

Bachelor's thesis

Information and Communications Technology

2019

Jesse Kiviniitty

# TEXT-INDEPENDENT OFFLINE WRITER IDENTIFICATION USING CONVOLUTIONAL NEURAL NETWORKS



BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information and Communications Technology

2019 | 54 pages

Jesse Kiviniitty

# TEXT-INDEPENDENT OFFLINE WRITER IDENTIFICATION USING CONVOLUTIONAL NEURAL NETWORK

As technology and hardware capabilities improve, the use of machine learning increases in every industry from stock market analysis to image processing. Also, the amount of data is exponentially increasing and forces to develop machine learning solutions to create value from that data. Offline writer identification is a machine learning solution to recognize a writer based on scanned handwritten text.

The main goal of this thesis was to research and implement a convolutional neural network model for identifying writers based on the characteristics of their writing styles. The model is built using the neural network library Keras and TensorFlow as backend.

During the project, many different writer identification techniques were researched, and a working offline writer identification model was developed. The Google Cloud computing platform was used for the project model training.

The project resulted in a model which can reliably identify a writer based on handwritten sentences. The possible next steps in the project is to implement an automated method of training and identifying additional writers.

## KEYWORDS:

writer identification, convolution neural network, machine learning, TensorFlow

Jesse Kiviniitty

## SISÄLLÖSTÄ RIIPPUMATON JA AVUSTAMATON KIRJOITTAJAN TUNNISTAMINEN KONVOLUUTIO- NEUROVERKKOJEN AVULLA

Teknologian ja laitteistojen suorituskyvyn noustessa koneoppimisen hyödyntäminen lisääntyy kaikilla aloilla pörssikursseista kuvien analysoimiseen. Myös datan määrä on jatkuvasti räjähdysmäisessä kasvussa, ja datan jalostaminen hyödylliseksi pakottaa kehittämään arvoa luovia koneoppimisen ratkaisuja. Avustamaton kirjoittajan tunnistaminen on koneoppimisen ratkaisu kirjoittajan tunnistamiseen skannatun käsialanäytteen avulla.

Opinnäytetyön tavoitteena oli tutkia ja toteuttaa konvoluutioneuroverkkomalli, joka tunnistaa kirjoitustyylin piirteiden perusteella kirjoittajan. Malli on rakennettu käyttäen Keras-neuroverkkojen koodikirjastoa ja taustajärjestelmänä TensorFlow-koneoppimishjelmistoa.

Opinnäytetyön aikana tutkittiin useaa erilaista kirjoittajan tunnistamismenetelmää ja kehitettiin toimiva avustamaton kirjoittajan tunnistamismalli. Projektin mallin opettamiseen käytettiin IAM-tietokantaa ja paikallisen laitteiston sijaan Google Cloud -ohjelmistoalustaa.

Opinnäytetyön tuloksena valmistui malli, joka pystyy luotettavasti tunnistamaan kirjoitustyylin piirteiden perusteella kirjoittajan. Mahdollinen seuraava askel projektissa on laajentaa olemassa olevaa mallia ja toteuttaa automaattinen tapa opettaa ja tunnistaa uusia kirjoittajia.

### ASIASANAT:

kirjoittajan tunnistaminen, konvoluutioneuroverkko, koneoppiminen, TensorFlow

# CONTENTS

<b>1 INTRODUCTION</b>	<b>6</b>
<b>2 MACHINE LEARNING</b>	<b>8</b>
2.1 Overview of Machine Learning	8
2.2 Learning process	9
<b>3 ARTIFICIAL NEURAL NETWORKS</b>	<b>12</b>
3.1 Overview of Artificial Neural Networks	12
3.2 Artificial Neuron	13
3.3 Activation Functions	15
3.4 Artificial Neural Networks	17
3.5 Backpropagation	20
<b>4 IMAGE RECOGNITION</b>	<b>24</b>
4.1 Overview of Image Recognition	24
4.2 Convolutional Neural Networks	27
4.2.1 Convolution layer	27
4.2.2 Pooling layer	30
4.2.3 Overfitting	31
<b>5 WRITER IDENTIFICATION PROJECT</b>	<b>32</b>
5.1 Overview of writer identification	32
5.2 Writer identification project tools	34
5.2.1 Data set	34
5.2.2 Software	35
5.2.3 Hardware	35
5.3 Research of Writer identification models	36
<b>6 DEVELOPMENT</b>	<b>39</b>
6.1 Preprocessing	39
6.1.1 Manipulating the data set	39
6.1.2 Splitting data set	40
6.1.3 Cropping the image samples	41
6.2 Generator function	42
6.3 Convolution neural network model	43

6.4 Training	44
6.5 Summary	45
<b>7 CONCLUSION</b>	<b>46</b>
<b>REFERENCES</b>	<b>48</b>

## APPENDICES

Appendix 1. Convolutional Neural Network Design of the Project

### FIGURES

Figure 1. Machine learning process [7].	9
Figure 2. A Basic Artificial Neuron [49].	14
Figure 3. Plot of Rectified Linear Unit [55].	16
Figure 4. Linear classification separating two species [32].	18
Figure 5. Small feedforward neural network with three layers and an input layer. [25]	19
Figure 6. A two-layer ANN with example weights and node values [26].	21
Figure 7. Gradient descent visualized [45].	22
Figure 8. 3x3 matrix resized into a 1x9 vector [28].	25
Figure 9. CNN architecture detecting a car [52].	30
Figure 10. Downsampling operation [52].	31
Figure 11. H.T. Nguyen et al.'s CNN architecture for extracting samples from a Japanese character [44].	38
Figure 12. Xing et al.'s CNN architecture for writer identification [2].	38
Figure 13. Splitting the data sets [54].	40

### EQUATIONS

Equation 1. Activation Function Equation.	15
Equation 2. Approximation function of an artificial neural network. [51]	17

### PICTURES

Picture 1. Hardware implementation of the first perceptron called 'Mark I Perceptron' 1958 [14].	13
Picture 2. Sample digits of the MNIST database [29].	26

Picture 3. Convolution operation [31].	28
Picture 4. A set of edge detectors applied on MNIST data set.	29
Picture 5. A sample of IAM database data sets [36].	34
Picture 6. Couple HWDB Chinese characters [2].	37
Picture 7. IAM data set sample sentence [36].	41
Picture 8. Patches cropped from sample sentence.	42

*Machine intelligence is the last invention*

*That humanity will ever need to make*

*Nicklas Boström*

## LIST OF ABBREVIATIONS

Abbreviation	Explanation of abbreviation
ANN	Artificial Neural Network
AN	Artificial Neuron
CNN	Convolutional Neural Network
DNN	Deep Neural Network
ReLU	Rectified Linear Unit
MNIST	Modified National Institute of Standards and Technology
CPU	Central Processing Unit
GPU	Graphics Processing Unit
GPGPU	General-Purpose computing on Graphics Processing Unit
GDDR6	Graphics Double Data Rate type Six
API	Application Programming Interface
NFC	Near-Field Communication
GCP	Google Cloud Platform
GB	Gigabyte

# 1 INTRODUCTION

Machine learning is a thriving field of data analysis and its implementations are exponentially increasing on every industry. New and novel solutions to problems are discovered all the time. This thesis focuses on image recognition, convolution neural network side of machine learning and how to build a model that analyzes characteristics of handwritten text and identifies the writer of the text.

Handwritten text can be categorized as a behavioral identifier in contrast to biometric identifiers such as fingerprint, iris or palm vein scanning. A Writer can be identified by capturing specific characteristics of handwriting. By analyzing the characteristics of a sample of handwriting, it can be assigned to a single writer out of a set of writers. Typical applications for this implementation are in the fields of forensic document examination and security, which without machine learning requires high level of expertise and heavy work [1].

Research in writer identification is split into two categories, online and offline identification [2]. Online writer identification requires a recording of the writing procedure where a sensor notices the movements and pressure of the pen-tip and when the pen touches paper. The recording is regarded as an unique digital representation of handwriting style [3]. Offline identification uses only scanned images of handwritten text and lacks the additional information about the writing style. Instead offline writer identification uses either text-dependent or text-independent method. Text-dependent methods require the writer to write a predefined text for identification. Text-independent methods can use any written text [4].

Deep learning-based approaches are effective handwriting feature extractors that provide accurate predictions for writer identification purposes as compared to traditional identification approaches [5].

The thesis has four parts, the first chapters (Chapter 2, 3) introduces the reader to the machine learning topic and the importance of convolution neural networks in image recognition. The following chapter (Chapter 4, 5) continues into the



image recognition and its sub-topic, writer identification. The last chapter (Chapter 6) includes the development work of the writer identification project.

## 2 MACHINE LEARNING

The chapter introduces to the themes of machine learning and is intended as a short overview of the whole topic.

### 2.1 Overview of Machine Learning

Machine learning is a subfield of artificial intelligence. The premise of machine learning is to program an algorithm that takes data as an input and uses data analyzing methods to predict an output. In other words, machine learning is about simulating intelligent behavior characterized by how humans independently learn and do things, though the machine learning implementations are not truly intelligent as they are built to solve strictly limited set of problems [6].

It is exceptionally difficult or impossible to solve even a common machine learning problem, such as recognition of handwritten numbers, without using machine learning algorithms. The algorithm would need to be explicitly programmed to consider every possible variable of the problem. Machine learning streamlines the problem solving as it eliminates the need to explicitly program every detail [6].

The world is flooded with data and it is being created rapidly every day all around the world. With large amounts of data about a topic, it is possible to explore that data in smart ways to find patterns. Machine learning algorithms examine often large amounts of data and then generate discovered patterns from it. The algorithm can also improve its performance with specific problem by training based on repetition of that problem. For example, a movie streaming service can be trained to give personalized movie recommendations by examining the watching history of the user base [7].

All implementations of machine learning have a common requirement of large amount of data specific to a problem. Depending on a problem the data can for example, be in form of pictures, text, sound or video. The types of machine

learning models and algorithms differ in their approach, the type of data they handle, and the type of problem that they are intended to solve. For each problem there are many possible solutions with a range of results. It would be best to compare the models and algorithms to different tools as some tools are better than others for solving a particular problem. Some creative machine learning applications are automatic text translation in images, handwriting generation and automatic description of image contents [8].

## 2.2 Learning process

The basic process of machine learning is generally the same. The process starts with raw data and ends up with a model generated from that data. This process is described in Figure 1.

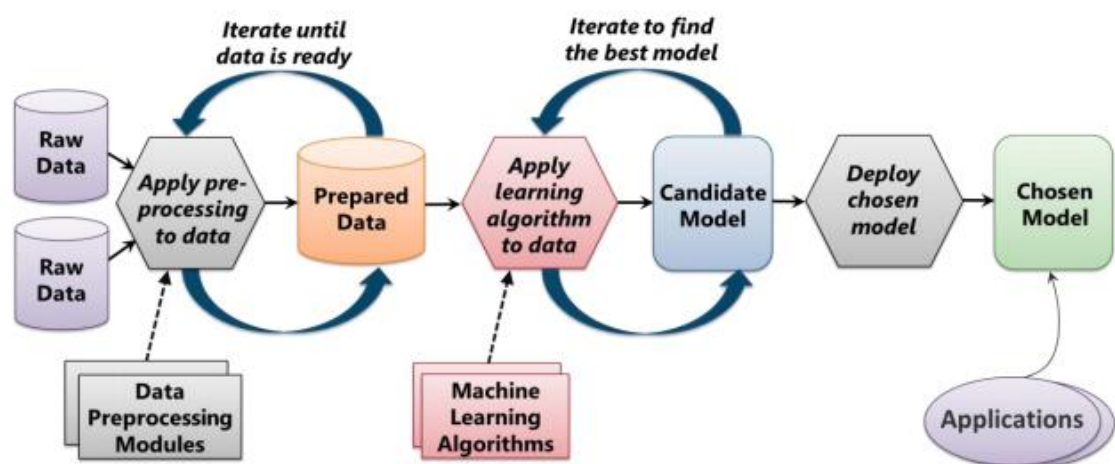


Figure 1. Machine learning process [7].

The more data is available for a project, the better the results will likely be. As we live in the era of big data and having more data available in any given field than ever before, this enables the use of machine learning on those fields. Since the process starts with raw data, it is important to preprocess and choose only the substantive data for the problem. [7]

Usually the chosen data is in a wrong format for the model to be of any use, and, therefore, the data needs preprocessing. Preprocessing data usually means removing of duplicate files or excel columns, resizing pictures or changing file types. In the case of pictures, this might also include removing inaccurate pictures from the data set or editing them, for example, by cropping or grayscaling. Sometimes the important data is difficult to separate from the unneeded data because of inconsistencies, thus preventing automatic preprocessing and slowing data preparation significantly. Preparing data is important as the machine learning algorithms require a certain format of data to work predictably and usually predictable data input leads to consistently useful output [9].

When the data has been prepared, it is time to apply learning algorithms to it. Usually algorithms perform some model of statistical analysis on the data. When choosing a machine learning algorithm, there is not a single algorithm that dominates the field. Some perform better with large data set and some with high dimensional data. The statistical models can be something basic such as linear regression or cluster analysis. For example, linear regression can be used to predict housing prices in a given area. Cluster analysis can be helpful in observing the taxonomy of species and define more accurate species groups [7].

Some quite different situations might require more complex algorithms, for instance an algorithm called random forest. Random forest can be useful in finance, where it is used to predict the behavior of stock-market prices. Experimenting with multiple algorithms is a great way to proceed in a situation in which it is not obvious which algorithm is the best for solving the problem [10].

A model is generated from the data that the algorithm is trained with. Therefore, the process is called 'training a model'. A model is a code implementation of a machine learning algorithm for recognizing a pattern in the data. The model is then applied to problems it is designed to give answers. For example, in case of this thesis' project the generated model predicts a writer's identity from the handwritten text sample. Of course, the model only applies to an already defined data set. The model gives a probability value between 0 and 1, instead of a simple 'yes' or 'no' answer. Usefulness of the probability value is estimated case by case.

Probability value of 0.8 for the writer's identity is plausible and useful, but the same value with medical application that determines patient's health, for example, MRI picture scan intended to find cancer, needs higher accuracy of probability [11].

Experimenting with different models and data preprocessing techniques is usual with any given machine learning problem. As stated earlier, the first model might not be the best suited for the problem. Then again there is not one solution or one approach that fits all. Experimenting with supposedly non-optimal models might reveal something completely unexpected that no one could predict. In addition, sometimes even a standard neural network at a first glance seems to work as intended, but when examined closely is solving an entirely different problem. For example, a neural network that is trained to recognize sheep, instead learns to recognize lush green hillsides the sheep graze on. When the most effective model is decided, the next step is applying the model to solve the problem [12].

### 3 ARTIFICIAL NEURAL NETWORKS

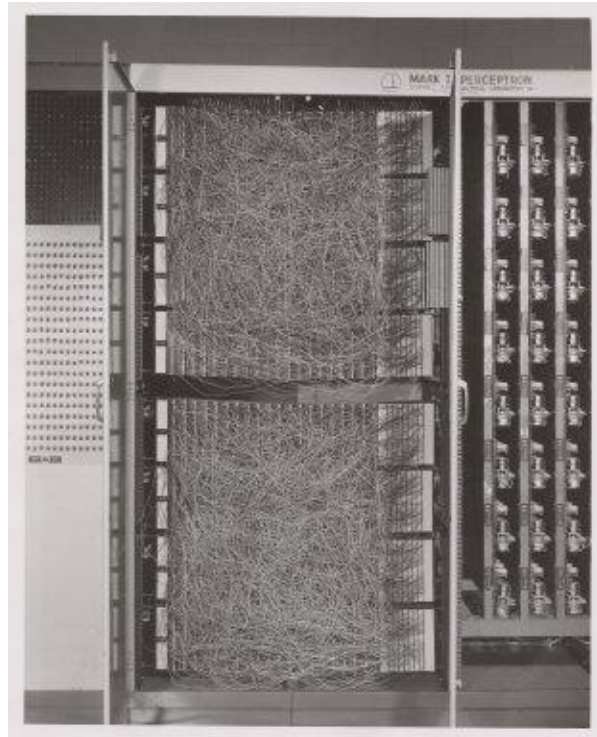
“Encoded in the large, highly evolved sensory and motor portions of the human brain is a billion years of experience about the nature of the world and how to survive in it. The deliberate process we call reasoning is, I believe, the thinnest veneer of human thought, effective only because it is supported by this much older and much more powerful, though usually unconscious, sensorimotor knowledge. We are all prodigious Olympians in perceptual and motor areas, so good that we make the difficult look easy. Abstract thought, though, is a new trick, perhaps less than 100 thousand years old. We have not yet mastered it. It is not all that intrinsically difficult; it just seems so when we do it.”

- Hans Moravec

#### 3.1 Overview of Artificial Neural Networks

Artificial neural network (ANN) design is inspired by the way human brains process information. The most fundamental unit of both brain and ANN is called a neuron. An artificial neuron (AN) takes an input, processes it and returns an output for the next layer of neurons if the input exceeds the set threshold value. The way how ANNs process information is considered as the next big step in the computing industry as it can in theory represent any mathematical functions [24].

The earliest work on a mathematical model of a single neuron was suggested by McCulloch and Pitts in 1943. They introduced the idea of using neural networks as computing machines. In 1958, the advancements in artificial neurons by Frank Rosenblatt, a research psychologist, resulted in single-layer neural network called ‘the perceptron’. His achievement was to show that artificial neurons could learn from data. This was implemented with a hardware system called ‘Mark I Perceptron’ (Picture 1) that could classify simple shapes correctly with 20x20 sized pixel input [14].



Picture 1. Hardware implementation of the first perceptron called 'Mark I Perceptron' 1958 [14].

Even though perceptron is a great model to classify certain simple patterns, as a simple piece of technology it lacks in efficiency and has many limitations. A notable obstacle for a single perceptron at the time was the 'exclusive or' or XOR problem. Still the perceptron had laid the foundations for the modern neural computing field [14].

### 3.2 Artificial Neuron

The artificial neuron is the basic building block of the neural network. Modern artificial neuron differs from the classic perceptron in a couple of additional features. Perceptron consists of multiple inputs that create the 'input layer' and a single output. Perceptron works well in the case of binary classification when a solution is simple yes-no answer.

An artificial neuron (AN) has four basic functions to simulate the likeness of a biological neuron. Compared to perceptron, AN has added weights and an

activation function. These functions together calculate if the AN should give an output and if it does, the output will be a number between 0 and 1. Figure 2 represents a basic artificial neuron.

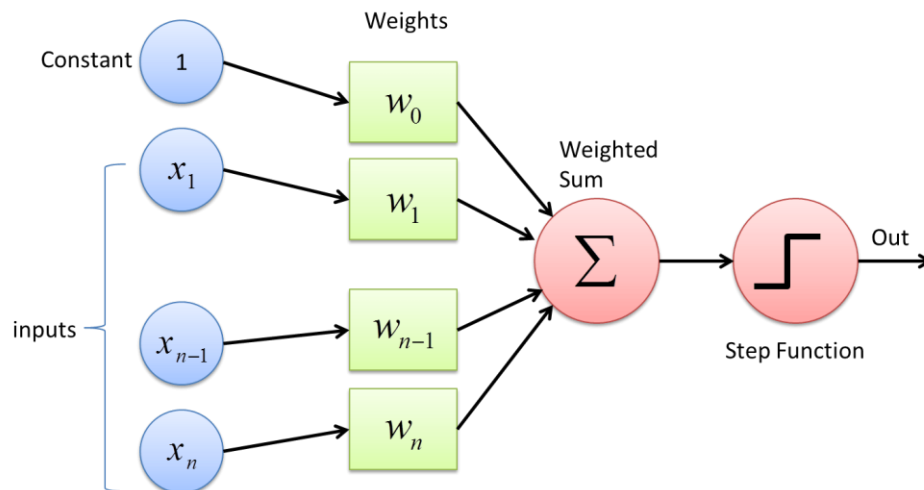


Figure 2. A Basic Artificial Neuron [49].

Examining the AN in Figure 2, the blue nodes on the left are the input values that make the input layer. The AN takes any positive or negative value as its input. A great example is to classify Iris flowers with the famous Iris data set from UCI Machine Learning Repository. If the AN is tasked with classifying Iris flowers, a couple of the inputs could be the length and width of the flower petals, and petal color [15].

The green colored boxes next to input nodes are weights in Figure 2. Weights are called as the parameters of the ANN model. The weights represent the relative importance of each input to the classification decision. Before the inputs reach the 'weighted sum', every single one is multiplied by a weight value associated with the input. Usually for the first round the weight is initialized as a random value. For the model to learn, it needs to repeat the same task countless times. The weights change every round as the ANN tries to find a solution for the classification that results in a more accurate prediction. This algorithm is called 'forward propagation'. In short forward propagation is a term used for passing forward the information from one layer to the next [16].



An example of the forward propagation with using the Iris flower database would be to classify different color Iris flowers. There are multiple Iris species that look the same except for their color. When weighting the characteristics of yellow and purple Iris', the length and the width of their petals does not matter, so those inputs will have low weights. In reality, only the color matters and that input will be weighted higher depending if the color is either yellow or purple [16].

The leftmost red circle in Figure 2 represents the weighted sum. The weighted sum takes the weighted input and sums them up to create one aggregate value which is fed into the activation function. Equation 1 below is a mathematical representation of a single artificial neuron's output. Equation takes  $x$  which is the input,  $n$  is the number of inputs and  $w$  which is the weight.  $\Phi$  refers to the activation function [17].

$$f(x, w) = \varphi \left( \sum_{i=1}^n w_i x_i \right)$$

Equation 1. Activation Function Equation.

The rightmost red circle in Figure 2 is the activation function of the artificial neuron and last part before the output. The activation function, also called transfer function, generates a classification decision according to its design. For example, the Iris is classified as 'Setosa' or 'Versicolor.' The activation functions are used to add desired non-linearity to the ANN. Creating more layers for the ANN does not increase the efficiency if the ANN is linear, unlike for non-linear ANN. There are multiple different activation functions with different designs that are regularly used with ANNs.

### 3.3 Activation Functions

There are three categories of multiple different activation functions with different designs that are regularly used with ANNs. The activation functions determine

whether an AN is activated or not. The three categories of activation functions are step, linear and non-linear.

The step function is activated when the weighted input is above the set threshold. The functionality is similar between perceptron and step function. Step activation works perfectly with binary classification as the output will be either 0 or 1, activated or not activated. The plot function has drawbacks, however. It does not fit with a problem that needs more than two classifications, like in the earlier Iris flower example. For Iris flowers, instead of binary value, the output should be a value between 0 and 1 that indicates 'partly activated'. Linear function partly solves this problem. [18]

The Linear function gives a linear range of activations instead of the binary 0 or 1. Though if only linear activation is used on every layer of the ANN, the output will be a linear transformation of the input, where inputs are multiplied with weight. Such a network is not able to output interesting behavior, as real-world problems are non-linear. Actually, with linear activation ANNs it does not matter how many layers there is as all the layers collapse into one and it simply becomes a linear regression model, which is the reason why linear activation is not used with ANNs. Also, linear activation cannot use backpropagation algorithms, which is the main way for ANN models to learn. More about backpropagation in chapter 3.5 [50].

Non-linearity allows the model to create complex connections between ANNs nodes. Complex connections are essential for learning from complex data, for example, data with hundreds of dimensions, video, images or audio. This thesis' project uses Rectified Linear Unit (ReLU) activation seen in Figure 3. ReLU is the most common activation function used in ANNs and usually is a great choice [55].

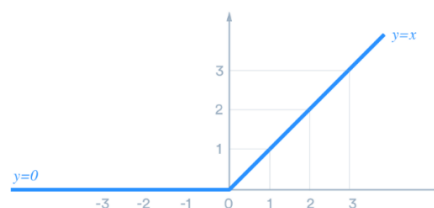


Figure 3. Plot of Rectified Linear Unit [55].

Figure 3 shows that ReLU activates linearly only with positive input values. That means it is cheap and fast to compute and activates sparsely. Sparsity is a desired functionality as it results in concise models that are more accurate predicting than non-linear that have more complex logistic calculations for every input, for example, sigmoid and tanh activation functions. [56]

Non-linear activation functions in ANN can approximate any function, depending on how many layers ANN has and how many neurons are on each layer. This is called the universal approximation theorem, a mathematical theory of ANNs. Essentially every layer of the ANN can be thought as a part of the larger function, and as the information flows through the layers the function gets more complicated. ANN can be described mathematically as in Equation 2 [25].

$$f(x) = \sum_{i=1}^n \varphi(w_i^T x + b_i)$$

Equation 2. Approximation function of an artificial neural network. [51]

Equation 2. depicting  $f(x)$  is a mathematical approximation of an ANN.  $\varphi$  is any sigmoid type activation function.  $w^T$  is a transpose of weight so that  $w^T x$  is the inner product of  $w$  and  $x$ .  $b$  is the bias of AN.  $n$  is the number of inputs.

### 3.4 Artificial Neural Networks

The ability of brain to learn from experiences is a natural proof that while some problems are beyond the scope of current technology, they are solvable for the brain itself. Advances in biological research show that brain stores information as patterns in its neural networks. It is known that knowledge is formed by connections between groups of neurons and the connections have a role in other cognitive functions such as memory. While there are myriad hypotheses, the mechanics how those patterns translate into knowledge is still fundamentally unknown [21].

Perceptron is essentially a neural network with only single layer. Interlinked perceptrons or rather perceptrons on multiple layers are called a 'neural network'. In a simple case, binary classification of a data set one layer is enough. Like the example in the Figure 4, a single layer ANN can solve simple problems such as separating two data groups with a single linear line. Once again, there are multiple different ways to implement the neural network, but this thesis only goes through the basic feedforward, deep neural and convolution network models [23].

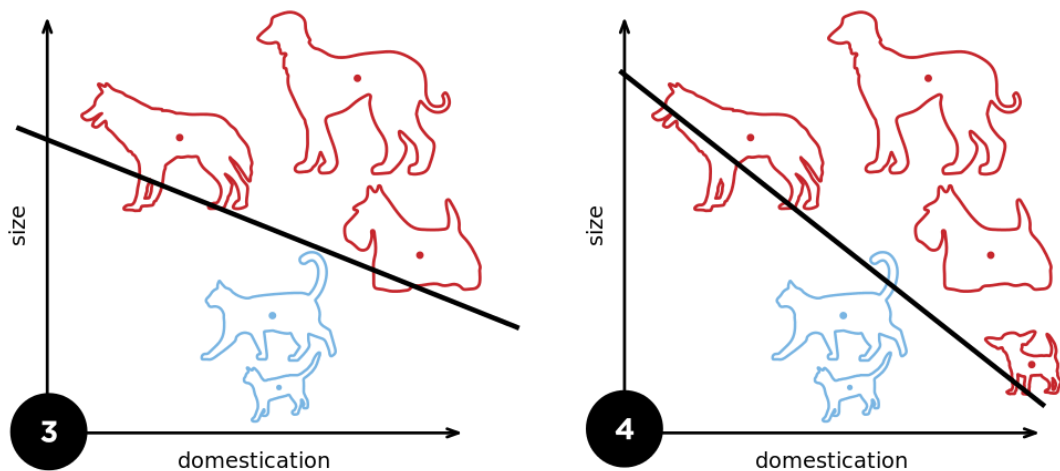


Figure 4. Linear classification separating two species [32].

ANNs are most valuable when solving hard or abstract problems that would not be possible otherwise, so one layer is not enough. Most of those problems are not possible to solve with simple linear separation. More ANN layers equal more features that can be handled. Features refer to the variables and attributes in the data set. In the case of Figure 4, the features are size and domestication values. Each feature is an additional dimension when creating a plot of the data. Two and three features can still be easily imagined, but when the problem has two or three hundred features, it becomes impossible to solve with traditional methods. This is the reason why multiple layers of ANN are so important [27].

Before attempting to explain how and why neural network functions, it is crucial to understand how ANs create a layer of ANN. As seen in the Figure 5, a layer is a collection of ANs usually represented visually as a straight column of

circles. Every circle is a functional AN like the one depicted in earlier chapter. The ANs take inputs from earlier layer and after processing passes an output to the next layer. A feedforward ANN is the basic model where information flows only “forward” from one layer to the next.

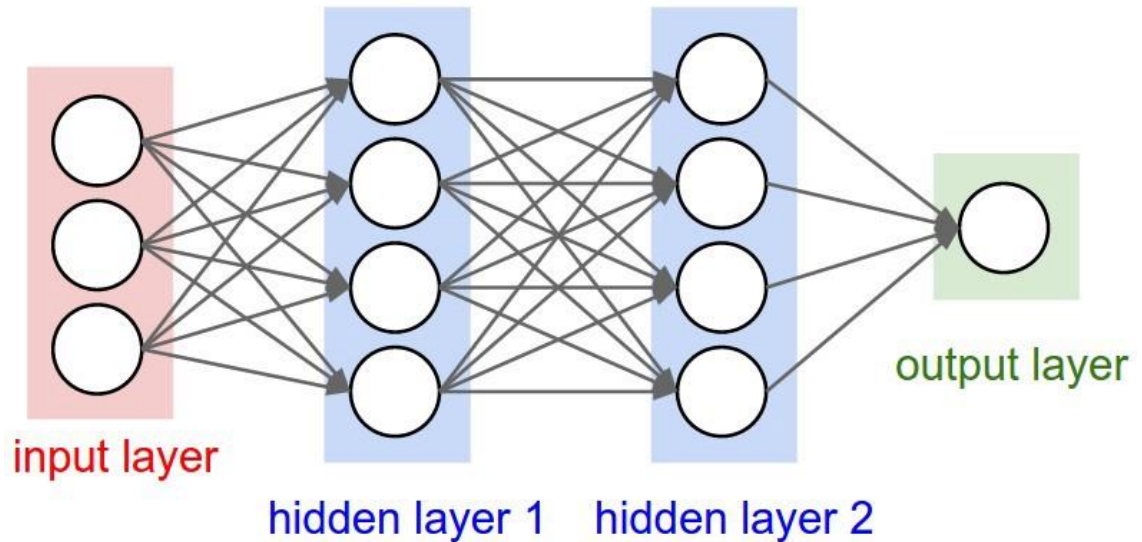


Figure 5. Small feedforward neural network with three layers and an input layer. [25]

As with a single AN, the leftmost layer is the input layer of the ANN. Each input node has weighted connections to every AN on the next layer, the ‘hidden layer 1’. The ‘hidden layer 1’ and every layer between the input and output layers are called ‘hidden layers’. These layers are hidden as they are not observed at all and will not give any insight on the structure of the function. The hidden layers are sometimes referenced as the ‘black box’. The rightmost layer is the output layer. All ANNs have an input and an output layers, but the number of hidden layers will differ. Usually when describing the number of layers, the input layer is not counted. More hidden layers mean that the ANN can solve more complex problems [19].

A hidden layer is a collection of ANs that have the same properties. That also means that it is possible for each of the hidden layers to have different activation functions. It is possible to discover better performing models by experimenting

with multiple activation functions inside the same ANN. There is a study on how different activation functions behave when combined [20].

### 3.5 Backpropagation

Artificial neural networks remain far from approaching the animal intelligence not to even mention human intelligence. One of the reasons is the fundamental difference between biological and artificial neural networks. Biological neural networks have a property called plasticity. Plasticity essentially means that the brain's neural synapses strengthen or weaken over time or entirely new connections are made based on their activity, but all previously existed connections will remain. Countless connections create intelligent and adaptable behavior. This is essentially how brain's memory works and learns, and brain needs memories to learn from its earlier mistakes. Supervised ANNs do not need memories, as the instructions to change parameters comes from "outside" of the ANN, from validating the prediction accuracy of the supervised data set. This separation of these functions creates structure for ANN and significantly reduces calculation time. ANNs use backpropagation algorithm which is the main factor of how AN parameters become modified and the ANN learns. [22].

One of the advantages of supervised learning is that we can use testing sets to get an objective measurement of learning performance. The connections between each AN layer change constantly when training, and the connections might disappear as in not being triggered, to have faster training period with the prediction algorithms. Additionally, as the ANN is supervised, the wanted solution is achieved with adjusting AN parameters and learning from earlier mistakes with backpropagation algorithm [22].

Backpropagation is the core algorithm for how supervised ANNs learn. When the ANN is supervised there are determined values that are wanted out of it. After a training cycle, the ANN suggests a value as a solution and the solution is compared to the set of right answers. The difference between target and calculated value is expressed as a logarithmic error rate. Backpropagation

algorithm fine-tunes the weights of ANs based on error rate given by a loss function. Error rate approximates the performance of the ANN and the confidence in the predictions it has made. In the example of Figure 6, the ANN has an ideal output value of 0, but after the training cycle the real output is 0.77 [25].

Just like in forward propagation, backpropagation weight adjustments are carried out at each layer, but in reverse order, first the weights between output and last hidden layer, then the weights between second-to-last and last hidden layer, and so on. Additionally, every single different training example in a data set and their ideal solution are compared to the prediction ANN gives as an output. The weight values are then updated. The mathematical properties of backpropagation are out of the scope of this thesis [26].

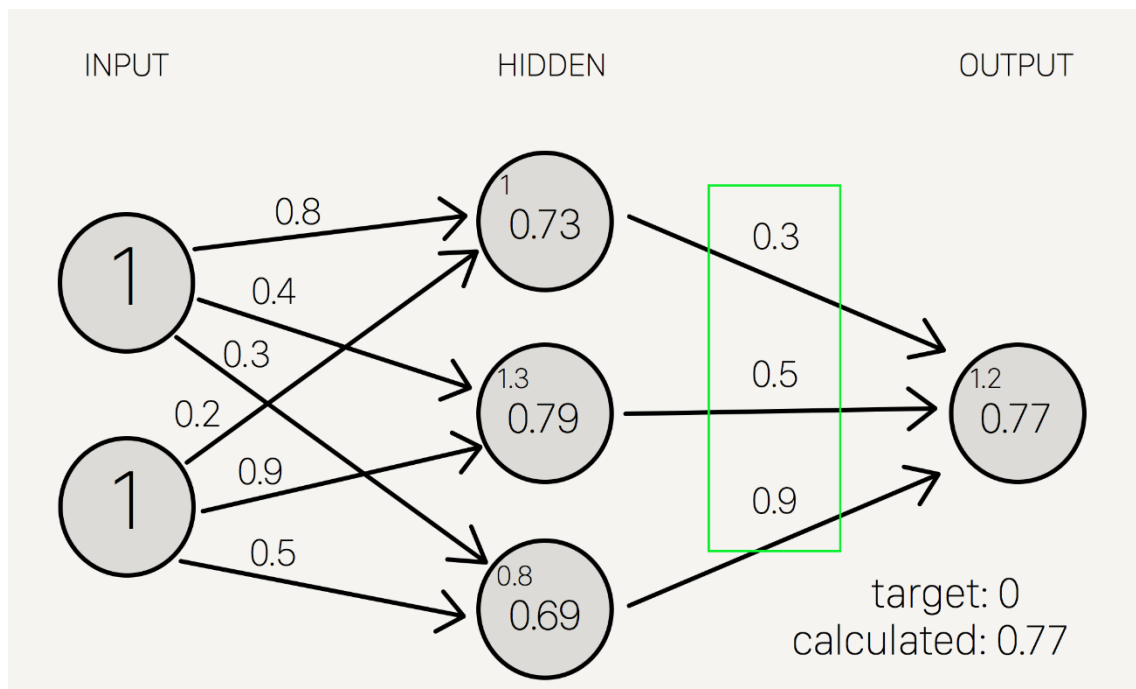


Figure 6. A two-layer ANN with example weights and node values [26].

To improve the model, it is essential to measure how wrong the predictions are between training cycles. In Figure 6 as the real output is 0.77, the adjustment is made to the weights marked with green box to make the model to output a value closer to 0.

An algorithm called gradient descent is designed to minimize the loss function and doing so it creates more accurate model. Gradient descent attempts to find the direction in which to change the weight values to reduce loss function value. The gradient descent algorithm iterates the weight values towards a situation where further tweaks will not change the loss function value. Figure 7. visualizes the iteration process [45].

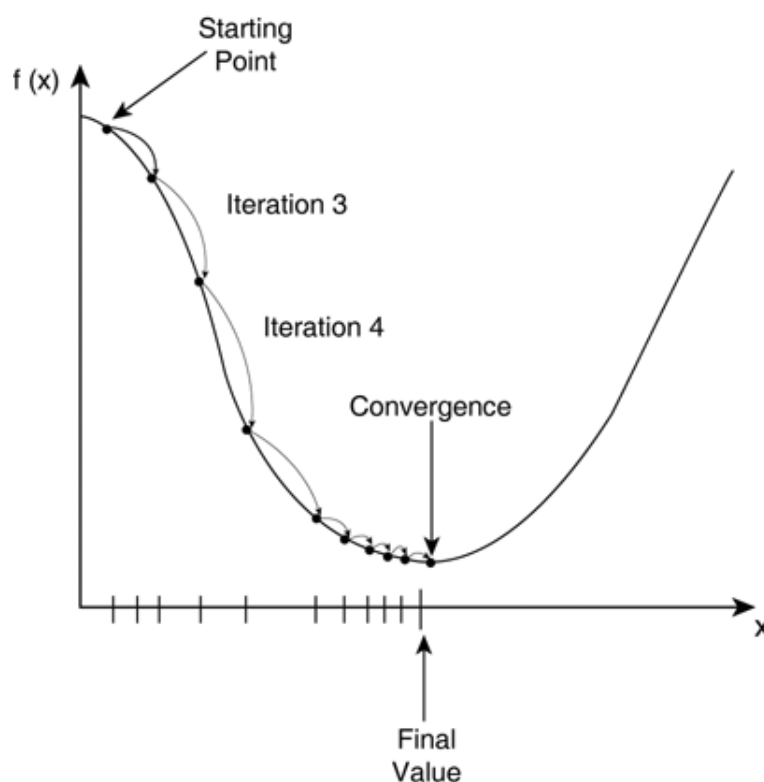


Figure 7. Gradient descent visualized [45].

Figure 7 shows  $f(x)$  as the cost function and  $x$  as the weight. The convergence signifies the point where no change in the weights is needed. The jumps between iterations comes from the learning rate. Learning rate controls the step size of every iteration. Learning rate cannot be too large as the point of convergence is missed, and if the rate is too small it takes significantly longer to converge. There are many different optimization algorithms to determine the learning rate size. This thesis' project uses well known algorithm Adam, which computes individual learning rate sizes for all weights [45].



To summarize, in machine learning, the focus is on learning from data. The learning happens through repetition and feedback. The model outputs predictions for the problem and continues the learning guided by the loss function. As loss function measures how wrong the model is, gradient descent finds weight patterns in the ANN to minimize the loss function. In short, training calibrates the parameters with repeating forward and backpropagation algorithms.

## 4 IMAGE RECOGNITION

“Almost any process you can imagine can be thought of as function computation. [Examples include] naming a piece of music based on a short sample of the piece [...], translating a Chinese text into English [...], taking an mp4 movie file and generating a description of the plot of the movie, and a discussion of the quality of the acting. [...] In principle, neural networks can do all these things.”

- Michael Nielsen

Human brains can intuitively recognize objects, their spatial relationships and conceptual structure. These are essentially the pieces of which all images are composed.

### 4.1 Overview of Image Recognition

The ANN variant most commonly used with computer vision to recognize content of images and video, is called convolutional neural network (CNN). CNN is a subset of deep neural networks (DNN). As with the earlier examples, ANN that has more than one hidden layer is usually called a DNN and their learning process is called ‘deep learning’. As images and video are a visual medium and are fundamentally different with data sets that consists only of numerical values and attributes, the learning becomes more abstract problem [28].

An image is essentially a matrix of pixel color values. Figure 8 has a general example how images can be modified to use them as an input in ANN. An image can be flattened from two-dimensional matrix into one dimensional vector like in the example of Figure 8 [28].

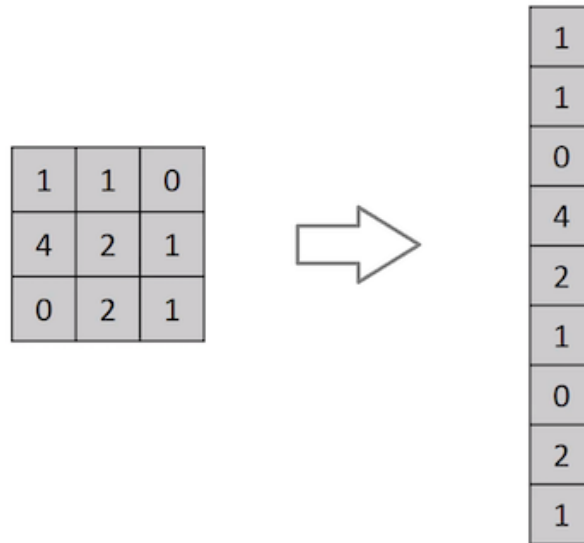


Figure 8. 3x3 matrix resized into a 1x9 vector [28].

The vector of pixel values can then be used as an input with ANN. Though there is one problem. Even if the Figure 8 vector is only theoretical example, each of the vector values is considered as a feature. Pixel value as a feature does not carry “meaning” in the same way a data set about attributes of animals or Iris flowers does. In case of image recognition, a regular feedforward ANN can be trained to recognize MNIST data sets of black and white ‘binary color’ images of handwritten numbers, but hardly anything more complicated [28].

A data set is a collection of data. Image data sets are used to train and validate object detection algorithms. MNIST is a database of handwritten digits. The database contents are used commonly to train different image processing systems. The handwritten digits have been preprocessed to contain only the shape of the digit and thickness of the pen. Picture 2 contains a sample set of from the MNIST database [29].



Picture 2. Sample digits of the MNIST database [29].

The digits in the Picture 2 are preprocessed to contain only the important information before they are given to the feedforward ANN. Considering the problem, most of the image content is irrelevant and so the digits are preprocessed to be as simple as possible. In this case this irrelevant content is called image noise as it distorts the important content.

The MNIST data set digits are originally scanned from paper and have all kinds of artifacts like noticeable texture of the paper surface and multiple shades of color. For feedforward ANN this absolutely matters, as every input is a pixel of equal importance. Regarding the problem though, only the shape of the digit and thickness matters, so the unnecessary properties needs to be left out. The scan is contrasted so only two colors remain. All the digits are isolated and centered into small 28x28 sized images. All digits need to be centered, because for example, a regular feedforward ANN would consider a centered seven and a seven that is closer to the border as two different digits. As the input is standardized to a format where the digits have clear shape without any noise or artifacts, the feedforward ANN outputs much more accurate predictions [30].

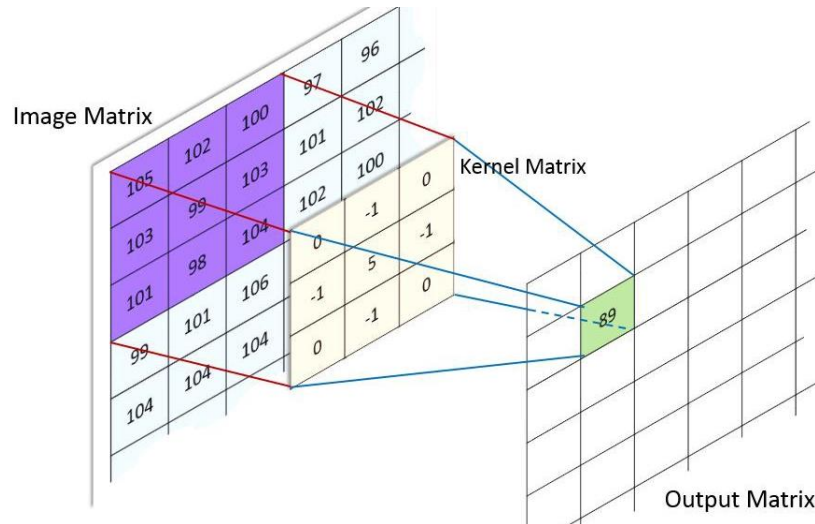
## 4.2 Convolutional Neural Networks

Using regular feedforward ANN will not work with full-colored normal sized pictures. Normally pictures are composed of three overlapping pixel matrix channels to simulate full range of color with RGB color model. Additionally, when considering a digital photo, for example, their size ranges for example, anywhere from small 614x874 images to large 7752x10320 images taken with a professional system camera. The size difference is huge compared to preprocessed MNIST data set digits.

CNN can be broken up into two main parts. The convolution layers that extract features from image data set and regular fully connected layers as the last layers before output layer, that classifies the extracted features. Fully connected layer uses the extracted features to predict the content of the image sample. CNN is comparable to regular ANN, but CNN must find the features contained in an image file before it can learn from those features [33].

### 4.2.1 Convolution layer

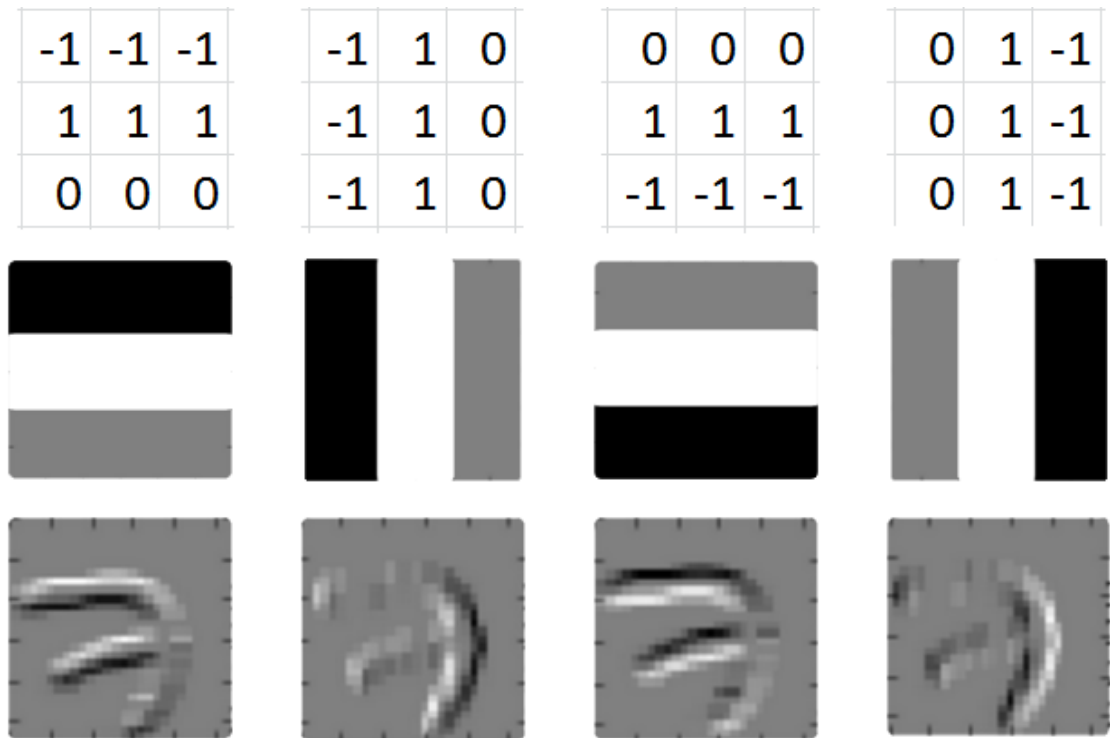
As images are essentially only matrixes, the features need to be created from the pictures and CNNs has the ability to learn these features. The name of convolutional neural networks comes from a special kind of hidden layer called the convolution layer. The concept of the convolution layer is shown below in the Picture 3 [31].



Picture 3. Convolution operation [31].

Convolution layer works as a filter for the image input. Picture 3 visualizes the convolution operation. In the picture a 3x3 sized kernel matrix makes dot product calculations with every different combination of 3x3 areas of the input image's matrix. After every convolution, the kernel outputs a value to an output matrix highlighted green in the Picture 3. Output matrix is also known as feature map. Essentially the kernel matrix 'slides over' the whole image matrix surface area step by step during the process and creates the feature map [33].

Depending on the kernel size and its values, the feature map highlights different patterns from the image matrix. The patterns of a single photo consist of all kinds of edges, shapes, textures, objects, curves, colors, etc. These are exactly the things the convolution layer detects. Each kernel matrix is specialized to detect some pattern. For example, this Picture 4 has a handwritten number seven from the MNIST data set and applied kernel matrixes that are designed to detect edges. [33]



Picture 4. A set of edge detectors applied on MNIST data set.

The edge detectors displayed on Picture 4 are some of the simplest filters. The picture shows edge detector kernel matrixes on top row and the middle row describes which kind of patterns are detected. The bottom row pictures show feature maps for all different detected edges of a handwritten number seven. The feature map for edge detection operation creates white pixels on those areas where the image color changes the way the filter is anticipating, thus detecting the edge.

As CNN is deep neural network, there are multiple layers of convolution. Actually, CNN has multiple specialized layers in addition to convolution layer. Pooling, normalization and normal fully-connected layers in different orders to detect different features. Edge detection is a simple filter and is located at the start of the CNN. These simple filters enable the use of more complex filters located in the deeper layers. Complex filters can detect more sophisticated patterns like the shape of an eye or a historical architectural design trend. The more layers of convolution the model has, the better it is detecting complicated shapes, for example, cars [33].

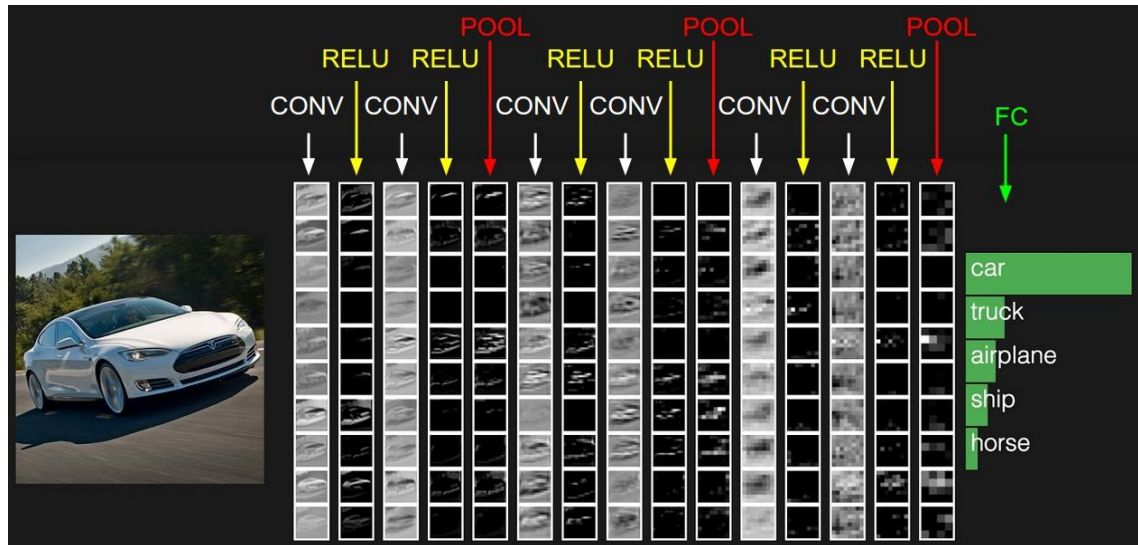


Figure 9. CNN architecture detecting a car [52].

Figure 9 shows a CNN architecture detecting a car. The model has 6 convolution layers in total. As an image of a car consists of multiple different shapes, each convolution layer creates multiple different feature maps detecting more complex shapes deeper in the CNN model. Finally, a fully connected layer catches the generated feature maps and from them predicts the most likely content of the original image file [52].

#### 4.2.2 Pooling layer

A common practice is to have an occasional pooling layer between two convolution layers. Pooling reduces the computation by decreasing the size of the input image and indirectly the number of parameters. This pooling operation is also called downsampling of the image that can be seen in Figure 10 [52].



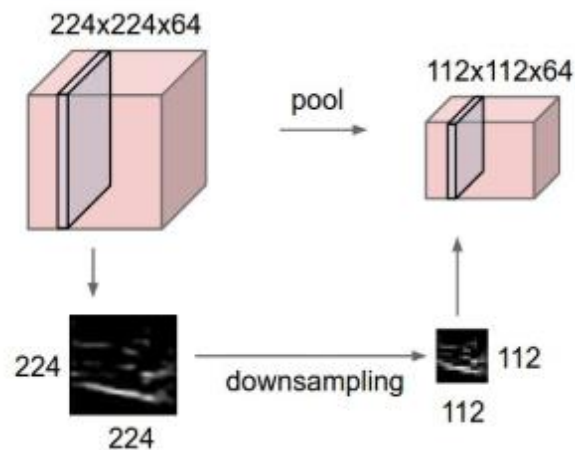


Figure 10. Downsampling operation [52].

In case of CNN as the input size gets smaller, more of the input fits inside the kernel. As more of the input fits inside the kernel, the picture still visually represents the same content, but with fewer pixels. This variance in size of a single input image file also reduces the overfitting [52].

#### 4.2.3 Overfitting

Deep neural network has lot of flexibility to handle variety of complex data sets. The number of layers becomes a weakness if the learning process lacks control in the ANN design. Overfitting is the situation where the DNN performs well on the training data set, but not with other similar data set. In this case the DNN model is “memorizing” the data set it is training on, and memorizing is not considered learning.

There are multiple ways to reduce overfitting by controlling different parts of the DNN. Later this thesis goes through some of the methods to that reduce overfit that were used in the project’s ANN design.

## 5 WRITER IDENTIFICATION PROJECT

The chapter explains the project scope, the tools and materials, which was used for creating the model.

### 5.1 Overview of writer identification

Human identification is an old field of study. Research in history has revealed that in 14<sup>th</sup> century Chinese practiced person identification by ink handprints, and early 19<sup>th</sup> century Paris Police listed criminals by measuring the parts of their bodies. Modern techniques are of course much more refined. There are three categories in human's identification. "*What he knows*" as in passwords or personal information. "*What he has*" as in NFC-tag or 2-step verification with a phone. "*What he is*" as in fingerprints, voice or signature. "What he is" category is also called biometrics [35].

Identification methods that can be stolen, shared, lost or duplicated are not perfectly safe. Identification methods that are not safe cannot be relied on. The most difficult identification method that can be stolen, shared, lost or duplicated is biometric data. Biometric techniques use person's own physical features such as eye's retina, face shape, fingerprints or behavioral features such as voice or writing [35].

The identification of a writer based on handwriting is a practical behavioral identifier. Usually analyzing someone's handwriting needs expertise and hard work. As the handwriting from a single writer may vary because of mood, writing speed or tool used, the problem becomes even harder. The implementation of machine learning can be a convenient tool in the research field of behavioral biometrics and in writer identification.

Writer verification is a subcategory of writer identification. Verification method is used in classification problems to determine if two samples of the same handwritten word are written by the same writer. For example, verification is used

when proofing if an autograph is real or fake. Writer identification method is used in categorizing large data sets of handwritten text, the method lists probable writers of every sample. Research made by Brink, Bulacu and Schomaker in 2009 demonstrates when given an identical data set, writer verification achieves much lower error rate than writer identification. The research demonstrates that developing writer identification systems is a more difficult challenge when compared to verification [39].

Researches in writer identification is split into two categories, online and offline identification [2]. Online writer identification requires a recording of the writing procedure where a sensor notices the movements and pressure of the pen-tip and when the pen touches paper. The recording is regarded as a unique digital representation of handwriting style [3]. Offline identification uses only scanned images of handwritten text and lacks the additional information about the writing style. As offline identification lacks information, it is therefore more complicated to solve and requires more samples [4].

Both writer identification categories use either text-dependent or text-independent methods. Text-dependent methods require the writer to write a predefined words or sentences for identification. As the text sample is predefined, the model is designed to expect a close match with the original sample. Text-dependent methods can be used, for example, in font recognition problems [40]. Text-independent methods can use any written text sample. In contrast to text-dependent method, text-independent method does not make assumptions about the samples and because of it has much broader range of applications [41].

Considering the lack of information and limitations of analyzing scanned paper documents, text-independent offline writer identification methods are the most difficult to develop methods out of all writer identification and verification methods. Previous research on text-independent writer identification has based on finding features designed by handwriting experts. As deep learning methods can successfully learn features independently and developing high-performing methods for those problems is difficult, it is a great field to study and research [44]. This thesis' project goes through the machine learning tools which help

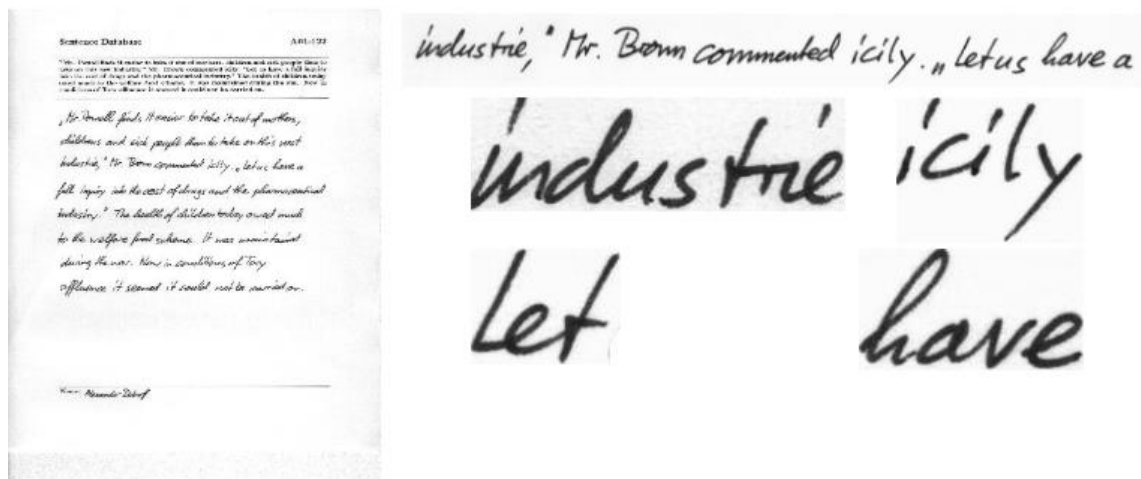
solving the problems of text-independent offline writer identification with CNN model.

## 5.2 Writer identification project tools

The main goal of this thesis was to research and implement a CNN model for identifying writers based on the characteristics of their writing styles. More specifically a text-independent offline writer identification implementation.

### 5.2.1 Data set

There are multiple databases of scanned handwritten documents just for researching machine learning implementations. The data set used for this project is from IAM handwriting database. IAM handwriting database contains 1 539 pages of scanned handwritten text and is maintained mainly for machine learning research purposes. IAM handwriting database has three different preprocessed data sets of the pages, which are entire scanned pages, cropped single rows of text and cropped words. Picture 5 has a sample of all three IAM data sets [36].



Picture 5. A sample of IAM database data sets [36].

The project used IAM data set that includes the cropped rows of text. As full scanned pages have too much empty non-essential information and single

cropped words are unnecessarily small to have multiple random patches cut out of them, images of the cropped sentences have a good balance between the data sets [36].

### 5.2.2 Software

The project model was built using Python language, neural network library Keras and TensorFlow framework as a backend. Currently many popular machine learning tools use Python language so it was a natural fit for this project. Python supports many excellent open source machine learning tools such as Google's TensorFlow and Keras. The machine learning tools Google has developed like TensorFlow are used on their own famous services, for example, with Google Translate's handwriting recognition functionality. Keras is an open source neural network library that has a capability of running on top of TensorFlow framework.

### 5.2.3 Hardware

Usually training a DNN model needs high amount of calculations, so the training requires a considerable hardware to run efficiently. For example, thesis' project model has over 800,000 parameters to be calculated per every image input. Graphics Processing Units (GPU) have thousands of concurrent hardware threads compared to Central Processing Units (CPU) and are specialized at handling multiple simple calculations in parallel. When considering the amount of simple calculations in the project, it makes sense to use GPUs instead of CPUs for training the DNN model. Offloading model training to multiple GPUs can reduce large DNN's training time from days to hours [37].

No suitable hardware was available locally for the project, so Google Cloud platform (GCP) was used for the project model training. GCP is a cloud computing service on which Google runs their own products like Google Translate, Search and YouTube. GCP can be rent by anyone to build their own projects. GCP has

also extensive support for wide variety of machine learning projects with complete high-performing models and hardware designed for machine learning tasks.

The service offers NVIDIA's Tesla line-up of high-precision GPUs with high video RAM intended for industrial scale deep learning and data analysis projects. As GPUs are specialized in calculating vectors used with graphics, a technique called GPGPU is more general parallel computing specialized in floating point calculations. This project used NVIDIA's Tesla T4 with 16 GB GDDR6 memory. The choices between different powerful GPUs are great for industrial scale deep learning projects, but with small scale projects it matters little which kind of GPU is used [38].

### 5.3 Research of Writer identification models

Researching and developing new high performing CNN architectures has been ongoing from 1989 when the first image recognition CNN design was created. Great way to start studying different CNN model designs is to study popular high performing models. Couple different CNN architecture designs was researched for the project, namely LeNet-5 and AlexNet designs.

In 1998, LeNet-5 model was the first CNN to learn automatically the data set features [42]. In 2012, AlexNet won the ImageNet database competition by a large error rate margin. It was the first high-performing CNN that combined ReLU activation function, dropout and data preprocessing to reduce overfitting, illustrating the benefits of CNN [43]. Those techniques are still used on modern CNN architectures.

By modern standards both of those models are linear and very simple, compared to the complex and high-performing architectures like ResNet or DenseNet, which bring totally new elements to the CNN architecture. LeNet-5 has 7 linear layers and about 60,000 parameters [42]. AlexNet has the same general architecture than LeNet-5 but is considerably larger as it has about 6,000,000 parameters [43].

Also couple different CNN writer identification research papers was researched. One is a paper from Xing et al. called *DeepWriter: A Multi-Stream Deep CNN for Text-independent Writer Identification* published in 2016. The team created text-independent offline writer identification method, which used modified AlexNet structure. The project also used HWDB database of written Chinese characters and preprocessed the images into cropped small square sized samples. Picture 6 shows couple HWDB Chinese characters [2].



Picture 6. Couple HWDB Chinese characters [2].

All images used in CNN should be of identical sizes, so the HWDB characters need to be resized. If all the characters are just resized to identical sizes, it would distort the shape of the handwriting. As the shape of the handwriting is the only feature offline writer identification can use, distortion needs to be avoided. The characters need to be resized with their aspect ratio maintained. Still the size difference problem exists [2].

The DeepWriter research paper proposes a way to train a CNN to learn writer specific features by randomly cropping small square samples out of the handwritten text. This way the CNN gets identical sized input and can be standardized [2]. Another paper from H.T. Nguyen et al.'s research group, *Text-Independent Writer Identification using Convolutional Neural Network* that was published in 2018 proposes the same method and has an excellent image showing the process in Figure 11 [44].

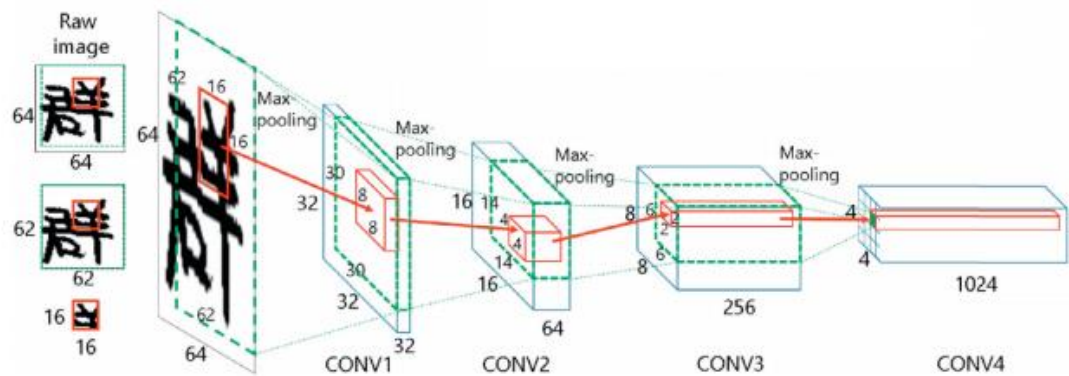


Figure 11. H.T. Nguyen et al.'s CNN architecture for extracting samples from a Japanese character [44].

Figure 11 visualizes the currently cropped sample with green. Another important reason to crop samples is the way how CNN model learns. If the input image would contain whole words or sentences, the model would learn the shape of the words. By cropping the images, the model will not learn the specific characters and words, the model sees only parts of the written characters it learns the writer specific features instead [44]. Figure 11 also shows part of the CNN model architecture H.T. Nguyen et al. used.

Figure 11 shows other details in addition to cropping. The model has 4 convolution layers with max pooling in between every layer. By comparing the model results of different layer compositions, H.T. Nguyen et al. determined that four convolution layers is the most efficient solution for the problem [44].

Curiously Xing et al.'s paper has over 4 times larger input images, and they use five convolution layers [2]. Sounds reasonable that the image size and details need a deeper CNN architecture. Figure 12 has a simple picture of Xing et al.'s architecture. Also, different kernel filter sizes were tested, and no clear difference was found between them.

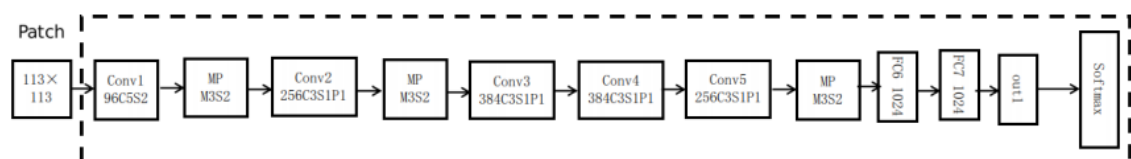


Figure 12. Xing et al.'s CNN architecture for writer identification [2].



## 6 DEVELOPMENT

The development of the project was done entirely on Jupyter notebook which combines Markdown and Python languages. As Jupyter notebook is popular in data science fields it has a functionality to run a part of the code and visualize output with Python's matplotlib library. The next sub chapters of this thesis will go through the projects Jupyter notebook. The whole project can be found from the project author's GitHub repository [59].

### 6.1 Preprocessing

Building an efficient CNN model requires consideration on both the architecture and input format of the data. Preprocessing refers to all manipulation of the raw data before a readied data set is given as an input to the CNN model. IAM database has already preprocessed binary color image files. The process is specified in chapter 5.2.1.

#### 6.1.1 Manipulating the data set

IAM data set used with the project is originally separated in folders depending on the writer and the sentence. The data set has 657 writers and 5,685 unique sentence image files. Every sentence file is labelled to its writer and an enclosed text file lists every single sentence image of the data set. This information can be used to map each sentence to its writer.

The project data set needed scaling because of the hardware limitations. The full data set was way too large so only 50 most active writers was selected as the project data set. First a dictionary was needed to pair all the writer with correct sentence IDs. The created dictionary was used to find the writers with most content. After the writers were found, an array was used to filter the data set. A Python program was written for the creation of the custom data set.

### 6.1.2 Splitting data set

Data set is usually divided into three parts, the training, validation and test data sets. Training data set contains the data that is used to train the model. Every iteration the model trains through the training data set, an accuracy value is given, and the model parameters are fine-tuned. A single iteration of training is called an epoch [53].

Validation data set is used as an unbiased evaluation of the model. After every epoch the model estimates training accuracy, it also checks the accuracy with the validation data. The model does not learn from the validation data set but is affected indirectly, as the validation accuracy is taken into consideration when fine-tuning the parameters between epochs [53].

Test set is like validation set but used only after all the training is done. The test set is generally used as a standard for evaluating the model's prediction accuracy. Usually the test set is curated to include all the different problem categories of the larger data set. In machine learning competitions, the test data set used as a standard, which every trained model is evaluated against. In this project all three data sets were created with randomized content without curation [53].



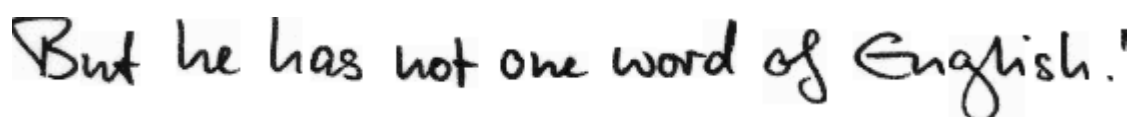
Figure 13. Splitting the data sets [54].

Figure 13 shows the approximated size ratio of the three data sets. The ratio is case specific and might need trial and error. A usual ratio is 4:1:1 which is the usual ratio many of the different DNN models use, and later the ratio is adjusted

if needed [54]. Xing et al.'s paper also used the same ratio, so the 4:1:1 ratio seemed to fit the thesis' project.

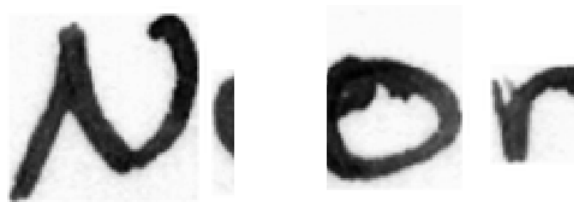
### 6.1.3 Cropping the image samples

The model follows the design of earlier mentioned research papers. The model takes identical sized square patches of images. As the IAM data set sentences are not squares, they first need to be resized with the aspect ratio maintained. Picture 7 shows a sample sentence from the data set.

A handwritten sentence in black ink on a white background. The sentence is "But he has not one word of English." The text is written in a cursive, slightly slanted style. A light blue rectangular highlight is placed behind the entire sentence.

Picture 7. IAM data set sample sentence [36].

The writing characteristics can be seen from a single letter and from the flow of the words, how single letters are related to each other. Xing et al.'s research paper raises the question of losing spatial relation between the letters and words in the process and thereby some of the handwriting characteristics. The paper requests two approaches. The elegant approach would be to input a pair of adjacent patches into the CNN model that has multiple convolution streams. The patch pair is merged together by element-wise sum before the categorization. The approach was not explained well enough to implement it. The second approach is to create a large number of patches per sentence, that would catch every letter and their relation to each other. The samples are inputs to a regular CNN model. This way the model would learn more characteristics from the writer than just cropping a sentence to a handful of patches. The accuracy difference between the approaches is less than one percent, elegant multi-stream CNN model being 99.01% and regular CNN being 98.23%. The thesis model follows the regular CNN architecture seen in Figure 12 [2].



Picture 8. Patches cropped from sample sentence.

Picture 8 shows couple of the patches cropped from the Picture 7 sentence. The program crops 116 patches at random from a single sentence. A Python program was written to crop identical 113x113 sized patches from the sentences. Xing et al.'s research paper had the same sized patches cut from Chinese characters and the IAM sentence image height averages on 150 pixels, so the patch size is suitable and resizing will not cause distortion.

## 6.2 Generator function

As image files require multiple times more memory than plain text when uploaded in the GPU, memory issues are a serious threat. Without generator function on a single training epoch the whole data set is uploaded in the GPU and might not stay within the hardware's memory limits. Generator function is an important part of the image recognition architecture [56].

Generator function will select part of the training data set as a batch and uploads only the batch to the GPU memory. Python language makes programming the generator function easy as it already has a support for the functionality. The model was not tested without the generator function as it was not clear if the Google Cloud hardware could manage the full data set load. The generator function was instead programmed and implemented only for educational purposes.

### 6.3 Convolution neural network model

In architecture of the earlier mentioned research papers, that inspired the development of this model architecture, had three to four convolution layers depending on the image sample size, max pooling between each of the convolution layers and two fully connected layers after the convolution layers. Xing et al.'s model had SoftMax classification as an output layer and the model was trained with the cross-entropy loss function [2].

The thesis' model was built using Keras library and was intentionally simplified. The model has only three convolution layers to scale down the calculation time. The rest of the Xing et al.'s paper's architecture is implemented unchanged. The model summary shows the specific architecture and can be found in Appendix 1.

Appendix 1 summary has columns for layer types that are named after Keras functions, the shape of the layer and how many parameters the layer has. The model is standard CNN architecture as it has ReLU activation functions after every convolution and two fully connected layer for categorization the feature maps. Flattening function is used after the convolution layers to transform the feature maps from matrixes to arrays so they are in a right input format for fully connected layers.

A fully connected layer makes most of the model's parameters. Checking the Appendix 1, total amount of parameters is 827,698 and the first fully connected layer is responsible for over two-thirds or 590,336 of those parameters. While training, the model's ANs develop co-dependency between each other, which leads to overfitting. Dropout layer is used between two fully connected layers to randomly shutdown a selected amount of ANs and it forces the neural network to develop more robust features. Usual practice is to shutdown half of the ANs of single fully connected layer and this practice was used with the project's model [57].

## 6.4 Training

The model trained for 40 epochs. One epoch is one full training cycle of the whole training set. After backpropagation and fine-tuning of the weights with training and validation accuracy, another epoch begins. The other research papers did not mention their epochs, so finding the right amount is purely trial and error. The model was first trained with 20 epochs and it gave approximately 80.34% accuracy. For the second training the epochs are doubled to 40, which gave the final accuracy score of 90.97%.

Figure 14 shows the training and validation accuracy scores through the whole training and how the accuracy developed. Training shows how the model is progressing in terms of its training. Validation shows how well the model is predicting a data set it is not been trained with. As can be seen, the validation accuracy is higher compared to training set, even though it is the training data set the model is learning. One reason for this might be that the training data set is learned through the dropout layer, but in the testing phase dropout is not used with validation data set. This is totally normal and corresponds with the point of dropout layer, as it should make the CNN model more robust and not memorize the training data set [58].

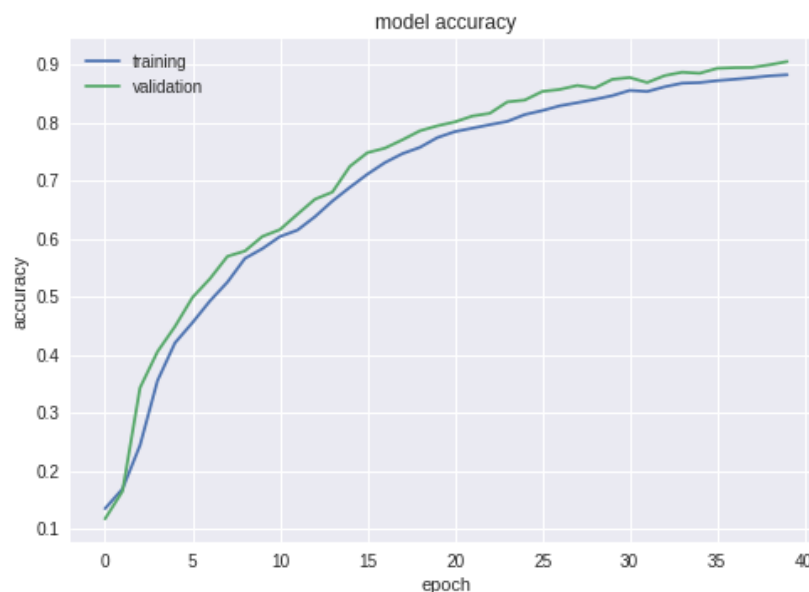


Figure 14. Plot of the project's model training and validation accuracy.

The final training and validation loss values give more insight how the accuracy should be interpreted. The final training loss value is 0.47 and validation loss is 0.31 after 40 epochs. Both loss function values are calculated with categorical cross-entropy algorithm that gives logarithmic values about the confidence of the model. Optimally the loss values should drop under 0.05. The backpropagation uses both accuracy and loss scores to fine-tune the parameters. The values do not have much meaning in itself and they need to be interpreted with the accuracy values.

It is possible for a model to have perfect score, because it takes the right decisions for the certain data set, but might still have high loss value. As the final prediction accuracy of the project model is about 90%, the model is predicting the data sets mostly correctly, but loss value is unusually high, it means the model is not absolutely confident about the predictions it is making.

## 6.5 Summary

The project resulted in a model which can reliably identify a writer based on handwritten sentences from IAM data sets. Training the CNN model with Nvidia's Tesla T4 GPU took about one and half hours. For the sake of science, the same model was trained also with local hardware's Intel i5-4300M CPU and it took 46 hours to complete. The possible next steps in the project is to implement an automated way of training and identifying additional writers.

## 7 CONCLUSION

The purpose of this thesis was to examine the machine learning field, focus on the image recognition subfield and to design and create an offline writer identification model. The decisions to use convolutional neural networks, offline identification, Keras, Python and specific data preprocessing techniques were all inspired mainly by H.T. Nguyen et al.'s and Xing et al.'s research papers and by the large machine learning community and their resources.

Before this project, the author had minimal experience with machine learning and had only implemented APIs of already finished machine learning projects. This project was the author's first neural network he has trained and designed from the start. The research, documentation, development and testing of the project took three months to complete.

The writer identification model was trained successfully and works as intended, though more training and tweaking of the model is needed for increased accuracy. The project was created mainly to support the learning process about neural network field. The created the model itself cannot independently work as a behavioral identifier application. More development work is required for the model to be used as a practical identification application. Training a new version of the writer identification model takes time and resources and more powerful hardware. The Google Cloud Computing platform is a good way to start a new project, but after the testing period the service introduces monthly payments, so it makes sense to move the project on a local hardware system.

Essential requirement for further development is a new hardware system. When developed further, the model can, for example, be automated to independently add new writers in the database. The database does not include the project authors own handwriting, but when added the model could be used as a biometric identifier for the mobile phone.

Technology and hardware capabilities are rapidly increasing the use of machine learning on every field. The amount of data is also exponentially increasing and



forces to develop machine learning solutions to create new value from the data. Research of new and innovative ways of using machine learning tools or just gaining general understanding of those methods is becoming each day more important.

## REFERENCES

- [1] SCHOMAKER, L., Writer Identification and Verification, 2008. [https://www.researchgate.net/publication/226888524\\_Writer\\_Identification\\_and\\_Verification](https://www.researchgate.net/publication/226888524_Writer_Identification_and_Verification) [Apr 7, 2019].
- [2] XING, L, QIAO, Y., DeepWriter: A Multi-Stream Deep CNN for Text-independent Writer Identification, 2016. <https://arxiv.org/pdf/1606.06472.pdf> [Apr 7, 2019]
- [3] HUANG, B., Q., ZHANG, Y., B., Preprocessing Techniques for Online Handwriting Recognition, 2007. [https://link.springer.com/chapter/10.1007/978-3-540-85644-3\\_2](https://link.springer.com/chapter/10.1007/978-3-540-85644-3_2) [Apr 7, 2019]
- [4] SCHLAPBACH, A., BUNKE, H., Off-line Writer Identification and Verification Using Gaussian Mixture Models, 2008. [https://link.springer.com/chapter/10.1007/978-3-540-76280-5\\_16](https://link.springer.com/chapter/10.1007/978-3-540-76280-5_16) [Apr 7, 2019]
- [5] REHMAN, A., NAZ, S., MUHAMMAD, I., R., Automatic Visual Features for Writer Identification: A Deep Learning Approach, 2019. <https://ieeexplore.ieee.org/document/8620507/authors#authors> [Apr 7, 2019]
- [6] SCHAPIRE, R., Theoretical Machine Learning, 2008. [http://www.cs.princeton.edu/courses/archive/spr08/cos511/scribe\\_notes/0204.pdf](http://www.cs.princeton.edu/courses/archive/spr08/cos511/scribe_notes/0204.pdf) [Apr 8, 2019]
- [7] CHAPPEL, D., Introducing Azure Machine Learning, 2015. [http://download.microsoft.com/download/3/B/9/3B9FBA69-8AAD-4707-830F6C70A545C389/Introducing\\_Azure\\_Machine\\_Learning.pdf](http://download.microsoft.com/download/3/B/9/3B9FBA69-8AAD-4707-830F6C70A545C389/Introducing_Azure_Machine_Learning.pdf) [Apr 8, 2019]
- [8] BROWNLIE, J., 8 Inspirational Applications of Deep Learning, 2016. <https://machinelearningmastery.com/inspirational-applications-deep-learning/> [Apr 8, 2019]
- [9] MALIK, F., Processing Data to Improve Machine Learning Models Accuracy, 2018. <https://medium.com/fintechexplained/processing-data-to-improve-machine-learning-models-accuracy-de17c655dc8e> [Apr 8, 2019]
- [10] DONGES, N., The Random Forest Algorithm, 2018. <https://machinelearning-blog.com/2018/02/06/the-random-forest-algorithm/> [Apr 8, 2019]
- [11] VIJAY, I., Probability and Statistics Explained in the Context of Deep Learning, 2018. <https://towardsdatascience.com/probability-and-statistics-explained-in-the-context-of-deep-learning-ed1509b2eb3f> [Apr 8, 2019]
- [12] SHANE, J., When Algorithms Surprise Us. 2018. <https://aiweirdness.com/post/172894792687/when-algorithms-surprise-us> [Apr 8, 2019]
- [13] MORAVEC, H., Mind Children: The Future of Robot and Human Intelligence. 1988. [Apr 9, 2019]
- [14] KURENKOV, A., A 'Brief' History of Neural Nets and Deep Learning <http://www.andreykurenkov.com/writing/ai/a-brief-history-of-neural-nets-and-deep-learning> [Apr 10, 2019]
- [15] FISHER, R., A., Iris Data Set, 1936. <https://archive.ics.uci.edu/ml/datasets/iris> [Apr 10, 2019]

- [16] BABS, T., The Mathematics of Neural Networks, 2018. <https://medium.com/coinmonks/the-mathematics-of-neural-network-60a112dd3e05> [Apr 10, 2019]
- [17] JUHOLA, M., Neural Network Basics, Neurocomputing, University of Tampere, 2017. <http://www.uta.fi/sis/tie/neuro/index/Neurocomputing2.pdf> [Apr 10, 2019]
- [18] SHARMA, A., Understanding Activation Functions in Neural Networks, 2017. <https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0> [Apr 10, 2019]
- [19] NIGAM, V., Understanding Neural Networks, From Neuron to RNN, CNN and Deep Learning, 2018. <https://towardsdatascience.com/understanding-neural-networks-from-neuron-to-rnn-cnn-and-deep-learning-cd88e90e0a90> [Apr 10, 2019]
- [20] NIKHIL, N., Is ReLU After Sigmoid Bad? 2018. <https://towardsdatascience.com/is-relu-after-sigmoid-bad-661fda45f7a2> [Apr 11, 2019]
- [21] KROTOV, D., HOPFIELD, J., Unsupervised Learning by Competing Hidden Units, 2019. <https://www.pnas.org/content/116/16/7723> [Apr 12, 2019]
- [22] RODRIGUEZ, J., IBM Draws Inspiration from the Human Brain to Build Better Neural Networks, 2019. <https://towardsdatascience.com/ibm-draws-inspiration-from-the-human-brain-to-build-better-neural-networks-ed41ace864b1> [Apr 12, 2019]
- [23] BROWNLEE, J., How to Configure the Number of Layers and Nodes in a Neural Network, 2018. <https://machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/> [Apr 12, 2019]
- [24] NIELSEN, M., Neural Networks and Deep Learning, Ch. 4., 2015. <http://neuralnetworksanddeeplearning.com/chap4.html> [Apr 13, 2019]
- [25] DABBURA, I., Coding Neural Network – Forward Propagation and Backpropagation, 2019. <https://towardsdatascience.com/coding-neural-network-forward-propagation-and-backpropagation-ccf8cf369f76> [Apr 13, 2019]
- [26] MILLER, S., Mind: How to Build a Neural Network (Part 1), 2015. <https://stevenmiller888.github.io/mind-how-to-build-a-neural-network/> [Apr 13, 2019]
- [27] SRIVASTAVA, P., About Neural Networks and Deep Learning, 2017. <https://towardsdatascience.com/about-neural-networks-and-deep-learning-8615fe366e37> [Apr 13, 2019]
- [28] SAHA, S., A Comprehensive Guide to Convolutional Neural Networks, 2018. <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53> [Apr 13, 2019]
- [29] LE CUN, Y., CORTES, C., BURGESS, C., The MNIST Database of handwritten digits, 1998. <http://yann.lecun.com/exdb/mnist/> [Apr 14, 2019]
- [30] LE CUN, Y., MATAN, O., BOSER, B., DENKER, J. S., HENDERSON, D., HOWARD, R. E., HUBBARD, W., JACKEL, L. D., BAIRD, H. S., Handwritten Zip Code Recognition with Multilayer Networks, 1991. <http://yann.lecun.com/exdb/publis/pdf/lecun-90e.pdf> [Apr 14, 2019]
- [31] SALERNOS, S., A Gentle Introduction to Convolution Filters, 2018. <https://medium.com/skylar-salernos-tech-blog/image-convolution-filters-explained-c878f1056e78> [Apr 14, 2019]

- [32] TERAQ, K., Introduction to Convolutional Neural Networks for Homogeneous Neutrino Detectors, 2017. <https://indico.fnal.gov/event/14597/>
- [33] ZEILER, M. D., Visualizing and Understanding Convolutional Networks, 2013. <https://arxiv.org/pdf/1311.2901.pdf> [Apr 14, 2019]
- [34] CHATTERJEE, S. Different Kinds of Convolutional Filters, 2017. <https://www.saama.com/blog/different-kinds-convolutional-filters/> [Apr 14, 2019]
- [35] BENSEFIA, A., TAMIMI, H., Validity of Handwriting in Biometric Systems, 2018. [https://www.researchgate.net/publication/328247881\\_VValidity\\_of\\_Handwriting\\_in\\_Biometric\\_Systems](https://www.researchgate.net/publication/328247881_VValidity_of_Handwriting_in_Biometric_Systems) [Apr 17, 2019]
- [36] BUNKE, H., IAM Handwriting Database, 1999. <http://www.fki.inf.unibe.ch/databases/iam-handwriting-database> [Apr 18, 2019]
- [37] BALTAZAR, G., CPU vs GPU in Machine Learning, 2018. <https://www.datascience.com/blog/cpu-gpu-machine-learning> [Apr 19, 2019]
- [38] KLEBAN, C., NVIDIA Tesla T4 GPUs Now Available in Beta, Google Cloud, 2019. <https://cloud.google.com/blog/products/ai-machine-learning/nvidia-tesla-t4-gpus-now-available-in-beta> [Apr 19, 2019]
- [39] BRINK, A., BULACU, M., SCHOMAKER, L. How Much Handwritten Text is Needed for Text-Independent Writer Verification and Identification, 2009. [https://www.researchgate.net/publication/224375589\\_How\\_Much\\_Handwritten\\_Text\\_Is\\_Needed\\_for\\_Text-Independent\\_Writer\\_Verification\\_and\\_Identification/download](https://www.researchgate.net/publication/224375589_How_Much_Handwritten_Text_Is_Needed_for_Text-Independent_Writer_Verification_and_Identification/download) [Apr 19, 2019]
- [40] ZHU, Y., TAN, T. WANG, Y., Font Recognition Based on Global Texture Analysis, 2001. <http://www.cbsr.ia.ac.cn/publications/yzhu/Font%20Recognition%20Based%20on%20Global%20Texture%20Analysis.pdf> [Apr 19, 2019]
- [41] YANG, W., JIN, L., LIU, M., DeepWriterID: An End-to-End Online Text-Independent Writer Identification System, 2015. <https://arxiv.org/ftp/arxiv/papers/1508/1508.04945.pdf> [Apr 19, 2019]
- [42] LECUN, Y., Gradient-Based Learning Applied to Document Recognition, 1998. <http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf> [Apr 19, 2019]
- [43] KRIZHEVSKY, A., SUTSKEVER, I., HINTON, G., ImageNet Classification with Deep Convolutional Neural Networks, 2012. <https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf> [Apr 19, 2019]
- [44] NAKAGAWA, M., NGUYEN, H., Text-Independent Writer Identification using Convolutional Neural Network [https://www.researchgate.net/publication/326602328\\_Text-Independent\\_Writer\\_Identification\\_using\\_Convolutional\\_Neural\\_Network](https://www.researchgate.net/publication/326602328_Text-Independent_Writer_Identification_using_Convolutional_Neural_Network) [Apr 19, 2019]
- [45] MCDONALD, C., Machine Learning Fundamentals: Cost Functions and Gradient Descent, 2017. <https://towardsdatascience.com/machine-learning-fundamentals-via-linear-regression-41a5d11f5220> [Apr 20, 2019]
- [46] ADAK, C., CHAUDHURI, B., BLUMENSTEIN, M., An Empirical Study on Writer Identification & Verification from Intra-Variable Individual Handwriting, 2019. <https://arxiv.org/pdf/1708.03361.pdf> [Apr 21, 2019]
- [47] SKALSKI, P., Preventing Deep Neural Network from Overfitting, 2018. <https://towardsdatascience.com/preventing-deep-neural-network-from-overfitting-953458db800a> [Apr 21, 2019]

- [48] BROWNLEE, J., A Gentle Introduction to Pooling Layers for Convolutional Neural Networks, 2019. <https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/> [Apr 21, 2019]
- [49] TAVARES, Y., Robot Navigation Using a Multilayer Perceptron Neural Network, 2017. <https://yangtavares.com/2017/08/17/robot-navigation-using-a-multilayer-perceptron-neural-network-2/> [Apr 23, 2019]
- [50] Seven Types of Neural Network Activation Functions, Neural Network Concepts, MissingLink, 2018. <https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/> [Apr 23, 2019]
- [51] CYBENKO, G., Approximation by Superpositions of a Sigmoidal Function, Chapter 1., 1989. <https://pdfs.semanticscholar.org/05ce/b32839c26c8d2cb38d5529cf7720a68c3fab.pdf> [Apr 23, 2019]
- [52] KARPATY, A., Layers Used to Build ConvNets, Convolutional Neural Networks, 2018. <http://cs231n.github.io/convolutional-networks/> [Apr 23, 2019]
- [53] BROWNLEE, J., What is the Difference Between Test and Validation Datasets?, 2017. <https://machinelearningmastery.com/difference-test-validation-datasets/> [Apr 24, 2019]
- [54] SHAH, T., About Train, Validation and Test Sets in Machine Learning, 2017. <https://tarangshah.com/blog/2017-12-03/train-validation-and-test-sets/> [Apr 24, 2019]
- [55] LIU, D., A Practical Guide to ReLU, 2017. <https://medium.com/tinymind/a-practical-guide-to-relu-b83ca804f1f7> [Apr 25, 2019]
- [56] AMIDI, A., A Detailed Example of How to Use Data Generators with Keras, 2018. <https://stanford.edu/~shervine/blog/keras-how-to-generate-data-on-the-fly> [Apr 25, 2019]
- [57] NELSON, D., Neural Net Dropout: Dealing with Overfitting, 2018. <https://www.datascience.us/neural-net-dropout-dealing-overfitting/> [Apr 25, 2019]
- [58] Frequently Asked Keras Questions, Keras Documentation. <https://keras.io/getting-started/faq/#why-is-the-training-loss-much-higher-than-the-testing-loss> [Apr 25, 2019]
- [59] KIVINIITY, J. Writer Recognition, GitHub, 2019. [https://github.com/jkiviniitty/writer\\_recognition](https://github.com/jkiviniitty/writer_recognition)
- [60] BOSTRÖM, N., What Happens When Our Computers Get Smarter Than We Are?, 2015. [https://www.ted.com/talks/nick\\_bostrom\\_what\\_happens\\_when\\_our\\_computers\\_get\\_smarter\\_than\\_we\\_are](https://www.ted.com/talks/nick_bostrom_what_happens_when_our_computers_get_smarter_than_we_are) [Apr 25, 2019]

## Convolutional neural network design of the project

Layer (type)	Output Shape	Param #
zero_padding2d_1 (ZeroPaddin	(None, 115, 115, 1)	0
lambda_1 (Lambda)	(None, 56, 56, 1)	0
conv1 (Conv2D)	(None, 28, 28, 32)	832
activation_1 (Activation)	(None, 28, 28, 32)	0
pool1 (MaxPooling2D)	(None, 14, 14, 32)	0
conv2 (Conv2D)	(None, 14, 14, 64)	18496
activation_2 (Activation)	(None, 14, 14, 64)	0
pool2 (MaxPooling2D)	(None, 7, 7, 64)	0
conv3 (Conv2D)	(None, 7, 7, 128)	73856
activation_3 (Activation)	(None, 7, 7, 128)	0
pool3 (MaxPooling2D)	(None, 3, 3, 128)	0
flatten_1 (Flatten)	(None, 1152)	0
dropout_1 (Dropout)	(None, 1152)	0
dense1 (Dense)	(None, 512)	590336
activation_4 (Activation)	(None, 512)	0
dropout_2 (Dropout)	(None, 512)	0
dense2 (Dense)	(None, 256)	131328
activation_5 (Activation)	(None, 256)	0
dropout_3 (Dropout)	(None, 256)	0
output (Dense)	(None, 50)	12850
activation_6 (Activation)	(None, 50)	0
=====		
Total params: 827,698		
Trainable params: 827,698		
Non-trainable params: 0		