DEGREE PROJECT IN INFORMATION
AND COMMUNICATION TECHNOLOGY,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, SWEDEN 2020

Interpretation of Swedish Sign Language using Convolutional Neural Networks and Transfer Learning

Gustaf Halvardsson & Johanna Peterson

KTH ROYAL INSTITUTE OF TECHNOLOGY
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Authors

Gustaf Halvardsson & Johanna Peterson
gustafha@kth.se & jpet6@kth.se
The School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology

Place for Project

KTH Royal Institute of Technology & Prevas AB
Stockholm, Sweden

Examiner

Benoit Baudry - KTH Royal Institute of Technology

Supervisors

César Soto Valero - KTH Royal Institute of Technology

Maria Månsson - Prevas AB

Abstract

The automatic interpretation of signs of a sign language involves image recognition. An appropriate approach for this task is to use Deep Learning, and in particular, Convolutional Neural Networks. This method typically needs large amounts of data to be able to perform well. Transfer learning could be a feasible approach to achieve high accuracy despite using a small data set. The hypothesis of this thesis is to test if transfer learning works well to interpret the hand alphabet of the Swedish Sign Language. The goal of the project is to implement a model that can interpret signs, as well as to build a user-friendly web application for this purpose.

The final testing accuracy of the model is 85%. Since this accuracy is comparable to those received in other studies, the project's hypothesis is shown to be supported. The final network is based on the pre-trained model InceptionV3 with five frozen layers, and the optimization algorithm mini-batch gradient descent with a batch size of 32, and a step-size factor of 1.2. Transfer learning is used, however, not to the extent that the network became too specialized in the pre-trained model and its data. The network has shown to be unbiased for diverse testing data sets.

Suggestions for future work include integrating dynamic signing data to interpret words and sentences, evaluating the method on another sign language's hand alphabet, and integrate dynamic interpretation in the web application for several letters or words to be interpreted after each other. In the long run, this research could benefit deaf people who have access to technology and enhance good health, quality education, decent work, and reduced inequalities.

Keywords

Sign language interpretation, machine learning, convolutional neural networks, transfer learning, image recognition

Abstrakt

Automatisk tolkning av tecken i ett teckenspråk involverar bildigenkänning. Ett ändamålsenligt tillvägagångsätt för denna uppgift är att använda djupinlärning, och mer specifikt, Convolutional Neural Networks. Denna metod behöver generellt stora mängder data för att prestera väl. Därför kan transfer learning vara en rimlig metod för att nå en hög precision trots liten mängd data. Avhandlingens hypotes är att utvärdera om transfer learning fungerar för att tolka det svenska teckenspråkets handalfabet. Målet med projektet är att implementera en modell som kan tolka tecken, samt att bygga en användarvänlig webapplikation för detta syfte.

Modellen lyckas klassificera 85% av testinstanserna korrekt. Då denna precision är jämförbar med de från andra studier, tyder det på att projektets hypotes är korrekt. Det slutgiltiga nätverket baseras på den förtränade modellen InceptionV3 med fem frysta lager, samt optimiseringsalgoritmen mini-batch gradient descent med en batchstorlek på 32 och en stegfaktor på 1,2. Transfer learning användes, men däremot inte till den nivå så att nätverket blev för specialiserat på den förtränade modellen och dess data. Nätverket har visat sig vara ickepartiskt för det mångfaldiga testningsdatasetet.

Förslag på framtida arbeten inkluderar att integrera dynamisk teckendata för att kunna tolka ord och meningar, evaluera metoden på andra teckenspråkshandalfabet, samt att integrera dynamisk tolkning i webapplikationen så flera bokstäver eller ord kan tolkas efter varandra. I det långa loppet kan denna studie gagna döva personer som har tillgång till teknik, och därmed öka chanserna för god hälsa, kvalitetsundervisning, anständigt arbete och minskade ojämlikheter.

Nyckelord

Teckenspråkstolkning, maskininlärning, convolutional neural networks, transfer learning, bildigenkänning

Acknowledgements

We want to thank those who have helped make this bachelor thesis possible.

Firstly, we would like to thank our supervisor César Soto Valero at KTH Royal Institute of Technology. César was always available for us when we encountered problems or when we needed guidance. This was truly helpful during the process of both coding and writing the thesis.

Secondly, we would like to thank our examiner, Professor Benoit Baudry. For guiding us through the first steps of the process and for encouraging us to conduct this project.

Further on, we would like to thank Prevas and especially our supervisor Maria Månsson who have helped us with feedback and knowledge. It was very helpful to discuss the problems and alternative solutions during every Friday meeting we had together.

Finally, we would like to thank Katarina, Patrik, and Sami, for helping us gather the Swedish Sign Language data set.

Acronyms

Adagrad	Adaptive Gradient Algorithm
Adam	Adaptive Moment Estimation
AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
ASL	American Sign Language
CNN	Convolutional Neural Network
DL	Deep Learning
DNS	Domain Name System
GCS	Google Cloud Services
GPU	Graphics Processing Unit
HTTP	Hypertext Transfer Protocol
ML	Machine Learning
MSE	Mean Square Error
NAG	Nesterov Accelerated Gradient
ReLu	Rectified Linear Unit
RNN	Recurrent Neural Network
SDGs	Sustainable Development Goals
SGD	Stochastic Gradient Descent
SSL	Swedish Sign Language
TLS	Transport Layer Security
UI	User Interface
VM	Virtual Machine

Contents

1 Introduction **1**

1.1 Background 1

1.2 Problem Formulation 2

1.3 Purpose and Objectives 3

1.4 Benefits, Sustainability, and Ethics 3

1.5 Methodology 6

1.6 Stakeholders 8

1.7 Delimitations 8

1.8 Outline 9

2 Theoretical Background **10**

2.1 Sign Language for Machines 10

2.2 Machine Learning (ML) 11

2.2.1 Types and Techniques 11

2.2.2 Artificial Neural Networks (ANN) 12

2.2.3 Backpropagation and its Optimization Algorithms 14

2.3 Image Recognition 17

2.3.1 Problem Characteristics 17

2.3.2 Image Recognition Methods 18

2.3.3 Convolutional Neural Networks (CNN) 18

2.4 Transfer Learning 21

2.4.1 General Method 21

2.4.2 Pre-Trained Models 21

2.5 Evaluation 22

2.5.1 Evaluation Methods 22

2.5.2 Training, Validation, and Testing Data 23

2.6 Related Work 24
2.6.1 Work not Based on Neural Networks 24
2.6.2 Work Based on Neural Networks 24
2.6.3 Work Based on Neural Networks and Transfer Learning 25
2.7 Summary 26

3 Methodologies and Methods 27

3.1 General Methodology 27
3.2 Research Process 27
3.3 Literature Study Phase 28
3.3.1 Literature Study 28
3.3.2 Pre-Study of Development Tools 29
3.4 Application Phase 30
3.4.1 Imported Pre-Trained Model 31
3.4.2 Generated Swedish Sign Language Data Set 31
3.4.3 Retrained Model on New Data Set 33
3.4.4 Testing 33
3.4.5 Improving Accuracy 34
3.4.6 Built Web Application 36
3.4.7 Quality Assurance 38
3.5 Summary 40

4 Results 41

4.1 Code and Data Set 41
4.2 Accuracy Testing 43
4.2.1 Optimization Algorithm and Pre-Trained Model 43
4.2.2 Hyper Parameters 45
4.3 Final Network Architecture 46
4.4 Web Application 49
4.5 Summary 51

5 Conclusions 52

5.1 Conclusions and Final Words 52
5.1.1 Were the Goals Met? 52
5.1.2 Analysis of Final Network Architecture 53
5.1.3 Analysis of Choice of Method 53

5.1.4 Analysis of Data and Target Group Ethics 54
5.1.5 Final Words 55
5.2 Future Work 55

References **57**

Chapter 1

Introduction

This chapter introduces the thesis. Section 1.1 presents the background of the problem. The problem itself is addressed in Section 1.2. Section 1.3 describes the purpose and goals. Further on, Section 1.4 presents who will benefit from the project, together with some ethical issues and sustainability aspects. Section 1.5 presents available theoretical methodologies as well as the methodologies used. Section 1.6 presents the stakeholders and 1.7 describes the delimitations for the project. Lastly, Section 1.8 presents the outline for the rest of the report.

1.1 Background

In 2018 is was estimated that 466 million people world wide had a disabling hearing loss [29]. It is not ascertain whether deafness should be considered *a gain, a loss*, or both [30]. Irrespective of which, being born deaf comes with difficulties [17]. When a person in a family turns deaf or is born with impaired hearing, several problems might emerge. Deaf people, whom often use sign language to communicate, are in many cases dependent on interpreters when for example seeking care. An application that automatically could translate sign language, could improve these people's quality of life, mainly in terms of increased social inclusion and individual freedom.

The problem is that just as with spoken languages, all sign languages differ; there is no global sign language shared over the world [10]. Therefore, just one generic translating solution is not enough in order to address this problem for all deaf people in the world. There are today no performant applications, as we know of, to help people interpreting sign language to text. If there would be one, a tool interpreting only American Sign Language (ASL) for example would not work on Swedish Sign Language (SSL). A sign language interpreting application, which is based on a solution that can easily be adjusted to interpreting many different sign languages, would be preferable.

An application to interpret sign language benefits from Artificial Intelligence (AI). AI is the science and engineering of building intelligent machines that can be fed raw data to learn on [13]. The machines can then make decisions in situations they have not encountered before. A large sub-area to AI is Machine Learning (ML) which specializes in recognizing patterns in data to continuously learn from feedback [33]. One commonly used ML architecture is neural networks. A neural network is built up of several layers of artificial neurons [20] that try to mimic the function and behaviour of biological neurons [34]. Each layer of neurons specialize in detecting different features; one layer could for example learn to detect edges when analyzing images. Since the differences between signs in sign languages largely consist of different patterns like hand movements and shiftings, using neural networks learnt on data sets of signs, could be used to develop a model to detect and interpret signs. Some studies on sign language interpretation are based on this specific idea [6, 27, 44].

1.2 Problem Formulation

People who have to use sign language to communicate with other people, are often unable to communicate effectively with people who do not know any sign language [10]. However, since sign languages are specific to each country, these people also face difficulties in communicating with all people whom do not know *their* specific sign language. An application to translate sign language to text would therefore be useful for many people.

As mentioned in Section 1.1, this type of application could be built on the idea of neural networks. The problem is that to be able train a neural network to perform adequately, large quantities of data are usually needed [3]. Regarding sign language, the only data sets available are based on ASL [1][2], there are none based on for example SSL that could

[1]The Sign Language MNIST image data set can be found at: `https://www.kaggle.com/datamunge/sign-language-mnist#amer$_$sign2.png` (based on ASL).

[2]The ASL Alphabet image data set can be found at: `https://www.kaggle.com/grassknoted/asl-alphabet#J1006.jpg`.

be used to solve this problem. While a few databases of SSL exist [3], their data only include one example per sign. To train a ML model, many examples per sign is needed, which makes it infeasible for this problem. A solution to this is to use a pre-built and pre-trained model applying transfer learning [23]. Transfer learning is a technique that is based on using models trained on one specific, large quantitative, data set. The first layers of this model is then reused in order to personalize new models based on a set of limited and specific data.

1.3 Purpose and Objectives

The purpose of this thesis is to investigate the hypothesis if transfer learning works well to interpret the hand alphabet of SSL. The goal of the project is thus to apply transfer learning to interpret the hand alphabet of SSL by building an application that can translate the signs into its corresponding form in text. The goal of the project is to fulfill the following tasks:

1. Conduct a literature study about the existing tools for translating and using sign language, and about which machine learning tools can be used to translate a static sign into text. More specifically, transfer learning is examined in order to find ways to adapt a model to SSL based on a limited set of data. Furthermore, studies about what has earlier been done in this area are examined, in order to improve the model.
2. Develop a ML model based on transfer learning that translates the static SSL hand alphabet into its corresponding text form, and test the model's accuracy. This task also involves building a web application where a static sign is entered as an input image by the user and its corresponding text form is displayed on the screen.

1.4 Benefits, Sustainability, and Ethics

After discussing the application of interpreting SSL to text with the Swedish National Association for Deaf People (Sveriges Dövas Riksförbund), several benefits in society can be distinguished [17]. This ranges from ideals of social inclusion to concrete

[3]The SSL dictionary provided by Stockholm's University can be found at: `https://teckensprakslexikon.su.se/sok/handalfabetet`

situations at the doctor's appointment. Deaf people might not have the same opportunities as hearing people, and when communication is a problem, meeting new people can be difficult. Thus, having a well developed infrastructure for deaf people mainly involves improving social sustainable development, and in it mainly to increase their global social inclusion.

Benefits and Sustainable Development Goals

The Sustainable Development Goals (SDGs) [4] for 2030 were set by world leaders at the United Nations in New York in 2015 [41]. The agreement consist of seventeen goals and 169 targets which comprise social inclusion, economic growth, and environmental protection. Research conducted after these goals were set shows that in order to be able to meet the goals focus should lie on inter linkages across sectors, across societal actors, and between and among countries at different levels of development.

If this thesis shows that transfer learning efficiently can be used to train small data sets of SSL, this idea could be replicable to other sign languages. An application that could aid communication for millions of deaf people world wide could gain several of the SDGs. These goals are:

- **Good Health and Well-Being** (goal 3). Even when an interpreter is not available, deaf people can communicate their need of care and understand the information needed to become and remain healthy.
- **Quality Education** (goal 4). All people should have the right to study, and all educations and all jobs should be available for all people. The application could aid students in their studies, with or without an interpreter available.
- **Decent Work and Economic Growth** (goal 8). Deaf people can with this application more easily work in *communication heavy* jobs.
- **Reduced Inequalities** (goal 10). The first three goals presented above all concludes to this goal. This goal will be fulfilled when all people have access to the same opportunities in society. This application could reduce inequalities by opening up situations that today are difficult for deaf people to participate in.

Connecting also to the research about interlinkages in society, this can be analysed such that the translation of sign language could simplify communication, since deafness

[4]To read more about the SDGs, visit: `https://sustainabledevelopment.un.org/?menu=1300`

is widespread in society. Easier communication could aid communication across different sectors, and increase social inclusion in civil society across societal actors such as local authorities. This in itself, is an important factor of democracy. Since this application based on transfer learning has the potential to be adapted for other languages, this application could also interlinkage low, medium, and high income countries.

Data Ethics

An important aspect of new technology and sustainable development is situations when sustainable development is not enabled, but inhibited [43]. One aspect is increased inequalities due to biased data used to train AI-models, or when the AI-model does things out of prejudice in data. To name an example regarding image recognition, the results are condemned to be biased when using data sets only based on images of people with one specific skin colour [28]. Some existing AI-models have been learned on data with people with white skin colour, and thus the applications have not worked well, or at all, on people with darker skin colour. Therefore, an important factor to include in this project is making sure the model works well on diverse data in order to enable, and not inhibit, reduced inequalities.

Target Group Ethics

As presented above, the idea of this application was discussed with the Swedish National Association for Deaf People [17]. The person we were in contact with was positive to the application and its potentials. However, this positivism does not have to apply for all deaf people. There might be a risk that this application is seen as a tool for a minority whom in this case are treated as handicapped. Another risk is that this tool is perceived as aiming to make life easier for hearing people and not for deaf people. However, we see this as an opportunity to, if the application is easy enough to use, make more room in society for deaf people and allow them to exist in rooms which they today are not a part of. The focus is not to speed up a process, but to allow these people to be in a greater part of society. Thus, a focus for us has been to make sure this product can be helpful in the long run, hopefully for example in acute care when no interpreter is available, but also in any place where deaf people today face challenges in communication.

If transfer learning is proven to work on a small data set of SSL then this idea could be replicable to many other sign languages. Thus, this research could in the long run benefit deaf people who have access to technology, and thus reduce inequalities, as long as the data used is unbiased. It could enhance four of the SDGs mainly with focus on good health, quality education, decent work, and reduced inequalities.

1.5 Methodology

The decision of methodology is important in a scientific study [14]. The methodology should be well suited to, and focused on, working towards the purpose of the project. Research methods are often defined as a spectrum of methods that aim for a search for knowledge. The research often starts with a literature study where focus is on understanding the background of the area and possible problems that need to be addressed. Further on, to reach the goals and get a result, a strategy for conducting the research is required. This is where different methodologies come in as guidelines to carry out the project. The decision is based on quantitative methodologies, supporting a phenomenon by experiments, or qualitative methodologies, studying a phenomenon to create theories or products. Based on the methodology chosen, philosophical assumptions, research methods, approaches, and strategies, data collection and analysis, and quality assurance methods need to be decided upon.

The methodology used in this project can be split into the literature study phase, and the application phase. During the literature study phase, focus is on what other researches have conducted in the area and their delimitations and analyses. The application phase focuses on using an existing pre-trained model and retraining the last layers of it in order to train it on new data, in order to get a result.

Methodology and Philosophical Assumption

The methodology used in this project is *quantitative* since experiments are conducted to support a phenomenon [14]; if transfer learning works well to interpret the hand alphabet of SSL. The philosophical assumption that is the basis of this project is *post-positivism*. The philosophical assumption is used to steer the research's assumptions. Post-positivism is positivist in the sense that it focuses on testing theories in a deductive manner to increase the predictive understanding. However, unlike positivism, post-positivism believes that the researchers' values can influence the observations. This is chosen since data used in a ML-model are chosen by individuals and commonly involves personal bias [28].

Research Method, Approach, and Strategy

The research method used in this project is *experimental* since the method focuses on finding relationships which improves the application's performance [14]. However, the research is also *applied* since it depends on transfer learning, and is thus based on other researchers' models and data. The research approach is *deductive* since it focuses on verifying hypotheses that are tested with fairly large data sets, and ending with specific accuracies. The outcome is a generalisation based on the collected data explaining the results as relationships between several variables. Further on, the research strategy is *ex post facto research* which is similar to experimental research but some part of the research is unable to control some independent variables as most of the data has already been collected. This corresponds well to this project since the data for the pre-trained model cannot be changed and is thus a factor that cannot be controlled.

Data Collection and Analysis

The data collection method used in this project is based on *experiments* and *questionnaires*. Experiments, since the data collected for the pre-trained model are based on large data sets [14]. Questionnaires, since the data collection of SSL is based on closed questions aimed to generate a generic data set. The analysis of the data is focused on *computational mathematics* since the focus is on modelling.

Quality Assurance and Presentation

The quality assurance is due to the research's deductive nature focused on *validity*, *reliability*, *replicability* and *ethics* [14]. Validity makes sure the models measure what is expected of them; that is, the code does what it is supposed to do. Reliability focuses on the consistency of the results, and thus how reliable the final results are. Replicability makes sure another researcher can conduct the same research and end up with similar results. Thus, if the method is described in clear manner so it can be repeated with similar results. Finally, the ethics is the moral principles used in planning, conducting and presenting the results. These are of focus both in choosing the pre-trained model and in collecting the data for SSL. Finally, the presentation follows the methodologies above throughout the thesis. The methods chosen, the data collection and analysis, and the quality assurance is of focus in Chapter 3. Chapter 4 focuses on the methodologies and will be shaped thereafter.

1.6 Stakeholders

This thesis is conducted in collaboration with Prevas AB. Prevas is a Swedish technical IT consulting firm focusing on several areas of industry such as energy, defence, and life science [31]. The idea of the project was presented by us to Prevas. Prevas is supporting us with knowledge and feedback from people working as consultants with technical solutions in the industry.

1.7 Delimitations

Due to the limited number of weeks available for this project, as well as our level of experience in the area, several delimitations are needed to minimize the scope of the project. The delimitations can be constricted to those regarding the sign language, and the model interpreting the signs.

Sign Language Delimitations

One delimitation regarding the sign language is that only the static signs of the SSL hand alphabet is used. Words, and four of the letters, in SSL are dynamic. They consist of several signs conducted in a specific order after each other. The remaining 25 letters in the hand alphabet consist of only one still hand gesture, and are thus static. The four letters that are dynamic are: Y, Å, Ä, and Ö. Å is for example the same sign as A, but moved around in a full circle. Any individual frame extracted from the sign Å would be interpreted as A. Y, is a sign shaped like a boat moved vertically down. Even though Y is dynamic, it can be interpreted statically since no other letter is shaped like a boat causing ambiguities, and Y is therefore included in the project. Å, Ä, and Ö, on the other hand, will not be included in the project.

When interpreting words and sentences, facial expressions are often involved when signing and this is not of focus in this project. The focus is only on hand gestures which works well for the hand alphabet. Another factor when signing is that the room and objects around the signer is often used as reference and commonly used to point at. This is not a critical factor for interpreting the hand alphabet and thus these references are not taken into consideration in this project.

Model Delimitations

One delimitation regarding the model used to interpret the signs is the use of transfer learning. This project focuses on using transfer learning for interpreting the hand alphabet of SSL, thus it does not necessarily suggest that the same thing can be done for the rest of the sign language, or for the hand alphabets of other sign languages. Furthermore, the project focuses on using particularly transfer learning, and thus other models with or without the basis of ML will not be tested. Finally, the data set used for SSL is specifically developed for this project based on us signing, and thus other possible data sources will not be used.

1.8 Outline

The thesis is organized as follows. Chapter 2 presents the theoretical background and theoretical decisions made for the thesis based on the research findings. The areas of sign languages, ML, image recognition, transfer learning, and evaluation methods, are presented. Chapter 3 describes the methodologies and methods used in the project. These include the two phases of literature study and application. Further on, Chapter 4 presents the results of the project, mainly focusing on the model, its accuracy, and the application. Finally, Chapter 5 discusses the thesis and the project, and presents some topics of future work in the area.

Chapter 2

Theoretical Background

This chapter presents the theoretical background of the thesis. Section 2.1 presents sign languages and the challenges when coding it to machines. Section 2.2 presents ML, including different types, models and subareas. Section 2.3 presents the area of image recognition; what characterizes the problem and the existing approaches to solve it. Section 2.4 discusses transfer learning. In Section 2.5, evaluation methods for ML models are presented. Section 2.6 focuses on previous research within sign language and ML. Finally, Section 2.7 summarizes the chapter.

2.1 Sign Language for Machines

The goal of this thesis is to make computers able to interpret signs of the SSL hand alphabet. In order to perform this, a model suitable for the task needs to be chosen as basis of the program. The model must overcome the following problems derived from the nature of sign languages:

1. Sign languages are modular in the sense that they depend on high level vision and high level motion processing systems for perception, and require refined motor systems of hands for production [10].
2. Signs can differ depending on who is signing [2]. The models used for automatic SSL interpretation thus need to be signer independent.

When processing sign languages by machines, these parameters need to be taken into account. Using algorithms focusing on special cases will not adapt well to different signers. Furthermore, this problem requires precise image processing. To solve this, one must probably use smart machines that can learn from data of signs and learn from patterns in the images. This is a suitable problem for ML. The signer independence can be received by training the ML model on diverse data.

2.2 Machine Learning (ML)

ML is a subarea within AI that specializes in recognizing patterns in data and by continuously learning from feedback it can generalise to new, previously unseen, data [33]. This section focuses on ML. Section 2.2.1 presents some learning types and some common ML-techniques. Section 2.2.2 presents the approach within ML that is Artificial Neural Network (ANN). Finally, Section 2.2.3 presents the most common ANN training algorithm, backpropagation, and some optimization algorithms commonly used on it.

2.2.1 Types and Techniques

ML is the art of machines learning from past experience and applying that knowledge on new situations [33]. ML is commonly divided into three subcategories: *supervised learning*, *unsupervised learning*, and *reinforcement learning* [33]. *Supervised learning* is when humans give the machine classified data. *Unsupervised learning* is learning without human supervision, and hence the data is unclassified. *Reinforcement learning* is when the machine learns from rewards. This project will rely on *supervised learning* since the data will consist of classified instances of a sign and its corresponding meaning.

Supervised learning thus learn from externally supplied instances to then make predictions about new, unseen, data [22]. There are many different supervised ML models that perform differently on different tasks and data [33]. The following discussion will only focus on some of the existing methods. One method is logic based algorithms such as decision trees and rule-based classifiers [22]. These methods are preferable when the data easily can be separated by a limited set of features [22]. There are also statistical learning algorithms such as Naïve Bayes classifiers and linear discriminant analysis. Statistical learning algorithms are suited to problems based on an explicit underlying probability model in the data. Furthermore, there are perceptron-based techniques, ANN. This model perform well on pattern recognition, and are thus to prefer in image recognition [20], which is the basis of sign language interpretation. The remaining part of Section 2.2 will hence focus on ANN.

2.2.2 Artificial Neural Networks (ANN)

ANN is a technique based on perceptrons [22]. The perceptron was first developed by Rosenblatt in 1958 as artificial neurons based on the functions of biological neurons [34]. This section focuses on ANN and begins by describing the perceptron. Further on, single and multi layered perceptrons are presented. Finally, the concept of training a neural network, and more precisely the algorithm of backpropagation, are presented.

The Perceptron

A perceptron, or artificial neuron, is a unit that mimics the behaviour of a biological neuron [34]. As presented in Figure 2.2.1, the perceptron has input feature values and a weight connected to each input value [22]. The output of the neuron is based on the sum of the weighted input. The net sum is calculated as:

$$netsum = \sum_{i=1}^{n} x_i w_i = x_1 w_1 + ... + x_n w_n$$

The threshold, or bias, is used to classify the output [22]. Every neuron has an activation function which transforms the activation level of the neuron into an output signal, based on the desired functionality [19]. Examples of common activation functions are: sigmoid, radial bases function, and conic section function.

Single and Multi Layered Perceptrons

Single layered perceptrons consist of one layer of several neurons [22]. Single layered perceptron can only classify data that is linearly separable. If the data is not linearly separable one must use multi layered perceptrons, ANNs.

Multi layered perceptrons, ANNs, have several layers of neurons [22]. The layers are interconnected with their corresponding output and input. The networks discussed in this thesis are feed forward networks where the information, or output, flows forward only. The other common type of network is called recurrent networks.

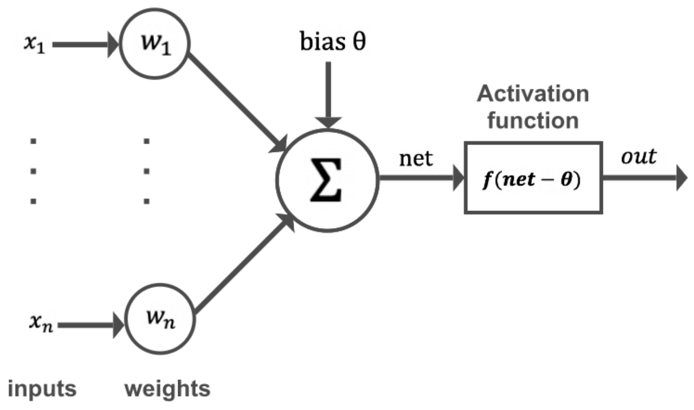


Figure 2.2.1: Figure representing the structure of a perceptron. On the left, the input values, x_i and corresponding weights w_i, are presented. These values form the net sum as they are summarized. This net sum is then, together with the bias, θ, used as input to the activation function, $f(.)$, which determines the behaviour of the perceptron. The activation function then generate the output of the neuron.

The layers in between the first input and the final output, are in ANN commonly called *hidden layers*, since their real functionality is hidden to the programmer [22]. Figure 2.2.2 shows the general architecture of a multi layered network with an input layer, two hidden layers, and an output layer. The figure was made with the help of the tool NN-SVG [1]. The more hidden layers a network has, the more complex boundaries can be formed, and thus the more complex data can be used. This is the area of Deep Learning (DL). A multi-layered perceptron network is a type of DL network [36]. Another DL technique is Convolutional Neural Network (CNN) which is particularly suitable for image recognition.

Training a Neural Network

From the start, a neural network is only perceptrons with corresponding input values, weights and biases [22]. The predictions on new data in the beginning is probably very poor. In order to achieve better accuracy the network must learn patterns that reside in the data. Networks learn by adjusting weights and biases. Learning is most commonly performed by the backpropagation algorithm [40]. Backpropagation utilizes the error in the output node and back propagates this error through the network in order to adjust the weights so the network becomes better adjusted to the patterns in data.

There are mainly two possible causes of failure in a network using the backpropagation algorithm [40]. The first is the fact that the algorithm is prone to converge to local

[1]The NN-SVG tool can be found at: `http://alexlenail.me/NN-SVG/index.html`

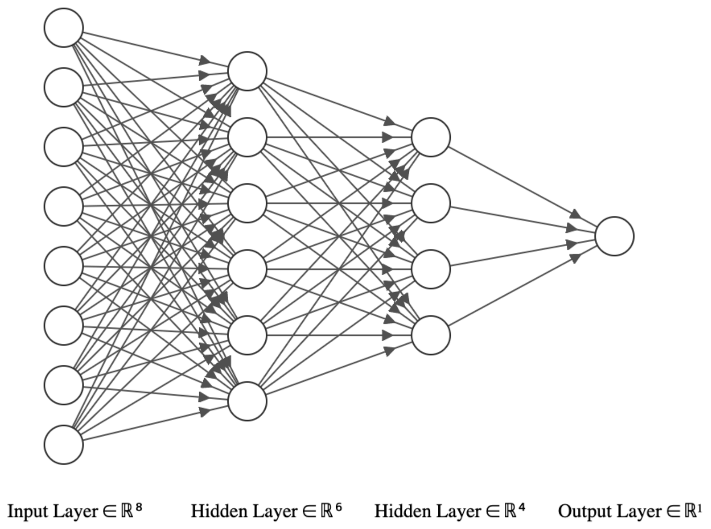

Input Layer $\in \mathbb{R}^8$
Hidden Layer $\in \mathbb{R}^6$
Hidden Layer $\in \mathbb{R}^4$
Output Layer $\in \mathbb{R}^1$

Figure 2.2.2: Figure representing a multi layered, feed forward, perceptron, an Artificial Neural Network. The nodes represent single perceptrons, and the lines represent connections between single perceptrons in the different layers. The first layer of neurons comprise the input layer. The two follow layers represent hidden layers. The final node represents the output layer of the network.

minima. The second is that it is sensitive to initial weights, and this might ruin convergence. This can be bypassed by using a well suited optimization algorithm on top of the backpropagation algorithm. There are several optimization algorithms used to optimize the backpropagation algorithm, these are discussed in Section 2.2.3.

There are other algorithms than the backpropagation algorithm used for neural networks, such as genetic algorithms [40]. These are however out of scope for this thesis, and since backpropagation is commonly used and can generally achieve satisfactory results, it will be used in this thesis.

2.2.3 Backpropagation and its Optimization Algorithms

This section focuses on the backpropagation algorithm and explores some of the optimization algorithms used on it. Firstly, the general objective of the backpropagation algorithm is presented. Secondly, the idea of gradient descent is described. Further on, the algorithms of momentum, Nesterov Accelerated Gradient (NAG), Adaptive Gradient Algorithm (Adagrad), AdaDelta, and Adaptive Moment Estimation (Adam) are briefly described.

Objective of Backpropagation

The general objective of the training algorithm backpropagation is to minimize the difference between the predicted output by the model, and the actual, desired, output [15]. The function to minimize in the algorithm is often referred to as the cost function. The cost function is minimized by adjusting weights throughout the network, starting from the output node and continuing all the way to the first layer of the network. Backpropagation is further specialized using an optimization algorithm. An iterative optimization algorithm used to reduce the cost function is gradient descent [35].

Gradient Descent

Gradient descent aims to minimize the cost function $J(\theta)$, where θ represents the model's parameters, $\theta \in \mathbb{R}^d$ in d dimensions [35]. Gradient descent updates the parameters in the opposite direction of the gradient of the cost function. Thus, movement is in the direction of the steepest descent defined by the negative value of the gradient in order to reach a minimum. In the following equations, η represents the learning rate. The bigger value of η, the greater steps are taken to reach the minimum. Furthermore, ∇_θ represents the gradient with respect to the parameters θ.

There are mainly three types of gradient descent algorithms, batch gradient descent, Stochastic Gradient Descent (SGD), and mini-batch gradient descent. Batch gradient descent uses the entire data set to compute the gradient of the cost function [35]. Due to this, it becomes slow and difficult to use for data sets that do not fit in memory. SGD, which updates the parameters for each data example $x^{(i)}$ and its corresponding label $y^{(i)}$ [35];

$$\theta = \theta - \eta \nabla_\theta J(\theta; x^{(i)}; y^{(i)})$$

SGD works faster than batch gradient descent [35]. Due to the frequent updates, the parameters will have high variance and thus the cost function will oscillate. This results in the possibility to reach local minima close to the global minimum. However, this also means there is a risk of never reaching the global minimum due to overshooting. Mini-batch gradient descent instead updates every mini-batch of n data [35];

$$\theta = \theta - \eta \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)})$$

Mini-batch gradient reduces the variance of the parameters and might thus result in a more stable convergence to the global minimum [35]. There are many ways to further optimize the gradient descent algorithm, and some of these are described below.

Momentum

Momentum handles the problem of SGD overshooting the global minimum by accelerating into the relevant directions and thus reducing the oscillations [35]. It takes the gradient of previous updates in consideration by adding a fraction of the update vector of the previous update.

Nesterov Accelerated Gradient (NAG)

NAG is a version of momentum where the probable future values of the parameters are approximated [35]. This update of the parameters allow the parameters to not follow a path up the gradient, and also slows down the steps.

Adaptive Gradient Algorithm (Adagrad)

The Adagrad algorithm focuses on the individual parameters and independently updates each parameter [35]. Adagrad modifies the learning rate for every parameter, just as momentum. To calculate the updated values for θ, a diagonal matrix is used. The values for the diagonal elements are the sums of the squares of ∇_θ. Due to the fact that the squared sum always grows throughout the training, the learning rate eventually shrinks so much that the algorithms ends up not acquiring new knowledge.

AdaDelta

The AdaDelta algorithms aims to solve the problem of the decreasing learning rate of Adagrad [35], by including a constant w which limits the total accumulated gradients. With Adadelta, η, is eliminated from the equations.

Adaptive Moment Estimation (Adam)

The Adam algorithm computes learning rates for each parameter [21]. It stores two factors that decrease exponentially: the average of past squared gradients, v_t, and the average of past gradients, m_t. The update rule is:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}} + \epsilon}\hat{m}_t$$

where β_1^t, β_2^t, and ϵ are constants, and

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

The Adam is well suited for problems that have a lot of data or many parameters [21]. It is also easy to use since it requires little tuning of the parameters. Further on, it typically requires a small amount of memory. Compared to other optimization methods it converges fast since the learning rate is high. Due to all of this, Adam will be used as main lead for the choice of optimization algorithm in this project.

2.3 Image Recognition

Image recognition is the technology of analysing patterns in images in order to classify the image as a particular object [39]. This section presents the area of image recognition starting with the problem's characteristics in Section 2.3.1. Further on, some methods of digital image recognition are presented in Section 2.3.2. Finally, Section 2.3.3 describes the structure and algorithms of the method CNN.

2.3.1 Problem Characteristics

Image recognition is about processing images and recognizing patterns that reside in it in order to recognize the image as a particular object [39]. An image recognition method generally includes four steps, as shown in Figure 2.3.1.

The first step, image acquisition, retrieves unprocessed images from a source [39] and defines class belonging for each image. These images and corresponding class will represent the data set for the model. The second step, image processing, performs some processing on the image, for example reduces the colour of the background, and finally represents every digital image frame as matrices of pixels. The third step, feature analysis, is the step where the method chosen comes in at most in analysing the features of the images [11], and finding patterns. The final step, image classification, classifies new, unseen, images to a class among the predefined classes.

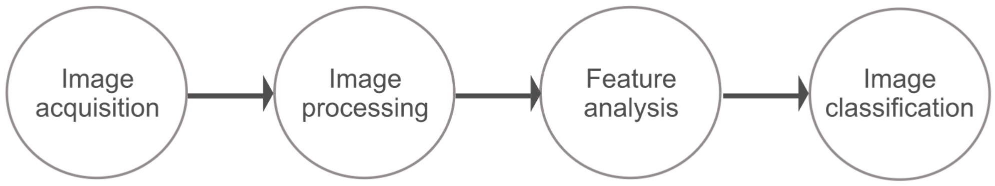


Figure 2.3.1: Figure representing the framework for image recognition technology. The first step is to acquire images for processing. The second step is processing the images. The third step analyses the different features of the images and finds patterns. The fourth and final step classifies new, unseen, images.

2.3.2 Image Recognition Methods

There are several image recognition methods, for example, statistical pattern recognition, fuzzy mathematical method, syntactic pattern recognition method, and CNN [39]. The statistical pattern recognition method is based on statistical mathematics and aims to find statistical patterns in the images [39]. The fuzzy mathematical method is based on the theory of fuzzy mathematics, that is, an extension of Boolean logic [18]. It works well when the classification needs to be fuzzy [39]. The syntactic pattern recognition method bases its analysis on the small structures in every image, such as points and lines [39]. These three methods have been widely used for image recognition, however, a common method used today is the DL method CNN [11].

The method of CNN is particularly suitable for image recognition since it automates the process of feature extraction from the images efficiently [11]. Another advantage of using CNN, as opposed to the other methods mentioned, is the fact that the other methods include feature vectors extracted using algorithms made by researchers. CNN on the other hand, does this extraction independently of the researcher, and can thus decrease bias introduced by the researchers. Since, CNN also outperforms other image recognition methods available today in accuracy [11], it will be used in this project.

2.3.3 Convolutional Neural Networks (CNN)

CNN is a DL method specialized in finding patterns in data [11]. This section will focus on CNN starting with its general structure, and continuing with the main parts of the method: convolution, pooling, flattening, full connection, and output. Finally, problems with the method are presented, and possible solutions.

General Structure

Training of a CNN is performed with the backpropagation method [37]. The general structure of a CNN is described in Figure 2.3.2. The figure was made with the help of the tool NN-SVG [2]. As seen in the figure, CNN performs feature extraction and classification. Firstly, the input images are split into small matrices of pixels. The feature extraction consist of the two processes of convoluting (and Rectified Linear Unit (ReLu)) and pooling that are applied repeatedly several times [11]. This enables the possibility to recognize specific geometrical patterns, with little data to train on [37]. Further on, the classification is performed through the three layer of flattening, full connection, and output. All of these steps are further described below.

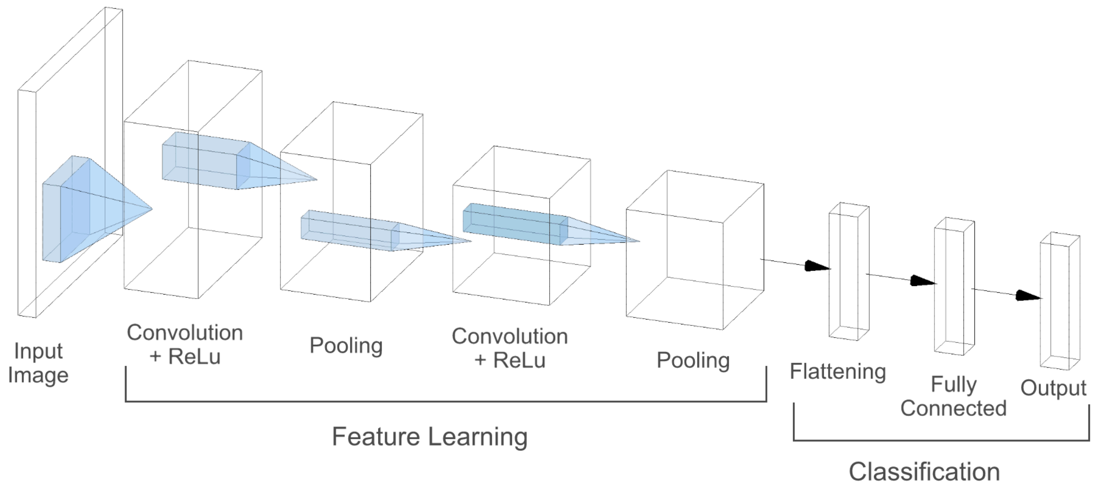


Figure 2.3.2: Figure representing the general architecture of a Convolutional Neural Network. Starting from the left in the figure, the input image is split into several smaller matrices that are used as input to the first convolutional layer. This layer performs convolving and the method of Rectified Linear Unit. The next layer performs pooling on matrices. These two steps are repeated several times and comprise the feature learning in the network. The final part, classification, consist of the methods of flattening, fully connection, and finally, output.

Convolutional Layer

The first time any input image is entering a convolutional layer, the image is split into many small images in matrix format [37]. The convolutional layer consists of a layer of

[2]The NN-SVG tool can be found at: `http://alexlenail.me/NN-SVG/index.html`

neurons each connected to a filter. A filter is a matrix of weights. The matrices from the input image are assigned to one neuron. All neurons in a layer apply the same filter on these matrices. This allows different neurons to activate on different patterns in the images. The filters perform dot multiplication with its assigned section of the input matrix. All of these scalar values are then stored as representatives for that section in a new matrix. The convolution layer then performs the step of ReLu which increases non-linearity and reduces the unwanted pixels. It replaces all negative values with zero. These values are put in a new matrix which is the new feature map that is transferred to the next layer, the pooling layer [11].

Pooling Layer

The output matrix of ReLu is passed onto the pooling layer [37]. The pooling layer further reduces the size of the convolved feature map [11]. It takes the input image and divides it into many small patches [37]. It then takes a specific value from every patch and places it in a new matrix. This specific value can either be the maximum value, minimum value, or average value. This layer downsizes the matrices and thus works as a regularizer for the network, and makes the network focus on main features. This new matrix is the new pooled feature map that is transferred to the next layer [11].

Flattening, Fully Connected Layer, and Output Layer

When all the steps of feature learning has taken place, the final pooled feature map needs to be flattened [46]. This is performed by transforming the feature map matrix into a one column matrix. The penultimate layer of the network is the fully connected layer [37]. It performs a multiplication with the flattened matrix and a weight matrix, and adds a bias vector in order to classify the images into the predefined classes. The final layer, the output layer consist of the probability of class-belonging to each class [11]. However, before outputting the results, the softmax function is used to make sure the probabilities of each class is between zero and one [46].

Problems with CNN

One problem with CNN is that big networks need high Graphics Processing Unit (GPU) performance which is often difficult to achieve on personal computers [1]. Without this, the learning will be slow. Another problem is the need for data. A possible solution for this is using a pre-trained model with the technique of transfer learning [38].

2.4 Transfer Learning

Transfer learning is a technique that leverages knowledge from one source to improve learning on another [38]. This section presents the approach of transfer learning. Firstly, in Section 2.4.1 the general method is described. Secondly, in Section 2.4.2 the concept of a pre-trained model is presented, together with several models considered for this project.

2.4.1 General Method

Transfer learning firstly uses a pre-trained DL model on some problem [23]. It does not have to be based on the same type of input data as the new source [38]. However, the performance of the new network will vary depending on which pre-trained model that is used. The first layers from this network are then frozen and put in front of some new layers that have not been trained on any data. The learned parameters from the pre-trained source are saved as a vector $\boldsymbol{\theta} = \{\theta_1, \theta_2, ..., \theta_n\}$ which is transferred to the new model together with new, specific data, for the model to train on. Transfer learning eliminates the need to train the entire network by transferring the knowledge of the pre-trained model and thus reducing the need for large quantities of data.

2.4.2 Pre-Trained Models

The pre-trained models considered for this project are all available for use with Keras [3], which is the DL Application Programming Interface (API) that will be used for this project. All models are specialized in image classification, trained on the data set ImageNet [4].

In Table 2.4.1 the models that are considered for this project are presented. The second column, after the name column, presents the models' accuracies on the ImageNet data set. This number is important to take in consideration when choosing model since the better the model performs on the ImageNet data set, the better starting conditions for the new model [7]. The third column in the table presents the number of parameters

[3]To read more about Keras, visit: `https://keras.io/applications/`

[4]For more information about the image data set ImageNet and its data, visit: `http://www.image-net.org/`

representing the model. The higher this number is, the more time and space will it take to train the new network. Too many parameters can lead to a slow and memory expensive process on personal computers, however, too few parameters will likely make the pre-trained model less fit for use on new data.

Table 2.4.1: Table showing eleven pre-trained models available from Keras. All models are trained on the ImageNet data set. Each model is presented with its accuracy on the ImageNet data set, and the number of parameters in the pre-trained model. The table is sorted on highest accuracy.

Model	**Accuracy** [**%**]	**Parameters**
InceptionResNetV2	80.3	55,873,736
Xception	79.0	23,910,480
InceptionV3	77.9	23,851,784
ResNet50V2	76.0	25,613,800
DenseNet121	75.0	8,062,504
ResNet50	74.9	25,636,712
NASNetMobile	74.4	5,326,716
MobileNetV2	71.3	3,538,984
VGG16	71.3	138,357,544
VGG19	71.3	143,667,240
MobileNet	70.4	4,253,864

2.5 Evaluation

This section focuses on the evaluation of an ANN. Section 2.5.1 presents some common evaluation methods for ML models. Section 2.5.2 presents the concept of splitting the available data into a training, validation, and testing set.

2.5.1 Evaluation Methods

Evaluating a ML model is about finding the difference between the predicted output by the model, and the actual output [15]. This defines the model's accuracy. One common evaluation method, or error function, used on ANN is the Mean Square Error (MSE) [8]. The MSE is calculated as the average of the accumulated squared difference between the predicted output and the desired output. Another common metric used on ML methods is the misclassification error on the data set [25]. This is calculated as the average number of correct classifications:

$$err(f, D) = \frac{1}{N} \sum_{i=1}^{N} lnd(f(\mathbf{x_i} \neq y_i)$$

where $lnd(f(\mathbf{x}) = 1)$ if x is true, otherwise $lnd(f(\mathbf{x}) = 0)$. One method commonly used on image recognition problems based on a DL network is to use a loss function [11]. A distance is here the distance between the feature vectors of the current pattern and the input image. The distance between two images of the same object is considered small, whilst the distance between two images of different objects is considered large.

Two metrics will be tested in the project to evaluate which one of them performs best on the data. The metric of misclassification error will be used to maximize the accuracy. The metric of using a loss function will be used to minimize the loss on the data. These two metrics were chosen since they are commonly used by other researchers [8]. The two metrics will be evaluated in the beginning of the testing process and the metric producing the highest accuracy on the data will be the one used for the rest of the project.

2.5.2 Training, Validation, and Testing Data

When training a network, two data sets are present [45]. The *training set* is used to train the network, and the *validation set* is used to improve the network's performance. Models trained on a set of data will if validated on the same data not give any relevant information. The data used for training and validation should be different of each other in order to improve the model. The size of the two sets is often determined with k-fold cross validation. This method makes sure the model does not become too specialized on training data with too many data examples in the training set, this is called overfitting [5]. It also makes sure the training data set is big enough for the model to learn the true patterns that reside in the data. A third data set is used when the network is finished training and the final accuracy on new, unseen data, is wanted [45]. This data set is often called the *testing set*.

2.6 Related Work

There have been several studies aimed at interpreting sign languages. Some have been based on gloves that can interpret signs, others on computer vision and statistical comparisons. Furthermore, there are studies focusing on both SSL and CNN. This section will present some of these studies. Section 2.6.1 focuses on studies performed without neural networks, while Section 2.6.2 focuses on studies with neural networks. Finally, Section 2.6.3 presents studies on sign language and transfer learning.

2.6.1 Work not Based on Neural Networks

This section focuses on three studies performed on interpreting sign language by human supervision.

The first study was based on the the signer wearing an AcceleGlove when signing [16]. The different angles per finger was then used as input to a computer program that analysed the positions. It was tested on different people and was able to recognize 30 one-handed signs with an accuracy of 98%.

The second project was based on computer vision and statistical comparisons [12]. Each sign was filmed in a controlled environment, the background was extracted, the image was cropped, resized and edge detected, and finally placed in an adaptive statistical database. To classify a sign as a particular word, the image was processes and then compared to all images in the statistical database.

The third study used a Kinect sensor to recognise SSL signs [2]. The signer used a RGB-D Kinect sensor placed in the hand. This allowed the backgrounds to be removed, helped with the resolution when the hand was placed in front of the face, and simplified the use of 3D signs. The classification was done through a statistical database. The results were different depending on who signed the signs. One signer received an accuracy of 77% while one had 94%.

2.6.2 Work Based on Neural Networks

The studies presented in this section are based on neural networks. The benefits of using a neural network is its ability to derive meaning from patterns too complex to be noticed by humans or traditional algorithms [42].

The first study focused on gesture recognition and proposed sign language translation as a possible application [44]. The project was conducted with a CyberGlove, a glove with virtual reality sensors. The analysis was conducted by a multi layered ANN, and the accuracy was close to 100% for some gestures.

The second project aimed at recognizing a stream of continuous signs in a video [27]. The computer architecture used was Recurrent Neural Network (RNN). The network thus had feedback connections which is suitable for video processing. The project reached an accuracy of 80% on a continuous stream of video data.

The third project aimed to translate signs filmed with a webcam [32]. The study aimed to translate the Auslan Sign Language alphabet. The data set for the Auslan alphabet was generated by extracting signs from YouTube videos and drawing boxes over the hands. Further on, CNN was used and the final accuracy was 86%.

The final research focused on translating a continuous stream of sign language sentences [6]. They specifically wanted to improve the interpretations when it comes to grammar. The project was built on CNN and attention based encoders and resulted in many sentences being correctly interpreted.

2.6.3 Work Based on Neural Networks and Transfer Learning

The studies presented in this section are based on neural networks and transfer learning. The data sets used on the pre-trained models have all been limited.

The first research was based on interpreting British Sign Language [26]. The data sets of British Sign Language used were a corpus for standard English with transcriptions to sign language and a pre processed corpus called Penn Treebank. The videos from the corpuses were split by sentences. The ML architectures used were both based on RNN and CNN. The study showed good results on words, however sentences were not interpreted grammatically correct.

The second research was based on interpreting Indian Sign Language [9]. The data used for Indian Sign Language was based both on images per sign but also depth images which reduced the pre-processing time and also allowed for better 3D processing. Further on, a pre-trained model based on the ImageNet data set was used to increase the accuracy of the limited data set. Then several methods, including CNN, was applied on the pre-trained model. The optimization algorithms AdaDelta and Adam were used.

They achieved an accuracy of 66%. However, when applying the pre-trained model their accuracy became lower and they concluded that they would have needed a larger data set (>1200).

2.7 Summary

This chapter presented the theoretical background for the project. Important parameters for sign language interpretation were presented including high level motion processing and signer independence. Based on these needs, the area of ML was introduced, focusing on supervised learning and ANN since it is a classification problem that involves pattern recognition. The aspects of perceptrons and training a neural network were presented together with the common training method, the backpropagation algorithm, and some of its optimization algorithms. Furthermore the problem characteristics and methods of image recognition were presented. Most focus was put on the suitable method of CNN, its structure, layers, and problems. Transfer learning was then presented as a solution to one of CNN's problem, namely the need for large amounts of data. Transfer learning was presented with its general method and some available pre-trained models. Further on, the area of evaluating a ML network was described, some common methods and the concept of using a separate training and validation set. The final part of the chapter was dedicated to related work in the area of sign language interpretation, both without neural networks, with neural networks, and with transfer learning.

Chapter 3

Methodologies and Methods

This chapter contains descriptions of the methodologies and methods used in the project, as well as the practical description of the work performed during the project. Firstly, Section 3.1 presents the general methodology for the project. Further on, Section 3.2 describes the research process, presenting the two phases of the project: the literature study phase, presented in Section 3.3 and the application phase, presented in Section 3.4. Finally, the chapter is summarized in Section 3.5.

3.1 General Methodology

The project's methodology was quantitative, since the whole project aimed at investigating the hypothesis if transfer learning works to interpret the SSL hand alphabet. The focus of the project was to build an application by using a pre-trained model by adding and retraining layers to train the network on SSL data. The philosophical assumption, post-positivism, was applied such that some different pre-trained models, optimization algorithms, and hyper parameter values, were tested in order to get a robust and reliable accuracy.

3.2 Research Process

The research process that permeated the project consisted on two phases: the literature study phase and the application phase. These phases, together with their corresponding activities are presented in Figure 3.2.1. As seen in the figure, the literature study phase consisted of two activities: the literature study and the pre-study of development tools. The application phase consisted of six activities, all of which included quality assurance. The first activity of the application phase was to import pre-trained models, then the SSL data set was generated. Further on, the model was retrained on the pre-trained model and the new data set. This model was then tested and its accuracy was improved in several cycles. Finally, the web application was built based on the model. The rest of this chapter is dedicated to these two phases.

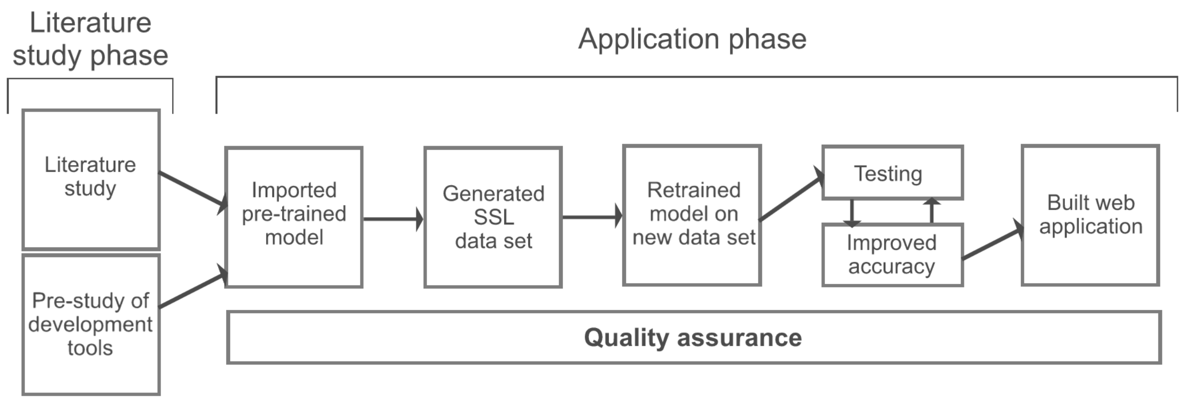


Figure 3.2.1: Figure presenting the research process of the project which consists of two phases: the literature study phase and the application phase. The literature study consisted of a literature study and pre-study of development tools. After this was completed, the application phase started. The first activity was to import pre-trained models. Then the Swedish Sign Language data set was generated. Following this, the new model based on the pre-trained model and the new data was trained. The model was then retrained and the accuracy was improved in several cycles. Lastly, the web application was built based on the model. The quality assurance aspects permeated the whole application phase.

3.3 Literature Study Phase

This section presents the activities conducted in the first phase on the project, the literature study phase. Section 3.3.1 presents the literature study, and Section 3.3.2 presents the pre-study of development tools used during the application phase.

3.3.1 Literature Study

The project started with a thorough literature study about the needs of better tools for translating sign language, what architectures and models can be used for translation, and what previous studies have performed and achieved in the area. The focus was on understanding the background to the problem together with possible difficulties that needed to be addressed in the project. Google Scholar [1], ScienceDirect [2] and IEEE Xplore [3] were the main sources for articles. The search words used to find articles were: sign language interpretation, artificial neural networks, gradient descent, image recognition, convolutional neural networks, transfer learning, pre-trained models, optimization algorithms, and evaluating machine learning models.

3.3.2 Pre-Study of Development Tools

The next step in the literature study phase consisted of a pre-study of suitable development tools. This section presents these tools. The argument to why most tools were used were if other computer scientists have used them and if the tools were well established. Since DL still is considered a new area, many models and tools are still untested. The following tools are the tools used in the project:

- **Flask.** Flask [4] is a web application micro-framework for Python. Flask was used since it easily can turn a ML model into an API which served as back end for the application.
- **GitHub.** GitHub [5] is a platform for hosting software engineering projects. This was used for version control and as a collaboration tool.
- **Google Kubernetes Engine.** Google Kubernetes Engine [6] is a platform for hosting serverless web applications in Google's cloud. This was used as hosting platform for the API created by Flask.
- **Google Colab.** Google Colab [7] offers an online Virtual Machine (VM) which allows developers to use their GPU. This was used for training the models in the project to make the training faster and not overwhelm our personal computers.
- **Heroku**. Heroku [8] is a platform that offers a hosting service for apps. This was used for hosting the frontend of the web application.

[1]Google Scholar can be found at: `https://scholar.google.se/`
[2]ScienceDirect can be found at: `https://www.sciencedirect.com/`
[3]IEEE Xplore can be found at: `https://ieeexplore.ieee.org/Xplore/home.jsp`
[4]For more information about Flask, visit: `https://flask.palletsprojects.com/en/1.1.x/`
[5]For more information about GitHub, visit: `https://github.com/`
[6]For more information about Google Kubernetes Engine, visit: `https://cloud.google.com/kubernetes-engine`
[7]For more information about Google Colab, visit: `https://colab.research.google.com/`
[8]For more information about Heroku, visit: `https://heroku.com`

- **OpenCV**. OpenCV [9] is an open source library for image processing. This was used to load and modify images to generate the data sets.
- **Python3.** Python is a programming language with support for ML in the form of tools and libraries. Python was used as language for the model and API since it is an established standard for ML development.
- **React.** React is a JavaScript-library used for frontend web applications. React was used to build the front end for the web application due to earlier experience in the library, and its simplicity to create a good-looking front end.
- **TensorFlow and Keras API.** TensorFlow [10] is an open source platform for ML. TensorFlow is well established and used by many researchers and developers. Keras [11] is a high-level API based on TensorFlow for ML-models. Keras was used to save time when developing since it is high-level, and there was good documentation for the API.
- **Weights & Biases.** Weights & Biases [12] is a visualization tool for monitoring the model and hyper parameters. It was used when training the model to help visualise which combination could yield the highest accuracy.

3.4 Application Phase

The second phase of the project, the application phase, is presented in this section. Firstly, Section 3.4.1 presents the use of pre-trained models. Section 3.4.2 presents the generation of the data set of SSL. Section 3.4.3 focuses on how the model was retrained on the new data set. Further on, Section 3.4.4 presents the testing of the model. Section 3.4.5 describes the actions taken for improving the model's accuracy. The building of the web application is presented in Section 3.4.6. Finally, Section 3.4.7 presents the steps taken for quality assurance that permeated the whole phase.

[9] For more information about OpenCV, visit: `https://opencv.org/`
[10] For more information about TensorFlow, visit: `https://www.tensorflow.org/`
[11] For more information about the Keras API, visit: `https://keras.io/`
[12] For more information about Weights & Biases, visit: `https://www.wandb.com/`

3.4.1 Imported Pre-Trained Model

The first step of the application phase was to choose and integrate a pre-trained model. The overall research method was both experimental and applied, and this part of the project was more oriented towards an applied research method. A pre-trained model was used as basis of the program to make better predictions on a small data set. With using the pre-trained model, it also established the usage of ex post facto research strategy since the pre-trained models basis could not be changed in the process. The data for the pre-trained model was already collected, processed, and the model had been trained on it; there were some variables, such as the type and characteristics of images sued for training the model, that could not be controlled. This was the part of the data collection which was based on experiments, since the pre-trained models considered all were based on a large data set.

The pre-trained models were imported as a pre-existing module through Keras. They were then possible to use in library calls, to for example test the model on data. The last classification layer was detached from the model and some layers were frozen in order to add new layers to specialize on the new data. The process of choosing and using the pre-trained models is further described in Section 3.4.5.

3.4.2 Generated Swedish Sign Language Data Set

The next step was to collect the data of SSL. This part of the data collection was based on questionnaires since, what would be referred to as the questions that were used in this case, were aimed to generate a generic data set. The first two steps in the framework for image recognition were conducted in this step of the process. The third and fourth steps, feature analysis and image classification, were performed when the model was retrained and tested with the new data set.

Image Acquisition

The first step, image acquisition, was performed by filming all the letters in the hand alphabet during 20 seconds, for a total of eight times by five people. All 26 letters were recorded eight times, and every recording was performed signing with the person’s right hand. The recordings were performed as presented in Figure 3.4.1. Only a part of the body and face were included, the person was centered, and their hand was centered in front of the chest. The figure was created using assets from Flat Icon [13].

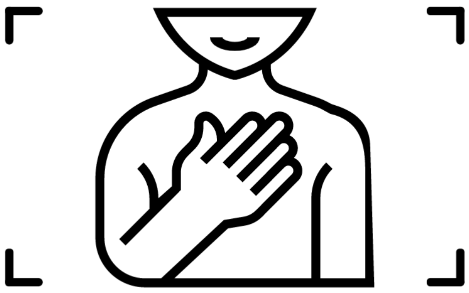


Figure 3.4.1: Figure representing how the signing videos were staged regarding the position of the person's body and hand.

The eight recordings became the basis of the training, validation, and testing data sets. The signs were based on the SSL dictionary provided by Stockholm's University [14]. The conditions between the recordings were varied by background and clothes. When filming, the hands were slightly rotated to allow for more variance. Each letter was then classified as belonging to a specific letter.

The data sets were thus collected by five different people. This part of the method connected to the consciousness of ethics in society and technology. This was one part of the project where the technology could become an inhibitor and not an enabler for sustainable development. By collecting diverse data, it could improve the ML models ability to generalise and different types of signers and hands.

Image Processing

The second step, image processing, was now performed on each video. A function was created to read all videos per letter and created one frame per 0.1 second so every recording became 200 images. These were then resized to a size of 224×224 pixels. 100 of these images were placed in the training set, 50 in the validation set, and 50 in the testing set. All images were put into repositories on GitHub, cloned into the code and then merged once on the Colab VM in order to be used as data. To prevent memory overflow, the images in the training data set were split up into batch sizes of 32 that were sent into the model in batches.

The training images were then augmented in order to allow for more noise in the data. The testing images were also augmented in order to create a more realistic real-life

[13]Figure 3.4.1 used assets from the icon made by Eucalyp from `https://www.flaticon.com/`
[14]The SSL dictionary can be found at: `https://teckenspraakslexikon.su.se/sok/handalfabetet`

testing experience. The steps performed for both data sets were rotation, shifting height and width, zooming, and tilting, to allow for more variations of signing. When shifting and tilting, the points now outside of the boundaries of the image were filled with the nearest pixel. The testing images were also altered on brightness. No image augmentation was performed on the validation data set. The image augmentation was performed with the Keras class ImageDataGenerator [15].

3.4.3 Retrained Model on New Data Set

When the pre-trained model was imported and the new data set generated, the training of the new model began. This was done by adding extra layers. Three convolutional layer were added. The first two consisted of 1024 nodes each, the third consisted of 512 nodes, and they all used ReLu. These values of the numbers of nodes were chosen since they were frequently used in other researches and yielded satisfying results. The pooling layers used global average pooling. Finally, softmax was used to ensure that the probabilities ended up between zero and one. The first 20 layers of the pre-trained model were frozen (model weights became immutable) in order to not retrain them, but keep their knowledge. Then the model was trained on the new data with the new layers in order to specialize on the new data. This was the step of feature analysis in the image recognition framework. The final step of image classification then occurred when testing the model and finally towards the end of the project, when using the application.

3.4.4 Testing

Before starting to improve the accuracy of the model, the evaluation method needed to be decided upon. This focused on the research's approach, deductive, which ends with specific accuracies of the model's performance. As the model used was a neural network, the outcome would be a generalisation based on the collected data over several runs explaining the results as relationships between several variables.

The model was trained on 50% of the data, validated on 25%, and tested on 25%. The evaluation of accuracy on the validation data was both performed through the loss function and the total misclassification error on the validation data. The models

[15]To read more about the Keras class ImageDataGenerator, visit: `https://keras.io/preprocessing/image/`

were non-deterministic and thus each run of the model would generate a slightly different accuracy. Therefore each run was redone a certain number of times in order to get an average accuracy. Several training sessions on ten epochs, with different parameters were tested, as will be presented in Section 3.4.5. These runs were used when comparing different networks and thus the differences between the networks were of importance, not its absolute performance. In order to present the final accuracy of the model, the final network architecture was trained during 30 epochs. This run was then tested on the testing data to present the final accuracy.

A dynamic video tool was developed to visualize the model's performance on single signs. This was also developed to be able to detect several signs in a row and help evaluate the different models accuracy and behaviour in real-life situations and with fast feedback. It worked by making predictions continuously on every frame from the webcam video stream and showing the result live. Some basic helper functions like backspace, delete, and reset, were also added. The tool was only developed to be run locally for testing purposes as deploying that functionality on the web was out of scope for this project.

3.4.5 Improving Accuracy

One part of the research that focused on the experimental aspect of the project's research method, was the choice of which pre-trained model to use, which optimization algorithms to use, and the tuning of hyper parameters. The method of improving the accuracy was conducted in two steps. Firstly, some pre-trained models were tested with different optimization algorithms. Secondly, two hyper parameters were tuned based on the combination of pre-trained model and optimization algorithm that had the highest accuracy form the previous step.

Pre-Trained Model and Optimization Algorithm

Since training the models would take a lot of time, approximately three hours per training session on Colabs GPUs, three out of all of the pre-trained models presented in Table 2.4.1 were chosen to be tested. These were decided upon based on their accuracy, thus InceptionResNetV2, Xception, and InceptionV3 were tested. This was performed in the code by changing the imported model as well as changing a function call as can be seen in Figure 3.4.2.

```
1 from tensorflow.keras.applications import InceptionV3, Xception,
     InceptionResNetV2
2 ...
3 pre_trained_base_model = InceptionV3(weights='imagenet', include_top=False,
     input_shape=(224, 224, 3))
```

Figure 3.4.2: Figure that shows the code for alternating between the different pre-trained models. On line one is the function call for importing the different pre-trained models from the tensorflow.keras module. On line three the models used for that run were changed according to the specific pre-trained model tested. Line two represents the multiple lines of code that lies between line one and three.

The three pre-trained models were then tested on two optimization algorithms. Due to the same reasoning as above, not all algorithms were selected to be tested. The two methods tested were mini-batch gradient descent and Adam since they were commonly used by other researchers. Thus, all three pre-trained models were tested on the two optimization algorithms, correspondingly. The tests were conducted using a fixed batch size for the optimization algorithm of 32, and on ten epochs.

Based on these tests, one architecture was chosen by using the Wilcoxon signed rank test. The Wilcoxon signed rank test is a non-parametric statistical test that investigates if two dependent populations are of the same distribution [24]. This test was appropriate to use in this project since it can determine, with statistical significance, if one model combination can be used over another based on its overall performance. The combinations of different pre-trained models were used in the tests to determine if the models performed as an equal distribution or not and thus which validation accuracies were the most *reliant*. The final combination with most statistical significance were chosen as the final combination. With the final combination determined, the tuning of the rest the parameters could be performed.

Tuning of Hyper Parameters

The tuning of hyper parameters began with one combination of pre-trained model and optimization algorithm. This was done with the help of the tool Weights & Biases [4]. It allows sweeping of several parameters in order to help find the parameter values that maximize the networks performance. The following two parameters were tuned: the number of frozen layers of the pre-trained model, and the step size factor. The step size factor is a factor which determines the relationship between the total number of training data examples and the batch size, which results in the learning rate for the back propagation algorithm. The number of frozen layers determines the impact the pre-trained model should have on the final outcome. The step size factor was tested on three values, namely 0.2, 0.7, and 1.2. Furthermore, the number of frozen layers were tested on three values, namely, 5, 20, and 50.

3.4.6 Built Web Application

After the final network architecture was in place, the process of building the web application was initiated. The application was divided into two components, a front end and a back end. Further more, a cloud architecture was used to provide fast predictions.

Front End

The front end development was done using the React library for JavaScript. Thus, code was written in JavaScript, HTML, and CSS. The front end's purpose was to serve as the interface for the model which was served by the back end.

The main focus of the front end was to make a flow and design that was easy to use and navigate through. This meant only focusing on the core features that the user would need in the app, and eliminating everything else. The core features were uploading an image and getting a prediction from that image. This minimal approach was also applied to the User Interface (UI) where only the UI-elements necessary to complete a prediction and a few helper elements where used. The application had three states (Start, Uploaded Image and Result) and with two different routes (image source from either uploaded fie or webcam). All UI-elements followed the same colour scheme of black for important-text, gray for less important, and blue for UI-elements like buttons. The application was stateless between predictions, meaning the app doesn't pertain any data between predictions or sessions.

Back End

The back end was written in Python3 as a API. This API was used for the front end, the client, to communicate with. Flask was used to assist the endpoints and server-modules. The code generating the model produced a .h5-file. The back end's job was to load that file, and use it to make new predictions over Hypertext Transfer Protocol (HTTP) requests. The back end had a single endpoint (/predict), in which a binary encoded image was accepted via a Transport Layer Security (TLS)-encrypted HTTP POST-request. The models top three predictions for that image, with respective confidence in percent, was returned as a response.

Cloud Architecture

In order to make fast predictions, the back end API was deployed from a Docker image [16] that was made for this project, in a Kubernetes cluster via Googles Kubernetes Engine as an always-running service. This way the model was pre-loaded on the server to avoid time-costly operations for every request. To be able to handle TLS-encrypted requests, some additional components were needed.

Firstly, a trusted Secure Sockets Layer-certificate was issued via Google Cloud Services (GCS) and connected to a domain name (sign-interpreter.com). The domain name was issued via Amazon Route 53 that served as the outer endpoint for the API. The domain name was then configured with Domain Name System (DNS) A-record to point to the IP-address of the ingress-controller which was hosted as a service on GCS.

The ingress controller handled all incoming requests. By using its verified certificate, it was used as a proxy to forward the request to another internal GCS-service, a load-balancer. The load-balancer was used to split the incoming traffic between three nodes that independently handled requests as well as exposed the nodes to the ingress-controller. The nodes were contained in a Kubernetes cluster to handle workload scaling. One out of three nodes was used to handle the request and return the prediction to the client. The full network diagram of the cloud architecture can be seen in Figure 3.4.3 which was done using the tool Lucid Charts[17].

The use case of the development of the application involved several cycles of development and reviews. A few persons were asked to do reviews on the application, mainly in the end of the process. This was done both in order to get a natural flow through the application but also to make sure the users understood how to navigate through the application.

[16]The Docker Image can be found at: `https://hub.docker.com/r/gustafvh/sign-interpreter-ssl-api`

[17]The diagram tool Lucid Charts can be found at: `https://www.lucidchart.com/`

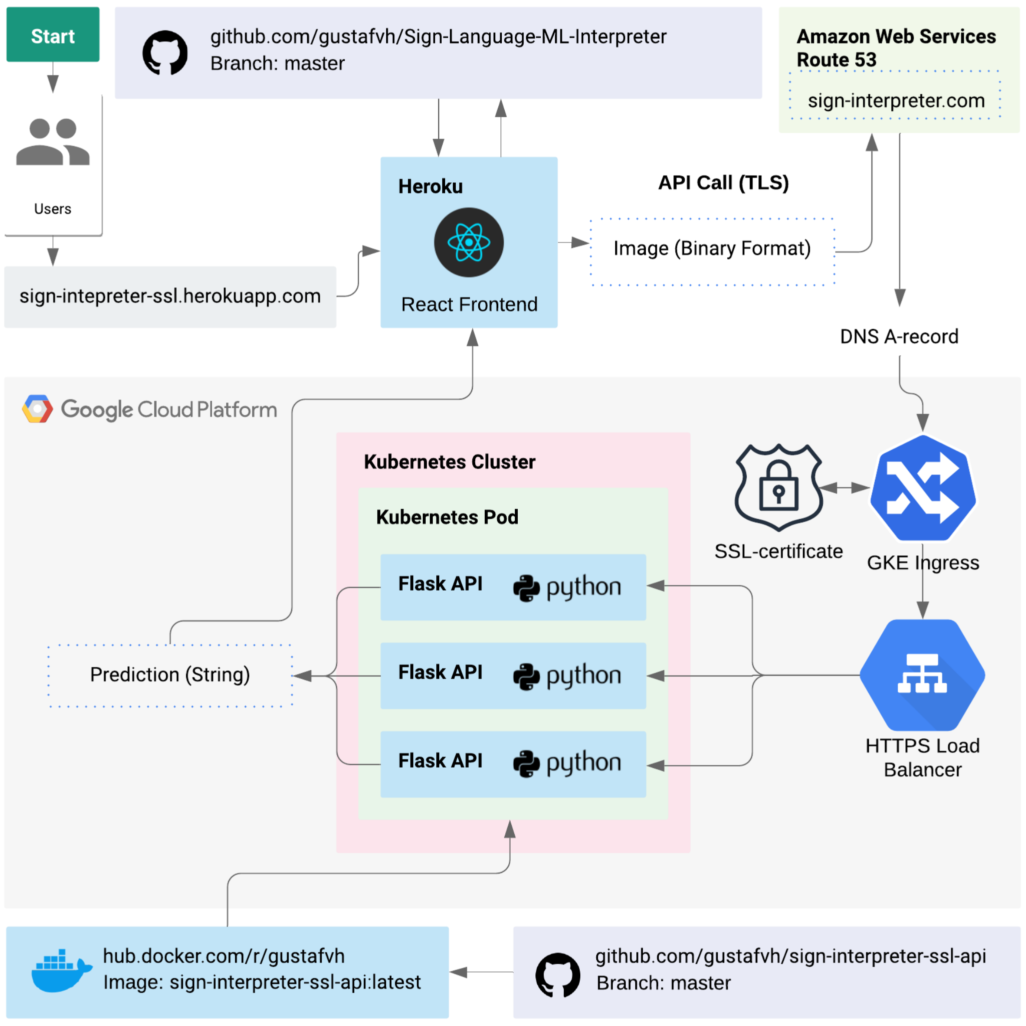


Figure 3.4.3: Cloud architecture of the web application. The arrows represent flow of data for an API request when a user wants to analyse a sign. Starting at the top left corner, the user enters the URL in a browser to the Heroku-hosted application which pulls its source code from GitHub. The front end (client) makes a POST request with the sign image now binary encoded, to the sign-interpreter.com domain, which points to the cloud services. That entry point is an ingress that validates the identity with a certificate to be able to handle the HTTPS-protocol. The request is then forwarded to the load-balancer which sends it to one of three Flask Python back ends that return a prediction to be returned to the client and displayed in the application. The Python back ends are based on Docker images living on Docker Hub and GitHub.

3.4.7 Quality Assurance

This section focuses on the quality assurance that permeated the application phase. Since the research was deductive, the quality assurance focused on validity, reliability, replicability, and ethics.

Validity

The focus on validity made sure the models measured what was expected of them; that is, the code did what it was supposed to do. Since DL models include hidden layers where the real functionality is hidden to the programmer, many aspects of how to develop DL relied on testing and experience of when to stop changing the model. The validity thus relied on repeated cycles of testing of the model.

Reliability

Reliability wise, the project focused on the consistency of the results, and thus how reliable the final results would become. The model created was non deterministic and thus one run of the model could produce slightly different results than another. By running the model several times per try, this uncertainty could decrease and thus increase the reliability of the network.

Replicability

A focus of replicability made sure another researcher can conduct the same research and end up with similar results. As presented above, the network was non deterministic, and thus total replicability was difficult to achieve. However, the method was described thoroughly in order to increase the possibility of replicability of the project. Another important factor of replicability is that the different runs on Google Colab did not guarantee the same VM, and thus not the same GPU which, although unlikely, might have affected the results. The replicability of the project was increased by the use of the Wilcoxon signed rank test.

Ethics

Finally, the ethics was of focus as the moral principles used when planning, conducting and presenting the results. These were of importance when the pre-trained model was chosen, when the data was generated, the final outputs of the network, and the availability of the application. The pre-trained model could have been trained on bias data and thus generating that bias to the final model. The SSL data set could also have been biased in the form of our decisions. Finally, the final output of the network might be difficult to trace back because of the hidden layers and backtracking in a big network. Thus, the importance of keeping these factors in mind throughout the process was significant. Another factor of ethics is availability. By deploying the application to the web, the availability was ensured for all with an internet connection and a webcam or camera. Finally, ensuring that the application was easy to use to make sure as many people as possible could use the tool.

3.5 Summary

The project consisted of two phases; the literature study and the application phase. The first phase consisted of both a literature study and a pre-study of development tools. The main development tools used were Flask, GitHub, Google Kubernetes Engine, Google Colab, Heroku, OpenCV, Python3, React, Tensorflow and Keras API, and Weights & Biases. The application phase consisted of six activities. Quality assurance permeated all of these activities. Firstly, a pre-trained model was imported and several layers of it were frozen. Secondly, the SSL data set was generated and the images were processed to allow for more difference in the data. Further on, the model was retrained on the new data set. Following this, several cycles of testing and improving the accuracy of the network was performed. This was mainly focused around determining which combination of pre-trained model, optimization algorithm, and values of hyper parameters to use. Finally, the web application could be built. Both the minimal frontend which acts as the interface as well as the backend that serves the model.

Chapter 4

Results

This chapter presents the results of the degree project. The chapter starts with Section 4.1 which presents the code for the model and application, and the final data set for SSL. Further on, Section 4.2 presents the results of the accuracy tests performed where the network with the highest accuracy was chosen as final. Section 4.3 presents what the final network architecture and its accuracy on the testing data. Section 4.4 presents the web application. Finally, the chapter is summarized in Section 4.5.

4.1 Code and Data Set

All code and data for the project was stored in three different GitHub repositories where each repository contains a README file with more information of its content. The code for the model, data generator, and front end for the application, can be found in one GitHub repository [1]. The code for the back end API for the application was stored separately and can be found in another GitHub repository [2].

The data set can be found in the GitHub repositories for data [3]. The data set was collected by five different people, in a total of eight recordings. Figure 4.1.1 presents one representative image per person, or recording, that was included in the data set. More information about each recording is shown in Table 4.1.1.

[1]The following link contains the GitHub repository for the model, data generator and front end for the application: `https://github.com/gustafvh/SignInterpreterSSL`

[2]The following link contains the GitHub repository for the back end API code: `https://github.com/gustafvh/SignInterpreterSSL_API`

[3]The following links contain the GitHub repositories for the SSL data set: `https://github.com/johannakin/SSL-data-1`, `https://github.com/johannakin/SSL-data-2`

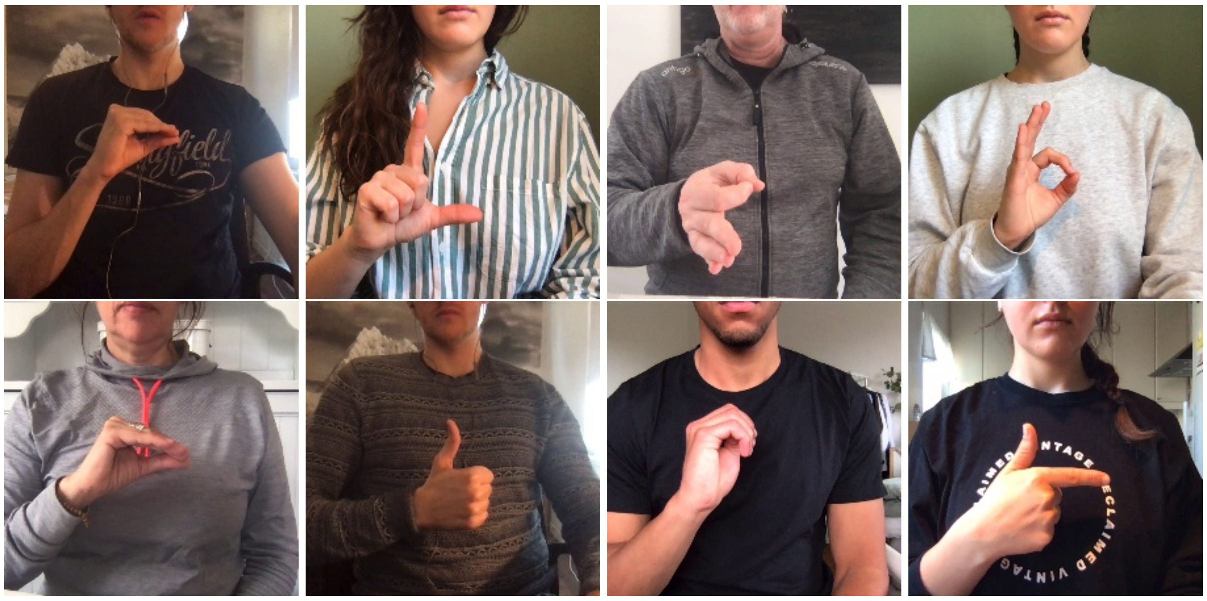


Figure 4.1.1: Figure displaying eight images in the data set for SSL. The images are showed in the same order as the recordings shown in 4.1.1. Each image represents one of the persons, or recordings, that was included in the data set. The images represent the letters A, L, P, H, A, B, E, and T.

Table 4.1.1: Table showing information about the eight recordings included in the data generation. Each row represents a recording. The columns represent the person's age, gender, and the total number of generated images for that recording.

Age	**Gender**	**No. of Generated Images**
22	Male	5 191
23	Female	5 200
53	Male	5 200
23	Female	4 401
53	Female	5 200
22	Male	5 077
24	Male	5 200
23	Female	5 134

The columns in Table 4.1.1 represent the person's age and gender and the total number of generated images for that recording. Due to the fact that different computers recorded with different frames per seconds rate (24 or 30), four out of the eight recordings did not sum up to 5 200 which they could have according to the method. Since the difference between the recordings, as shown in Table 4.1.1, is not significantly large, this was not regarded as something that needed to be changed. The final data set was split up into three parts; training, validating, and testing. All recordings were represented in each part. The parts consisted of, in order, 20 700, 10 334, and 9 569, images.

4.2 Accuracy Testing

This section focuses on the tests conducted to find the tuning of the network with the highest resulting accuracy. This was performed in two steps, first with three different pre-trained models and two different optimization algorithms, presented in Section 4.2.1, and then by tuning the hyper parameters, presented in Section 4.2.2.

4.2.1 Optimization Algorithm and Pre-Trained Model

These first tests were conducted using two different metrics as stated in Section 2.5.1. Firstly, the metric of using a loss function was used, aiming at minimizing the validation loss. Secondly, the metric of misclassification error was used, aiming at maximizing the validation accuracy. Minimizing the validation loss resulted on average in 20% lower accuracies on the data. Therefore, maximizing the validation accuracy was used for the rest of the project.

The results of the tests of the pre-trained models and optimization algorithms are presented in Table 4.2.1. The table shows that the differences in accuracy between the different combinations are small. Thus, the Wilcoxon signed rank test was used to validate which combination proved to perform best on the data.

Table 4.2.1: Result, measured in accuracy, of the testing of pre-trained models and optimization algorithms. The columns correspond of the three pre-trained models tested, and the rows correspond to the two optimization algorithms. The results were obtained by basic layers added to the pre-trained models, and the new Swedish Sign Language data set.

	Pre-trained models		
Optimization algorithms	InceptionResNetV2	Xception	InceptionV3
Adaptive Moment Estimation (Adam)	88.51%	94.77%	91.77%
Mini-batch gradient descent	87.98%	91.63%	94.78%

The Wilcoxon Signed Rank Test

Since the runs with InceptionResNetV2 produced significantly lower accuracies than the other pre-trained models, they were excluded from the Wilcoxon test. However, Xception and InceptionV3 produced similar results and thus the test was used to see if there were any statistical significance in the data that could be used to determine which combination performed better.

The first test kept Xception static while altering the two optimization algorithms. The second test kept InceptionV3 static while altering the two optimization algorithms. The data used for the tests were the validation accuracies received per epoch of training for the different combinations. The null hypotheses for both tests were that the combinations were of the same distribution. The code and corresponding p-value outputs are presented in Figure 4.2.1.

```
1  from scipy.stats import wilcoxon
2
3  Inc_adam = [0.2590, 0.6840, 0.7553, 0.7047, 0.5302, 0.5226, 0.7160, 0.7684,
       0.0449, 0.9177]
4  Inc_sgd = [0.8275, 0.9223, 0.9047, 0.9435, 0.9474, 0.9163, 0.9363, 0.8756,
      0.9191, 0.8849]
5
6  Xce_adam = [0.3528, 0.6779, 0.67791, 0.7688, 0.8358, 0.9233, 0.8558,
      0.9477, 0.7798, 0.8602]
7  Xce_sgd = [0.7542, 0.8007, 0.8842, 0.8937, 0.9163, 0.8593, 0.8484, 0.9100,
      0.9009, 0.8747]
8
9  pValue_Inc = wilcoxon(Inc_adam, Inc_sgd)
10 pValue_Xce = wilcoxon(Xce_adam, Xce_sgd)
11
12 print('P-Value of InceptionV3 test:', pValue_Inc) # prints: ('p-Value of
      InceptionV3 test:', 0.006910429807814799 5)
13 print('P-Value of Xception test:', pValue_Xce) # prints: ('p-Value of
      Xception test:', 0.046853284788147 15)
```

Figure 4.2.1: Figure that shows the code used to perform the Wilcoxon signed rank tests on the two pre-trained models InceptionV3 and Xception. Line one imports the necessary package from SciPy. Line three to seven include four arrays containing the the validation accuracy received per epoch for the corresponding combinations. Furthermore, line nine and ten presents the calculation of the p-values for the two tests. Finally, line twelve and thirteen prints the p-values for the tests, which are shown as comments at the end of each line.

As seen in Figure 4.2.1, the p-value of the test with InceptionV3 was approximately 0.0069 and thus the null hypothesis could be rejected at a confidence level of over 99%. This meant that with InceptionV3, the optimization algorithms were not of the same distribution with a certainty of over 99%. This means that the combination that provided the highest accuracy could be used. Since mini-batch gradient descent performed a higher accuracy than Adam for InceptionV3 as shown in Table 4.2.1, it passed on to the next step of the evaluation. Approximately the same reasoning could be used regarding the test with Xception since it produced a p-value of approximately 0.0469. Its null hypothesis could also be rejected, however here at a confidence level of over 95%. Since Adam performed a higher accuracy than mini-batch gradient descent as shown in Table 4.2.1, it was passed on to the next step of the evaluation. The two final combinations to evaluate were thus InceptionV3 with mini-batch gradient descent, and Xception with Adam. As seen in Table 4.2.1, they received a validation accuracy of 94.78% and 94.77% respectively. Thus, further tests were needed to show which performed better on the data. To evaluate this, a box plot was used to show the variance and consistency in validation accuracies for the different epochs.

Evaluating Spread and Consistency of Validation Accuracy

As seen in Figure 4.2.2, InceptionV3 with mini-batch gradient descent had a low spread in the validation accuracies per epoch and produced consistently high accuracies. Xception with Adam however, showed a large variance in the results as well as consistently lower accuracies than the other combination. This shows that Adam was more prone to get stuck in local minima. Thus, even though the two combinations performed equally as stated in Table 4.2.1, InceptionV3 and mini-batch gradient descent was shown to be more reliant since the validation accuracies per epoch was more consistent. This increased reliance was an advantage since it simplifies the tuning of hyper parameters as every run was more prone to deliver good results. This combination also increased the replicability of the project. Thus, the combination of InceptionV3 and mini-batch gradient descent was used as basis for the final network architecture.

4.2.2 Hyper Parameters

The hyper parameters for the network based on InceptionV3 and mini-batch gradient descent was now tuned. Two hyper parameters were in focus; the step size factor and the number of frozen layers of the pre-trained model. The results from these tests are presented in Figure 4.2.3. The figure shows that the two combinations with the highest validation accuracy was with step size factor 1.2 and five frozen layers, and with step

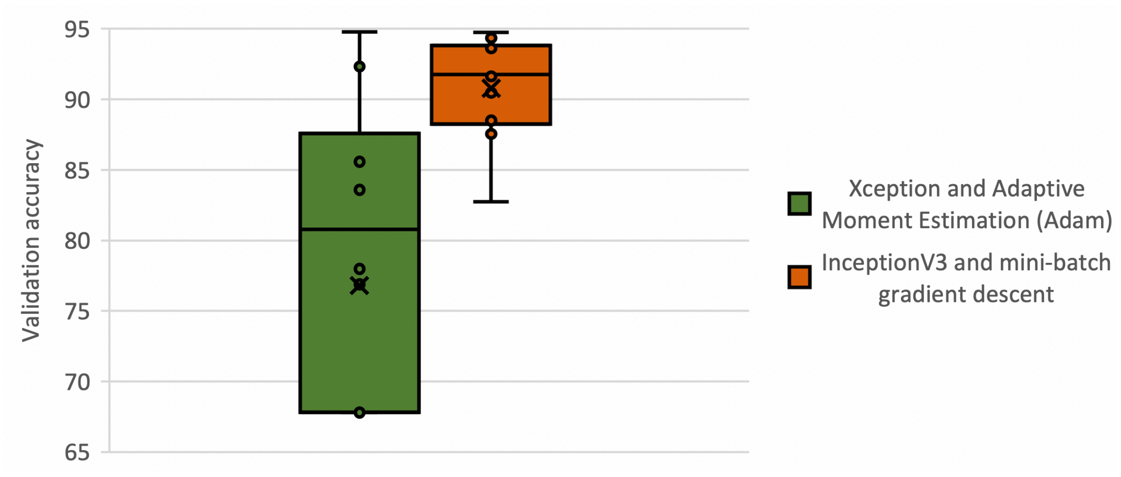


Figure 4.2.2: Box plot displaying how the validation accuracies of the two combinations, Xception and Adaptive Moment Estimation (Adam), as well as InceptionV3 and mini-batch gradient descent, varied over the ten epochs. The boxes include 50% of the data points and thus shows the main distribution. The lines over and under the boxes represent the highest and lowest validation accuracies received for the different combinations.

size factor 0.7 and 50 frozen layers. These two combinations were trained two times more each in order to find the average accuracy over several runs. This resulted in an accuracy of 96.46% for the combination of a step size factor of 1.2 and five frozen layers, and an accuracy of 93.90% for the combination of a step size factor of 0.7 and 50 frozen layers. Thus the final values of the hyper parameters was five frozen layers and a step-size factor of 1.2. The value of the step size factor, corresponds to a learning rate close to, but bigger than the number of training data samples divided by the batch size.

4.3 Final Network Architecture

The accuracy tests ended with one final network architecture with the highest validation accuracy. It consisted of the following properties: the pre-trained model was InceptionV3, the optimization algorithm was mini-batch gradient descent, the batch size was 32, the step-size factor was 1.2, and the number of frozen layers of the pre-trained model was five.

To come up with the final architecture, several decisions were made and several tests were conducted. Firstly, only three pre-trained models were tested, based on the highest accuracy as presented in Table 2.4.1. Then, mini-batch gradient descent and

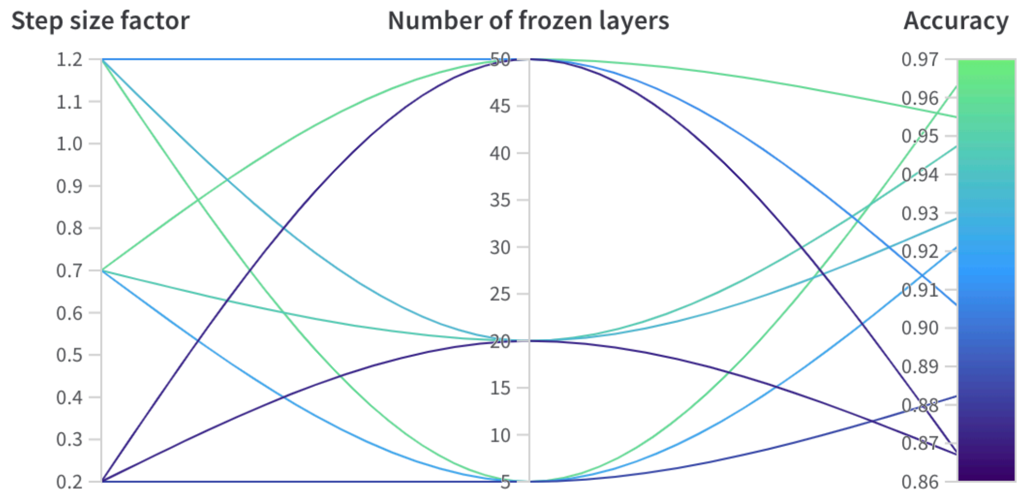


Figure 4.2.3: Figure displaying the results from the tuning of hyper parameters based on InceptionV3 and mini-batch gradient descent. Two hyper parameters were tuned, the step size factor, and the number of frozen layers of the pre-trained model. The graph includes nine different combinations of the hyper parameters and presents the combinations' validation accuracies after training. The figure was made automatically by the tool Weights & Biases [4].

Adam were chosen as leads on optimization algorithms. Mini-batch gradient descent was chosen since it is a middle way between the basic approaches of SGD and batch gradient descent and is more stable than the others to converge to the global minimum. Furthermore, Adam was chosen since it generally converges fast and is well suited for problems with many parameters. The combination of InceptionV3 and mini-batch gradient descent was then chosen since it produced the most reliable results based on the Wilcoxon signed rank test as presented in Section 4.2.1. Three values of step-size factor and number of frozen layers were tested. These numbers were based on the recommendation from Keras, and then one smaller number and one bigger. The hyper parameter values of a step-size factor of 1.2 and five frozen layers were chosen since they performed better than other values on the combination of pre-trained model and optimization algorithm as presented in Section 4.2.2

On top of the pre-trained model, three convolutional layer were added. The first two with 1024 nodes, the third with 512 nodes, and they all used ReLu as activation function. The pooling layers used global average pooling. Finally, softmax was used to ensure that the probabilities ended up between zero and one. In order to present the final accuracy of the network, it was trained one last time, this time during 30 epochs. The final model, as it was a convolutional DL network consisted of 25 488 698 parameters and 316 layers [4]. The training and validation loss and accuracy over the 30 epochs can be seen in Figure 4.3.1. As seen in the figure, the training and validation loss softened after approximately twelve epochs. The overfitting seen in epoch two and nine, was eliminated with more epochs. The final model was then tested on the testing data set. The final testing accuracy was 85%.

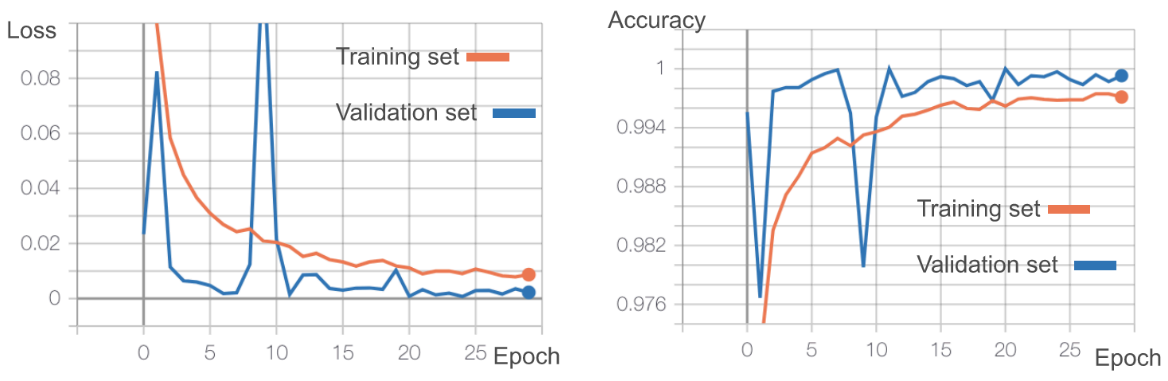


Figure 4.3.1: Figure presenting the loss and accuracy of the training and validation set per epoch for the final model. The first figure presents the loss, and the second figure presents the accuracy on the training and validation set. The figures were made automatically by the Tensorboard tool in the tool Weights & Biases [4].

The dynamic video tool that was used when testing the models' performances can be seen in Figure 4.3.2. This particular picture was taken based on the final model.

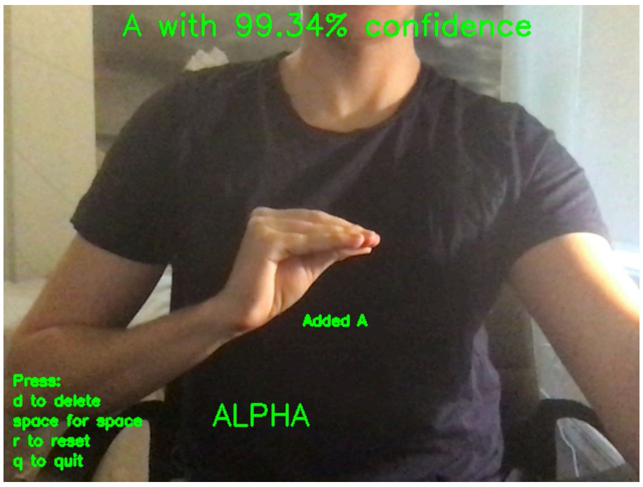
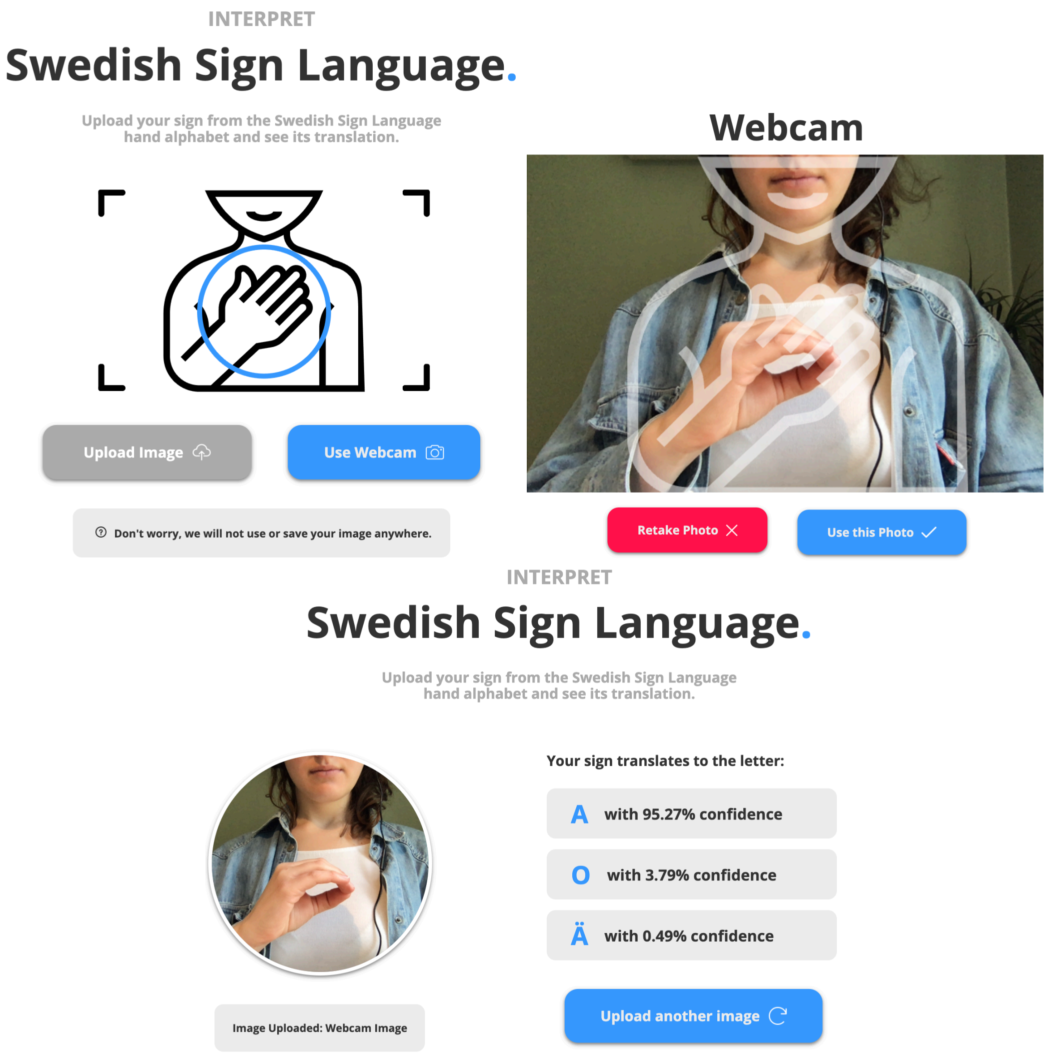


Figure 4.3.2: Figure representing a caption of the video tool used for testing. This picture presents how a list of letters were signed and added to a word. The bottom left of the picture shows the basic helper functions included to be able to sign words.

[4]All layers and parameters for the network can be found at `https://app.wandb.ai/sign-interpretor/sign-interpreter/runs/y11of78x/files/model-best.h5`

4.4 Web Application

Based on the final model of the SSL interpreter, the web application was built [5]. Some screenshots of the application are presented in Figure 4.4.1 and Figure 4.4.2. Figure 4.4.1 presents the steps of the application when the user wishes to take a picture of a sign with their webcam. Figure 4.4.2 presents the steps of the application when the user wishes to use an existing picture from their gallery. The home page, presented as first image in both figures contains information in text and the picture shows how the sign should be signed to be able to interpret it correctly.


INTERPRET

Swedish Sign Language.

Upload your sign from the Swedish Sign Language hand alphabet and see its translation.

Don't worry, we will not use or save your image anywhere.

Webcam

INTERPRET

Swedish Sign Language.

Upload your sign from the Swedish Sign Language hand alphabet and see its translation.

Your sign translates to the letter:

A with 95.27% confidence

O with 3.79% confidence

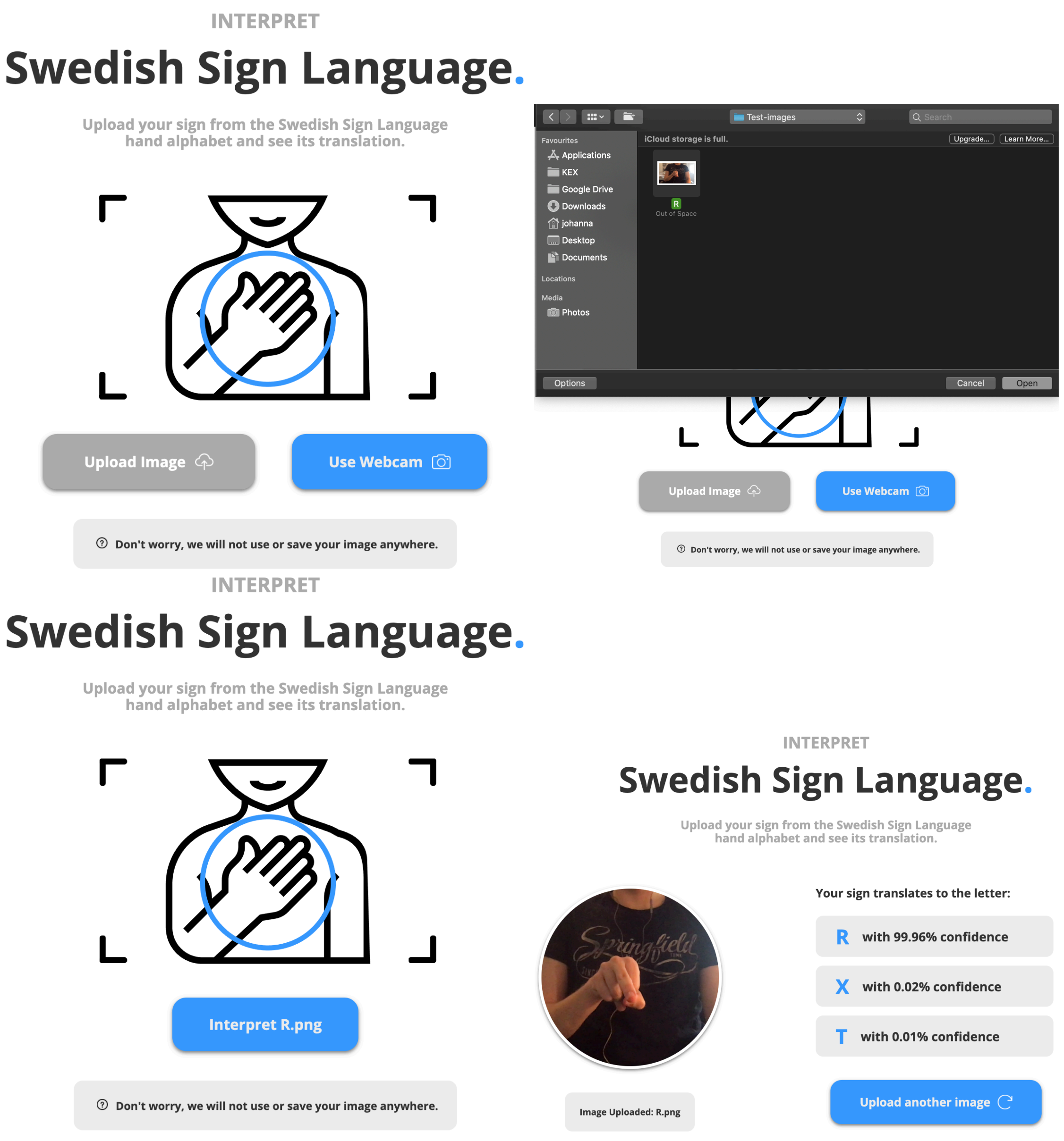
Ä with 0.49% confidence

Image Uploaded: Webcam Image

Figure 4.4.1: Figure displaying the user flow of the web application when using the webcam to take a picture. The first image presents the home page where a button for using your webcam is placed on the bottom right. The second image presents the page of the webcam where the user can take, and re-take, a picture. The final image presents the results page that shows the sign with highest confidence on the top.

[5]The final version of the application can be found at: `https://sign-interpreter-ssl.herokuapp.com/`

In order to make the web application user friendly, several aspects were considered. Firstly, all unnecessary steps and information was excluded in order for the application to be as easily used as possible. Secondly, a layer was placed on top of the webcam recording, as seen in the second image in Figure 4.4.1, in order to show how the image should be taken for the best results.


INTERPRET

Swedish Sign Language.

Upload your sign from the Swedish Sign Language hand alphabet and see its translation.

Don't worry, we will not use or save your image anywhere.

iCloud storage is full.

INTERPRET

Swedish Sign Language.

Upload your sign from the Swedish Sign Language hand alphabet and see its translation.

Don't worry, we will not use or save your image anywhere.

INTERPRET

Swedish Sign Language.

Upload your sign from the Swedish Sign Language hand alphabet and see its translation.

Your sign translates to the letter:

R with 99.96% confidence

X with 0.02% confidence

T with 0.01% confidence

Image Uploaded: R.png

Figure 4.4.2: Figure displaying the user flow of the web application when uploading an image. The first image presents the home page where a button for uploading your image is placed on the bottom left. The second image presents the pop-up that appears for the user to pick their image of a sign. The third image presents the page that displays a button to interpret the uploaded image. The final image presents the results page that shows the sign with highest confidence on the top.

4.5 Summary

The absolute results of the project can be split into the accuracy tests and the final network architecture. The accuracy tests showed highest performance when using InceptionV3 and mini-batch gradient descent as basis. The tuning of the hyper parameters with the highest accuracy was a step size factor of 1.2 and with five frozen layers. The final network architecture reached a testing accuracy of 85%. This network consisted of 316 layers and 25 488 698 parameters. Based on the final model, trained during 30 epochs, the web application was built.

Chapter 5

Conclusions

This chapter concludes the thesis. Section 5.1 presents analyses and final words, and Section 5.2 discusses future work that can be conducted on the basis of the thesis.

5.1 Conclusions and Final Words

This section analyses and concludes the project. Section 5.1.1 states how the goals were met. Section 5.1.2 analyses the final network architecture. Section 5.1.3 presents the method used and its advantages and disadvantages. Further on, Section 5.1.4 discusses the ethics of the generated data set and the target group. Finally, Section 5.1.5 presents some final words on the project.

5.1.1 Were the Goals Met?

The first goal of the project was to conduct a literature study. The second goal was to develop a ML model based on transfer learning and test its accuracy. This second goal also included building a web application that uses the model to make predictions.

The literature study was conducted and the questions of how to manage the difficulties of interpreting sign languages were answered by using supervised ML and specific optimization algorithms, image processing styles, layers, pre-trained models, and evaluation types.

The model based on transfer learning was developed. The final network reached a testing accuracy of 85% and thus supported the initial hypothesis that transfer learning can be used to translate the hand alphabet of SSL. Furthermore, the problem of having only a small data set of SSL-data proved to be solvable by using transfer-learning with a pre-trained model. The web application was created and works accordingly.

5.1.2 Analysis of Final Network Architecture

The final network architecture was presented in Section 4.3. There are several parameters in this network that are important to analyse in order to understand the network and its performance. The final network architecture was based on five frozen layers of InceptionV3. Thus, transfer learning was used, however not to the extent that the network became too specialized on the pre-trained model and its data.

Since the network is a DL network, and thus non-deterministic, consisting of 316 layers, some conclusions and relationships might be difficult to see or analyse. Therefore, the accuracy of the network is important in order to evaluate the model. It is also important to minimize the non-deterministic effect by including repeated cycles of testing of the model. This was performed in order to ensure the validity and reliability of the project.

The accuracy of a network is based on the data. If the testing data is too similar to the training data, nothing can be said about how the model performs in real-life situations. If it does not perform well, this could be due to the model failing to generalise over new unseen data. To allow for diverse testing data, several activities were performed. The person recording the signs for the data varied the angle and position frequently to allow for more variance in the data. That was combined with data augmentation to make the data as diverse as possible. Also, the data sets were separated, so when the network was tested on the testing data, it was on images it had not seen before. Another way of generating diverse testing data could have been to gather more data in many different situations and by many different people. The choice of testing data greatly affects the networks reliability and is therefore important to make separate from the training data.

5.1.3 Analysis of Choice of Method

Based on the difficulties of sign language interpretation presented in Section 2.1 regarding high level motion processing and signer independence, ML and specifically CNN were chosen as basis architectures for the project. Further on, since the data set generated was limited, the method of transfer learning was integrated in the project. The data set for SSL was generated by including five different people. Different optimization algorithms of the backpropagation algorithm for CNN were examined. Furthermore, several different pre-trained models were also tested, and as well as different values of the hyper parameters, step-size factor and number of layers frozen.

Section 2.6 presented several studies in the area of sign language interpretation. Previous research on sign language interpretation without neural networks presented accuracies of 98% [16], 77% [2], and 94% [2]. Research with neural networks presented accuracies of 86% [32], 80% [27], and close to 100% [44]. Finally, research with neural networks and transfer learning presented accuracies of 66% [9]. Thus, our accuracy of 85% compares well to other studies and is thus shown as a valid source of interpreting the hand alphabet of SSL.

By using neural networks, the impact of human supervision is minimized and some patterns too complex for human's perception can be derived meaning from. However, the use of neural networks also means adding an abstraction layer which humans have no control in understanding or changing. This means the algorithm might do things it is not supposed to or derives too much meaning out of a binary situation. Since the previous studies presented above that did not use neural networks also presented high accuracies, these methods must not be neglected on the basis of new and more complex techniques. Methods including both transparency as well as complex analyses would therefore be the most preferable.

5.1.4 Analysis of Data and Target Group Ethics

There are several aspects to consider regarding ethics in this project, mainly based on data and target group. Regarding the data, it is important to generate data without bias in order for the network to be able to correctly classify different types of hands and conditions in the image. The choice of testing data is important in order to be able to evaluate the model based on this. The testing accuracy showed that the model developed in this project works well on some diverse data and is thus not biased for one type of hand.

Regarding the target group, making the application easy to use and available was important. A possible disadvantage of these types of application might however be an excessive confidence in technology. The technology might produce inaccurate interpretations and without a critical approach, misinterpretations might aggravate communication. Another critical factor of developing interpretation tools is if they become widely used, and many people rely on them instead of learning the sign language, personal contacts might suffer.

5.1.5 Final Words

Since transfer learning is supported to work on a small data set of SSL, this idea could perhaps be replicable to many other sign languages. Thus, this research could in the long run benefit deaf people who have access to technology. It could enhance good health, quality education, decent work, and reduced inequalities.

If we had known everything we know now, after the project has been conducted, there are some things we would have done differently. We would have gathered more data, and especially from more people and in more various situations, to be a part of the generated data set. This in order to make sure the data is not now specialized on the five different hands used in this project. Another factor we would have considered is allowing for left hand signing in order to make the model less signer dependent. Furthermore, we would have considered using a more rigorous statistical comparison of the most appropriate network structure using for example the Friedman's test base.

5.2 Future Work

This thesis supported that transfer learning can be used to translate the hand alphabet of SSL. There are several aspects of this work that has been left undone and thus there are some suggestions for future work in the area.

One suggestion is to integrate dynamic signing in the method perhaps with the help of visual tracking in order to interpret words, and following that, sentences. One important factor to remember when doing this is that SSL is for example not Swedish *on hands*, the grammar is completely different. Focus when interpreting must lie on epentheses (insertion of sound in a word) and phonological parameters unique to specific sign languages. Since several previous studies on continuous interpretation are based on RNN, that might be a good lead for this work.

Another suggestion is to test to use the same network architecture, but use data for another sign language's hand alphabet and see if transfer learning works well on that sign language as well.

The final suggestion is to adapt the application to allow for a continuous video stream of live interpreting which would allow spelled words to be interpreted. This could be performed by deploying the same functionality the local video testing tool provided, to the web application. This would be helpful in order to make the application more useful in a signer's every day life with faster interpretation of words and sentences.

Bibliography

[1] Ahmadvand,
P., Ebrahimpour, R., and Ahmadvand, P. “How Popular CNNs Perform in Real Applications of Face Recognition”. In: *24th Telecommunications Forum* (2016). DOI: 10.1109/TELFOR.2016.7818876.

[2] Akram, S., Beskow, J., and Kjellstrom, H. *Visual Recognition of Isolated Swedish Sign Language Signs*. 2012. URL: https://arxiv.org/abs/1211.3901.

[3] Atkinson, P.M. and Tatnall, A.R.L. “Introduction Neural networks in remote sensing”. In: *International Journal of Remote Sensing* 18 (4 1997). DOI: 10.1080/014311697218700.

[4] Biewald, L. *Experiment Tracking with Weights and Biases*. Software available from wandb.com. 2020. URL: https://www.wandb.com/.

[5] Briscoe, E. and Feldman, J. “Conceptual complexity and the bias/variance tradeoff”. In: *Cognition* 118 (1 2011), pp. 2–16. DOI: 10.1016/j.cognition.2010.10.004.

[6] Camgoz, N.C., Hadfield, S., Koller, O., Ney, H., and Bowden, R. “Neural Sign Language Translation”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 7784–7793.

[7] Canziani, A., Paszke, A., and culurciello, E. *An Analysis of Deep Neural Network Models for Practical Applications*. 2016. URL: https://arxiv.org/abs/1605.07678.

[8] Christiansen, N.H., Torbergsen Voie, P.E., Winther, O., and Hogsberg, J. “Comparison of Neural Network Error Measures for Simulation of Slender Marine Structures”. In: *Journal of Applied Mathematics* 2014 (2014). DOI: 10.1155/2014/759834.

[9] Dhiman, M. *Sign Language Recognition*. 2017. URL: `https://edu.authorcafe.com/academies/6813/sign-language-recognition`.

[10] Emmorey, K. *Language, Cognition, and the Brain: Insights From Sign Language Research*. Lawrence Erlbaum Associates Inc, 2002. ISBN: 1135664811.

[11] Fujiyoshi, H., Hirakawa, T., and Yamashita, T. “Deep learning-based image recognition for autonomous driving”. In: *IATSS Research* 43 (4 2019), pp. 244–252. DOI: `10.1016/j.iatssr.2019.11.008`.

[12] Glenn, C., Mandloi, D., Sarella, K., and Lonon, M.K. *An Image Processing Technique for the Translation of ASL Finger-Spelling to Digital Audio and Text*. 2005. URL: `https://pdfs.semanticscholar.org/faa5/35314f389c63d02a4ea790b64ca50d3064ba.pdf`.

[13] *Lecture 1 in course ID1214 Artificial Intelligence and Applied Methods*. KTH Royal Institute of Technology, Oct. 31, 2018.

[14] Håkansson, A. “Portal of Research Methods and Methodologies for Research Projects and Degree Projects”. In: *Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering FECS’13* (2013), pp. 67–73.

[15] Hecht-Nielsen, R. “Neural Networks for Perception - Computation, Learning, and Architectures”. In: ed. by H. Wechsler. Elsevier Inc, 1992. Chap. III.3 - Theory of the Backpropagation Neural Network.

[16] Hernandez-Rebollar, J., Kyriakopoulos, N., and Lindeman, R. “A new instrumented approach for translating American Sign Language into sound and text”. In: *Sixth IEEE International Conference on Automatic Face and Gesture Recognition* (2004).

[17] *Interview with Stockholms Dövas Riksförbund*. Nov. 7, 2019.

[18] Jovanović, I., Miljanović, I., and Jovanović, T. “Soft computing-based modeling of flotation processes – A review”. In: *Minerals Engineering* 84 (2015), pp. 34–63. DOI: `10.1016/j.mineng.2015.09.020`.

[19] Karlik, B. and Olgac, A. “Performance Analysis of Various Activation Functions in Generalized MLP Architectures of Neural Networks”. In: *International Journal of Artificial Intelligence And Expert Systems* 1 (2 2011), pp. 111–122.

[20] Kaur, T. “Implementation of Backpropagation Algorithm: A Neural Network Approach for Pattern Recognition”. In: *International Journal of Engineering Research and Development* 1 (5 2012). ISSN: 2278-067X.

[21] Kingma, D.P. and Lei Ba, J. “Adam: a Method for Stochastic Optimization”. In: *ICLR* (2015).

[22] Kotsiantis, S.B. “Supervised Machine Learning: A Review of Classification Techniques”. In: *Informatica* 31 (2007), pp. 249–268.

[23] Liang, H., Fu, W., and Yi, F. “A Survey of Recent Advances in Transfer Learning”. In: *2019 IEEE 19th International Conference on Communication Technology (ICCT)* (2019). DOI: 10.1109/ICCT46805.2019.8947072.

[24] Liu, L. “Biostatistical Basis of Inference in Heart Failure Study”. In: *Heart Failure: Epidemiology and Research Methods* (2018), pp. 43–82. DOI: 10.1016/B978-0-323-48558-6.00004-9.

[25] *Lecture 3 in course DD2421 Machine Learning*. KTH Royal Institute of Technology, Jan. 24, 2020.

[26] Mocialoc, B., Hastie, H.F., and Turner, G. *Transfer Learning for British Sign Language Modelling*. 2018. URL: https://pdfs.semanticscholar.org/f2a8/9db300ea75f0e720b3cb4ee33ef5ea833d82.pdf?_ga=2.94496341.890869255.1587537639-702972717.1584439496.

[27] Mocialov, B., Turner, G., Lohan, K.S., and Hastie, H. *Towards Continuous Sign Language Recognition with Deep Learning*. 2017. URL: https://pdfs.semanticscholar.org/f24c/82e85906bc7325b296d37370febd65833fdd.pdf?_ga=2.233293527.890869255.1587537639-702972717.1584439496.

[28] Noble, S.U. *Algorithms of Oppression: How Search Engines Reinforce Racism*. NYU Press, 2018. ISBN: 9781479837243.

[29] Organization, World Health. *Addressing the rising prevalence of hearing loss*. 2018. URL: https://apps.who.int/iris/bitstream/handle/10665/260336/9789241550260-eng.pdf?sequence=1&isAllowed=y.

[30] Paul, P.V. “What’s It Like to Be Deaf? Reflections on Signed Language, Sustainable Development, and Equal Opportunities”. In: *American Annals of the Deaf Gallaudet University Press* 163 (4 2018). DOI: 10.1353/aad.2018.0030.

[31] Prevas, AB. *Om Prevas, Hello Possibility*. URL: `https://prevas.se/om_prevas.html`.

[32] Quirk, T. and Kamaal, K. *How we used AI to translate sign language in real time*. 2018. URL: `https://blog.coviu.com/2018/09/21/how-we-used-ai-to-translate-sign-language-in-real-time/`.

[33] Ray, S. "A Quick Review of Machine Learning Algorithms". In: *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)* (2019).

[34] Rosenblatt, F. *Principles of Neurodynamics*. New York: Spartan, 1962.

[35] Ruder, S. *An overview of gradient descent optimization algorithms*. 2016. URL: `https://arxiv.org/abs/1609.04747`.

[36] Schmidhuber, J. "Deep learning in neural networks: An overview". In: *Neural Networks* 61 (2015), pp. 85–117. DOI: `10.1016/j.neunet.2014.09.003`.

[37] Shanthi, T., Sabeenian, R.S., and Anand, R. "Automatic Diagnosis of Skin Diseases using Convolution Neural Network". In: *Microprocessors and Microsystems* (2020). DOI: `10.1016/j.micpro.2020.103074`.

[38] Shen, S., Sadoughi, M., Li, M., Wang, Z., and Hu, C. "Deep convolutional neural networks with ensemble learning and transfer learning for capacity estimation of lithium-ion batteries". In: *Applied Energy* 260 (2020). DOI: `10.1016/j.apenergy.2019.114296`.

[39] Shu, Y., Chen, Y., and Xiong, C. "Application of image recognition technology based on embedded technology in environmental pollution detection". In: *Microprocessors and Microsystems* 75 (2020). DOI: `10.1016/j.micpro.2020.103061`.

[40] Siddique, M.N.H. and Tokhi, M.O. "Training neural networks: backpropagation vs. genetic algorithms". In: *IJCNN'01. International Joint Conference on Neural Networks* 4 (2001), pp. 2673–2678.

[41] Stafford-Smith, M., Griggs, D., Gaffney, O., Ullah, F., Reyers, B., Kanie, N., Stigson, B., Shrivastava, P., Leach, M., and O'Connell, D. "Integration: the key to implementing the Sustainable Development Goals". In: *Sustainability Science* 12 (2016), pp. 911–919. DOI: `10.1007/s11625-016-0383-3`.

[42] Stergiou, C. and Siganos, D. “Neural Networks”. In: *urveys and Presentations in Information Systems Engineering (SURPRISE)* 96 (1996).

[43] Vinuesa, R., Azizpour, H., Leite, I., Balaam, M., Dignum, V., Domisch, S., Felländer, A., Langhans, S.D., Tegmark, M., and Nerini, F.F. “The role of artificial intelligence in achieving the Sustainable Development Goals”. In: *Nature Communications* 11 (233 2020). DOI: 10.1038/s41467-019-14108-y.

[44] Weissmann, J. and Salomon, R. “Gesture recognition for virtual reality applications using data gloves and neural networks”. In: *IJCNN’99. International Joint Conference on Neural Networks* 3 (1999), pp. 2043–2046.

[45] Wong, T. “Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation”. In: *Pattern Recognition* 48 (9 2015), pp. 2839–2846. DOI: 10.1016/j.patcog.2015.03.009.

[46] Xu, W., Zhang, X., Yao, L., Xue, W., and Wei, B. “A multi-view CNN-based acoustic classification system for automatic animal species identification”. In: *Ad Hoc Networks* 102 (2020). DOI: 10.1016/j.adhoc.2020.102115.

TRITA-EECS-EX-2020:200