned from being born and one that comes naturally. That's why a human can identify environment and objects surrounding him without thinking. When he sees something, he can characterize it and give the name to the objects immediately. We can't share skills such as quite fast recognizing objects, adapting to different environments with our machines effectively [7].

When processing data with computer, everything is different: when it gets an image, it's represented as an array of pixel values, which has height, width, depth. The issue for computer is to recognize, what is in the picture or, at least, to make a prediction, what can be in the image.

The first innovation in this field made Alex Krizhevsky in 2012. He used neural network in the image classification and dropped classification error significantly from 26% to 15%. At that time it was a real breakthrough. Right now this technique is used by apps like Facebook, Google, Pinterest and others.

As it was mentioned above, convolutional neural network is used for the tasks related with images. This convolutional neural network is based on the features of a human brain. The key idea in CNN is the alternation of convolution layers and pooling layers. The structure of network is multilayer. For teaching of neural network back-propagation or backward propagation of errors is often used. The "backwards" part of the name stems from the fact that calculation of the gradient proceeds backwards through the network, with the gradient of the final layer of weights being calculated first and the gradient of the first layer of weights is being calculated last. Partial computations of the gradient from one layer are reused in the computation of the gradient for the previous layer [8]. But such name this architecture got is due to convolution operation being used. This operation will be described in detail in section 1.3.2.

The function of convolutional neural network is always interpreted as a transition from specific features to abstract details and after that to more abstract until there will be selected high-level concepts. The important thing is, that network is setted up with itself and produces necessary hierarchy of abstract features, filtering unimportant details and keeping significant ones. On the other side, when network is big enough, it's difficult to understand how all features are produced, and it's not recommended to understand them or fix somehow. It's better to improve the architecture of network or its structure. If the system can't give good results, it means, that probably there's not enough training data or structure has some limitations.

1.3.2 Architecture

CNN is made up of neurons, which has weights and biases, but unlike regular neural network, neurons are arranged in three dimensions: width, height, depth (in the case of image input data). Convolutional neural network has input and output layers and hidden layers: convolutional, pooling and fully connected. The CNN's architecture is shown in Fig. 1.1.

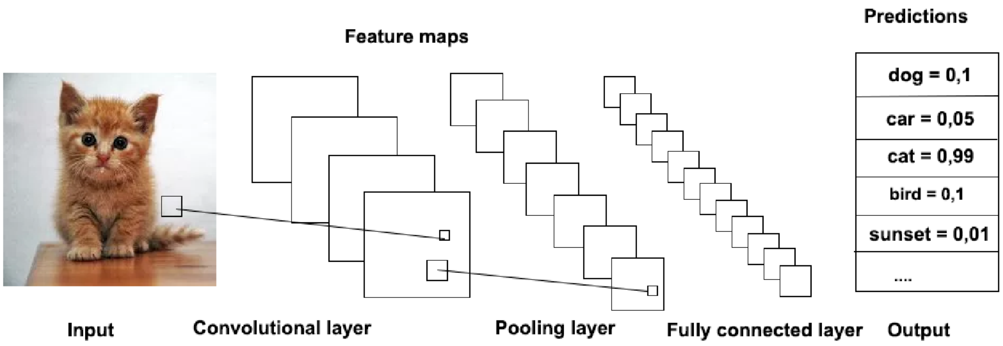


Fig. 1.1: Architecture of CNN.

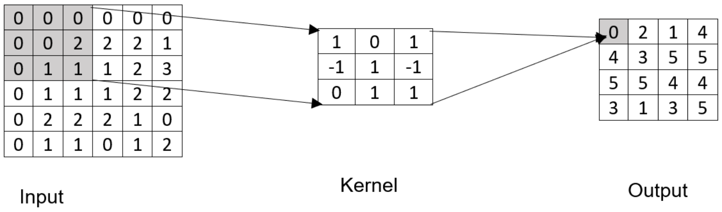
The network receives input data, transforms it while passing hidden layers and from the output layer the result can be read. The result can differ depending on the task e.g. class of the object in the image, boolean value etc.

Input layer

This layer contains the input data, for example raw pixel values of image (width, height, depth).

Convolutional layer

Usually this is the first hidden layer. The purpose of this layer is to extract features from input data. For example, let's consider, that input data is a picture, as it's the main target of this kind of the neural network. For easier understanding of a convolutional layer, let's imagine that there's a flashlight, that goes step-by-step over the image and every step is just one pixel. In this way, flashlight will go through all the image. Turning to technical terms, this flashlight is called filter or kernel. It's an array of numbers (weights), size of it is smaller than that of the input image. The "shined" area in the image is called the receptive field. The depth of filter is the same as input image. The operation, that is provided for each receptive field, is multiplying the values in the filter with original pixel values. After that they are all summed up and written as a new pixel value. For example, if there is a filter of size $5 \times 5 \times 3$, 75 multiplications will be performed and after summing up, just one value is written. When this procedure is applied on each receptive field, the output of this layer will be an activation map. It's important to note, that filters act as feature detectors from the original input image [7]. The operation of convolution is shown in Fig. 1.2. There's an array selected with shadow colour of size 3×3. Each value of pixel will be multiplied with the value of kernel array on the same position as follows: $0 \cdot 1 + 0 \cdot 0 + 0 \cdot 1 + 0 \cdot (-1) + 0 \cdot 1 + 2 \cdot -1 + 0 \cdot 0 + 1 \cdot 1 + 1 \cdot 1 = 0$ and this result will be written to the output array.



0	0	0	0	0	0
0	0	2	2	2	1
0	1	1	1	2	3
0	1	1	1	2	2
0	2	2	2	1	0
0	1	1	0	1	2

1	0	1
-1	1	-1
0	1	1

0	2	1	4
4	3	5	5
5	5	4	4
3	1	3	5

Fig. 1.2: The convolutional operation.

ReLU layer

ReLU is the abbreviation of Rectified Linear Units. This layer is always used after the convolutional layer. The purpose of this layer is to tell the network whether the information received by a neuron is relevant or not. This can be achieved using activation function. There are two types of activation function: linear and non-linear. Activation functions make the back-propagation possible since the gradients are supplied along with the error to update the weights and biases [9]. Without a differentiable non-linear function, this would not be possible.

The linear function $y = ax$ has the main disadvantage: the derivation of it is a constant, it means, that gradient has no dependence on x. If there is an error in prediction, the changes made by back-propagation is a constant and doesn't depend on the changes in input which are labeled as Δx. It means, that if all layers are linear, the final activation function is the same as a linear function of the input layer. That's the reason to use non-linear function instead [10].

The most popular and widely used non-linear functions are sigmoid, tanh, and ReLU functions.

Sigmoid function. The function has the following mathematical formula:

$$f(x) = \frac{1}{1 + \mathrm{e}^{-x}}, \tag{1.1}$$

The plot of this function is depicted in Fig. 1.3.

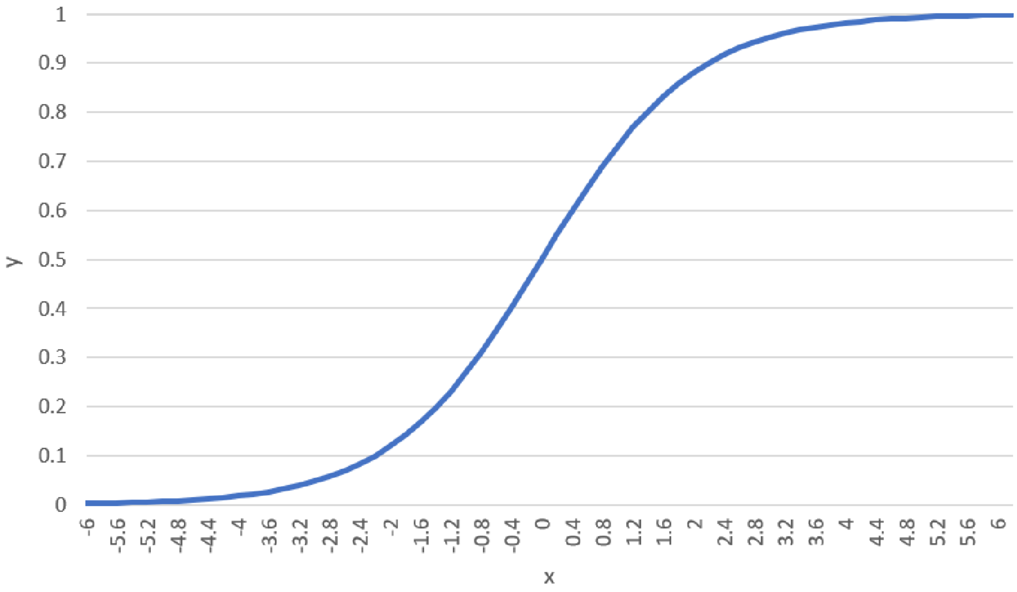


Fig. 1.3: Sigmoid function.

The advantage of this function is its non-linearity, which means, that derivation of this function will be different depending on x. It also has range between 0 and 1 which is a good advantage and doesn't need further normalization. If it's required to have a value of probability as an output, this activation function is a good choice. But on the other side, with growing x the number of changes in y value decreases, which can lead into small gradient changes. As it has been mentioned, during calculations of backward propagation the final gradient depends on the current gradient, consequently, if the changes are small, the network will be trained very slowly or will not be trained at all. Anyway, there are ways to work around this problem and this activation function is actively used in practice.

Tanh. The tanh function is very similar to the sigmoid function. The plot is shown in Fig. 1.4 The mathematical formula is as follows:

$$f(x) = \frac{2}{1 + \mathrm{e}^{-2x}} - 1, \tag{1.2}$$

The main difference in comparison with the sigmoid function is that its output ranges from -1 to 1, but all other properties are the same. The tanh function has the problem with "vanishing gradient" too. The choice between sigmoid and tanh function depends on the problem that needs to be solved.

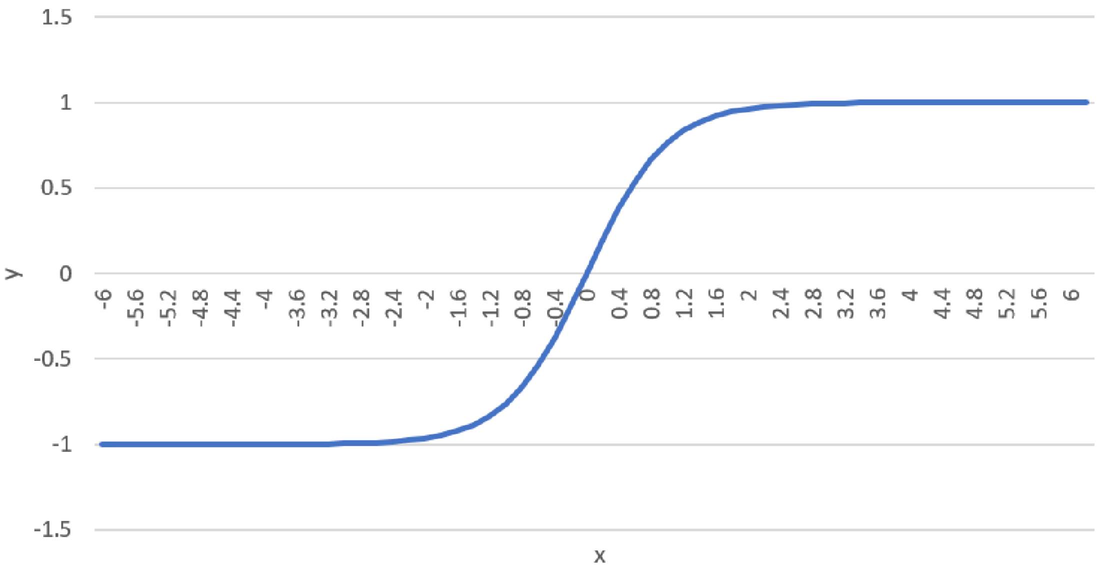


Fig. 1.4: Tanh function.

ReLU. ReLU is the most used activation function. It is defined as $f(x) = max(0, x)$. The plot of this function is shown in Fig. 1.5. This function seems to be linear, but it's not. That's why there's no problem with back-propagation errors. The derivation of ReLU can be 1 (if $x > 0$) or 0 otherwise. It means that gradient will be either zero or remains the same value. In sigmoid and tanh functions it could be higher or lower, or the sign can be changed, depending on value of activation function. In ReLU this problem is solved. Moreover, thanks to converting the negative inputs and zero, it is prevented from activating all neurons. The advantage of this is that getting the result is easy for computation. On the other hand, it can lead to "dying ReLU" problem, that means, some neurons will never be activated for all inputs [9].

Pooling layer

The usual approach is to put pooling layer into convolutional neural network periodically. This technique reduces the number of parameters and computation in network, which leads into reduction of the size of image. It works as follows: group of pixels is reduced to the single pixel using non-linear transformation, usually the Max function. The transformation takes just disjoint rectangles, which will be compressed into one pixel each, depending on pixel's maximum value. The aim of pooling layer is that if there are any features detected before the layer, it won't be necessary to

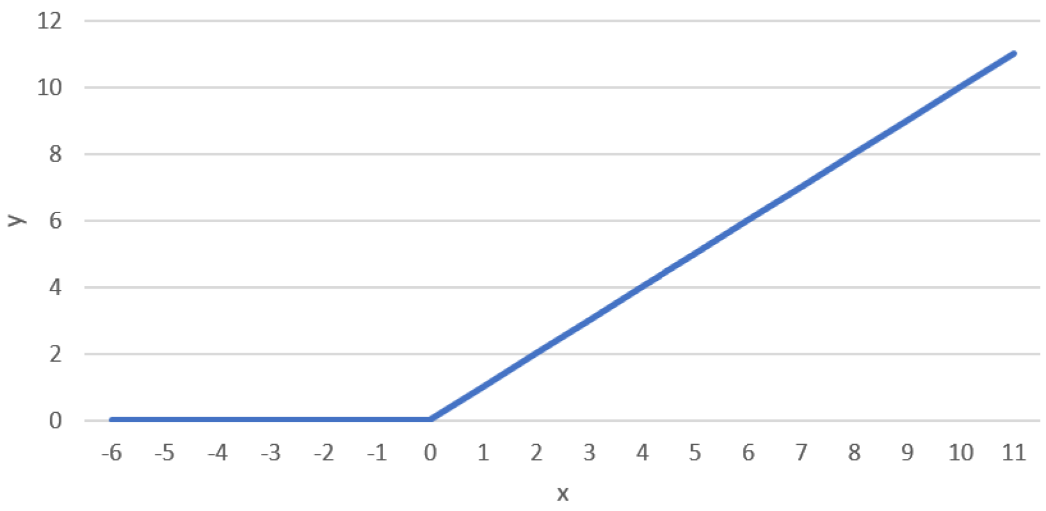


Fig. 1.5: ReLU function.

keep so detailed information from the input image. Saving just parts of the image will suffice.

Fully connected layer

The last layer in this architecture is a fully connected layer. After the input image passed through all convolutional and pooling layers, the output represents high-level features. The fully connected layer is fed by those features and N-dimensional vector, where N is a number of classes, from which algorithm will choose. The fully connected layer usually uses a softmax function. The aim of this function is to "squash" the values of output vector to be between 0 and 1. But it also divides each output so that the total sum of the outputs is equal to 1 [11]. As the value is higher, the probability of this correct class is also higher. The softmax function can be written as follows:

$$\sigma(z)_j = \frac{\mathrm{e}^{z_j}}{\sum\limits_{k=1}^{K} \mathrm{e}^{z_k}}, \tag{1.3}$$

where

- z – vector of the inputs,
- j – index of output units,
- K – number of the output values.

The softmax function can be used for any number of classes. That’s why it’s useful in an object recognition, where hundreds of classes can occur.

1.3.3 Application

As it was mentioned above, CNNs are mainly used in computer vision. Thanks to the architecture of this network, "instead of feeding each into the neural network as one grid of numbers, the image is broken down into overlapping image tiles that each fed into a small neural network" [12]. It makes it easier to work with images.

Image classification. One of the most usual tasks for CNN is image classification. This is about "assiging an input image one label from a fixed set of categories" [13]. The process of image classification should overcome following issues:

- **Viewpoint orientation.** One object on the image can be rotated, that's why it's complicated to detect the object.
- **Deformation.** Logically, nothing can't stay in the same position, and bodies can be deformed in an extreme way.
- **Background.** It's easy to loss object in background clutter, for example, sleeping white cat on the white blanket.
- **Just a part of an object is visible.** In the situation, when we can't fully see the object, but only part of it. For a human it's not a problem to recognize or guess, what it can be, but for a computer this is a big problem. That's why all of these issues make image classification harder.

Face recognition. The interesting application of convolutional neural network is a face recognition. Of course, it requires a large dataset of different people, anyway, people always upload a mass of various pictures to public network, that's why it's possible to get such dataset. Face recognition is actively used in security systems, such as verifying payments or getting access to some devices. Apple Inc., Samsung Electronics Co. and Xiaomi Corp. install biometric features on their smartphones. Facebook Inc. has developed a technology that can identify people even when their faces aren't clearly shown in pictures [14]. Face recognition can be used in a forensic science for identification of criminals. In the USA and Great Britain, there is a system developed and tested, that allows to find criminals using recordings from cameras.

Speech recognition. A few years ago, Deep Neural Networks (DNNs) were used for speech recognition, but after research of the Microsoft Corporation, which indicated, that CNN gave better results than DNN for different tasks including speech recognition, CNN became to be widely used. Nowadays speech recognition is used in live subtitling of television broadcast, also in applications, which allows user to input voice commands. For example, Apple's application Siri uses speech recognition to receive commands from phone's owner. Another interesting thing, that voice can be used for, is authentication. This application is patented and the patent belongs to Apple [15].

Scene labeling. Scene labeling is the task of fully labeling an image pixel-by-pixel with the class of the object each pixel belongs to. This task is very challenging, as it incorporates resolution of detection, segmentation and multi-label recognition problems [16].

Document analysis. This field of computer vision works with handwriting texts. This is one of the typical tasks for CNN. People actively use it with scanned documents and of course it's wide-spread in applications for phone or PC. But to achieve a good results from this system, having just CNN is not enough and using other tools is required [17].

2 KEYWORD DETECTION

Keyword detection is a temporary field of research. It's getting more actual, since the amount of information and documents increases very fast. There's a need to analyse all of it quickly and in reasonable time. Extracting keywords from texts is one of approaches that help to achieve these goals, because there's no need in reading full texts. With this approach it's enough to find the keywords and based on them understand the main idea of the text and the kind of text we are dealing with. Nowadays two main ways exist how to detect text in the image: using Optical Character Recognition and using Deep learning. In this work I focus on keywords detection that uses Deep Learning instead of Optical Character Recognition.

2.1 Optical Character Recognition

"Optical Character Recognition (OCR) is a technology that is capable of converting different types of documents, such as scanned paper documents, PDF files or images captured by a digital camera into editable and searchable data" [18].

Nowadays, the most of applications use algorithms, which consist of the following steps:

- pre-process the image (clear the noise, remove lines etc.),
- detection of text lines and words,
- recognition of characters.

OCR systems usually use a dictionary to improve the quality of recognition. But these systems have some disadvantages. Firstly, OCR can't work good enough with handwritten documents and with images, that has poor quality, consequently, there's a problem with the accuracy. Secondly, they have a problem to differentiate such characters as "O" (letter) and "0" (zero). Thirdly, a human should check the output of OCR systems for mistakes, because, as it was stated above, the OCR output can contain errors and isn't 100% accurate. Finally, it's relatively time consuming and takes a lot of time to perform a recognition of big amount of images containing text. That's why it can't be applied in fields as analysing data in a network, because the speed is crucial in network data analysis.

However, the problem of OCR is a challenging one nowadays and last years there are attempts to use artificial intelligence for text recognition.

2.2 Keyword detection in security

The most wide-spread application of keyword detection in security is spam and phishing detection. Phishing is a method which is being used to obtain sensitive information from users. Spam is the use of messaging systems for sending unwanted messages e.g. emails. The most of spam messages contain useless information such as advertisement, propaganda etc. The one of the main functions of email services is to filter spam messages, which always contains external links, leading to pages with viruses. That's why Google always improves its systems and filters using deep learning to protect users from endless phishing messages and spam.

Keyword detection can be also applied in detection of attack-specific keywords in network traffic using existing "database" of signatures of known attacks. Obviously, this method has a disadvantage: it can't detect new attacks, which are not in a database of signatures. That's why such systems need machine learning to be able to learn about new kind of attacks and detect them. But even with machine learning, it's impossible to react on something new immediately.

Also, there's another issue with monitoring a network. There can be a situation, when someone connects illegally into private network and tries to download, for example, scanned documents with sensitive and private data. It won't be easy for computer to detect such intrusion. If some algorithm is able to analyse images passing through network, it can detect private documents containing images. For such analysis we need a fast detector of keywords in images since speed is crucial in network data processing. That's why artificial intelligence can be used as a solution of this problem.

Moreover, keyword detection is also getting more actual since terrorism-related incidents happen more often than before. Right now, social media take that fact in account and must control messages and information, that goes through their services. Obviously, it's impossible to process all traffic, that's why there is a need of system, that will trigger an "alarm" when some keywords appear.

In this work, two methods for detection keywords were used: Object Detection and Image Segmentation.

2.3 Object Detection

Object detection is a problem of finding and classifying objects in the image. Images can contain unpredictable number of objects. Nowadays, the object detection is also actual problem in computer vision field, because there are still some problems waiting for solution, for example, variable number of objects (the model needs fixed-size number of classes, which should be detected). Another issue is a different size of objects. Some objects can be really small and for this case a relatively slow sliding window with different sizes is used.

2.3.1 Older methods of object detection

There's no doubt, that researchers tried to develop methods, which are able to detect objects in the image. During last 10–15 years a lot of approaches were tested. The most of them use sliding window and try to extract features suitable for classification.

Since deep learning became very popular 5 years ago, researchers tried to apply it in a computer vision including detection of objects. Nowadays exist a bunch of new methods based on neural networks.

The first attempt to perform detection of objects with neural network was in 2013. The developers proposed method with multi-scale sliding window using CNN called OverFeat.

In 2014 a new way for object detection called Region Convolutional Neural Network (R-CNN) was introduced. The key idea is to extract possible objects using a region proposal method. In the next step method extracts features of objects using CNN. Finally classification of each region is performed [20]. This method gave good results and outperformed OverFeat.

R-CNN wasn't perfect and needed some improvements. Fast R-CNN, better version of R-CNN, was created in 2015. According to the name it's easy to understand that it was faster than previous one. In R-CNN the slowest part was the extraction of object proposals independently and this action took a lot of time, but Fast R-CNN was accelerated by sharing computations across all proposals. This improvement saves memory, that is needed for training the model.

Lately, the approach called YOLO (You Only Look Once) was invented. It's different from the previous methods of object detection. YOLO uses single neural network for predictions of bounding boxes and class probabilities directly from full images in one evaluation. In short, there's a grid over processed image and every cell of the grid has bounding boxes with different sizes. For each bounding box neural network computes the confidence if object in bounding box belongs to some class [21]. YOLO is much faster than previous models and it can be applied in detection of objects in video.

2.3.2 Single Shot Multibox Detector

The next step of evolution of object detection is Single Shot Multibox Detector (SSD) [19]. It's relatively new, and it bases on YOLO model, but gives better results and works faster.

SSD uses convolutional neural network. The main difference from other models is that SSD has layers from VGG-16 architecture, and after them there are additional convolutional layers. After each block of convolutional layers, there are feature maps of different sizes, that allows to detect small and big objects. These feature maps go through detector and classifier. In this step the network generates bounding boxes, localizes them and predicts the class of object in bounding boxes. After all these operations it performs filtration of found objects basing on their confidence. Bounding box with confidence higher than 0.5 will be marked as positive. SSD uses non-maximum suppression to group together with highly-overlapping boxes into a single box to avoid showing bounding boxes for one object. Softmax function is used as a classifier. Softmax allows to compute "probabilities" for all labels. Model is depicted in Fig. 2.1.

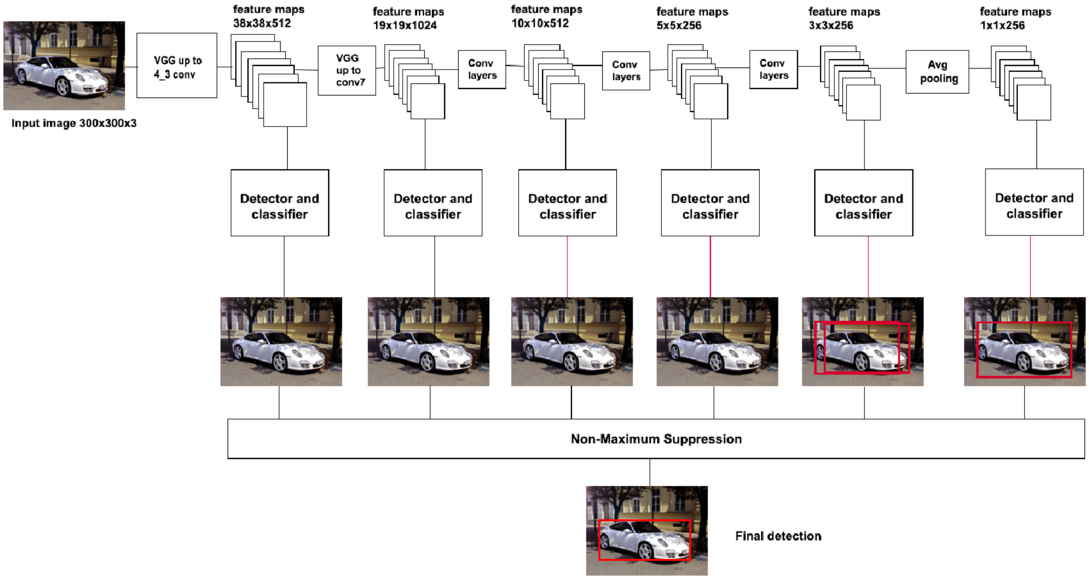


Fig. 2.1: Architecture of Single Shot Multibox Detector.

2.3.3 Application

People often meet with systems which use object detection even without noticing it. They can meet with object detection methods in their smartphones, on the Internet or in security systems.

One of the most well-known application of object detection is face detection. Nowadays the most phones have an application, that helps to detect face or smile and do auto-focus on it. This technique was very popular 7–10 years ago, but now it even doesn't surprise regular users. Moreover, the application of face recognition can be also found in security. For example, Samsung uses face recognition for unlocking smartphones or Apple added technology Face ID to their new iPhone X, that also works for unlocking a phone. Apple's technique is more advanced, as it uses a lot of points of face to keep 3D picture of it in memory. This approach should decrease the chances to fool the system with photos etc.

Also, the most popular application is visual search engine. The best example is search engine in Pinterest. Pinterest is social web, which helps people to share and find creative ideas. But the main feature of this project is that it helps to find similar products, which is done with object detection.

However, just a few of applications of object detection were mentioned above. Since people always try to create something new, the number of applications is still growing [20].

2.4 Image segmentation

Image segmentation is the process of dividing the image into multiple parts, which contains different objects. This is typically used for identification of objects or other relevant information in digital images [22]. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain visual characteristics [23]. Each pixel in a segment have the same properties, for example colour or intensity. Image segmentation is actual in computer vision, for example, in image recognition and image analysis. It can be usually met in medicine in analysis of medical images, also it's used in face recognition.

2.4.1 Existing methods

Region based segmentation

The aim of region based segmentation is to separate image into regions.

The simplest method is **threshold segmentation**. Thresholding creates binary images from greyscale ones by setting the value of all pixels with value below some threshold to zero and all pixels with value above it to one [24]. This method can be very effective, if the image has high levels of contrast. The disadvantage of this method is that it analyses the intensity of pixels, but doesn't use relationship between different pixels. It can cause that not all pixels will be included into some region. But at the same time the calculation is simpler, and the operation is faster.

Another method is **regional growth**. This is the most typical method of region-based segmentation. The main idea is to choose seed pixel, find nearest pixels with similar properties and group them to form a region. The advantages of this method are:

- it's possible to define the criteria, even multiple criteria for region detection, that makes it determined,

- it needs just a few seed points to define required property and to grow the region,
- it can work even with images containing noise.

The main disadvantage of this method is a high demand on resources and longer runtime. Nowadays exist a lot of improved approaches to segmentation, based on the region growing.

Edge Based Segmentation

Edge detection is identifying points in a digital image at which the image brightness changes sharply. These points are organized into set of curved line segments – edges [25]. The edge can be defined as significant local changes in the image intensity. This method reduces amount of information in the image and has important role in analysing, that's why this is the field for an active research. The most methods of edge detection are based on derivative operation of image intensity. The algorithm for edge detection has following steps:

1. **Filtering**. This is the step for reducing the noise as much as possible. But there should be taken in account that with more noise reduction chances to loss the edges in image increase.
2. **Enhancement**. The aim is to emphasize pixels in regions with a significant change in local intensity values.
3. **Detection**. In this step it is determined, which pixels belong to edges and which ones are noise.
4. **Localization**. The determination of location of edges [26].

Clustering Based Segmentation

This method is based on techniques which segment the image into clusters having pixels with the similar characteristics. The most used method in clustering segmentation is K-means clustering. It's simple and computationally fast. The main steps are:

1. Select k number of clusters and initial centre.
2. Calculate the distance from each pixel to each cluster centre.
3. Assign all the pixels to the nearest centre based on distance.
4. Recalculate new position of the centre.
5. Repeat steps until it satisfies the tolerance or error value [27].

The result of this algorithm depends on initial centres, that's why they should be carefully chosen. Moreover, the computation complexity of it depends on chosen k and number of data elements.

2.4.2 Segmentation with deep learning

For purpose of image segmentation Fully Convolutional Network (FCN) can be used . The FCN is extended version of classical CNN. The main difference between them is that CNN can be used for image classification – defining the class of image. As a result it produces numbers, for example, values of probabilities. In other words, there's information about image, that can be changed to a required form. The output of FCN is the segmented image, that has the same dimensions as the input image.

The architecture of fully convolutional network, shown in Fig. 2.2, can be divided into two parts:

1. **"encoding"** – reducing the size of image (combination of convolutional and poooling layers).
2. **"decoding"** – scaling up image and returning to input size, after that segmented part of output image will be matched with input of 1.

Upsample layer that scale up image is used for matching.

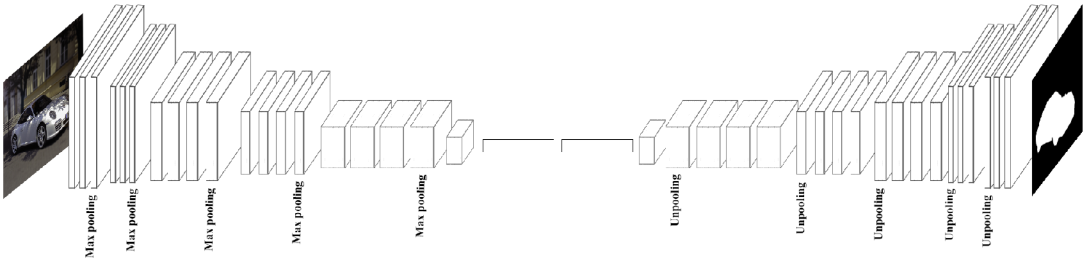


Fig. 2.2: Architecture of fully connected network.

For network training it's necessary to create a dataset with images and for each image to create the mask. Mask is a monochrome image, where each colour defines the group of objects. For example, in Fig. 2.3 there's an object with white colour in mask and background with black colour in mask.

For better results there are some points to be taken in consideration:

1. The training dataset should contain different images.
2. The location of objects should be defined as much accurate as is possible.
3. There's no universal model, that can be used for every task. Everything should be adapted for the required task.

Nowadays some implementations of fully connected network for image segmentation exists, for example U-net, SegNet, RefineNet and others [28].

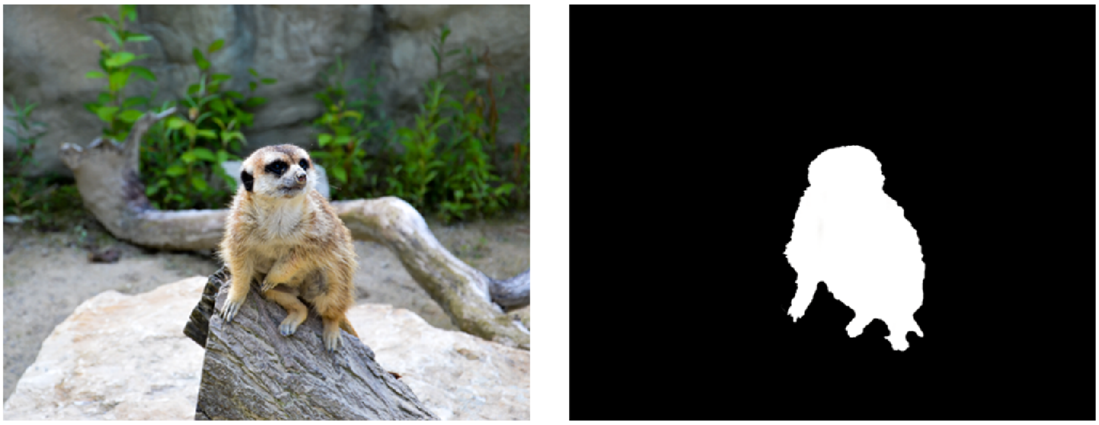


Fig. 2.3: Input image with a mask for training.

2.5 Frameworks and libraries that can be used in keyword detection

Obviously, researchers, working in computer vision and deep learning, focus more on algorithms and ideas, but less on tools which can be used for this. That's why it's necessary to use already created libraries and frameworks, which can be applied in implementation of those ideas. The most famous and wide used libraries and frameworks, which can help to implement the application for keyword detection are described in this part.

2.5.1 Deep learning frameworks

TensorFlow. The most famous library from engineers and researchers who work in Google. This library is designed for numerical computations and it's often used in machine learning, for example, in neural networks. TensorFlow is implemented in different programming languages: Python, C++, Java, but mainly it's used in Python. The advantage of TensorFlow is that it can be used not only on CPU, but also on GPU.

Keras. Keras is a high-level neural network API for Python. It is capable to run on top of such libraries as TensorFlow, Theano. It allows to do fast experimentations. The key feature of this API is friendliness, that means, user needs to do minimum actions for common cases and it provides understandable feedback in case of errors. Easy extensibility, which allows user to add new functions and modules is also one of main advantages of the library [29].

Caffe. Another library for deep learning is Caffe. It is implemented in Python and C++. Caffe is more focused on image classification and segmentation. It's also fast, that allows it to be used in an industry deployment. Nowadays also Caffe2 exists, which is based on original Caffe, but with some improvements, focused more on performance and speed.

2.5.2 Object detection

Single Shot Multibox Detector (SSD). New one in the world of object detection, SSD firstly appeared in 2016. It's not library, but framework, that allows to quickly detect objects in pictures, but it's also enough fast to detect objects in videos. This framework was originally written upon Caffe library, but implementations with Keras also exist.

Open-CV. It's library for processing images. The aim of it is to work with real-time computer vision. It's cross platform, supports Linux and Windows. It can be used in Java, C++, Python and C. This library has more than 2500 optimized algorithms, which can be used in different computer vision tasks: face detection, identification of objects, moving objects tracking etc.

2.5.3 OCR

Tesseract. Tesseract is optical character recognition engine, which is mainly used in extraction printed or handwritten texts. It can be used directly or via API. It also works in Linux, MacOS and Windows and supports more than thirty languages.

3 EXPERIMENT DESCRIPTION AND APPLICATION DESIGN

The aim of this work is to figure out if object detection can be applied in keyword detection. This application can be used in, for example, detection of confident information in images. For sure, using some keywords can be defined from the context of document, that's the reason to analyse document in this way.

The characteristics of used computer for this experiment are:

- CPU – Intel(R) Core(TM) i7-4700HQ 2.40GHz,
- RAM – 8GB,
- Grafic Processing Unit (GPU) – NVIDIA GeForce GT 750M (with CUDA support), RAM 4GB.

Two algorithms were chosen for this experiment: Single Shot Multibox Detector and Fully Connected Network. For achieving better results, especially speed, it was necessary to use libraries, which are the best in this field. I decided to use Keras and TensorFlow libraries. Keras is fast and simple, TensorFlow focuses on effective computing, especially in neural networks. These libraries are compatible with programming language Python and that's the reason why I use this language in this work.

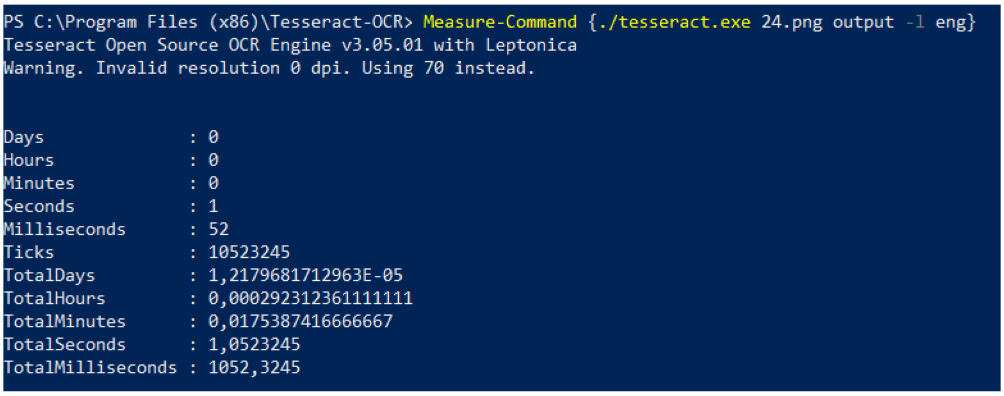
For simplifying management of necessary packages, I decided to use Anaconda. It's a free distribution for Python, that includes a lot of packages for large-scale data processing and scientific computing. For better comparability with TensorFlow, there were used Python 3.5, Keras 2.0.8, TensorFlow 1.3.0. TensorFlow for GPU 1.3.0 and Open-CV 3.4.0 were installed additionally.

Taking into the account that a word is a very specific object in an image, some requirements during training must be met. For example, it should be good differentiable with detector from other words.

Since there were used two different models, it's necessary to use different datasets. For generating training and validation datasets, I implemented a generator in Java, which can generate datasets for both models.

3.1 Experiment with Optical Character Recognition

Before starting the experiments with neural networks, it's better to test the way with OCR. For testing was used existing engine Tesseract, version 3.05.01 for Windows. The used image has dimensions $512 \times 512 \times 3$. The basic operation, text recognition, took 1052 ms, it's shown in Fig. 3.1. As it's expected, OCR takes more than one second to make recognition, which isn't suitable for fast detections.



```
PS C:\Program Files (x86)\Tesseract-OCR> Measure-Command {./tesseract.exe 24.png output -l eng}
Tesseract Open Source OCR Engine v3.05.01 with Leptonica
Warning. Invalid resolution 0 dpi. Using 70 instead.


Days              : 0
Hours             : 0
Minutes           : 0
Seconds           : 1
Milliseconds      : 52
Ticks             : 10523245
TotalDays         : 1,2179681712963E-05
TotalHours        : 0,000292312361111111
TotalMinutes      : 0,0175387416666667
TotalSeconds      : 1,0523245
TotalMilliseconds : 1052,3245
```

Fig. 3.1: The result of image recognition with OCR.

3.2 Experiment with Single Shot Multibox Detector

In the first part of experiment the SSD algorithm was tested. SSD was described in subsection 2.3.2.

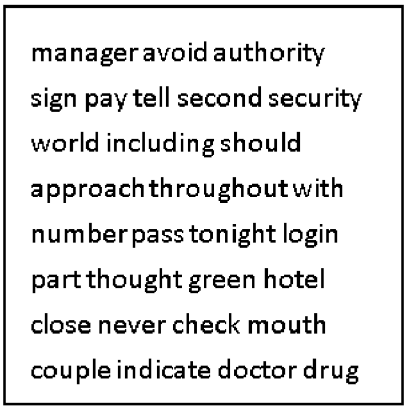
According to official article it works with dimensions $300 \times 300 \times 3$. Technically, when program gets the input image, it will be compressed and resized to this size. For such objects, as a cat, a dog, a car, a person, it's not so critical, the features for each object are very different and can be extracted even without perfect quality of image or with small size, but when it gets image with text, after compressing and resizing it doesn't look like readable even for a human. That was the reason to use dataset of images with required dimensions to avoid compression and loss of the image quality.

The other important fact to consider is that computer has memory limits, that's why it is necessary to divide dataset into smaller parts, but not to provide the neural network with the whole dataset. `Generator` included into Keras library can do the division of dataset and the number of images in each part can be defined in a parameter called `batch_size`. In addition, image preprocessing can be implemented in `Generator`. In this work such modifications as brightness, saturation and contrast were implemented. It was done for feeding model with different images, since all modifications in preprocessing of `Generator` are made randomly.

Another important parameter in neural network is a number of epochs. It defines the number of times that model goes through all dataset during a training. In this experiment, there were 60 epochs.

3.2.1 Training data

I attempted to detect in images such words as "PIN", "CardID", "password", "login". Images were generated with own generator, the format of all images is PNG because files compressed with this format are lossless. Every image has just one keyword. Example of a training image is in Fig. 3.2.



manager avoid authority
sign pay tell second security
world including should
approach throughout with
number pass tonight login
part thought green hotel
close never check mouth
couple indicate doctor drug

Fig. 3.2: Example of training image for SSD.

Dataset of 250 images was generated for this part: 200 images are for training and 50 of them are for validation. Also, there were used different sizes and fonts such as Arial, Calibri, Times New Roman, Verdana. Each keyword has different number of training images:

- CardID - 61,
- PIN - 67,
- login - 72,
- password - 50.

As it was described in official paper, images should have dimensions 300 × 300 × 3. Also, this model requires, that every image must have the XML file assigned, where there would be defined the following information about a picture:

- name – name of the image file, which belongs the XML file to,
- label – name of class, which belongs the image file to,
- width – width of the image,
- height – height of the image,
- xmin, ymin – coordinates of top left corner of object,
- xmax, ymax – coordinates of bottom right corner of object.

The example of XML file is shown below.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<image id="001">
      <name>001.png</name>
      <label>login</label>
      <width>300</width>
      <height>300</height>
      <coordinates>
        <xmin>205</xmin>
        <ymin>158</ymin>
        <xmax>254</xmax>
        <ymax>186</ymax>
      </coordinates>
</image>
```

3.2.2 Test data

After the image pass through model, as result of detection there will be confidences with bounding boxes. Anyway, there's no need to show bounding boxes with small confidence, that's why it will show detected object, if the confidence is higher than 0.6. Generated images similar to training and validation images were used as test images. There are the same fonts, size, dimensions.

3.3 Experiment with Fully Connected Network

As in the first part of experiment, there was attempt to detect keywords: "PIN", "CardID", "password", "login", but using Fully Connected Network. The architecture of it is shown in Fig. 3.3. As activation function in the first layers ReLU and sigmoid functions were used. Moreover, in difference from the previous model, `ImageDataGenerator` is used, which is also included in Keras library. I decided to use this tool to generate more images, but with some transformations, for example, rescale, rotation, flip horizontally and so on. In this experiment I used rotation of images. The network performs 33 iterations with 5 epochs, after each epoch it saves weights and uses them for the next epoch. The result is a mask with black area, that defines the location of a keyword.

3.3.1 Training and validation data

The dataset for training a network was also automatically generated. This model doesn't require XML file, but another image – mask, that defines the location of object (keyword). Location is represented by a black colour area on a white background. The principle of model is described in subsection 2.4.2.

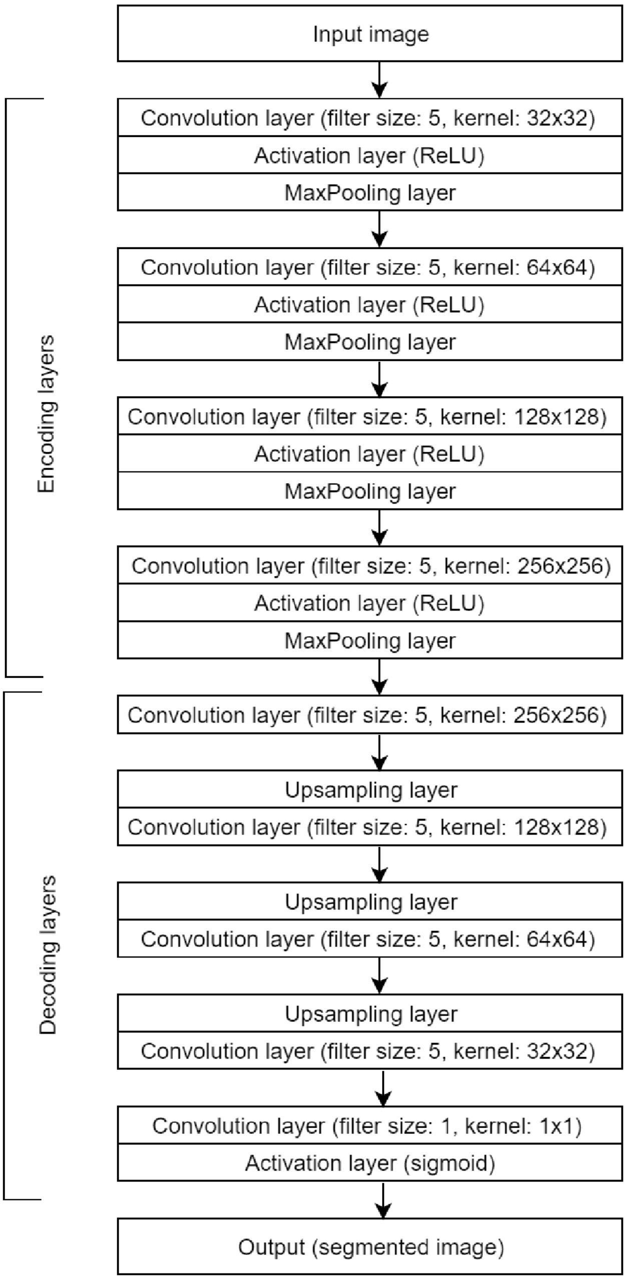
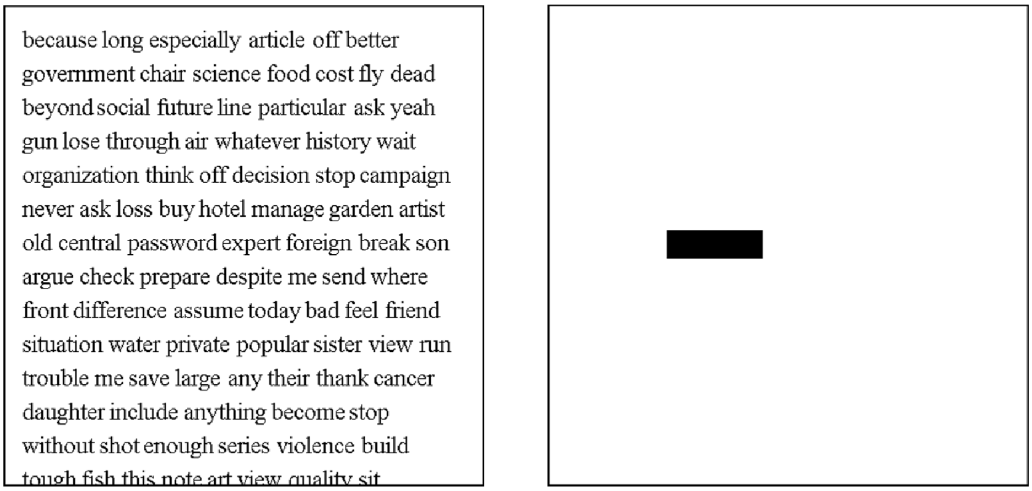

Input image
Convolution layer (filter size: 5, kernel: 32x32)
Activation layer (ReLU)
MaxPooling layer
Convolution layer (filter size: 5, kernel: 64x64)
Activation layer (ReLU)
MaxPooling layer
Convolution layer (filter size: 5, kernel: 128x128)
Activation layer (ReLU)
MaxPooling layer
Convolution layer (filter size: 5, kernel: 256x256)
Activation layer (ReLU)
MaxPooling layer
Encoding layers
Convolution layer (filter size: 5, kernel: 256x256)
Upsampling layer
Convolution layer (filter size: 5, kernel: 128x128)
Upsampling layer
Convolution layer (filter size: 5, kernel: 64x64)
Upsampling layer
Convolution layer (filter size: 5, kernel: 32x32)
Convolution layer (filter size: 1, kernel: 1x1)
Activation layer (sigmoid)
Decoding layers
Output (segmented image)

Fig. 3.3: The architecture of used Fully Connected Network.

Moreover, this model works with images with dimensions $512 \times 512 \times 3$. For purposes of this work, images with keywords "PIN", "CardID", "password", "login" were generated - 50 images for each word with different fonts (Arial, Calibri, Times New Roman, Verdana), sizes and colours. The example of training image is shown in Fig. 3.4.

This model requires special folder structure with training and validation images: each class has its own folders for input and label.



because long especially article off better
government chair science food cost fly dead
beyond social future line particular ask yeah
gun lose through air whatever history wait
organization think off decision stop campaign
never ask loss buy hotel manage garden artist
old central password expert foreign break son
argue check prepare despite me send where
front difference assume today bad feel friend
situation water private popular sister view run
trouble me save large any their thank cancer
daughter include anything become stop
without shot enough series violence build
tough fish this note art view quality sit

Fig. 3.4: Training image (left) and mask (right) for FCN.

3.3.2 Test data

For testing the trained model I used two datasets:

1. dataset with black text only.
2. dataset with different colours of text.

These datasets are also generated automatically and similar to the training images.

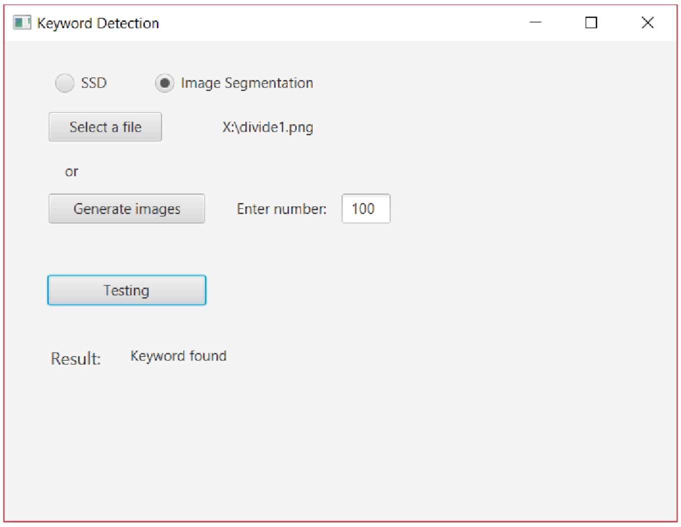
3.4 Design of application

Since the project includes different functional parts: two models of neural network and generator of training and validation images, it was necessary to connect them. I created an application in Java, that features graphical user interface, generator of training and validation datasets and class for dividing input images into smaller pieces in case of bigger input images.

3.4.1 GUI

For implementation of graphical user interface JavaFX was used. JavaFX is a set of graphics and media packages that allows developers to design, create, test, debug, and deploy client applications [30].

The application user interface clean design is shown in Fig. 3.5. It has radio buttons for choice of using model, button for easy uploads of image and button for testing input image.



○ SSD ● Image Segmentation

X:\divide1.png

Enter number: 100

Result: Keyword found

Fig. 3.5: Graphical user interface of application.

User can choose, which images will be used for test: automatically generated or uploaded. Important to note, that detections with uploaded images can be not so accurate as generated, because of limited training dataset, which were used for both models. If user chooses testing with FCN (radio button "Image Segmentation"), the following methods will be called:

- method for creating folders,
- method for dividing image into smaller parts, which have required dimensions,
- method for running python script for testing (using model with Fully Connected Network).

In case user decides to use testing with SSD (radio button "SSD"), methods for creating folders and dividing image will be called too, but another python script for testing (using model Single Shot Multibox Detector).

3.4.2 Generator of training images

Since generating images manually is time-consuming, I implemented generator of datasets in Java. It can generate datasets for both SSD and FCN model, also there can be defined fonts, size, colours. The position of keyword is chosen randomly and other words that compose "text" in image are also randomly chosen from predefined dictionary (contains 1000 words). First of all the program creates folders for images. In case of FCN it creates folders for training each of the classes (input, label, transforms) and for validation (input, label, transforms). In case of SSD model, it creates folder, that contains folders with images and XML files.

Also, there can be created images for testing inside a separate folder `testing images`.

3.4.3 Launch Python scripts using Java

It was necessary to run Python scripts using Java. To achieve this I used class for running operation system processes `ProcessBuilder`. The command for launching script is as follows:

```
Process p = new ProcessBuilder("py", "-3.5", path).start();
```

As it can be seen, the new process was created with such arguments as name of application to run (python), version (3.5), and path to file. Important to note, that Cuda 8.0 and all packages (Tensorflow 1.3, Keras 2.0.8, OpenCV 3.4.0) for Python should be installed in the system. It can be done with package manager `pip`. For better compatibility it's better to install even older version of packages, the latest version can cause the error, because of changes in names of some functions and arguments.

4 RESULTS AND DISCUSSION

4.1 Single shot multibox detector results

During the process of training the model, following results were achieved: accuracy and validation accuracy are less than 0.35 mAP, loss is 0.30, validation loss is 0.86. All values are shown in Fig. 4.1 and 4.2.

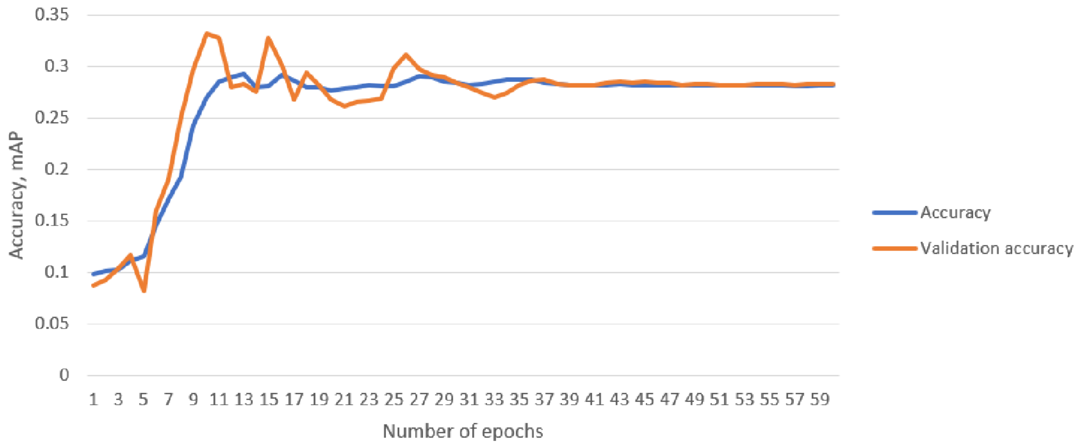


Fig. 4.1: Accuracy of SSD training process.

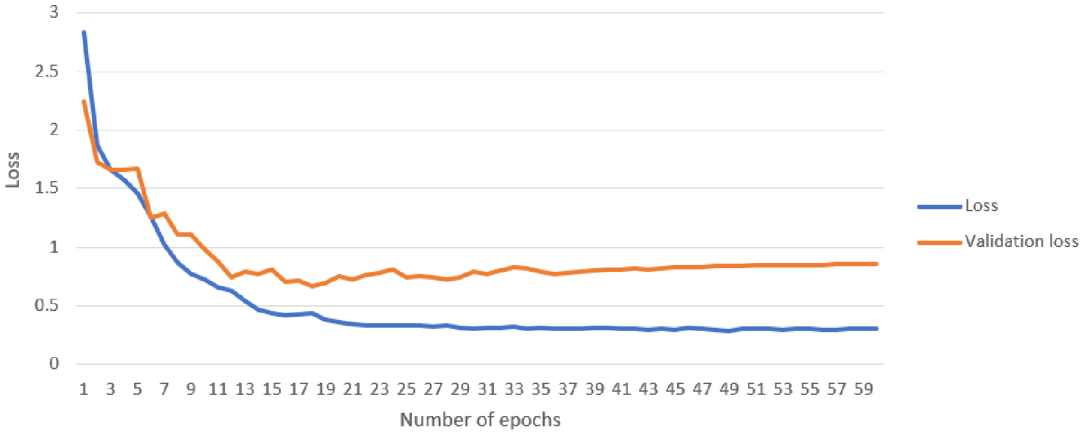


Fig. 4.2: Loss of SSD training process.

As it can be seen on the Figures 4.1 and 4.2, after 35 epochs the loss and an accuracy don't change so much and achieved results are not so good. The reason why this happened can be for example a relatively small dataset, not perfect parameters of neural network for this task.

The results for testing images are written in Table 4.1. In this table in column "Keyword in image" there is the word, that was detected with bounding box and in column "Keyword detected" there is the word, that was labeled with the neural network.

Despite low results during training, testing results are not so bad. As it can be seen from Table 4.1 the confidence is 100% for the most images. But number of correctly labeled images is just 21 of 35. Important to note, that for 1 image detection of keyword took just 0.3367 s. Anyway, the keyword can be detected, but not correctly labeled. The factors that can affect the results are: the number of training and validation dataset, preprocessing of images, parameters of model (activation function, number of layers etc).

The SSD model can be applied in a keyword detection: it's capable to extract features of words and to differentiate it from other words, but it still needs improvements.

4.2 Fully Connected Network results

The second part of this work was testing the model based on Fully Connected Network. There were two datasets:

1. dataset that contains images just with black text and white background.
2. Dataset which has images with different colours of text (black, light grey, grey, dark grey, blue).

As it can be seen on Fig. 4.3, Fig. 4.4, Fig. 4.5 and Fig. 4.6 the accuracy is higher (0.99 mAP) and loss is lower (0.0065) comparing to SSD.

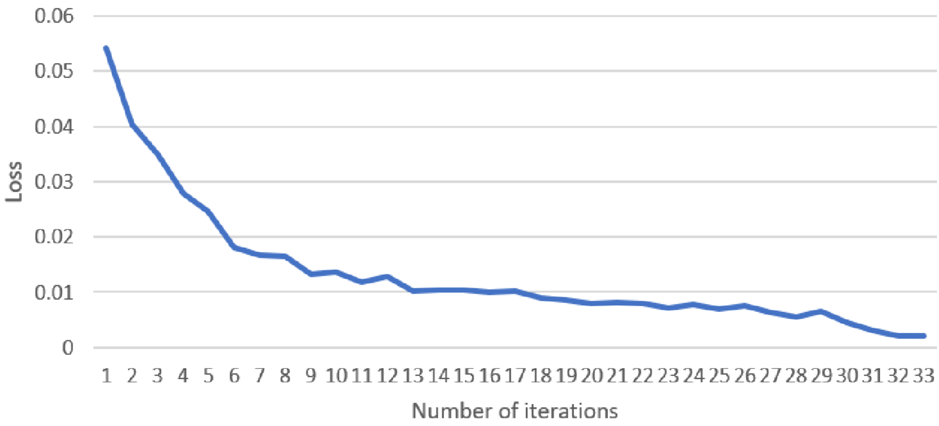


Fig. 4.3: Loss of FCN training process.

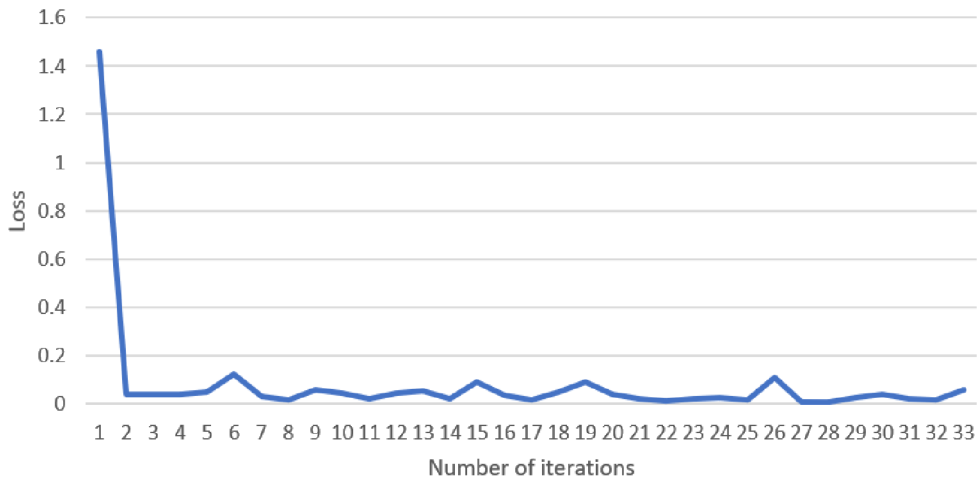


Fig. 4.4: Validation loss of FCN training process.

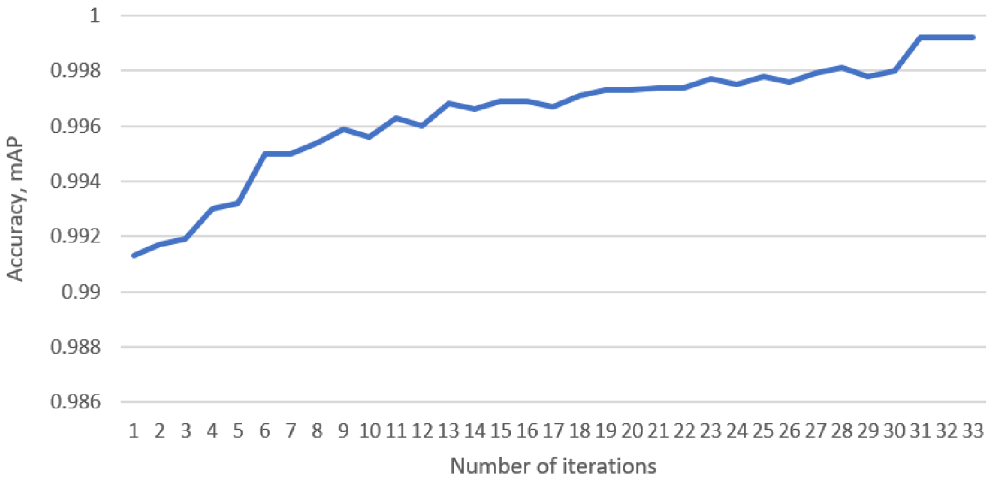


Fig. 4.5: Accuracy of FCN training process.

The time needed for testing each image is around 0.15 s for both datasets. The testing dataset with black colour gave better results: in 72 of 100 images it detected keywords and just in two cases it resulted with false positive (in detected area there wasn't a searched keyword). In dataset with text of different colours in 63 of 100 images was detected keyword and just in one image there was false positive – another word was detected as a keyword. The example of result is shown in Fig. 4.7 and in Fig. 4.8. It means, that results of this model depend on number of images in dataset: if each colour has more images, than better results could be expected. But it's an important finding that this model is able to work even with coloured images.

In real application there are some points to consider:

1. The neural network should get much more training images, where keywords

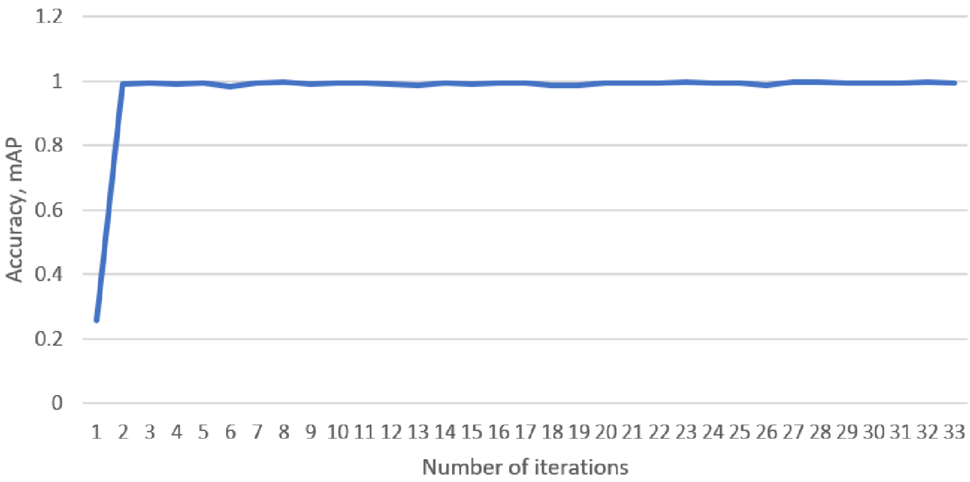
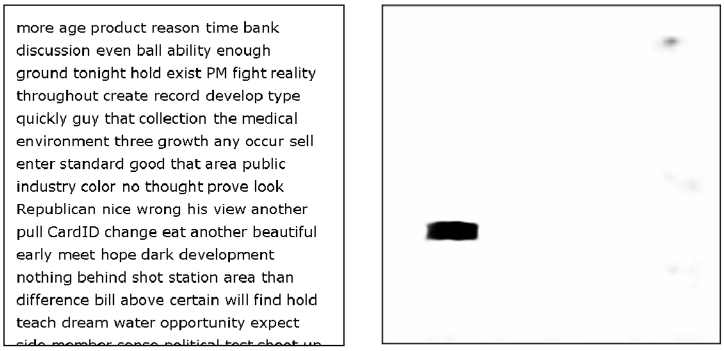
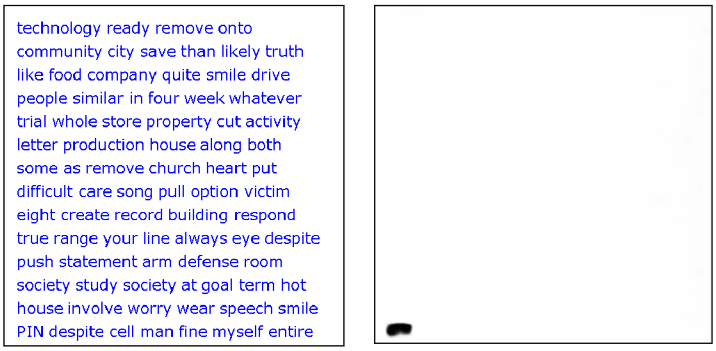


Fig. 4.6: Validation accuracy of FCN training process.



more age product reason time bank
discussion even ball ability enough
ground tonight hold exist PM fight reality
throughout create record develop type
quickly guy that collection the medical
environment three growth any occur sell
enter standard good that area public
industry color no thought prove look
Republican nice wrong his view another
pull CardID change eat another beautiful
early meet hope dark development
nothing behind shot station area than
difference bill above certain will find hold
teach dream water opportunity expect
side member sense political test shoot up

Fig. 4.7: The testing image (left) and output of model - mask (right).



technology ready remove onto
community city save than likely truth
like food company quite smile drive
people similar in four week whatever
trial whole store property cut activity
letter production house along both
some as remove church heart put
difficult care song pull option victim
eight create record building respond
true range your line always eye despite
push statement arm defense room
society study society at goal term hot
house involve worry wear speech smile
PIN despite cell man fine myself entire

Fig. 4.8: The testing image with blue text (left) and output of model - mask (right).

would have different sizes, fonts, colours, position.

2. The model works better, when compression and resizes of images are minimal. That's why, when there is document with bigger size, an image should

be divided into smaller parts to avoid severe distortion. Probably, the text detection will be slower in this way, but the hardware (such as GPU, CPU) is always being improved and RAM is getting more available, so modern PC would have more memory than now, that's why in nearest feature it won't be such problem and the speed of detection can be better than now.

Despite the mentioned above limitations, the neural network can be used in keyword detection, and as it can be seen, the Fully Connected Network gives better results.

Name	Size of image	Keyword in image	Keyword detected	Accuracy
001.jpg	300 × 300	-	-	-
002.jpg	300 × 300	CardID	CardID	0.95
003.jpg	300 × 300	CardID	CardID	1.00
004.jpg	300 × 300	CardID	CardID	1.00
005.jpg	300 × 300	login	CardID	1.00
006.jpg	300 × 300	password	CardID	1.00
007.jpg	300 × 300	password	CardID	0.89
008.jpg	300 × 300	CardID	CardID	1.00
009.jpg	300 × 300	password	CardID	1.00
010.jpg	300 × 300	CardID	CardID	1.00
011.jpg	300 × 300	PIN	PIN	1.00
012.jpg	300 × 300	login	CardID	1.00
013.jpg	300 × 300	PIN	PIN	1.00
014.jpg	300 × 300	PIN	PIN	1.00
015.jpg	300 × 300	CardID	CardID	0.99
016.jpg	300 × 300	PIN	PIN	1.00
017.jpg	300 × 300	PIN	PIN	1.00
018.jpg	300 × 300	coach	CardID	1.00
019.jpg	300 × 300	PIN	PIN	1.00
020.jpg	300 × 300	login	-	-
021.jpg	300 × 300	login	CardID	1.00
022.jpg	300 × 300	CardID	CardID	1.00
023.jpg	300 × 300	CardID	CardID	1.00
024.jpg	300 × 300	login	CardID	1.00
025.jpg	300 × 300	PIN	PIN	1.00
024.jpg	300 × 300	PIN	PIN	1.00
025.jpg	300 × 300	password	CardID	1.00
026.jpg	300 × 300	PIN	PIN	1.00
027.jpg	300 × 300	PIN	PIN	1.00
028.jpg	300 × 300	password	CardID	1.00
029.jpg	300 × 300	PIN	PIN	1.00
030.jpg	300 × 300	CardID	CardID	0.92
031.jpg	300 × 300	CardID	CardID	1.00
032.jpg	300 × 300	login	CardID	1.00
033.jpg	300 × 300	password	CardID	1.00
034.jpg	300 × 300	login	CardID	1.00
035.jpg	300 × 300	login	CardID	0.99

Tab. 4.1: Results of testing SSD.

5 CONCLUSION

The objectives of this work were to study the problem of object detection and try to find keywords in images containing text using neural network.

In the scope of this work a research was conducted, I studied how deep learning can be applied in information security and how it can be applied in detection of escape of private information, that is contained in images, for example, in scanned documents.

In the next part of my work, I used two architectures: Single Shot Multibox Detector and fully connected network to perform detection of text in images. For training and testing models I implemented generator of images containing texts and created datasets with it. Each dataset for training contained 250 images. The implemented application has two parts: the first part can generate and divide images into smaller parts (preparing data for neural network), it was implemented in Java. Second part are the neural networks implemented in Python, that take prepared data and perform prediction or training (depends on user's requirements).

The achieved results are quite optimistic: the both models are able to extract features from images with texts and find keywords. The FCN shown better results than SSD during training and testing. Moreover, the speed of detection is 150–300 ms, that is much faster, than detection with OCR (1052 ms). SSD correctly detected 22 of 35 images (62%), FCN correctly detected 62 of 100 images (62%) – dataset with different color images) and 70 of 100 images (70%) – dataset with black images). Also dataset for FCN with just one text color (black) gives better results.

The main contribution of this work is usage of neural network (fully connected network) in keyword detection instead of OCR: it's much more faster.

This fact allows to apply it in practice, for example, in analysing network traffic of firm or company, where always a lot of documents go through it. OCR isn't suitable for this task, it's not fast enough, but using deep learning it's possible to control information, that is received and sent over the network, especially using computers with more performing hardware than one used in this experiment.

BIBLIOGRAPHY

[1] ZHU, Y.; YAO, C.; BAI, X. Scene text detection and recognition: Recent advances and future trends. *Frontiers of Computer Science*, 2016, p.19–36.

[2] MARR, B. *What Is The Difference Between Deep Learning, Machine Learning and AI?* [online]. 08. 12. 2016 [cit. 11. 11. 2017]. Available: <`https://www.forbes.com/sites/bernardmarr/2016/12/08/what-is-the-difference-between-deep-learning-machine-learning-and-ai/#7ad1798926cf`>.

[3] ROUSE, M. *Definition: machine learning* [online]. 30. 06. 2017 [cit. 11. 11. 2017]. Available: <`http://whatis.techtarget.com/definition/machine-learning`>.

[4] FEHRENBACHER, K. *How Tesla is ushering in the age of the learning car* [online]. 16. 10. 2015 [cit. 05. 04. 2018]. Available: <`http://fortune.com/2015/10/16/how-tesla-autopilot-learns`>.

[5] PASCANU, R., et al. Malware classification with recurrent networks. In: *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on.* IEEE, 2015, p. 1916–1920.

[6] BAHNSEN, A. C., et al. Classifying phishing URLs using recurrent neural networks. In: *Electronic Crime Research (eCrime), 2017 APWG Symposium on.* IEEE, 2017, p. 1-8.

[7] DESHPANDE A. *A Beginner's Guide To Understanding Convolutional Neural Networks* [online]. Last actualization 20. 06. 2016 [cit. 14. 11. 2017]. Available: <`https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/`>.

[8] PERSSON, S. Application of the German Traffic Sign Recognition Benchmark on the VGG16 network using transfer learning and bottleneck features in Keras. 2018.

[9] *Fundamentals of Deep Learning – Activation Functions and When to Use Them?* [online]. Last actualization 23. 10. 2017 [cit. 15. 04. 2018]. Available: <`https://www.analyticsvidhya.com/blog/2017/10/fundamentals-deep-learning-activation-functions-when-to-use-them/`>.

[10] SHARMA V, A. *Understanding Activation Functions in Neural Networks* [online]. Last actualization 30. 03. 2017 [cit. 15. 04. 2018].

Available: <https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0>.

[11] JI Y. *ReLU and Softmax Activation Functions* [online]. Last actualization 11. 02. 2017 [cit. 20. 11. 2017]. Available: <https://github.com/Kulbear/deep-learning-nano-foundation/wiki/ReLU-and-Softmax-Activation-Functions>.

[12] BHANDARE, A., et al. Applications of Convolutional Neural Networks. *International Journal of Computer Science and Information Technologies,* 2016, p. 2206–2215.

[13] KARPATHY, A. *CS231n: Convolutional Neural Networks for Visual Recognition* [online]. Last actualization 11. 10. 2017 [cit. 16. 11. 2017]. Available: <http://cs231n.github.io/classification/>.

[14] *Macau's ATMs Are Using Facial Recognition to Help Follow the Money* [online]. Last actualization 28. 06. 2017 [cit. 16. 11. 2017]. Available: <https://www.bloomberg.com/news/articles/2017-06-28/macau-atms-need-face-time-before-payout-to-help-follow-the-money>.

[15] CHEYER, Adam J. *Device access using voice authentication.* U.S. Patent No 9,262,612, 2016.

[16] PINHEIRO, P.; COLLOBERT, R. Recurrent convolutional neural networks for scene labeling. In: *31st International Conference on Machine Learning (ICML).* 2014.

[17] ISRAFILOV, H. Application of neural network in handwriting text recognition. *The young scientist*, 2016, no. 29. p. 24–27.

[18] *What is OCR and OCR Technology* [online]. cit. 20. 11. 2017. Available: <https://www.abbyy.com/en-ee/finereader/what-is-ocr/>.

[19] LIU, Wei, et al. Ssd: Single shot multibox detector. In: *European conference on computer vision.* Springer, Cham, 2016. p. 21-37.

[20] REY, J. *Object detection: an overview in the age of Deep Learning* [online]. Last actualization 30. 08. 2017 [cit. 26. 11. 2017]. Available: <https://tryolabs.com/blog/2017/08/30/object-detection-an-overview-in-the-age-of-deep-learning/>.

[21] REDMON, J., et al. You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2016. p. 779-788.

[22] *Segmentation methods in image processing and analysis* [online]. [cit. 15. 04. 2018]. Available: <https://www.mathworks.com/discovery/image-segmentation.html>.

[23] *Segmentation* [online]. Last actualization 06. 03. 2017 [cit. 15. 04. 2018]. Available: <http://imagej.net/Segmentation>.

[24] MORSE B.S. Lecture 4: Thresholding, Brigham Young University, 2000, [online]. Available: <http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCALCOPIES/MORSE/threshold.pdf>.

[25] SHASHANK S.; SHIVANGI S.; KIRTI J. Review on Effective Edge Detection Techniques. *International Journal of Engineering Development and Research (IJEDR)*, November 2017, vol. 5, issue 4, p. 826-831. ISSN:2321-9939. Available: <http://www.ijedr.org/papers/IJEDR1704134.pdf>.

[26] JAIN, Ramesh; KASTURI, Rangachar; SCHUNCK, Brian G. *Machine vision.* New York: McGraw-Hill, 1995.

[27] DHANACHANDRA, N.; MANGLEM, K.; CHANU, Yambem Jina. Image segmentation using K-means clustering algorithm and subtractive clustering algorithm. *Procedia Computer Science*, 2015, 54.2015: p. 764-771.

[28] OZHIGANOV I. *Object detection using fully connected network* [online]. Last actualization 24. 11. 2016 [cit. 12. 05. 2018]. Available at: <http://ai-news.ru/2016/11/poisk_obekta_na_izobrazhenii_s_pomoshu_polnosvertochnyh_nejronnyh_setej.html>.

[29] *Keras: The Python Deep Learning library* [online]. [cit. 26. 11. 2017]. Available at: <https://keras.io>.

[30] *What Is JavaFX?* [online]. Last actualization: April 2013 [cit. 15. 05. 2018]. Available at: <https://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm>.

LIST OF SYMBOLS, PHYSICAL CONSTANTS AND ABBREVIATIONS

DNN	Deep Neural Network
CNN	Convolutional Neural Network
R-CNN	Region Convolutional Neural Network
SSD	Single Shot Multibox Detector
VGG-16	Model of the 16-layer network
GPU	Graphic Processing Unit
CPU	Central Processing Unit
XML	Extensible Markup Language
OCR	Optical Character Recognition
ReLU	Rectified Linear Unit
FCN	Fully Connected Network
mAP	Mean Average Precision
GUI	Graphical User Interface

LIST OF APPENDICES

A THE CONTENTS OF THE ATTACHED CD

```
/
├── 185934.pdf ............................................................ Thesis
└── project
    ├── Keyword Detection GUI ..................... Folder with generator and GUI
    │   ├── src
    │   │   └── sample
    │   │       ├── Controller.java
    │   │       ├── CreateXML.java .... Class for creating XML files for training model
    │   │       ├── GeneratorImages.java ................ Class for generating images
    │   │       ├── ImageDivider.java
    │   │       ├── Main.java
    │   │       ├── sample.fxml
    │   │       ├── SSDProcessor.java ........... Class for generating images for SSD
    │   │       └── UnetProcessor.java ......... Class for generating images for FCN
    │   ├── words.txt ..................... Dictionary for generating text for images
    │   └── Keyword Detection.jar ..................... Running file for application
    ├── SSD ............................................... Folder with SSD model
    │       ├── checkpoints - 300x300
    │       │   └── weights.59-0.86-0.28.hdf5
    │       ├── main.py ................................. Script for model testing
    │       ├── prior_boxes_ssd300.pkl
    │       ├── SSD_arch.py
    │       ├── ssd_layers.py
    │       ├── ssd_training.py
    │       ├── ssd_utils.py
    │       ├── train.py ................................ Script for model training
    │       └── xml_reader.py
    ├── text_segmentation .............................. Folder with FCN model
    │       ├── weights
    │       │   ├── weights32.hdf5 ........... Weights for text with different colours
    │       │   └── weights41.hdf5 ............... Weights for text with black colour
    │       ├── main.py ................................. Script for model training
    │       └── test.py ................................. Script for model testing
    └── readme.txt
```