Czech Technical University in Prague
Faculty of Information Technology

ASSIGNMENT OF BACHELOR'S THESIS

Title:	Automated solver of image based CAPTCHA
Student:	Vojtěch Mach
Supervisor:	Ing. Martin Kopp
Study Programme:	Informatics
Study Branch:	Software Engineering
Department:	Department of Software Engineering
Validity:	Until the end of winter semester 2018/19

Instructions

The image based CAPTCHA is the next generation of reverse Turing tests where the main challenge is pattern recognition. The output of this work should be a working prototype of the automated CAPTCHA solver using computational intelligence. The main goal is to show the weak spots of current CAPTCHA solutions.

1) Review existing image annotation solutions and the state-of-the-art literature.
2) Experiment with image CAPTCHA solving solutions based on textual annotation of images using tools found in the previous step.
3) Design and implement your own working prototype of automated CAPTCHA solver using artificial intelligence.

References

Will be provided by the supervisor.

Ing. Michal Valenta, Ph.D.
Head of Department

prof. Ing. Pavel Tvrdík, CSc.
Dean

Prague March 1, 2017

Czech Technical University in Prague

Faculty of Information Technology

Department of software engineering

Bachelor's thesis

Automated solver of image based CAPTCHA

Vojtěch Mach

Supervisor: Ing. Martin Kopp

16th May 2017

Acknowledgements

I would like thank my supervisor Ing.Martin Kopp for his positive attitude and for the opportunity to work on an interesting topic. I also want to appreciate my faculty for gained knowledge, teachers for creating a friendly environment and lastly all my close ones for their support throughout my studies.

Declaration

Citation of this thesis

Mach, Vojtěch. *Automated solver of image based CAPTCHA*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2017.

Abstrakt

Práce se zaměřuje na strojové rozpoznávání moderních obrázkových CAPTCHA systémů pomocí technik umělé inteligence. Cílem bylo poukázat na nedostatky tohoto zabezpečení jeho prolomením a otestovat vhodnost různých přístupů k problému. Pro tento účel byl navržen software, jenž shromažďuje více postupů rozpoznávání a klasifikace obrázků pro vybrané druhy vstupních úloh. Dle povahy úlohy jsou uživateli k výběru nabínuty různé řešící algoritmy. Mezi využívané techniky řešení patří předtrénované neuronové sítě komerčních webových služeb a algoritmus KNN. Srovnání úspěšnosti každého z algoritmů je uživateli přehledně zobrazena po dokončení výpočtů. Součástí práce je rovněž rešerše, zhodnocení průměrné přesnosti vybraných technik a dokumentace projektu.

Klíčová slova CAPTCHA, počítačové vidění, strojové učení, rozpoznávání obrazu, neuronové sítě, KNN algoritmus

Abstract

This thesis focuses on machine recognition of modern image-based CAPTCHA security systems using techniques of artificial intelligence. The goal was to show weaknesses of these systems by breaking through it and to test suitability of various approaches. For this purpose a software was designed which collects a few procedures of image classification and pattern recognition for different input tasks. According to the nature of the input task, applicable algorithms are presented to the user. The techniques utilized to solve this matter are commercial services using pretrained neural nets and KNN algorithm. An accuracy of each algorithm is presented to user after computations are finished. A research of this topic, precision evaluation of selected solutions and project documentation are also included in the thesis.

Keywords CAPTCHA, computer vision, machine learning, pattern recognition, neural nets, KNN algorithm

Contents

List of Figures

List of Tables

Introduction

CAPTCHA is a security mechanism which is widely used in computing to recognize human users from computers by various challenges. It falls into category of reverse Turing tests, which is a subcategory of regular Turing tests. Reverse Turing tests unlike regular Turing tests have a computer judge, which means that the testing system must be able to decide righteously on its own whether the tested subject is a human or a computer. CAPTCHA performs the test by presenting a user with any challenge of such properties that it should be easy to solve by a human but difficult to solve by a computer.

Motivation

There are many CAPTCHA schemes and new ones are still being created. Growing computing power and improving abilities of AI both carry a threat of breaking the challenges which stimulates a continuous development of more secure schemes. Older schemes have already been proven vulnerable to attacks of this matter and thus found unsafe. For this reason they were replaced by modern image-based schemes which provide better resistance against attacks while maintaining convenience of handling. A subset of chosen image-based challenges will be the focus of my work.

Goal of the thesis

The goal of the thesis is to create a prototype of an automated image-based CAPTCHA solver in a form of desktop application which will incorporate different computer vision techniques in order to break a given challenge. The application should serve as a benchmark for testing different approaches to the problem and will also provide a framework allowing addition of new ones. I will implement selected existing on line image classifiers as well as my own implementation of a classifier to demonstrate abilities of the framework and finally carry out a comparison of their accuracy in experiment.

CHAPTER 1

Analysis

This chapter analysis the thesis from both theoretical and practical point of view. In the first section I summarize the most commonly used CAPTCHA schemes and introduce the problem and its possible solutions. The second part focuses on my approach to the solution, analyses utilized tools and techniques and reviews selected solutions in detail. It also describes a creative process and explains the main structure of the application prototype.

1.1 CAPTCHA schemes

Many different CAPTCHA schemes were developed through the time. We can ultimately divide them to four groups described in following subsections:

- Visual - most common scheme, challenge presented in mostly static visual form
  - Text-based
  - Image-based
  - Mixed
- Auditory - targeting visually impaired people (see 1.1.2)
- Logical - a simple task requiring some intellectual process
  - Mathematical
  - General knowledge
- Other - infrequently used, minigames etc.

The most used schemes are reviewed closely in the next sections.

1.1.1 Visual scheme

In the world of CAPTCHA security, visual challenges surely have been the most used type since they are easy to generate while being relatively difficult to break until recently. Google is probably making the greatest progress in this area, they continuously improve their own CAPTCHA system called ReCaptcha which evolved from being a simple text-oriented challenge mechanism to present day sophisticated complex security system as described in the following subsections.

1.1.1.1 Older version of ReCaptcha

Challenges of ReCaptcha v1 were mostly text-based and generally used a short randomly generated sequences of characters and numbers transformed into an image with some sort of distorting features such as noisy background, twisted, deformed or even crossed out figures, random lines etc. (figure 1.1). These features are crucial for the security of this scheme, because it greatly complicates its exploitation by hackers. A common OCR software is unable to recognize deformed figures, human however is very successful at this. A reasonable explanation for this is that people learn handwriting since their youth so throughout their lives they already have seen many interpretations of the same character so a letter deformed by a computer will still be recognizable for them.

This scheme's biggest disadvantage is that there is only the set of characters and numbers is limited. Once a classifier that is able to recognize all symbols and numbers is developed, this security system becomes extremely vulnerable. This situation actually happened in 2014 when Google announced that their AI, originally purposed to read house numbers in Street View [1], managed to get astonishing accuracy of 99.8% when solving these types of challenges [2]. This was the impulse which lead to development of Google's newest version of ReCaptcha system implementing image-based challenges in combination with NoCaptcha security method as described further.

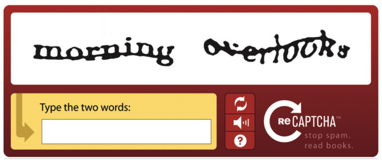


Figure 1.1: An old version of a ReCaptcha challenge

Mixed challenges (partly text, partly image) were introduced in late 2000's by Google which has brought an interesting new purpose to the world of CAPTCHAs. They effectively turned people into very advanced OCR machines. As seen on fig. 1.2, a user was presented with a text-based challenge and with an actual image of something, possibly a distorted word from a physical book which was being digitized by Google at the moment or a photography of a house number. If majority of users recognized the generated text challenge correctly there was a great chance they recognized both things correctly thus performing a reliable image recognition.

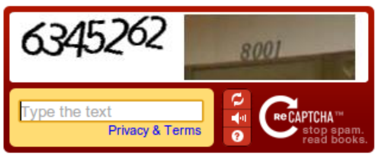


Figure 1.2: Newer ReCaptcha version.

This version of ReCaptcha has been deprecated since May 2016 and fully replaced by modern ReCaptcha v2.

1.1.1.2 Modern ReCaptcha

Current ReCaptcha v2 introduces a completely different approach to CAPTCHA security in a form of challenges based on real-world images. A user has to manually select a subset of presented images which are somehow related to the task of the challenge.

There are two basic variations of this challenge at the moment depending on specification of the task:

- specified by keyword (fig. 1.3)
- specified by reference image (fig. 1.4)

These two variations will be the focus of the practical part of this thesis..

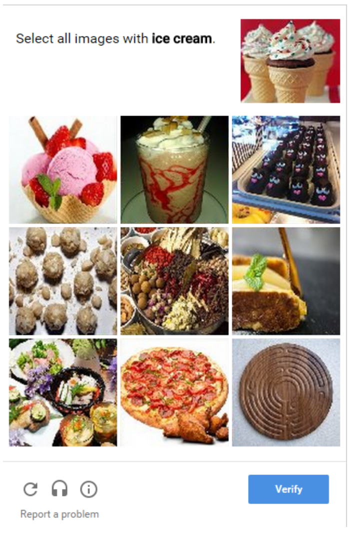


Figure 1.3: Example of ReCaptcha v2 challenge specified by a keyword.

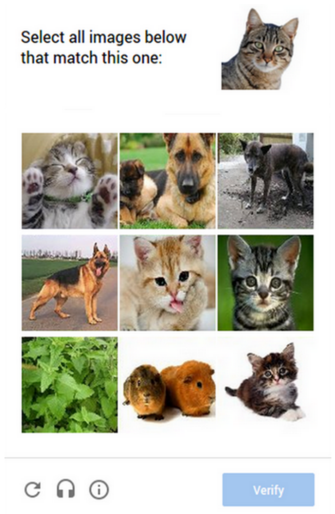


Figure 1.4: Example of ReCaptcha v2 challenge specified by a reference image.

There is currently one more variation, which is a little different from the previous ones. It prompts user with a single image of some object divided in a few parts resembling square puzzle pieces and asks user to select those pieces that have the object on them. For better understanding see fig. 1.5.

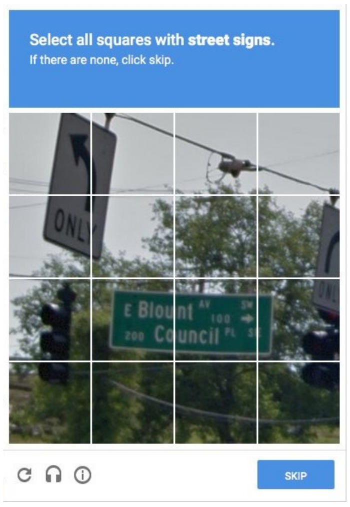

Select all squares with **street signs.**
If there are none, click skip.
SKIP

Figure 1.5: Example of third type of ReCaptcha v2 challenge

1.1.1.3 NoCaptcha technology

An important part of Google's ReCaptcha v2 is the NoCaptcha technology which adds another layer of security to the system. It uses browser cookies to perform a behavioral analysis and studies user's actions on a website. The API provided by Google contains a single check box with *I'm not a robot* prompt text 1.6. After clicking the check box, the user is presented with a CAPTCHA challenge only if he is found suspicious or if no relevant cookies are available yet. In practice, this actually works very well, there is an official website which demonstrates the system's functionality [3].

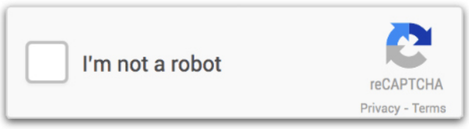


Figure 1.6: Google's NoCaptcha ReCaptcha system

Recently this year Google started advertising a newer version of NoCaptcha called Invisible ReCaptcha where the whole authorization is done in background [4]. This means that ideally every human visitor of a website protected with this system will have to complete the CAPTCHA challenge just once and not again until he erases the browser cookies.

1.1.2 Auditory scheme

Auditory challenges were created for people suffering from vision disorder and are usually bundled with the other schemes as an alternative for authorization. They usually contain a sequence of letters and numbers which are read to the user. The sound must be clear enough for the content to be recognizable by a human but noisy as well to be resistant to exploitation by bots. They are practically irreplaceable at the moment and that is the problem because simpler implementations show serious weakness against attacks as proven for instance by a bachelor's thesis from last year [5]. There were many attempts to bypass them using existing publicly available speech recognition technologies, which were so successful they initiated an improvement in this area [6, 7]. Despite that, audio challenges still remain the weakest point of CAPTCHA security system.

1.1.3 Logical and other schemes

Logical challenges often consist of a simple mathematical task or a general knowledge question and are typically found on Internet forums, blogs and smaller websites where top-level security is not needed. As stated in [8], these challenges are less resistant to hacks plus they may be language specific and therefore not universally applicable. There are other schemes that may contain an interactive challenge, e.g. a minigame. Interactive challenges are resistant to AI attacks but may be broken a script. These are relatively least popular of mentioned schemes.

1.2 Problem overview

In this thesis we tried to point out weaknesses of mentioned types of image-based CAPTCHA, by breaking it with artificial intelligence. This task eventually reduces to a problem of image recognition, i.e. passing unlabeled image to an image classifying software and obtaining labels of all objects in the image. Image recognition is a discipline of machine learning using computer vision algorithms to detect real-world objects in images. It is sometimes referred to as image classification, labeling or tagging, because it returns a set of labels (also tags or classes) corresponding to objects found in the image. An illustration of this process is shown in the following figure, where every circle represents one label returned for the input image in the middle. Note that the hierarchy of classes in the figure goes from right to left from general to more concrete ones.

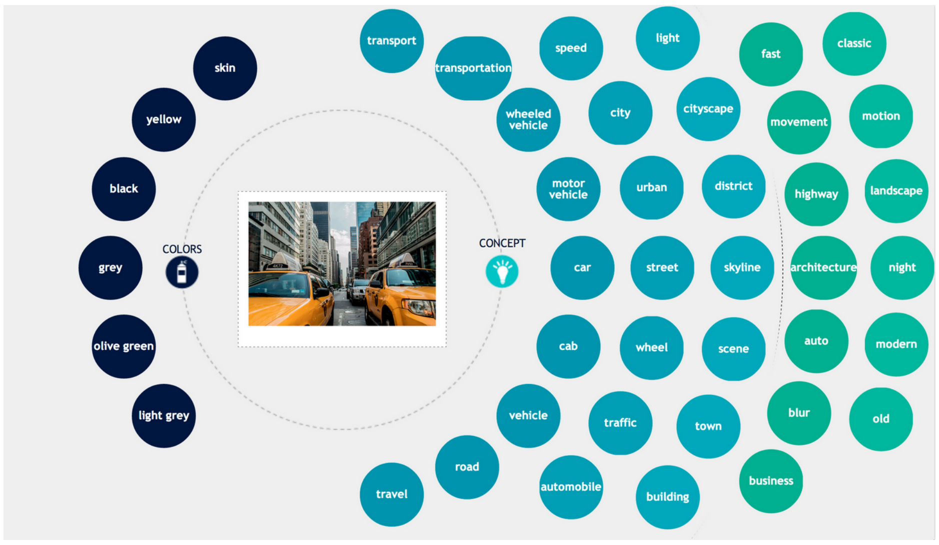


Figure 1.7: A simplified example of possible class hierarchy of larger image classification model

1.3 Examined algorithms

Many algorithms were invented for this purpose, however a rapid growth of computational power has put some models and methods in favor, especially deep learning, a class of algorithms which extract features from inputs on many levels by using multiple layers of functional units. In following subsections I review some of today's best performing models used for image recognition. The last subsection describes a general idea of KNN algorithm, which I have implemented in a custom classifier.

1.3.1 Artificial neural nets

Artificial neural network (ANN) is a general computational model used in machine learning to solve problems where ordinary programming techniques cannot be applied. Its structure resembles structure of a biological brain and simulates cooperation of brain neurons.
Artificial neurons are ordered in multiple cascaded interconnected layers where front layer takes inputs in any form (e.g. an image) and end layer deliver outputs. Every neuron combines its inputs in linear combination and applies a mathematical function known as activation function which determines the output value of the neuron as seen on fig.1.8. A list of examples of the most common activation functions is in fig.1.9

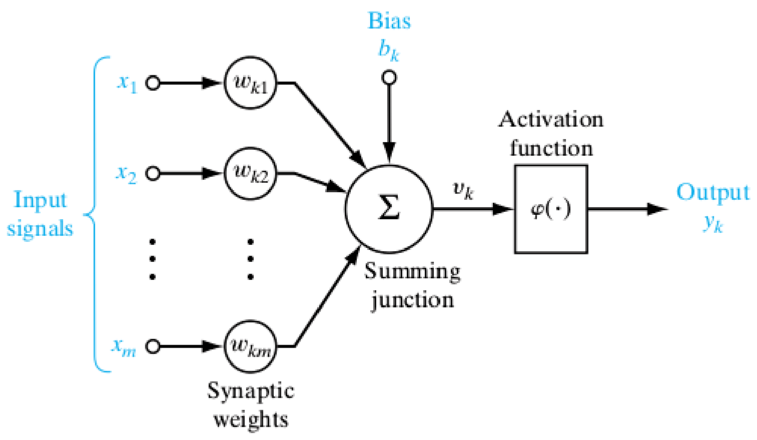


Figure 1.8: A symbolic schema of an artificial neuron

Since ANN is a classification and regression model, its purpose is to classify the input as one of previously defined classes. Each output neuron represents one class and delivers a decimal number from interval (0,1) measuring the model's confidence with the input belonging to that class. The output layer of neurons is finite which implies that the set of defined classes needs to be finite as well and previously defined. Large commercial models implement hierarchy of classes, similar those in 1.7, in order to produce correct guesses even for ambiguous or unclear inputs.

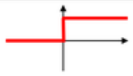
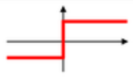
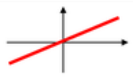
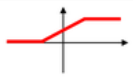
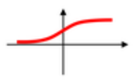
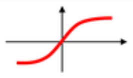
Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer NN	

Figure 1.9: Common activation functions used in various machine learning models

A powerful property of this model is that it does not require any explicitly defined rules to evaluate inputs, it rather modifies its structure to generate as accurate outputs as possible. This is done by learning by trial and error from input data where desired output is known and penalization is applied if a different output is returned. This process is called training of the net. The evaluation of outputs is not binary, it uses a special dedicated function called cost function to measure the progress of the training and applies penalization depending on the difference of accuracy between real output and desired output. The goal of the training is to modify the network's structure in such a way that its produced outputs are as accurate as possible on the set of training data and maintain more or less similar accuracy for completely new input data.
For the sake of example of the network's function in practice, consider a network trained to labeling images. Its output neurons then produce scores representing network's confidence in each label. An illustrative example of the network's outputs is shown on fig.1.10.

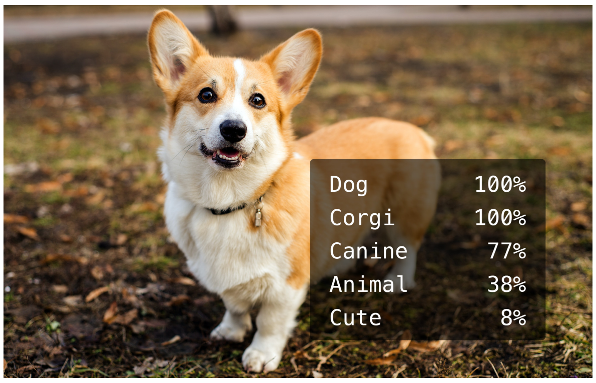


Figure 1.10: Typical output of an image labeling model

Advantages

If trained properly some variations of neural networks designed specifically to image recognition perform remarkably well and are therefore currently viewed as a state-of-art approach the area. They gained their popularity in last decade when accelerated training of the nets using powerful GPUs became possible, especially when run in parallel. This turned out to be a big step forward in this area since the training of the net is the key feature determining the model's accuracy[9]. Also the basic model definition is very general and allowed creation of many various implementations featuring different activation functions, cost functions, neuron connections and even special training methods. Some of the models that achieved especially good results in image recognition, like convolutional neural networks and deep belief networks are described further in this subsection.

Disadvantages

Probably the main issue with models of machine learning in general is gathering enormous quantities of data for the training of the model. Also the training itself must be done carefully in order to keep the structure sufficiently general. Theoretically it is possible for the network, if trained long enough, to achieve near 100% accuracy on a particular training set. This phenomenon is called overfitting and is highly unwanted, because it means that the net became an expert in distinguishing unimportant details of concrete inputs of the training set but disregards defining features of inputs as general concepts. Overfitted networks are often characterized by excellent results for training inputs and poor results for new unknown inputs.

Convolutional neural networks (CNN)

Convolutional neural nets are a modification of multilayer ANN performing especially well in areas of natural language processing and image recognition. I will describe a CNN's functionality in case of image classification, i.e. assigning a labels to unlabeled image. Following is a list of the network's layers and their function.
A common CNN consists of these four types of layers:

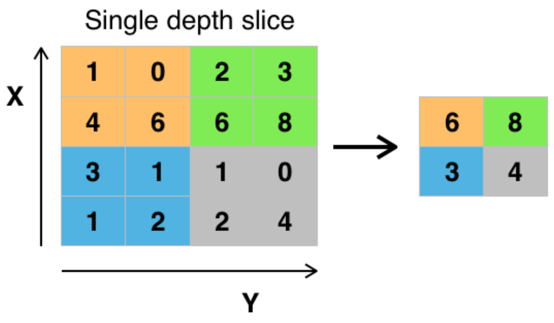
1. **Convolutional layer**
   Neurons in this layer are responsible for finding a certain pattern in specified region of an input image. They use mathematical operation of convolution to approximate their responses. Every pattern creates a so called feature map which represents a presence of the feature on each position in the input. That means the dimensions of the input grows with each pass through this layer as seen on fig. 1.14. This issue is solved by incorporating pooling layers.

2. **Pooling layer**
   This layer performs downsampling of the input from previous layer. Different functions may be applied, a typical example of is a *maxpooling* function. This function divides input in non-overlapping rectangles and the selects the maximal value from each of them. The process is shown in fig. 1.11.

3. **Normalization (ReLU) layer**
   ReLU, which stands for *rectified linear units*, is a layer of neurons that apply an activation function defined as $f(x) = max(0, x)$ to normalize the input feature map. This function is called rectifier and replaces all negative values of input with zeros as shown in fig. 1.12. According to [10], ReLU is the most popular activation function since 2015, despite its simplicity. Common alternative functions are sigmoid and hyperbolic tangent (fig. 1.9).
   Some models omit this layer as it has shown little contribution to network's overall accuracy in certain cases.

4. **Fully connected layer**
   A sequence of common ANN layers performing classification of the inputs into predefined categories

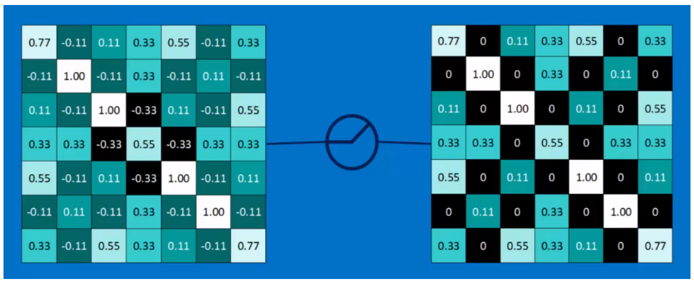
Layers 1., 2. and 3. are responsible for feature extraction and are often inserted in the network multiple times in chain behind each other as shown in fig. 1.13. The fully connected layers are always present at the end of the network and perform the actual classification and delivery of output labels with their respective scores. Figure 1.14 presents a simplified schema of the



1	0	2	3
4	6	6	8
3	1	1	0
1	2	2	4

6	8
3	4

Figure 1.11: Maxpooling function applied on a feature map



0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	0.77

Figure 1.12: ReLU function applied on a feature map

network as a whole. The advantage of CNNs, if properly trained, is very accurate outputs (be it a pattern recognition or another use case). On the other hand its complicated structure may be viewed as a disadvantage apart from already mentioned issues of machine learning.

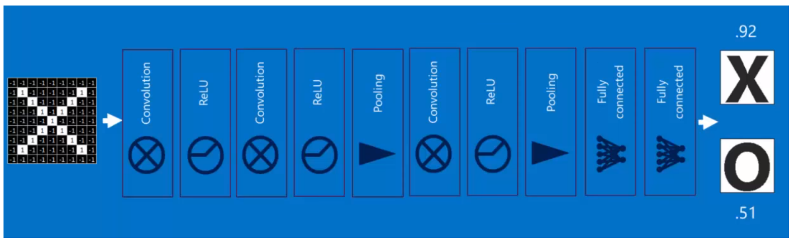


Figure 1.13: A symbolic image of layer chaining in CNN

Google's implementation of a CNN called GoogLeNet is currently one of the best performing image recognition model in the world. In 2015 it contained

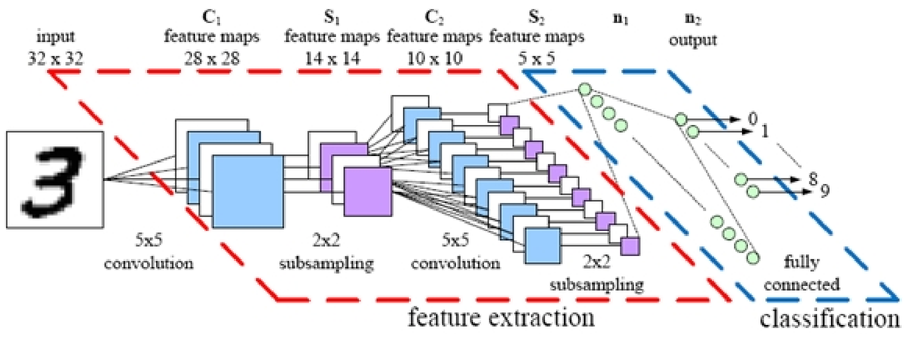


Figure 1.14: A simplified schema of convolutional neural network

22 layers [11] and it is still being developed. For illustration of a real-world functional network, a schematic picture of GoogLeNet's layers is shown on figure 1.15.
GoogLeNet is the winning model of ILSVRC[1] 2014 [12, 13] and also a foundation of the famous computer program DeepDream[14, 15].

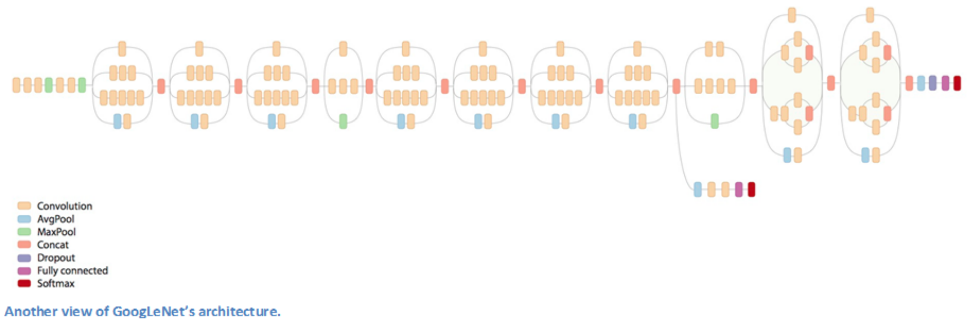


Figure 1.15: A structure of GoogLeNet

Deep belief neural networks (DBN)

A decent alternative to convolutional neural networks could be a deep belief networks. These networks consists of stack of autoencoders such as restricted Boltzmann machines (RBM). Autoencoder is a neural network model used in feature extractions and input encoding. A restricted Boltzmann machine is two-layer autoencoder where every neuron from one layer is connected to every neuron of next layer but no two neurons of the same layer are connected.

[1]ImageNet Large Scale Visual Recognition Competition

The training of RBM is done in several iterations by passing the input through the net and encoding it in a set of numbers, then passing the numbers back and reconstructing the input as accurately as possible. An important aspect of RBM is that the data do not need to be labeled manually, the auto-encoder rather extracts the important features of the input and stores them in its own structure. This way it can automatically sort unlabeled data in categories. In deep belief network, a hidden layer of each RBM acts as an input layer of next RBM.
Deep belief network's biggest advantage lies in its ability to be trained with relatively smaller amount of data compared to the case of CNN, which also implies a shorter training time.

1.3.2 K nearest neighbors (KNN)

K nearest neighbors is one of the simplest machine learning classification algorithms. The general idea of the algorithm is that every element of data has a set of defining features distinct from other elements. We wish to extract those features from each element to so called feature vectors, elements of n-dimensional vector space, and measure the distances between them using a distance function. A form of the features as well as the measurement technique are arbitrary and implementation dependent.
The classification of an input element, or precisely its feature vector, is done by finding its K (already classified) nearest neighboring vectors and determining their majority class. If majority of the neighbor vectors share the same class, this class is then assigned to the input vector as well, as illustrated in fig. 1.16. In the opposite case a different distance function may be selected and a new set of neighbors is found, or some specifying technique is used, for example weighted voting. Weighted voting takes into consideration the actual distances between the input vector and its neighbors, which helps determine the closest of the neighbors and also its class.
Changing the parameter K may have impact on the result as seen on fig. 1.16 where Class 2 is the majority class for $K = 3$ but for $K = 5$ it is Class 1.

From the previous text is clear that unlike pretrained models, for example artificial neural nets (1.3.1), the KNN algorithm has no fixed inner structure and it needs a classified set of data to operate on.

Advantages

An advantage of KNN algorithm is its simplicity as it basically only states that an element is defined by its surroundings. It is then upon consideration, if provided data are likely to fulfill this rule or not. In a positive case (e.g. a highly clustered data), this method will yield fairly accurate results. Its generality also allows for the basic idea to be widely modified to fit a variety of tasks.

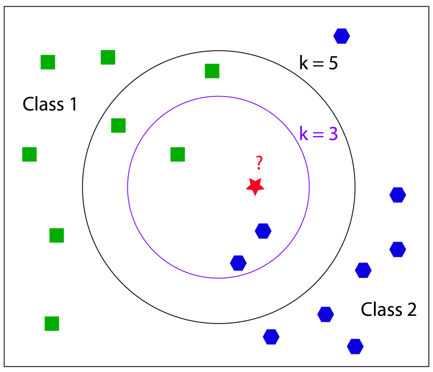


Figure 1.16: An illustration of KNN algorithm

Finally it is very easy to implement in its basic form and provides a good introduction to machine learning. This was one of the reasons I have chosen to incorporate its implementation to the prototype. It also acts as a counterpart to other selected solutions which are described in 1.3.3 since, my implementation of KNN as a classification model, is fundamentally different from them.

Disadvantages

The algorithm is rather useful in other disciplines of machine learning using rather low-dimensional vector spaces, for example some areas data mining. As stated it is very simple and does not define which input features to compare and how to extract them, that is an issue of individual implementations.
Another already mentioned disadvantage is that this model needs to access classified data, stored either locally or remotely. Either way at some point of the process the data must be loaded to memory, which in consequence means that the amount of time needed for the computation is dependent on the size of the dataset and speed of the storage medium it is saved on (providing no optimization techniques are used).
Tested implementation of this algorithm is closely described in section 2.5.

1.3.3 On line services

There are many commercial on line services performing various machine learning tasks including image analysis. They differ from each other in minor details, but all of them are developed by large research teams and fine-tuned to achieve as high precision as possible. By using state-of-art technologies, these services provide the most accurate solutions for tasks requiring machine learning algorithms and certainly represent the best choice for any business

case.
Following is the list of image labeling services used in this project:

- Google Vision[16] - provided by the Google Cloud Platform
- Watson Visual Recognition[17] - a part of IBM's cognitive system
- Microsoft Computer Vision[18] - Microsoft's computer vision service
- Clarifai[19] - an AI company focused on visual recognition and related machine learning services
- Imagga[20] - a company providing image operations requiring AI (tagging, auto-cropping, colorization etc.)

All of these services use computational models based on deep neural nets and deep learning which is currently the most successful approach in the area of pattern recognition and yields very accurate results.

The services offer registration of paid accounts as well as free trial accounts. Trial accounts restrict usage of their service with a limited number of requests a day and set period of the account validity. The most strict of them provides a maximum of one hundred requests a day, which is sufficient for the purpose of demonstration of the project. Also, an expired validity of the trial account will result in malfunction of provided service and therefore classification of images in the prototype application. For these reasons I recommend creating paid accounts if increased or prolonged usage of the services is required.

1.4 Tools and technologies

Technologies used in creation of the project's practical part are described in this section.

Python

Python 2.7 scripting language was chosen for implementation of the core algorithms and for communication with the on line image classifiers. Python API libraries provided by the respective on line services are used to communicate with their servers, sending requests and receiving responses. Responses are encoded in JSON[2] file format and their decoding is performed using functions from built-in Python library *json*[21].
The implementation of KNN algorithm uses *numpy* library[22] providing various mathematical operations on arrays and matrices. Lastly, *sys* library[23] was used to process command line arguments.

[2]JavaScript object notation

An authorization of on line image recognition services is done by their API developer keys, identification strings, which are included in the code in order to make it portable and user independent. This also allows a simple change of the API keys, should any of the used service's account be replaced.

Java

JavaSE [3] is used in the application for managing user inputs, loading data, executing the scripts and processing results. The presentation layer is created with JavaFX (a part of JavaSE 8 SDK[4]) which is a set of Java libraries providing a platform independent GUI development. I have chosen JavaFX over older alternatives for the sake of effectiveness, modern graphical appearance and convenient handling.
My initial intention was to encode and save generated CAPTCHA challenges in HTML[5] files. However this file format turned out to be unfitting since the HTML files are only displayable in JavaFX by loading them to a `WebView` container, which is purposed to display a real-world Internet webpages. Displaying the HTML files by this container's methods seemed to be very slow, so I decided to use `ImageView` instead, a JavaFX image container fully optimized for these purposes.

Other tools

The whole project was developed in NetBeans IDE[6] and versioned with Git VCS[7]. Its current version is stored in GitHub repository.

[3]Java Standart Edition
[4]Software Development Kit
[5]HyperText Markup Language
[6]Integrated development environment
[7]Version control system

Chapter 2

Implementation

This chapter mentions steps taken in creation of the practical part of this thesis such as collecting sample data and defining image classes for the challenge. It then explains the project's modular structure along with responsibilities of each component and also the project's extendability options in latter sections. Last section is dedicated to description of my custom implementation of image classifier based on previously introduced KNN algorithm.

2.1 Obtaining input data

The initial intention was to gather real world CAPTCHA challenges and run selected scripts on them. This turned out to be difficult to accomplish because firstly there is no truly reliable source of challenges, secondly the challenges could not be acquired by automation easily and lastly the modern NoCaptcha system (mentioned in 1.1.1.3) greatly complicates the automation as well. Therefore we decided to generate our own custom image-based challenges of predefined classes. The process is described in detail in 1.

In order to reach a certain level of quality and variance of our custom challenges, we had to collect a large dataset of sample images. As a primary source of categorized images I selected the ImageNet[24] on line database. Downloading of the images was automated by a custom Python script which can be found on the attached CD and its code is shown in the end of this section.
Every category of images in the ImageNet database carries is assigned a unique identification string called *wnid*. The website, upon request, displays a list of URLs[8] of all images from a certain category specified by its *wnid* in the request. I used this functionality to automate the downloading and sorting the images to appropriate folders.

[8]Uniform Resource Locater

Over 10000 images of ten classes were downloaded and stored in the dataset using this process. It can be, however, extended at will as described in 2.4.1.

The download script:

```python
import urllib
import urllib2

count = 0

#an html page displaying URLs of images from category defined by
    wnid
root = "http://www.image-net.org/api/text/imagenet.synset.geturls?
    wnid="

#a wnid string representing category "tree"
wnid = "n11621547"

#specifies name of the folder in the dataset to saved images to
imageclass = "tree"

strimageclassupper = imageclass.upper()
data=urllib2.urlopen(root + wnid)

#every line in list of URLs represents one image
for line in data:
  print "Downloading: " + str(line)
  try:
    urllib2.urlopen(line)
    filename = imageclass + "_" + str(count)+ ".jpg"
    urllib.urlretrieve(line, "D:\BAP\DATASET\\" +
    strimageclassupper + "\\" + filename )
  except:
    print ("Exception occured")
    count=count+1
    continue
  else:
    print "Saved to: " + str("D:\BAP\DATASET\\" +
    strimageclassupper + "\\" + filename)
    count=count+1
```

2.2 Definition of image classes

As seen in figures 1.3 and 1.4, CAPTCHA challenges of both types have a task of a certain class. We therefore decided to define a set of image classes used for generation of our custom challenges. At first we selected ten image classes to be used in the prototype. This idea had to be reconsidered though, when minor complication was introduced by combining outputs of different on line classifiers. Each of the classifiers return a different set of labels for the same input image. The hierarchy of these labels also differ, some classifiers

tend to return more general labels with high scores whilst others are more concrete possibly sacrificing some score points. The goal is not to underscore a classifier for returning labels which may be correct but were not defined in the program. This requirement lead to an idea of introducing synonymous classes for already defined classes.
The current set of image classes with their respective synonyms is as follows:

- CAT
- DOG
- HORSE
- BIRD
- BUILDING : HOUSE
- CAR
- BOAT : SHIP
- FACE : PORTRAIT, PERSON
- TREE
- FLOWER

For example, if two classifiers are given an image of a building, one may assign it with a label *house* while the other one with a label *building*. Both answers are viewed as correct.
This set of image classes and their synonyms is not fixed and can be edited as described in section 2.4.1.

2.3 Project structure

The application provides a certain level of modularity by being divided in three mutually independent modules. Following list summarizes these modules and their roles:

- Locally stored structured dataset of images
- Python scripts providing image classification
- Java application handling user interactions and executions of scripts.

The dataset is a locally stored directory containing sample images that are organized in subdirectories. Every subdirectory contains all images of a specific class which are used in a generation of challenges. This process is explained in 1.

A configuration text file containing a list of all image classes used in the program is located in the root folder of the dataset. The structure rules of both the dataset and the configuration file are described in detail in 2.4.1.

The Python scripts are executed from the Java application and provide image classification by an arbitrary computational model. The image is passed to the script by its absolute path and a list of labels with their respective scores is then collected from the standard output of the script. More information about the mandatory structure of the scripts are listed in 2.4.2.

Operations performed by the Java application itself as well as its design and graphical interface are subjects of the next chapter.

2.4 Extendability

The application is prepared for extension of its capabilities in a number of ways. Both the dataset of sample images and the set of image classes with their synonyms may be extended if certain rules are followed. In this section I discuss these rules for expansion in two separated subsections.

2.4.1 Extending the inputs

All image classes and their respective synonyms are defined in a resource file which is located in the root of *DATASET/* directory. Following lists define rules for the resource file and the dataset.

Structure of the image class resource file:

- Name of the resource file must match the string constant `CONFIG_IMAGE_CLASSES` in the project's *src/resources/Constants.java* file. A value of the constant is editable.
- Every image class name is defined by exactly one word
- Every line contains a name of an image class. A list of its synonymous classes separated by commas may follow and should be separated from the image class by a colon. A symbolic notation of the format is:

  ```
  {name of the class}[:synonym1[,synonym2[,synonym3]...]]
  ```

Structure of the dataset:

- Every class of images has its own folder in the *DATASET/* directory, named after the class of images it contains (e.g. *horse/*). Letter case is arbitrary as well as the file format of sample images and their naming conventions are arbitrary.

- Creating a dedicated folder with at least one sample image for every image class in the resource file is mandatory and can result in unstable behavior if disregarded.

2.4.2 Extending the outputs

If addition of a new solving method is required, one can do this by providing the solver script written in Python scripting language, saving it to the */src/scripts* folder of the project and creating its respective Java wrapper class in */src/utility/solvers* package. An instance of this wrapper class must be then added to the array of available solvers in Java class of whichever challenge it solves.

The script must obey following rules:

- The script outputs several tags (labels) and their respective scores for a given image in this format:

  ```
  {label}:{score}
  ```

- Each label and score pair is printed on a single line with no upper limit for number of the lines printed

- The script uses standard output to print its output

- The score of the label must be a decimal number in a range $(0, 1)$

Note that these rules are required by a general implementation of `solve` method performing script execution. This method is a member of `Solver` abstract Java class and can be overriden in the application's code to perform the script execution differently. My implementation of KNN algorithm, for instance, overrides this method in order to take more command line parameters. Refer to program documentation located on the attached CD for more details on implementation.

2.5 Custom image classifier

In this project I tried to implement KNN algorithm to provide an image classification of real-world images. Unlike models with inner structure, like mentioned neural nets, which can extract input features automatically by training, I had to define input features manually in order to be able to compare feature vectors. This is a difficult task with no simple yet satisfactory solution, which was the reason I have chosen to represent a feature vector of an input image as a matrix of its pixel values in all three RGB[9] color channels. Even though raw pixel values of an image presumably bare little connection to actual objects in the image, I wanted to present a different approach to image recognition and examine its precision on various categories of inputs.
This approach should be hypothetically fairly accurate on category of images that share similar color distribution. For instance photos of boats and planes often include a dominating blue color (of a sky or a sea). That means a new image of a plane in the sky is very likely to be classified as one of these two classes. Currently an application of this algorithm in the prototype is restricted to just those CAPTCHA challenges specified by a reference image in order to show the program's ability of containing different sets of solvers for different challenge types. This restriction is not fixed and may be removed with a small edition of a source code. The whole procedure is described in chapter 2.4 mentioning all other options of extendability of the program.

Implementation specifics

To measure distances between feature matrices, I included two commonly used functions, Euclidean distance (2.1) and Manhattan distance (2.2).

$$d_e(V_1, V_2) = \sqrt{(V_1 - V_2)^2} \tag{2.1}$$

$$d_m(V_1, V_2) = |V_1 - V_2| \tag{2.2}$$

An observation revealed a bottleneck of this implementation being the speed of storage medium of the images. Every image must be loaded to memory at some point of the process which drastically slows down the whole procedure. I tried to optimize this issue in two ways. Firstly, for the sake of space and speed efficiency the input images are scaled down in run time. Secondly, the loading of the dataset images is done just once as it stores feature matrices of all images in a binary file. Every other execution of the implementation script checks whether this file exists, and in a positive case proceeds directly to comparing the feature matrices.
The prototype is currently unable to detect changes in the dataset or in the image classes resource file as it would require creating a database to keep track

[9]Red, Green, Blue

of these information. This means the binary file is valid until for a certain state of dataset and image class resource file. If any of those are edited by external action, the binary file should be removed manually.

The scaling factor and the distance function, as well as the parameter K, can all be adjusted by user in the application dialog before execution of the KNN classifier script. Refer to subsection 3.3.3 for more information on parameter setting.

Chapter 3

Software design

This chapter introduces all features of the application. Functional features provide the application's functionality, usability and outputs and are a subject of first section. Nonfunctional features define its user interface and its connection to mentioned functionality and are described in second part of the chapter.

3.1 Functional features

This section outlines features that define the functionality of the application in an enumeration list. Every element of the list mentions one key functionality.

1. **CAPTCHA challenge generation**
   As mentioned before, this project focuses on two types of image challenges - specified by keyword and specified by image. Both types are generated quite similarly, either a keyword or a reference image is selected and assigned to the task of the challenge. Then 4 to 6 random images of the class in task are loaded and put on random coordinates in the payload grid. These are further referred to as the correct images.
   Rest of the grid is then filled with images of other random classes. Provided that the size of the dataset is large enough, this mechanism ensures uniqueness of every generated challenge and therefore variety of obtained results. This functionality is controlled from the challenge edition window of the application described in 3.3.2.

2. **Selection of solvers**
   Both types of CAPTCHA present their sets of available solvers for selection to the user. These sets may be extended by new solvers, as explained in 2.4.2. Some solvers support passing additional parameters. More information on this functionality is given in 3.3.3, which also describes a controller window of this functionality.

3. **Results comparison**
   All selected classifiers exhibit their results when the classifications are finished. An accuracy of every result is rated in percent by following formula:

$$accuracy = |SELECTED \cap CORRECT| \div |CORRECT|,$$

   where

   SELECTED = set of the challenge images selected by a solver as equivalent to the challenge task

   CORRECT = set of the actually correct images in the challenge.

A dedicated window responsible for presenting the results is described in 3.3.4.

3.2 Nonfunctional features

A review of the application's nonfunctional properties, in particular a graphical user interface, design and hierarchy of Java classes is offered in this section. It also characterizes the main Java classes and explain their functionality in the code.

3.3 GUI

The application consists of four main windows. Their respective descriptions are found in the following subsections. Each window contains buttons allowing navigation back and forth through the application and is dedicated to some functionality described in 3.1.

3.3.1 Challenge type selection window

The two selected variants of the image CAPTCHA are displayed in this window. The user selects one of them by clicking the respective radio button. There is also a possibility to specify and fix the keyword of the challenge or the class of the reference image explicitly by typing it to the newly appeared text box bellow the radio button. The typed string specifying the class must match one of the defined image classes (not synonyms), in arbitrary letter case. If a nonexistent class is specified it is ignored. Fig. 3.1 shows this window.

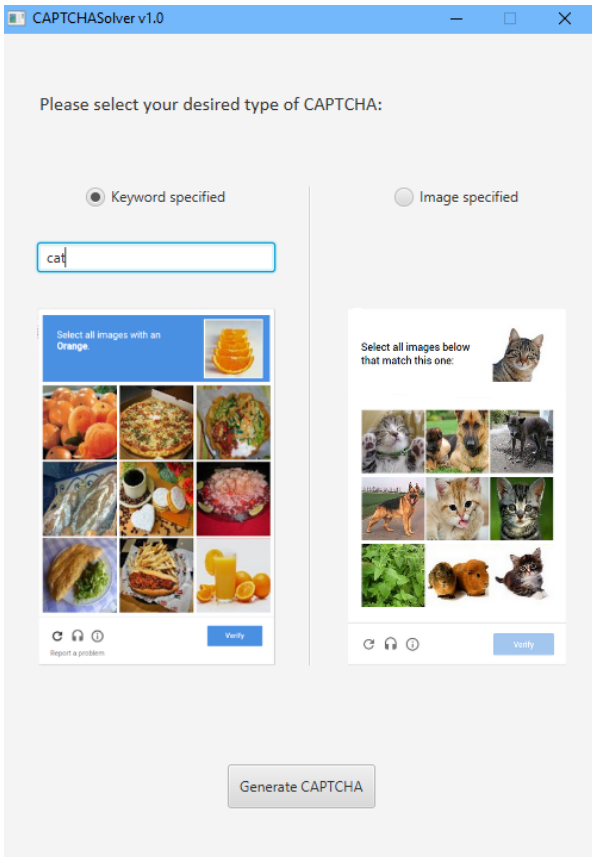

CAPTCHASolver v1.0
Please select your desired type of CAPTCHA:
Keyword specified
Image specified
cat
Select all images with an Orange.
Select all images below that match this one:
Verify
Report a problem
Verify
Generate CAPTCHA

Figure 3.1: Challenge type selection window

3.3.2 Challenge edition window

The user is able to edit the task by clicking the button in upper right corner of the window. The button is dynamically generated depending on user's actions in previous window. A completely new challenge is generated if he did not specify and fix the challenge class earlier, otherwise only a new payload is generated. In both cases the newly generated payload is completely independent from the old one, it has a new random number of correct images with new positioning. The randomness is, of coarse, determined by Java's pseudorandom number generator and the size of a dataset. Following figures shows edition windows of both challenge types. Note the different button labels in upper right corner indicating its functionality according to fixation of the class.

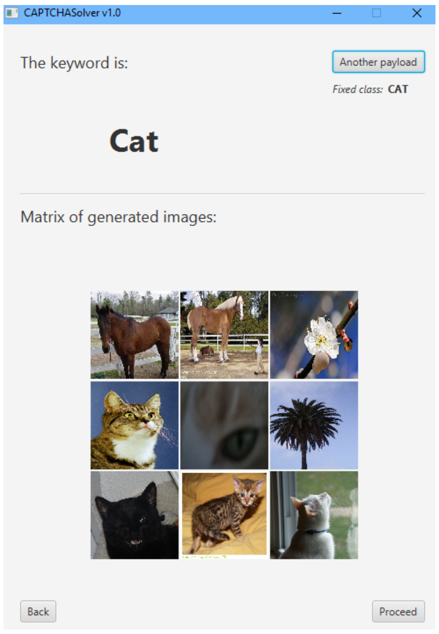


Figure 3.2: Keyword-specified challenge edition window

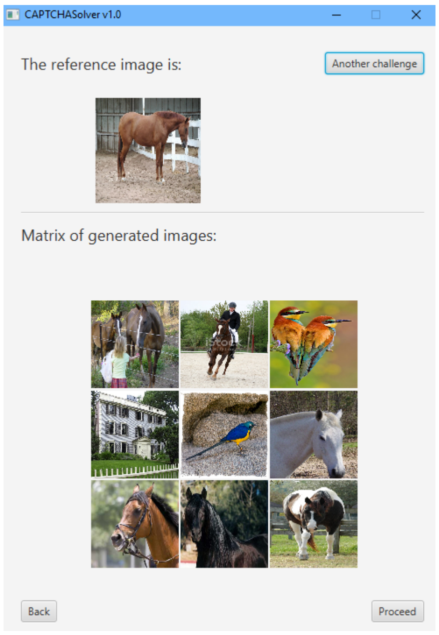
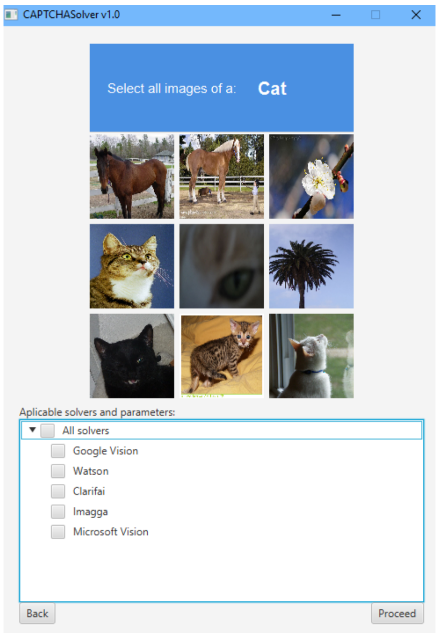


Figure 3.3: Reference image-specified challenge edition window

3.3.3 Solvers window

Every challenge holds its set of applicable solvers which is now presented to the user in a check box tree view hierarchy. Figures 3.4 and 3.5 show the difference between the two types of challenges and their available solvers.
After clicking the Proceed button a computation is initiated and a dialog window appears showing a time estimation and a progress bar. When the process is done the results are displayed automatically in the results window.
One should bare in mind that selecting many solvers results in a long computation time, user is therefore presented with a dialog informing about the progress to avoid it being mistaken for a computer lag. There is also a possibility to stop the computation anytime with a button. In that case the application cancels the computation and returns to the selection of solvers.

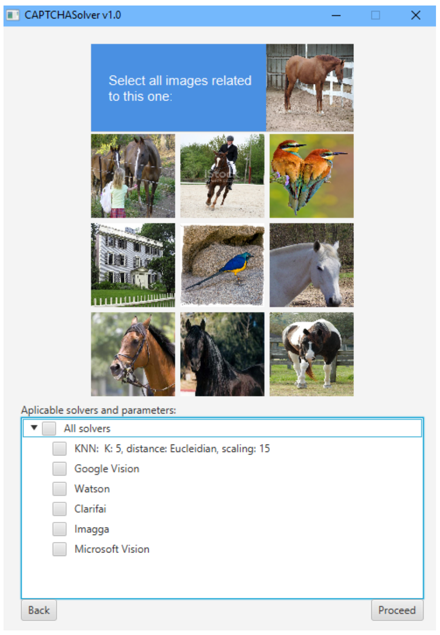

CAPTCHASolver v1.0

Select all images of a: Cat

Aplicable solvers and parameters:

- All solvers
  - Google Vision
  - Watson
  - Clarifai
  - Imagga
  - Microsoft Vision

Figure 3.4: Keyword-specified solvers window


CAPTCHASolver v1.0

Select all images related to this one:

Aplicable solvers and parameters:

- All solvers
  - KNN: K: 5, distance: Eucleidian, scaling: 15
  - Google Vision
  - Watson
  - Clarifai
  - Imagga
  - Microsoft Vision

Figure 3.5: Reference image-specified solvers window

If a solver supports additional parameters, they may be edited by righ-clicking the row of the solver in the list. This action opens a dialog window allowing setting values to offered parameters as seen in fig. 3.6.
This functionality rather serves as a demonstration though and is currently only available for my custom implementation of KNN classifier.

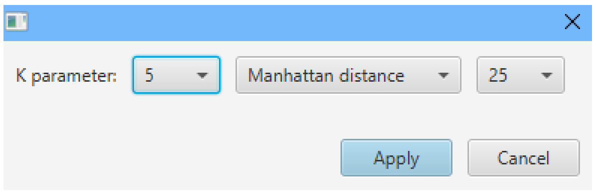

K parameter: 5 Manhattan distance 25

Figure 3.6: Dialog allowing parameters setting for included KNN classifier

3.3.4 Results window

Results are graphically displayed in a scrollable window along with their respective information and an accuracy mark. This may serve as a good comparison of different models or as a benchmark for newly added one.

Besides that, every item shows the generated CAPTCHA with a graphical interpretation of the result. Images highlighted by a green filter are those selected by the solver as equivalent to the challenge task. Images in green frames represent actual correct solution of the challenge. Figure 3.7 shows an example of a populated result window.

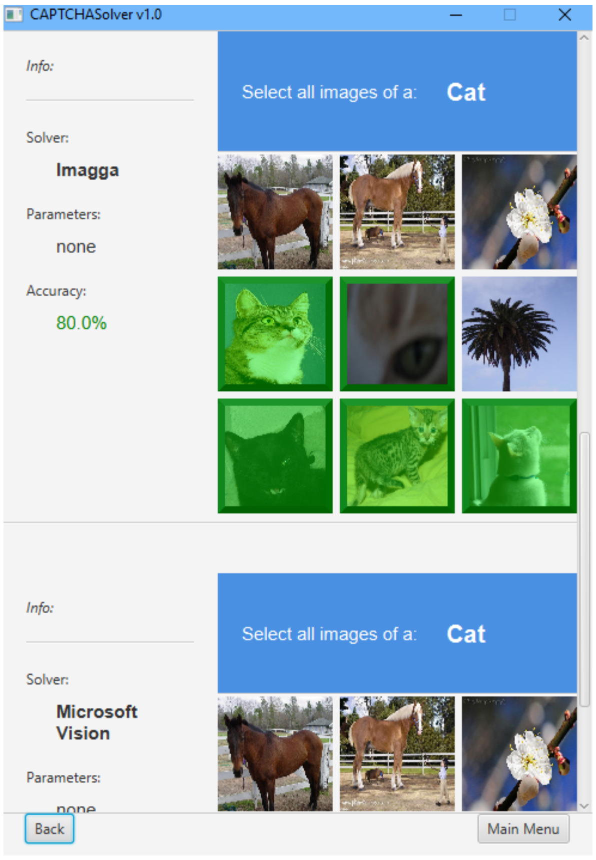


Figure 3.7: Populated result window of a solved keyword-specified challenge

Evaluation of accuracy of challenges specified by a reference image is tightened by an additional condition. If the solver classifies the reference image in the challenge incorrectly, its accuracy is automatically set to 0% and the program proceeds to next solver. Incorrectly classified reference image is then highlighted with red filter in the results window.

Chapter 4

Experiments

Selected classifiers were tested in iterations on challenges of all image classes in order to determine which performs best and would be therefore suitable for breaking real-world challenges. Results of this experiment are carried out in this chapter.

4.1 Testing process

Every classifier was executed in ten iterations for each of currently defined image classes. This process was applied only on challenges specified by a reference image. The reason for this is explained later in the chapter. An arithmetical averages of their individual rounded accuracies in percents for both challenge types combined were then calculated. Results are presented in following table (names of services are written in shortened form for space efficiency).

The results showed that the best average result across all image classes was achieved simultaneously by Google's and Clarifai's models which both reached approximately 93% precision.

The KNN classifier is in the current version of the prototype only applicable on challenges specified by a reference image. This restriction was created on purpose for reasons explained in 2.5. Tests of all other classifiers were performed only against this type as well, because otherwise those classifiers would have a clear advantage. Failing to classify a reference image correctly is followed by setting 0% accuracy mark to the solver and canceling its further solving process. This mechanism was already mentioned in 3.3.4.
Slightly better results were achieved using setting the parameter K to values higher than 20. The improvement was not significant enough to be mentioned in more detail though. Applying different distance function to the measurements did not result in a notable improvement either.

	Google	Watson	Microsoft	Clarifai	Imagga	KNN
CAT	97	91	86	96	92	2
CAR	89	83	79	90	85	3
DOG	96	93	85	96	93	2
BIRD	95	89	84	94	92	5
BOAT	93	92	82	92	89	6
BUILDING	90	84	80	91	87	0
FACE	92	85	78	92	89	0
FLOWER	94	85	80	95	93	4
HORSE	93	83	84	93	92	3
TREE	87	86	82	90	90	7
overall precision	**93**	87	82	**93**	90	18

Table 4.1: Precisions achieved solving keyword-specified challenges

4.2 Evaluations

This experiment proved, according to initial hypothesis, that an image classifying method based on the image's color features is not suitable for image recognition and unusable in practice. The poor results of the KNN classifier clearly imply that extraction of the defining features from inputs is one of the key factors determining accuracy of the method. It, however, served as a counterpart to the other methods and provided some insight to basics of machine learning.
The results received upon testing are slightly biased by simplified testing conditions, since the image classes defined in this project are more easily recognizable than those currently used in real-world CAPTCHA challenges. In practice, real challenges are intentionally designed to contain classes of images that are difficult to recognize by even the most advanced models of today.

Conclusion

The goal of this theses was to point out weaknesses of image-based CAPTCHA system by breaking them with algorithms of artificial intelligence. A research of the topic was provided and possible solutions were examined. The most successful algorithms, revealed in an experiment, may pose a thread for current image-based challenges. A desktop application, designed as a framework allowing addition and comparison of image recognition algorithms, was successfully developed and tested. Also, for purposes of generation of CAPTCHA challenges, a large dataset of images was collected and incorporated to the project in a way allowing its further expansion. A creative process of this thesis also contributed to author's knowledge in areas of machine learning which will become useful in later studies.

Future work

I envision this project to become a benchmark for testing various image classifiers against currently used types of CAPTCHA challenges. It provides a foundation which might be further extended, possibly in a master's thesis, to encompass future types of CAPTCHA.

Bibliography

[1] Google Street View. [online]. Available from: https://www.google.com/streetview/

[2] Vinay, S.: Street View and reCAPTCHA technology just got smarter. [online], April 2014, [cited 2017-05-06]. Available from: https://security.googleblog.com/2014/04/street-view-and-recaptcha-technology.html

[3] Google NoCaptcha ReCaptcha demo. [online]. Available from: https://www.google.com/recaptcha/api2/demo

[4] Google Invisible ReCaptcha. [online]. Available from: https://www.google.com/recaptcha/intro/comingsoon/invisible.html

[5] Jarušek, P.: Rozpoznávání znaků v CAPTCHA. Praha, 2016. Bakalářská práce. Fakulta informačních technologií, České vysoké učení technické v Praze.

[6] Googin D.: How a trio of hackers brought Google's reCAPTCHA to its knees. [online], May 2012, [cited 2017-05-06]. Available from: https://arstechnica.com/security/2012/05/google-recaptcha-brought-to-its-knees/

[7] Debasish M.: Attacking Audio "reCaptcha" using Google's Web Speech API. [online], April 2014, [cited 2017-05-07]. Available from: http://www.debasish.in/2014/04/attacking-audio-recaptcha-using-googles.html

[8] Captcha alternatives and thoughts. [online], December 2015, [cited 2017-5-06]. Available from: https://www.w3.org/WAI/GL/wiki/Captcha_Alternatives_and_thoughts

[9] Black, A., Kokorin, V.: Distributed Deep Learning, Part 1: An Introduction to Distributed Training of Neural Networks. [online], October 2016, [cited 2017-05-06]. Available from: `http://engineering.skymind.io/distributed-deep-learning-part-1-an-introduction-to-distributed-training-of-neural-networks`

[10] Mittal, G.: Convolutional Neural Networks (CNN). [online presentation], slide number 44. November 2015, [cited 2017-05-06]. Available from: `https://www.slideshare.net/GauravMittal68/convolutional-neural-networks-cnn`

[11] Deshpande A.: The 9 Deep Learning Papers You Need To Know About (Understanding CNNs Part 3). [online], August 2016, [cited 2017-05-06]. Available from: `https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html`

[12] Results of ILSVRC2014. [online], 2014, [cited 2017-05-06]. Available from: `http://image-net.org/challenges/LSVRC/2014/results`

[13] Szegedy Ch. et al.: Going deeper with convolutions. [online], 2015, [cited 2017-05-06]. Available from: `https://research.google.com/pubs/pub43022.html`

[14] DeepDream. [online]. Available from: `https://deepdreamgenerator.com/`

[15] Øygard A.: Visualizing GoogLeNet Classes. [online], July 2015, [cited 2017-05-06]. Available from: `https://www.auduno.com/2015/07/29/visualizing-googlenet-classes/`

[16] Google cloud vision API. [online]. Available from: `https://cloud.google.com/vision/`

[17] Watson visual recognition API. [online]. Available from: `https://www.ibm.com/watson/developercloud/visual-recognition.html`

[18] Microsoft computer vision API. [online]. Available from: `https://azure.microsoft.com/cs-cz/services/cognitive-services/computer-vision/`

[19] Clarifai image and video recognition API. [online]. Available from: `https://clarifai.com/`

[20] Imagga image recognition service. [online]. Available from: `https://imagga.com/`

[21] json library documentation. [online]. Available from: https://docs.python.org/2/library/json.html

[22] numpy library documentation. [online]. Available from: https://docs.scipy.org/doc/numpy/reference/index.html

[23] sys library documentation. [online]. Available from: https://docs.python.org/2/library/sys.html

[24] ImageNet. [online]. Available from: https://www.image-net.org/

Appendix A

Acronyms

API	Application programming interface
CAPTCHA	Completely automated public Turing test to tell computers and humans apart
GUI	Graphical user interface
GPU	Graphics processing unit
KNN	K nearest neighbors
ANN	Artificial neural network
CNN	Convolutional neural network
DBN	Deep belief network
RBM	Restricted Boltzmann machine
ReLU	Rectified linear units
JSON	JavaScript object notation
URL	Uniform resource locater

Appendix **B**

Contents of enclosed CD

```
readme.txt ........................ the file with CD contents description
javadoc ......................... documentation of the application code
DATASET .................................... the directory of a dataset
src ................................. the directory of all source codes
    project .................... the directory of application source codes
    script ........................... the directory of helper scripts
    thesis .............. the directory of LaTeX source codes of the thesis
text ........................................ the thesis text directory
    thesis.pdf ........................ the thesis text in PDF format
```