Bachelor's project

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Nikita Sumahers

Identification of tired, sleepy, or passed out vehicle drivers.

Blackout Protocol

Bachelor's project in Computer Engineering
Supervisor: Kjell Inge Tomren

December 2020

NTNU
Norwegian University of Science and Technology

Nikita Sumahers

Identification of tired, sleepy, or passed out vehicle drivers.

Blackout Protocol

Bachelor's project in Computer Engineering
Supervisor: Kjell Inge Tomren
December 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

NTNU

Kunnskap for en bedre verden

Identification of tired, sleepy, or passed out vehicle drivers. (Blackout Protocol)

IE303612 Bacheloroppgave

Totalt antall sider inkludert forsiden: 32

Ålesund, 21.12.2020

NTNU
Kunnskap for en bedre verden

Obligatorisk egenerklæring/gruppeerklæring

Den enkelte student er selv ansvarlig for å sette seg inn i hva som er lovlige hjelpemidler, retningslinjer for bruk av disse og regler om kildebruk. Erklæringen skal bevisstgjøre studentene på deres ansvar og hvilke konsekvenser fusk kan medføre. Manglende erklæring fritar ikke studentene fra sitt ansvar.

Du/dere fyller ut erklæringen ved å klikke i ruten til høyre for den enkelte del 1-6:		
1.	**Jeg/vi erklærer herved at min/vår besvarelse er mitt/vårt eget arbeid, og at jeg/vi ikke har brukt andre kilder eller har mottatt annen hjelp enn det som er nevnt i besvarelsen.**	**JA**
2.	**Jeg/vi erklærer videre at denne besvarelsen:** ● ikke har vært brukt til annen eksamen ved annen avdeling/universitet/høgskole innenlands eller utenlands. ● ikke refererer til andres arbeid uten at det er oppgitt. ● ikke refererer til eget tidligere arbeid uten at det er oppgitt. ● har alle referansene oppgitt i litteraturlisten. ● ikke er en kopi, duplikat eller avskrift av andres arbeid eller besvarelse.	**JA**
3.	**Jeg/vi er kjent med at brudd på ovennevnte er å betrakte som fusk og kan medføre annullering av eksamen og utestengelse fra universiteter og høgskoler i Norge, jf. Universitets- og høgskoleloven §§4-7 og 4-8 og Forskrift om eksamen §§14 og 15.**	**JA**
4.	**Jeg/vi er kjent med at alle innleverte oppgaver kan bli plagiatkontrollert i Ephorus, se Retningslinjer for elektronisk innlevering og publisering av studiepoenggivende studentoppgaver**	**JA**
5.	**Jeg/vi er kjent med at høgskolen vil behandle alle saker hvor det forligger mistanke om fusk etter høgskolens studieforskrift §31**	**JA**
6.	**Jeg/vi har satt oss inn i regler og retningslinjer i bruk av kilder og referanser på biblioteket sine nettsider**	**JA**

Publiseringsavtale

Studiepoeng: 20

Veileder: Kjell Inge Tomren

Fullmakt til elektronisk publisering av oppgaven

Forfatter(ne) har opphavsrett til oppgaven. Det betyr blant annet enerett til å gjøre verket tilgjengelig for allmennheten (Åndsverkloven §2).

Alle oppgaver som fyller kriteriene vil bli registrert og publisert i Brage HiM med forfatter(ne)s godkjennelse.

Oppgaver som er unntatt offentlighet eller båndlagt vil ikke bli publisert.

Jeg/vi gir herved NTNU i Ålesund en vederlagsfri rett til å

gjøre oppgaven tilgjengelig for elektronisk publisering: **X ja** **☐nei**

Er oppgaven båndlagt (konfidensiell)? **☐ja** **X nei**

(Båndleggingsavtale må fylles ut)

- Hvis ja:

Kan oppgaven publiseres når båndlegging perioden er over? **X ja** **☐nei**

Er oppgaven unntatt offentlighet? **☐ja** **X nei**

(inneholder taushetsbelagt informasjon. Jfr. Offl. §13/Fvl. §13)

Dato: **21.12.2020**

PREFACE

People like to travel and are usually trying to get from A to B in the shortest time possible, often neglecting safety. When you must pay attention over a long duration, for example, while traveling by car, you experience fatigue. It is easy to forget the limitations of our bodies, particularly our eyes when concentrating for extended periods. We often neglect, or do not recognize, the signs of impairment.

World Health Organization reports that approximately 1,35 million people die in traffic accidents annually worldwide. It is important to continue to develop new technologies to mitigate these risks and make the roads safer for everybody.

In my project, I focus on developing a technology that could inform the driver, or an on-board computer, about the driver's tiredness. Alerting to any impairment or an inability to control the vehicle, leading to fewer traffic accidents.

I hope that this project will help to improve current technologies and will inspire more development in this sphere. As well as bring more awareness to a problem and find solutions through the application of these technologies.

I want to thank all the professors and teachers I have learned from at NTNU. Also, my family and wife, who have supported me along the way.

Nikita Sumahers

CONTENTS

TERMINOLOGY

Terms

Convolution	is a mathematical operation on two functions (f and g) that produces a third function (f * g) that expresses how the shape of one is modified by the other. The term convolution refers to both the result function and to the process of computing it.
Haar cascade	Haar Cascade is a machine learning object detection algorithm

Abbreviations

IDE	Integrated Development Environment
AE	Auto Encoder
PCA	Principal Component Analysis
AI	Artificial Intelligence
CNN	Convolutional Neural Network
ML	Machine Learning
TPU	Tensor Processing Unit
GPU	Graphics Processing Unit
ReLU	Rectified Linear Unit

SUMMARY

The Blackout Protocol project is about developing a machine learning system or a concept that would bring attention to a problem and the danger of fatigue on professional and everyday drivers. The goal is to try to create a system that can predict the state of the eyes by inspecting an image of a person's face.

The results of the current model are showing an accuracy level of over 98%, with a very low error rate. To achieve results, I have used a dataset of 85 thousand images, a Tensorflow library, and a feature extraction method. To extract the image of the eyes from the web camera feed I have used OpenCV and Haar Cascade algorithm of object detection. The model is trained on the grayscale images and can be improved to work with infrared cameras.

I also discuss the different methods of feature extraction, different models I have tried, I will discuss the structure of my model and will explain every layer, its objective, and how it works, I will also mention the importance of a good dataset when working with deep neural networks.

1 INTRODUCTION

The idea for this thesis first came to me from my father who is a professional truck driver. He has stressed to me how dangerous fatigue is for drivers, on a route over a long period. He said that it could be a great improvement and could save a lot of lives if we had some technology that could be integrated into trucks that would be able to stop vehicles if the driver has fallen asleep.

I was intrigued by this idea and decided to choose this for my thesis and try to develop something similar to that, which might help the development of a system for this cause and to try to attract more attention to this problem.

The problem for this thesis comes from the lack of technologies that could prevent a tired or a person that has lost conciseness during the control of a vehicle from creating an accident on the road.

In my solution I try developing a system that will prove the concept, this system will perform binary classification of eye images from live camera stream. The images will be classified into 2 classes, open and closed. The system will also be able to recognize when the system fails to detect any face or eyes in that case the system cannot perform prediction and should inform the user about it.

While the system does detect the face and the eyes it should be able to predict the state of the eyes, the system should give the user-readable feedback about the state of the eyes.

The system implements pre-trained models, Haar cascade for face and eye detection, and ResNet50 for feature extraction from images. The rest of the model is a CNN model designed for binary classification of images that I have trained with the help of the MRL Eye dataset which includes over 80,000 images of eyes of different ethnicities, genders, with or without glasses, lighting conditions, and which camera image was taken with.

The final solution should be able to predict the state of the eyes with reasonable accuracy in a reasonable time frame.

2 THEORETICAL BASIS

2.1 Overview

During the research about this technology, I have come across multiple experiments that have been done by the USA government [1], car manufacturers [2], and other studies [3]. This issue is a big and important subject for road safety and therefore has been taken seriously by many researchers. Since over the past decade there has been a very big improvement in AI and ML technology [4], I have decided to take a machine learning approach to solve this problem like in the study mentioned above.

To create my deep learning model, I will be using CNN models, as they prove to be very sufficient and easy to implement for image classification. Neural networks require a large number of parameters which can be very computationally heavy and tuning so many parameters can be a very huge task, CNN largely diminishes the time taken to tune parameters.

CNN's are fully connected feed-forward neural networks. It is very effective in reducing the number of parameters without losing quality. Images have high dimensionality (each pixel considered as a feature) which suits the described abilities of CNNs.

In CNN dimensionality reduction is achieved using a sliding window with a size less than that of the input matrix. This computation is performed using the convolution filters present in all the convolution layers. You can intuitively think of this as reducing your feature matrix from a 3x3 matrix to 1x1. A bias is also added to the convolution result of each filter before passing it through the activation function. [5]

To recognize faces and eyes on the images I have chosen to use Haar Cascade [6], which is an object detection algorithm based on the concept of features proposed by Paul Viola and Michael Jones. It is a machine learning-based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images. [7]

For achieving high accuracy in image classification, I have chosen to use the transferred learning method, where we use a pre-trained ImageNet model for feature extraction from images. I have chosen to use ResNet50 [8] model as it has performed well in image classification tasks with a low error rate.

Transfer learning is a machine learning technique that enables data scientists to benefit from the knowledge gained from a previously used machine learning model for a similar task. This learning takes humans' ability to transfer their knowledge as an example. If you learn how to ride a bicycle, you can learn how to drive other two-wheeled vehicles more easily. Similarly, a model trained for autonomous driving of cars can be used for autonomous driving of trucks [9], and in this case, we use a model trained to classify images of eyes

CNN deep learning models require a good and preferably large dataset, it is a huge factor of how well the model will perform, the bigger dataset for training is provided the better

the results can be expected. To train my model I have chosen to use MRL Eye Dataset [10] which has over 80,000 different images of eyes of people of different ethnicities, gender it also includes eyes with and without glasses, sun glare, different lighting conditions, and of different quality. This has proved to be a very good dataset with enough images to train, validate, and test my model.

My knowledge of ML comes from Ibrahim A. Hameed's course in Intelligent Systems at NTNU [11], the book "Artificial intelligence: a guide to intelligent systems" by Michael Negnevitsky [12], and the Internet, lately there have been growing interest in ML subject in the world, which have made many more resources available.

2.2 CNN model

Half of my CNN model is made of Convolutional 2D layers and Max Pooling 2D layers with ReLU activation function while the other half is Dropout, Flatten, and two Dense layers.

The model has a total of 10 layers (ResNet50 is counted as 1 layer)

The first lambda layer does not perform any machine learning activity but resizes the input image to the size that ResNet50 requires.

The rest of the 8 layers are the CNN model I have written to classify images. The first 4 layers are the convolutional and max-pooling layer which reduces dimensions and performs convolution. The last 4 layers consist of dropout to fight to overfit, flatten to reduce the 2D array to 1D, and final dense layers to perform classification of the array.

Layer (type)	Output Shape	Param #
lambda_8 (Lambda)	(None, 224, 224, 3)	0
resnet50 (Functional)	(None, None, None, 2048)	23587712
conv2d_15 (Conv2D)	(None, 5, 5, 32)	589856
max_pooling2d_15 (MaxPooling	(None, 3, 3, 32)	0
conv2d_16 (Conv2D)	(None, 1, 1, 64)	18496
max_pooling2d_16 (MaxPooling	(None, 1, 1, 64)	0
dropout_7 (Dropout)	(None, 1, 1, 64)	0
flatten_7 (Flatten)	(None, 64)	0
dense_14 (Dense)	(None, 128)	8320
dense_15 (Dense)	(None, 1)	129

Validation accuracy of this model after 12 epochs is 98,96%

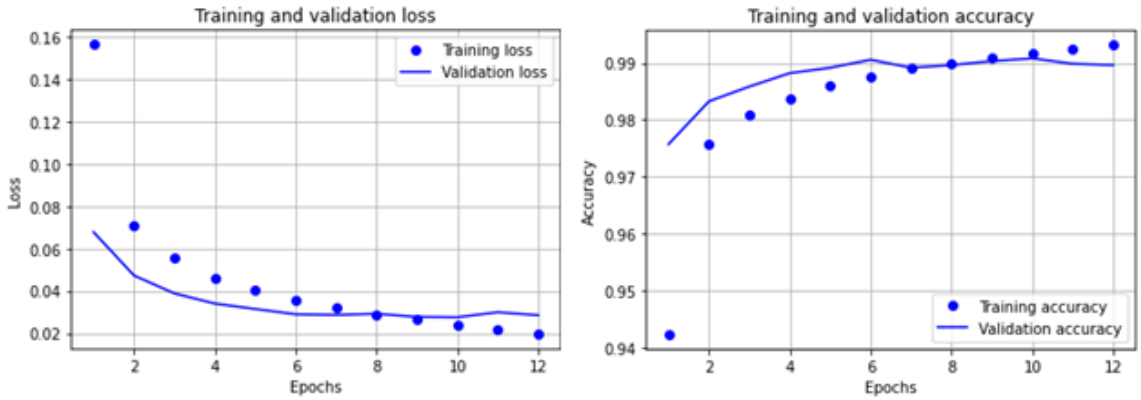


Figure 1: Training and validation (loss and accuracy)

This is the confusion matrix for the validation of the model. From 2155 images of closed eyes 27 was predicted as open and of 2090 images of open only 17 was predicted as closed. The system has not seen the images that were used for the validation during its training.

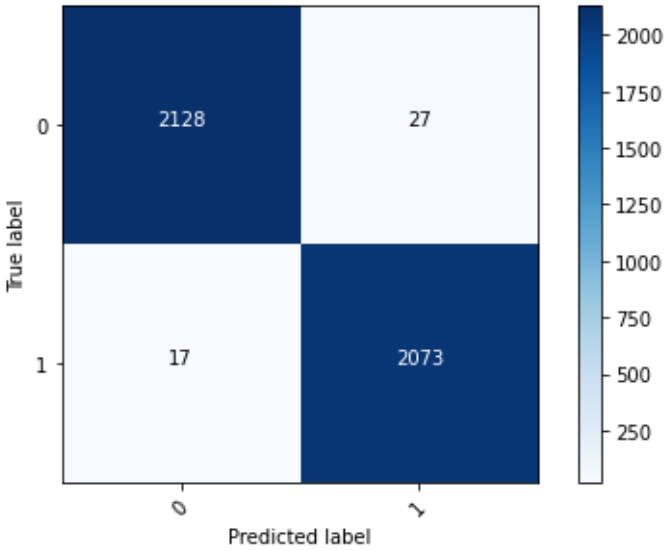


Figure 2: Confusion matrix

2.2.1 Convolutional 2D layer

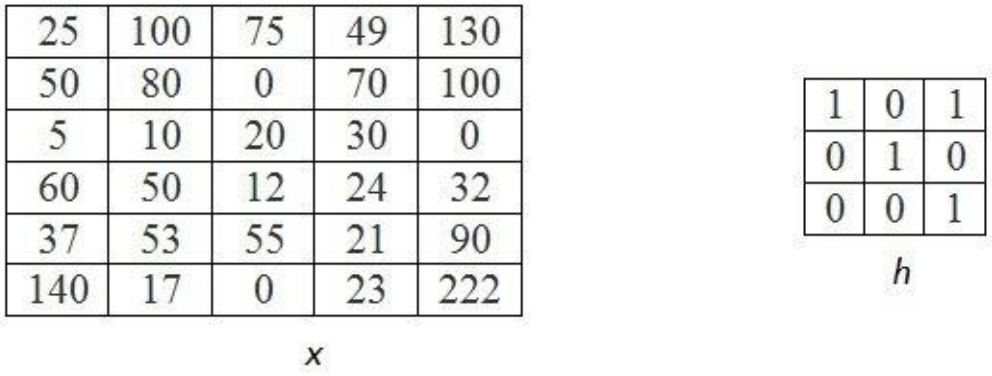
Convolutional 2D layer, convolution involving one-dimensional signals is referred to as 1D convolution or just convolution. Otherwise, if the convolution is performed between

two signals spanning along two mutually perpendicular dimensions (i.e., if signals are two-dimensional), then it will be referred to as 2D convolution.

This concept can be extended to involve multi-dimensional signals due to which we can have multi-dimensional convolution. [13]

In our model, we have a convolution that spans along two dimensions therefore we perform a 2D convolution.

A convolution layer extracts features from a source image by “scanning” the image with a filter of, for example, 5×5 pixels. For each 5×5 pixels region within the image, the convolution operation computes the dot products between the values of the image pixels and the weights defined in the filter. [14]
Example of 2D convolution:



25	100	75	49	130
50	80	0	70	100
5	10	20	30	0
60	50	12	24	32
37	53	55	21	90
140	17	0	23	222

x

1	0	1
0	1	0
0	0	1

h

Figure 3: Input matrices, where x represents the original image and h represents the kernel. Image created by Sneha H.L.

2.2.2 Max Pooling 2D layer

Max Pooling 2D is a type of operation that is typically added to CNN's following individual convolutional layers.

The way the max-pooling layer works is that it combines pixels into a bigger sector then finds the highest value in each sector and creates an output which is a smaller image with the most valued sectors.

This method allows us to reduce dimensionality, and leave only significant values, this is good for removing backgrounds from the image and leaving only the object we need to process and predict.

When added to a model, max-pooling reduces the dimensionality of images by reducing the number of pixels in the output from the previous convolutional layer. [15]

Example of max pooling:

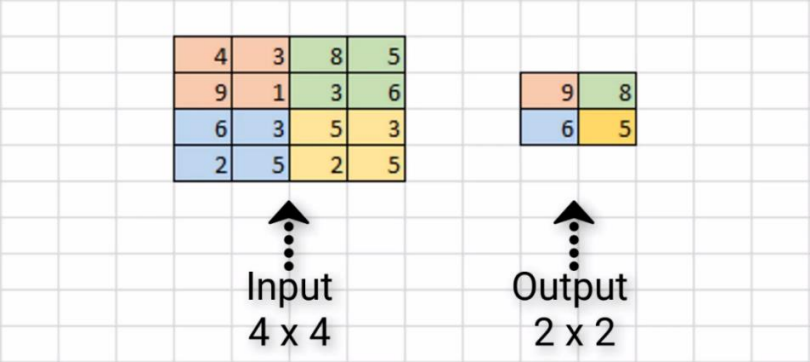


Figure 4: Max Pooling example by deeplizard.com

2.2.3 Dropout layer

The Dropout layer is a mask that nullifies the contribution of some neurons towards the next layer and leaves unmodified all others. We can apply a Dropout layer to the input vector, in which case it nullifies some of its features; but we can also apply it to a hidden layer, in which case it nullifies some hidden neurons.

Dropout layers are important in training CNNs because they prevent overfitting on the training data. If they aren't present, the first batch of training samples influences the learning in a disproportionately high manner. This, in turn, would prevent the learning of features that appear only in later samples or batches. [16]

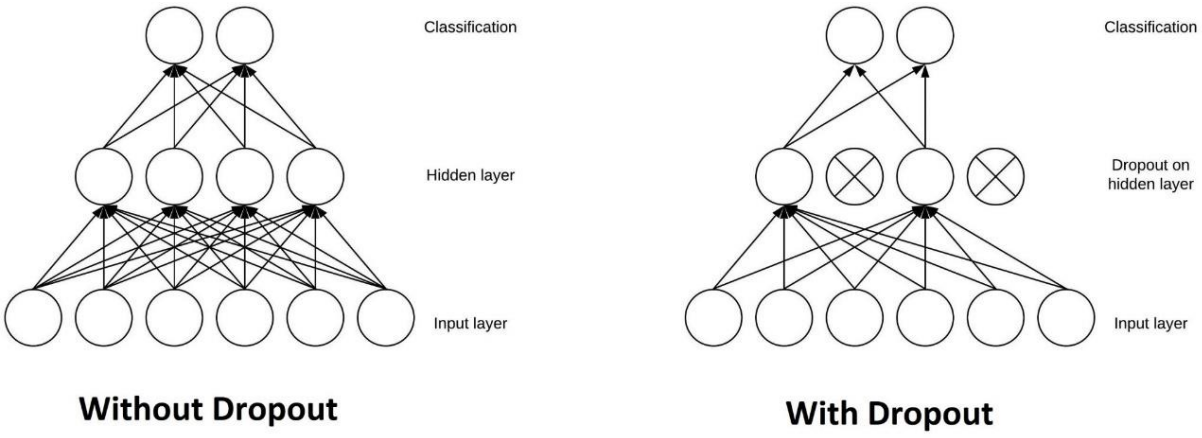


Figure 5: Dropout layer example

During the development of the model I have faced that the model sometimes would start to quickly overfit, first I have tried to make the model more complex and then I tried to make it less complex, but the overfitting problem was either not solved or the model would perform worse, then I was introduced to the Dropout layer during Ibrahim's class [11]. I

applied this layer to my model and was able to solve the overfitting problem and improve performance significantly.

2.2.4 Flatten layer

The Flatten layer is used to flatten the input. For example, if flatten is applied to a layer having input shape as (batch_size, 2,2), then the output shape of the layer will be (batch_size, 4) [17], in this case, this layer flattens the output after all the convolutions, max pooling and dropout, from 2D image array we get a 1D array that we pass to final layers.

2.2.5 Dense layer

Dense layers are the regular deeply connected neural network layer. It is the most common and frequently used layer. The layer operates below and returns the output.

`output = activation(dot(input, kernel) + bias)` [18]

In my model, I use a Dense layer as a hidden layer with 128 neurons that takes the output of the Flatten layer and outputs to the final Dense layer with 1 neuron and sigmoid activation function which finally outputs a prediction between 0 and 1.

2.2.6 ReLU activation function

ReLU activation function for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance. [19]

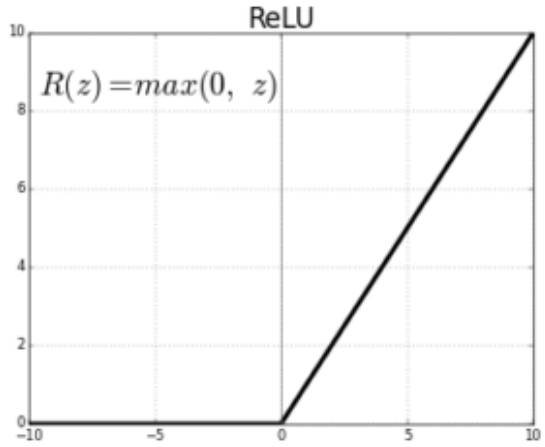


Figure 6: ReLU activation function

2.2.7 Sigmoid activation function

The sigmoid activation function is very simple which takes a real value as input and gives the probability that is always between 0 or 1. It looks like an ‘S’ shape.

In this model we use the sigmoid function only in the very last layer, ReLU function works well on all of our layers when we work with a deep neural network, but because our output is a binary class we need to use Sigmoid to get a prediction between 0 and 1.

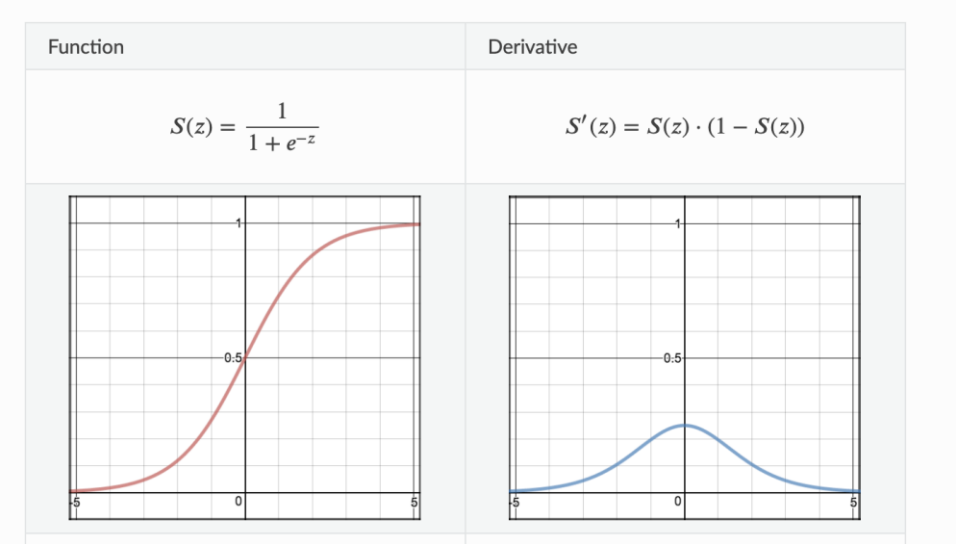


Figure 7: Sigmoid function and it’s derivative

It is non-linear, continuously differentiable, monotonic, and has a fixed output range. The main advantage is simple and good for a classifier. [20]

3 METHOD AND MATERIALS

3.1 Method

3.1.1 First tests

I have started by trying to make a simple perceptron to classify images, it was quickly clear that images have too many features for a simple neural network. The network was very inaccurate and was not able to predict. To solve this problem, I started by looking into dimensionality reduction, the two methods we have learned and therefore I decided to test were: AE and PCA.

After implementing both feature reduction methods I have tested my still simple neural network again and compared the results. Tests and validations seemed to start to produce results and sometimes would even give quite a good accuracy percentage. I was able to get the best results with AE and quite well with PCA too, but PCA was very resource inefficient and required quite a lot of RAM to work.

PCA would quickly run out of all the available RAM in Google “colab” and crash my runtime. Therefore, I decided to continue tests only with AE, as it seemed like a better solution, more resource-efficient and provided better accuracy. When I was satisfied with its performance in colab I started testing it on a local machine with footage from the webcam. The results from those tests were bad and the system was very inaccurate.

3.1.2 Improving neural network

At this point I decided that I should improve the neural network to try to achieve better results, I scraped my perceptron and went for CNN and Tensorflow framework. I looked up existing guides of Convolutional Neural Networks online and wrote one following those guidelines.

When the CNN was implemented I was quickly met with a new problem, overfitting, AE would reduce features so much that my new neural network would easily remember the dataset in just 2 epochs, overfit, and wouldn’t be able to predict.

To fight to overfit there were different possible approaches, removing AE, and usefully not reduced images, or trying to reduce the complexity of CNN. This is when I got a suggestion from Ibrahim to try one more thing, feature extraction.

3.1.3 Implementing feature extraction

I did some research to find out how to implement feature extraction and found a good online resource where they wrote about how to implement it and compared accuracy and efficiency on different ImageNet models for feature extraction. ResNet50 model was the one that seemed to have provided the best results in the classification of dogs and cats images. Dogs and cats are of course not eyes, but it is also a binary classification, and dogs and cats can often be very similar, and then it just little features that make the difference, therefore I thought that this should work well with my problem as well.

I completely removed AE from my network and added ResNet50 to the top layer, then I implemented one more layer on top of it, the lambda layer that would resize the input image to the right size for ResNet50 to work.

After running new tests of the improved network, I was met with signifiable better results, when the network was tested on the live images now, it was able to correctly predict most of the time. There was still a lot of room for improvement but, I was on the right track now.

3.1.4 Improving CNN

Now that I had a functioning network, I focused on improving it by trial and error. I would adjust the number of layers, neurons, dataset size, number of epochs, and implement early stopping, to improve network accuracy.

First, I tried to improve the network by making it more complicated, with more layers, and more neurons, I was surprised to see that this made the network more inaccurate and eventually network started to overfit.

Instead of making it more complicated now, I tried to reduce complexity, removed the number of layers and neurons, and achieved better results, the overfitting problem was solved, and the accuracy improved, reaching almost 99% accuracy, and performing quite well on the live feed tests.

3.1.5 Working with live footage

To perform tests of my network along the way validation tests were not sufficient for me, I wanted to see how my system performs in closer to a real-life situation. I had to develop a system that would be able to extract images of eyes in real-time so that I can pass those images to my CNN and get a prediction.

I choose to use the OpenCV framework for this implementation as this framework has all the tools, I needed to do this. By using this framework, I was able to easily extract feed from the webcam, change it to grayscale and then pass it through Haar Cascade.

```
while True:
    ret, frame = video_capture.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    if len(faces) < 1:
        cv2.imshow('img', frame)
    else:
        for (x, y, w, h) in faces:
            img = cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
            roi_gray = gray[y:y+h, x:x+w]
            roi_color = img[y:y+h, x:x+w]
            eyes = eye_cascade.detectMultiScale(roi_gray)
        for (ex, ey, ew, eh) in eyes:
            cv2.rectangle(roi_color, (ex, ey), (ex+ew, ey+eh), (0, 255, 0), 2)
            # extraction of eye and reshaping
            image = cv2.resize(roi_gray[ey:ey+eh, ex:ex+ew], (100, 100), interpolation=cv2.INTER_AREA)
```

Figure 8: Code for extracting faces and eyes

3.2 Language and environment

3.2.1 Python

As it is an ML project my choice of program language has quickly fallen on Python since it is considered the best programming language choice for ML projects [21]. During the intelligent system course [11] we have used Python to learn to program models and Google Collaborations is using Python language as well.

Another language that sometimes is used for ML is MATLAB but, it does not support many existing libraries that make deep learning models programming much easier and more efficient, therefore in this project, everything was written in Python.

3.2.2 IDE

For local programming I have used IntelliJ IDEA by JetBrains, this IDE provides good support of Python language, I have good experience with this software, and it proved to be fast and efficient to write code in it. I have used a student license to obtain and use the software.

I have also used Google Colaboratory for hosted development. Google Collaborate, or simply for short called “colab”, is a new web-based python development environment, allowing multiple people to work on the same code simultaneously. It has no installation requirements, and all code compilation and execution are done on Google servers, which house massive GPU banks which include Nvidia K80s, T4s, P4s and P100s, and even TPUs. [22]

This made it possible to train our model with the trial and error method, for comparison to train and test the same model on my local machine I would need over an hour, while with the hosted GPU I was able to train and test the model in about 20 minutes. This let me do many trials efficiently and make a lot of adjustments to my model.

3.2.3 Dependencies and libraries

3.2.3.1 Itertools

A module for iteration made entirely for python. It contains a lot of fast, powerful tools for iteration, which is very useful when one must iterate through an array consisting of over 80 000 elements. In our solution though it's only used for creating a confusion matrix.

3.2.3.2 NumPy

NumPy is a mathematical library, adding support for handling large multi-dimensional arrays and matrices, as well as high-level mathematical functions to handle these arrays and matrices. It is a powerful tool for machine learning and is in all probability one of the most used scientific tools in the Python toolbox. Most of the dataset manipulations are done with the help of this library, especially very handy becomes *reshape* and *repeat* functions:

```
X_train = np.repeat(X_train[..., np.newaxis], 3, -1)
```

```
X_train = X_train.reshape(len(X_train), 100, 100, 3)
```

These functions let me add channels and reshape 1D array to 2D efficiently. In this case, it is done to fit the input to my model.

3.2.3.3 Matplotlib

Matplotlib is a large array of visualization tools. It is a great library for visualizing matrices and arrays and helps me to visualize the results from testing the model which

helps to make sense of results and compare them with each other. To determine improvements or decline in performance.

3.2.3.4 Tensorflow and Keras

Tensorflow is an open-source library dedicated to machine learning, with a focus on training deep neural networks, and statistical inference of those networks. Keras is a deep learning API.

Built on Tensorflow 2.0, Keras is fast, powerful, convenient, and works on pretty much any scale, from a basic laptop CPU to massive GPU clusters, or dedicated TPUs Keras is the submodule of Tensorflow. This is probably the main library in our project as this library let us implement a feature extraction model, create the CNN model, apply layers, and perform evaluations of the model.

3.2.3.5 Scikit-learn

Scikit-learn, also known as sklearn, is another machine learning library, featuring algorithms for regression, clustering, and classification. In my project, this module was useful to quickly split the dataset into train and validation sets.

```
X_train,X_test,y_train,y_test = train_test_split(X,y,train_size=0.
95)
```

Train_size = 0.95 means that we use 95% of the dataset for training and 5% for validation. Since the dataset I use had 84 898 images this gives us 80 653 images for training and 4245 images for validation and testing.

3.2.3.6 Pickle

Pickle is a library containing protocols for binary serialization, and deserialization, or as they prefer calling it: pickling and unpickling. Pickling is the process of storing a data structure as a restorable byte-stream, while unpickling is the process of restoring the data from the byte-stream. It is a very space-efficient way of storing and sharing large amounts of data and functional models, like a NumPy array or a trained machine learning algorithm.

3.2.3.7 OpenCV

OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the open-source Apache 2 License. Starting with 2011, OpenCV features GPU acceleration for real-time operations. [23]

In the solution, it is used to operate the Haar Cascade classifier, get webcam video input, and manipulating images. Most of the functions in demonstration software are performed by the OpenCV framework and are referred to as cv2.

3.3 Models

3.3.1 ResNet50

ResNet50 is a deep learning neural network model, with depth up to 152 layers, this model has won the first prize on the ILSVRC 2015 classification task. Authors: Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun [8]

ResNet50 model is being used as a feature extraction model on top of my CNN model, the output from that model is input to my CNN model.

During early development, I have experimented with different feature extraction methods like autoencoder and PCA, but I have not been able to achieve sufficient results, then I got a proposal from Ibrahim A. Hameed to try to use pre-trained feature extraction models. That is when I decided to use an ImageNet model for feature extraction in this project. On the internet, there is a big range of different ImageNet models, VGG, ResNet, Inception, and more.

I have chosen to go with ResNet50 because it has performed quite notably well and better than other mentioned models, here is a comparison of ResNet over other models, where they classify images of cats and dogs:

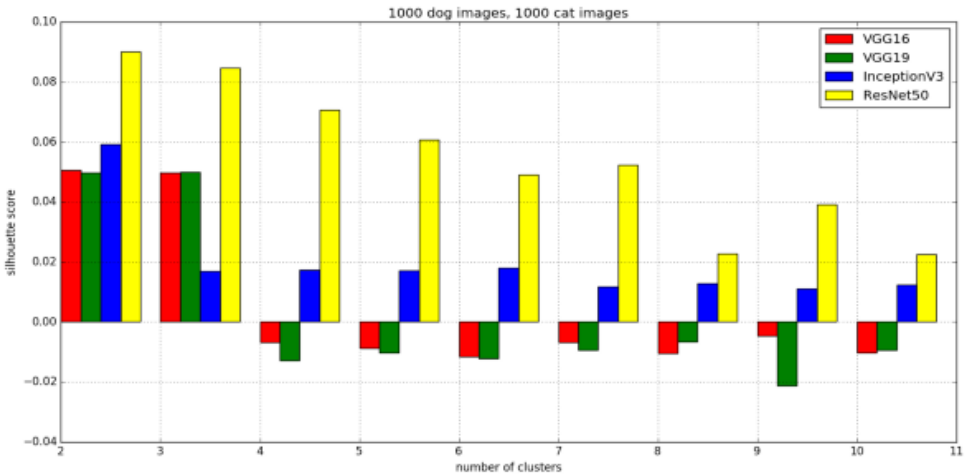


Figure 9: comparison of ImageNet models, score over the number of clusters

3.3.2 Haar Cascade

Haar Cascade is a machine learning object detection algorithm used to identify objects in an image or video and based on the concept of features proposed by Paul Viola and

Michael Jones in their paper "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001.

It is a machine learning-based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.
The algorithm has four stages:

1. Haar Feature Selection
2. Creating Integral Images
3. Adaboost Training
4. Cascading Classifiers

It is well known for being able to detect faces and body parts in an image but can be trained to identify almost any object. [6]

Haar Cascade models are divided into 2 models Haar Cascade Frontal Face model and the Haar Cascade Eyes model. The first model searches for faces in the image and creates a smaller image of a face, the second model searches for eyes in the image from the previous model and creates another smaller image of just eyes.

3.3.3 CNN

To create my CNN model, I have used tools provided by the Keras framework. Keras provides great tools for the implementation of model layers, which makes development fast and easy. It also makes it easy to import ResNet50, and to adjust layers for a quick trial and error development, as well as making it easy to implement early stopping, save history, and save the model.

Keras lets us implement each layer in a form of a simple function where we just have to provide some parameters like activation function, number of outputs or inputs, padding, or the size of the input.

To train the model we use final functions *compile*, and *fit* where we define the final parameters that will determine the way the model will be trained, since I have implemented an early stopping function as well, I choose to have a high number in the epoch parameters, because the model will stop itself if the validation loss will stop to improve.

As a loss function, I choose "*binary_crossentrapy*" because this is a binary classification problem, and this function fits our problem best, and for the optimizer, I chose the RMSprop function.

```
model = models.Sequential()
model.add(Lambda(lambda image: tf.image.resize(image, to_res)))
model.add(res_model)

# Convolutional layer and maxpool layer 1
model.add(Conv2D(32,(3),activation='relu'))
model.add(MaxPool2D(2, padding='same'))

# Convolutional layer and maxpool layer 2
model.add(Conv2D(64,(3),activation='relu'))
model.add(MaxPool2D(2, padding='same'))

# Dropout layer
model.add(Dropout(.4))

# This layer flattens the resulting image array to 1D array
model.add(Flatten())

# Hidden layer with 128 neurons and Rectified Linear Unit activation function
model.add(Dense(128,activation='relu'))

#Here we use sigmoid activation function which makes our model output to lie between 0 and 1
model.add(Dense(1,activation='sigmoid'))

model.layers[1].trainable = False

earlyStop = callbacks.EarlyStopping(monitor='val_loss', patience=2)

model.compile(loss='binary_crossentropy',
              optimizer=optimizers.RMSprop(lr=2e-5),
              metrics=['accuracy'])
history = model.fit(X_train, y_train, batch_size=128, epochs=30, verbose=1, callbacks=[earlyStop],
                    validation_data=(X_test, y_test))
model.summary()
model.save("/content/drive/My Drive/Datasets/16.11.12")
```

Figure 10: Source code of the CNN model

This framework limits how much you get to adjust the layers, but the way the layers are implemented there, I do not think they need to be tweaked much in most of the cases. If I would have chosen to write each layer myself then I would probably choose the PyTorch framework to write my model.

3.4 Dataset

The dataset we used is the MRL Eye Dataset [10], which contains 84 898 infrared images from 37 different people in multiple resolutions. In addition to the eye open/closed values, we needed it also has an annotation for gender, whether glasses are used, light conditions, if it contains reflections, and what kind of sensor/camera it was taken with.

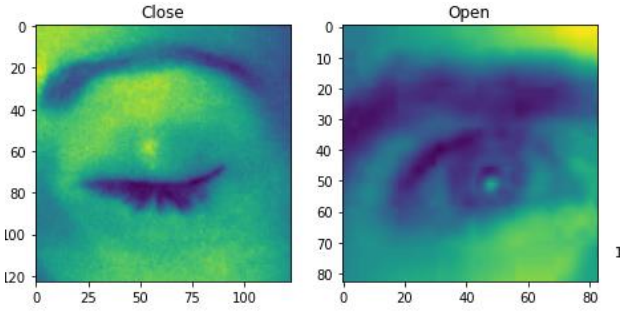


Figure 11: Sample from the dataset

On their page, they have also described the tools they have used to create this dataset, which was also the same Haar Cascade model that I use in the demonstration. Therefore, the method for extraction of eyes in the dataset and live tests are the same.

Images in the dataset had various pixels size, therefore my model has a layer that resizes images to a fixed 100x100 pixels size.

This dataset is exactly what I needed for this project, and I am very thankful to MRL for creating it and providing it for free for everyone who wants to use it in research.

4 RESULTS

4.1 Status

I find the result, personally, quite satisfying. The model has a quite high accuracy by itself and its weakest points right now is the Haar Cascade model that identifies eyes and a low-quality webcam. I have tested the system with different people in different lighting conditions with the same web camera. The web camera that was used for testing is an integrated camera on the Asus FX505DY laptop.

The results are best when the lighting condition is good and when a person looks straight at the camera. Systems have proved to work well with different ethnicities and skin color too.

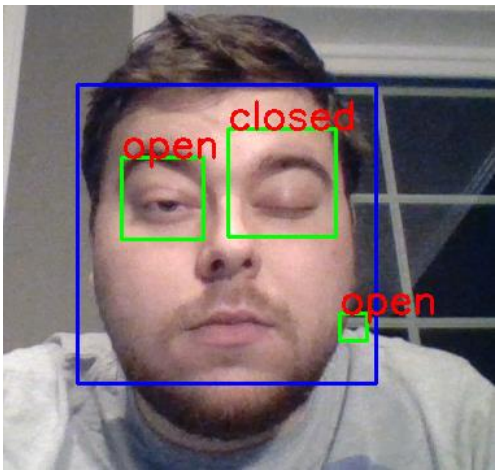


Figure 12: Picture of me testing the system

As you can see the system has predicted the eye states correctly, but Haar Cascade has also found the third eye in the background. The system manages the predictions very well, during tests I have noticed that Haar Cascade often having trouble to locate the eyes when they are closed, and then the system cannot predict.

The system will as well work when there are multiple faces detected, therefore the system can do prediction on as many eyes as the Haar Cascade would detect, but the current

system is not resource-efficient, and it would take a lot of computational power to predict a big number of images simultaneously.

It is important to note that the model is trained and tested on grayscale imagery therefore the system does not need to see colors in the image to do a prediction. This is done so that an infrared camera could be implemented easily. And that the system would work in bad lighting conditions.

4.2 Improvements

To improve the system, I would first go for implementing an infrared camera, system might need some adjustments, but it should not take much to make it possible to predict on infrared images, as the model was trained on the grayscale images from the beginning. An infrared camera would make it possible to make predictions in dark conditions and make it more useful inside a vehicle as well as eliminate the need for good lighting conditions to work.

I would also suggest finding a better model for finding eyes in the image, it could be a better trained Haar Cascade or its deep learning model for finding eyes on an image. There might be as well possible to improve it by tweaking my code.

The CNN model layers can be adjusted, and maybe some layers could be removed or added, to gain better results, as well as the model, can be retrained to work with other image sizes. With a good webcam, the model could be retrained to predict 200x200 pixel images, for example, this would double the number of features and possibly make predictions even more accurate.

Another possible improvement would be to implement image transformation and create an even bigger dataset with images augmented, by making quality a little bit worse, or adding artificial smudges and light rays. This would create an even bigger dataset and possibly a more robust model for bad environmental conditions.

4.3 Findings

With this research, I have found that using transfer learning is a very efficient way of feature extraction and can provide much better results than AE and PCA feature extraction methods, especially when working with deep learning models.

I learned about the Haar Cascade algorithm for object detection, how it works, and how it can be applied to detect objects in an image. I have also learned about convolutional, max pooling, and drop-out layers, how they work, and how they process images.

I have also learned to program in Python, to accomplish this project the code had to be written in Python, and I have not used Python before starting this project, therefore I had to learn on the go, thankfully knowing other programming languages like for example Java, helped a lot, and I was able to quickly adapt to Python syntax and write the code.

I found out that a simple model can be more efficient than a complicated one, and it is more robust from overfitting. Overfitting can be a huge problem in deep learning when working with a “small” number of features.

After working with CNN I have learned what convolution means and what it does, and have a better understanding of CNN models and how they work. I found image classification a very interesting task and would like to continue to learn more about it.

I learned a lot about creating and choosing a dataset, in the first project I have tried to create a dataset myself, but it did not provide enough data as the one from MRL. I learned the importance of a good dataset in ML, even if you have a great efficient model, it is useless without the right dataset.

4.4 Result

The current system has a high accuracy of 98.9% and very good results from my testing. I am very satisfied with the results I have achieved. As I mentioned earlier the system does see the difference between open and closed eyes quite well and performs well in well-lighted conditions.

The Haar Cascade detects faces very well but sometimes having trouble to detect eyes, or can detect them where there are none, the way it works is that I find objects, and if something might have a similar shape to an eye the Haar Cascade will detect there an eye.

I think using transfer learning helped a lot in this project, the systems performance improved significantly when transfer learning was introduced to the model. I have also tried to train bottom layers of the ResNet model to connect it with my model “better” but the model would become too complicated and would overfit quickly, therefore leaving ResNet completely untrainable worked best in my situation when trained and tested.

Can the system be used in a real-life situation? No, the system is still not ready, it is a proof of concept and it proves that such a system is possible to develop. The current system is still not efficient and not accurate enough to be used in real-life scenarios. Further development and improvements are required to achieve such a result.

Does the system prove the concept? I think it does, it manages predictions very well and shows that we are not far from being able to develop a similar system with close to none errors and higher efficiency.

All in all, this project is a success, and it has taught me a lot about machine learning, artificial intelligence, teamwork, and keeping the focus on your goals. I learned a lot about different ML models, how they work, and how to develop them. I hope I will be able to apply this knowledge in the future in other projects.

As the next step for this project, I would think that it should be improved with the technologies mentioned in the improvements section, and it could be then implemented in a small computer with an infrared camera and tested inside a car, it could for example make a sound when it detects that the eyes are closed over a long time.

5 DELIBERATION

5.1 Scraping

Throughout my thesis, I was once again proved that you should not be afraid of scraping your work and starting again. Even if your system did not completely fail, but had some major issues, it can be healthy to start all over.

My first system was very different from the current one, and I learned a lot from it. I knew what worked well and what did not work at all, and I took those findings with me in this system. I knew that I need a better dataset, I found out that the BLOB extraction of a face that I was using was not working well, as well as there were some other issues with it too.

Therefore, I decided to scrap all those things that were not working or were not working well enough and found other methods to do the same function but better or took a different approach to solve the problem.

5.2 Demonstration

For the demonstration of the system originally, I had big plans to use the car simulator that we have at our university. I was trying to come in contact with people responsible for the simulator, but it was bad communication as one and Covid-19 made it even harder to be at the campus, making the rules around public spaces stricter.

In the end, I didn't have enough time left to try to implement this and had to scrap the idea, I wish thou to have been able to do this, as I think it would be fun and interesting to see the system tested on a simulator.

To demonstrate my system, I have recorded a short video that I include together with this report. In the video, it can be seen how the system responds and predicts. The framerate of the video is very bad, this is not due to the video quality but due to the computational time, it takes to predict eye states.

5.3 Teamwork

For most of my thesis I have worked together with one more student on this project, I like to have good teamwork, but through this project, I found that sometimes if the team doesn't work you should do it yourself.

Sadly, for me, my teammate did not take this project seriously enough and did not put work into it. I have tried to talk with him to find a solution for us, but things have not gotten better, I ended up doing everything myself and had to cut him off my project.

After I have done that things have felt easier and went much better, the only regret I had been that I have not cut him off earlier.

6 CONCLUSION

6.1 My opinion

I think that the system does what it was meant to do, and proves the concept, that it is possible to create a system that could efficiently detect and predict eyes, that could be used to improve the safety on the road. The system provides a good insight into how technology can be used in the area and its abilities. It could as well be used in other places where it is important to keep attention or to track the state of a patient.

As the current state of the system, it is not efficient and accurate enough to be used in real situations and must be further developed and improved. I hope that I will get a chance in the future to further develop it or to work on similar projects in ML.

I have learned a lot of new technologies during my work with this project, I have gained more understanding of deep learning and convolutional neural networks, It was really fun to work with image classification and I wish I will be able to do more work in that sphere in the future. This project was good training in programming, problem-solving, and creativity.

All in all, I am very happy with my project, I am glad I have chosen this theme, I feel that I have learned a lot, and were able to show my skills in programming researching and problem-solving.

6.2 Feedback

The system has been demonstrated to a few people and professors and has received quite positive feedback. Notable feedback was received from my Professor Ibrahim of Intelligent Systems course, who was quite impressed with the system. I received a lot of help from him throughout the process of developing, and I am very thankful for his help and support.

The project was also chosen to be shown at a conference about machine learning, but as of today I am still waiting for more information about that, therefore I can’t say much about it here right now.

I have also used this project as my exam presentation for the intelligent systems class and have received very great feedback for it.

7 REFERENCES

[1] FMCSA, "fmcsa.dot.gov," 11 February 2016. [Online]. Available: https://www.fmcsa.dot.gov/safety/research-and-analysis/driver-fatigue-and-distraction-monitoring-and-warning-system. [Accessed 9 September 2020].

[2] V. C. Corporation, "Volvo cars," Volvo, 20 March 2019. [Online]. Available: https://www.media.volvocars.com/global/en-gb/media/pressreleases/250015/volvo-cars-to-deploy-in-car-cameras-and-intervention-against-intoxication-distraction. [Accessed 20 April 2020].

[3] L. C. Valeriano, P. Napoletano and R. Schettini, "ieeexplore," 17 December 2018. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8576183. [Accessed 20 September 2020].

[4] A. Team, "Azati," 17 September 2020. [Online]. Available: https://azati.ai/image-detection-recognition-and-classification-with-machine-learning/. [Accessed 5 October 2020].

[5] P. Mishra, "Medium," 27 May 2019. [Online]. Available: https://medium.com/datadriveninvestor/why-are-convolutional-neural-networks-good-for-image-classification-146ec6e865e8. [Accessed September 2020].

[6] W. Berger, "willberger," [Online]. Available: http://www.willberger.org/cascade-haar-explained/. [Accessed 7 September 2020].

[7] OpenCV. [Online]. Available: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html. [Accessed September 2020].

[8] Keras, "Kaggle," 17 December 2017. [Online]. Available: https://www.kaggle.com/keras/resnet50. [Accessed 10 October 2020].

[9] B. Ozgen, "aimultiple," 11 August 2020. [Online]. Available: https://research.aimultiple.com/transfer-learning/. [Accessed November 2020].

[10] M. R. Lab, "MRL," [Online]. Available: http://mrl.cs.vsb.cz/eyedataset. [Accessed 15 August 2020].

[11] NTNU, "NTNU," [Online]. Available: https://www.ntnu.edu/studies/courses/IE303312#tab=omEmnet. [Accessed 10 10 2020].

[12] M. Negnevitsky, Artificial Intelligence: a guide to intelligent systems, 2018.

[13] S. H.L., "All about Circuits," 30 November 2018. [Online]. Available: https://www.allaboutcircuits.com/technical-articles/two-dimensional-convolution-in-image-processing/. [Accessed 10 September 2020].

[14] Tensorflow, "missinglink.ai," [Online]. Available: https://missinglink.ai/guides/tensorflow/tensorflow-conv2d-layers-practical-guide/. [Accessed October 2020].

[15] "Deeplizard," [Online]. Available: https://deeplizard.com/learn/video/ZjM_XQa5s6s. [Accessed 10 September 2020].

[16] Baeldung, "Baeldung," 13 August 2020. [Online]. Available: https://www.baeldung.com/cs/ml-relu-dropout-layers. [Accessed November 2020].

[17] Keras, "tutorialspoint," [Online]. Available: https://www.tutorialspoint.com/keras/keras_flatten_layers.htm. [Accessed November 2020].

[18] Keras, "tutorialspoint," [Online]. Available: https://www.tutorialspoint.com/keras/keras_dense_layer.htm. [Accessed November 2020].

[19] J. Brownlee, "machinelearningmastery," 20 August 2020. [Online]. Available: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/. [Accessed September 2020].

[20] M. Chaudhary, "medium," [Online]. Available: https://medium.com/@cmukesh8688/activation-functions-sigmoid-tanh-relu-leaky-relu-softmax-50d3778dcea5. [Accessed November 2020].

[21] Pythonbasics, "pythonbasics," [Online]. Available: https://pythonbasics.org/why-python-for-machine-learning/. [Accessed 6 May 2020].

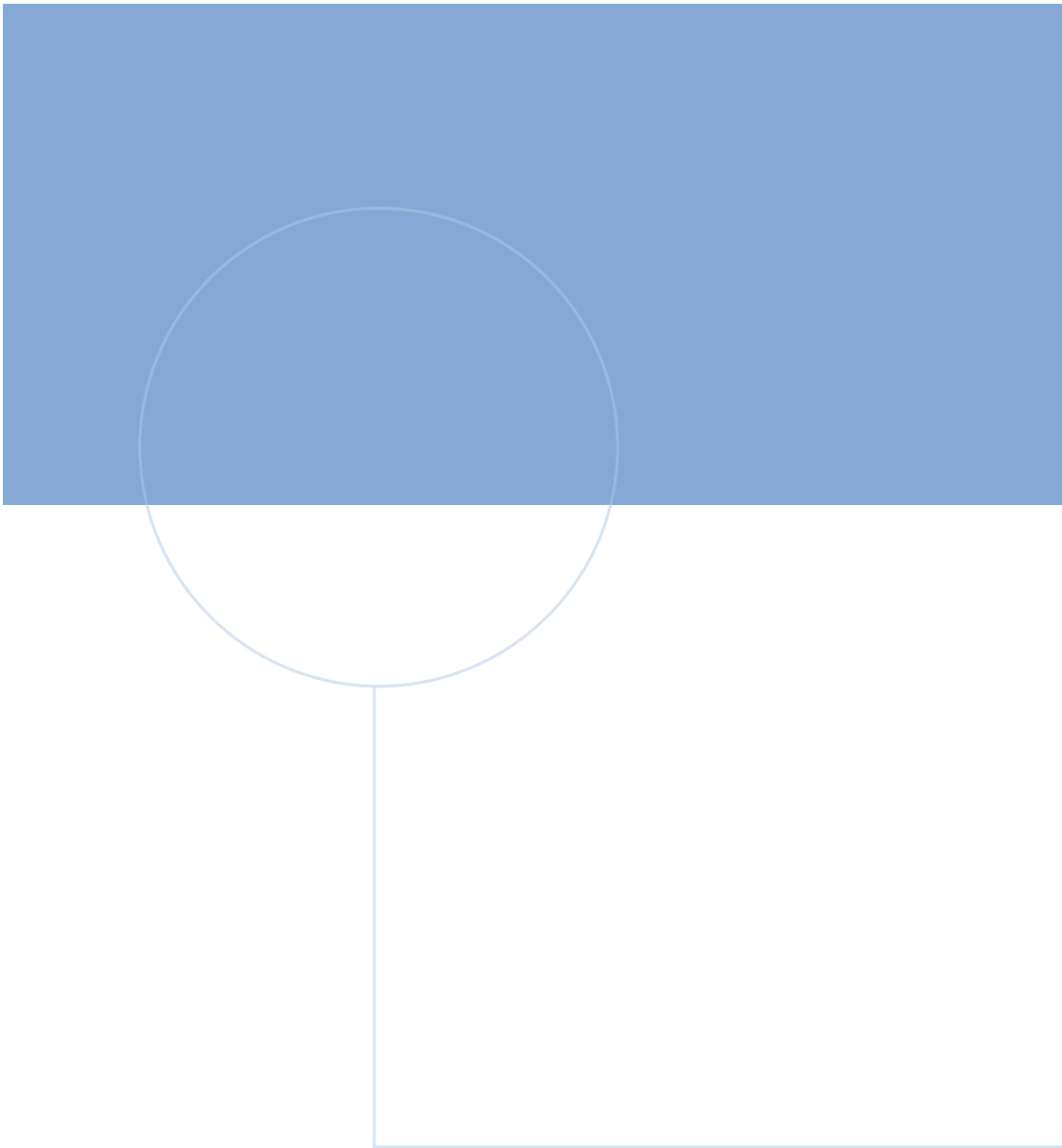
[22] Google, "Colaboratory," [Online]. Available: https://colab.research.google.com/.

[23] Wiki, "Wikipedia," 11 November 2020. [Online]. Available: https://en.wikipedia.org/wiki/OpenCV. [Accessed November 2020].

[24] Google, "Colaboratory," [Online]. Available: https://colab.research.google.com/. [Accessed 2020].

8 FIGURES

9 APPENDICES



NTNU

Norwegian University of
Science and Technology