Utrecht University

Master's thesis

Driver Handheld Cell Phone Usage Detection

Jannes W. Elings

supervised by

dr.ir. Ronald Poppe

September, 2018

ICA-4165411

Abstract

The usage of cell phones by car drivers leads to a lack of attention to the road and an increased chance of accidents. The Dutch police is tasked with fining these drivers. Current fining methods require drivers to be caught red-handed. In this work, it is demonstrated that application of computer vision techniques can lead to a massive decrease in man-hours necessary by automating the phone usage detection process. 2038 images of drivers were collected and classified into risky (phone usage) and non-risky (no phone usage) behavior. A straightforward Convolutional Neural Network approach and a more intricate combination of phone, hand and face detection and hand classification were compared on this task. The combined approach performed best, with an accuracy of 86.4% and an F-Score of 0.70 (precision: 0.70, recall: 0.70). The study revealed that it is achievable to detect driver phone usage using computer vision.

Contents

1 Introduction

Use of a handheld phone while driving is dangerous and illegal. Research shows that a driver that uses his phone suffers from reduced peripheral detection abilities (50) and a slower reaction time (5; 28).

Statistics around the world confirm the danger of using a phone while driving. A US study (13) estimated that the risk of an accident occurring increases by a factor of 2.2 while handheld calling and a factor of 6.1 while texting. There is no specific data on the number of traffic accidents involving distraction from mobile phones in The Netherlands. However, from 2014 to 2016, the number of deaths in traffic involving cars and trucks have gone up from 202 to 260 per year (7), after years of steady decrease. In a 2016 poll (8), 98.7% of Dutch drivers said they perceive texting while driving as dangerous. In the same poll, however, 12% of them said they occasionally text while driving themselves. The poll also revealed that 10% of people occasionally make handheld calls while driving. Dutch drivers appear to be aware of the dangers of handheld phone usage, but they seem to lack self reflection.

The Dutch police force plays a large role in the public road safety in The Netherlands. It is their task to discourage people from using their phone while driving. They do this by issuing fines. Currently, drivers using their phone have to be caught red-handed, holding their phone while driving, to get a fine. This leads to drivers trying to get out of a ticket by simply putting away their phone temporarily when they see a police vehicle. The driver avoids the ticket and phone usage is not meaningfully discouraged. Moreover, the Dutch police force does not have the manpower to enforce this way of fining on a large scale.

For this reason, there is an interest in a way of detecting phone usage automatically. A well-functioning automatic system for detecting phone usage of drivers could lead to more people being caught, and more importantly, more awareness among drivers. The aim of this thesis project is to develop prototype software that uses computer vision to automate this detection. The software should be able to classify whether drivers are holding their cell phone with sufficient accuracy. A human will still need to confirm if the software was correct by looking at the footage, but the number of drivers that have to be checked should be significantly reduced when compared to watching the full footage.

This document introduces the goal, requirements and challenges of the thesis. The relevancy of the research will be discussed. With these factors in mind, related research will be discussed, along with the techniques that underlie them. Once these have been discussed, the data that will be used is reviewed. Then, the research questions are introduced. An overview is given of the pipeline approach, the main software that was developed for this thesis. The pipeline approach will be used as a tool for the experiments, which are described next. The results of the experiments will be laid out and discussed. Finally, conclusions and suggestions for future work will be given.

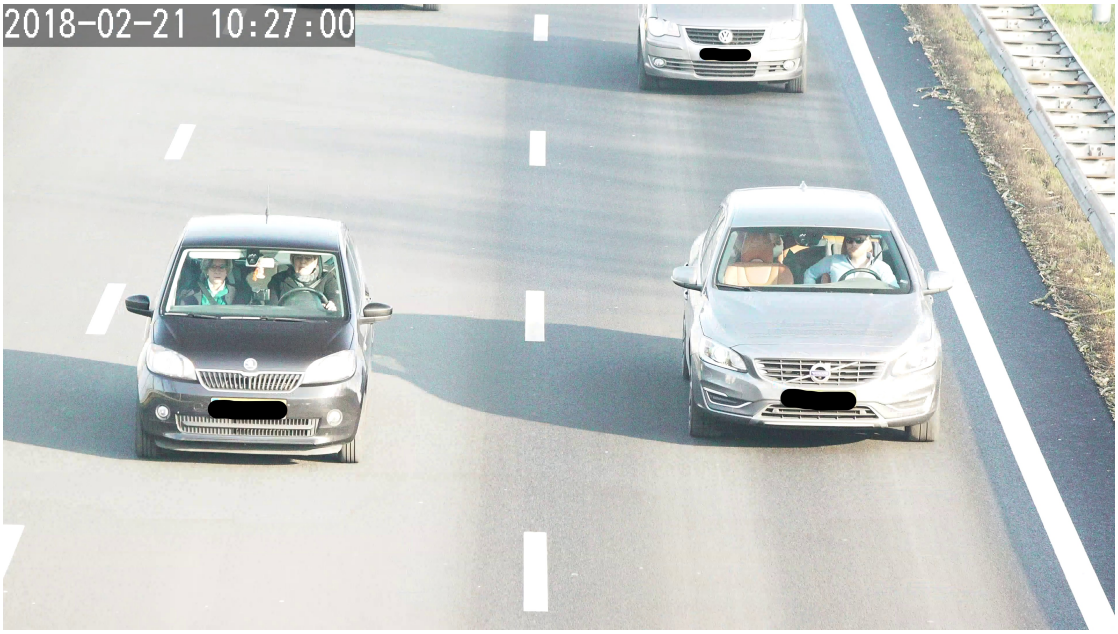


Figure 1: An example of the data that will be used.

1.1 Goal, requirements and challenges

The goal of the thesis will be to develop software that can detect if a driver is using his handheld phone using computer vision. The input will consist of videos from a camera mounted above the highway. The software should output images of drivers of which there is a high probability that they are using their phone. These images can then be reviewed by an employee of the Dutch police. Ideally, the number of drivers that have to be reviewed will only be a fraction of the total number of drivers. The software should be usable as a tool to answer the research questions provided in Section 4.

Previous research by Groot and Jiang (22) in the same setting showed that detecting phone usage is not a trivial task. The aim of this project is to build upon their research by starting with their method as baseline and adding improvements, all the while evaluating their impact. The end result should be a multi-stage algorithm consisting of techniques to subtract useful information from the images ended with classification that can detect phone usage more intelligently. Alternatively, it could end with the result that the baseline was the best performing method that was tested.

Requirements

The aim is to make use of off-the-shelf algorithms that can be combined to work for this use case. This approach is chosen because developing custom programs is costly in terms of time spent. Besides, the focus of the project should not be on writing code. It should instead be on experimenting with different scientific methods to solve

a practical problem. Utilizing publicly available algorithms will increase the amount of experimenting that can be done.

Algorithms that require large amounts of training data to learn from have been a recent trend in computer vision. This data is time consuming to gather. When using such algorithms, the preference will be given to publicly available training data over gathering custom data. This will not always be possible. For example, the final classification has to be trained on custom data regardless. Nevertheless, for intermediate algorithms, used for detection of for instance faces, pre-trained models will be preferred.

The drivers in the video were categorized by hand into a few different classes. The chosen classes are:

Handheld calling Driver that has his phone in his hand, next to his head.

Handheld phone usage Driver that is holding his phone and is using it for something other than calling.

Holding something else Driver whose hand or hands are visible and at least one of his hands holds something that is not a phone.

Hand close to head Driver whose hand or hands are visible and at least one of his hands is close to his face (hand on chin, cheek, behind head or similar).

Hand off the wheel Driver whose hand or hands are visible and at least one of his hands is not touching the steering wheel.

Non-risky behavior Driver that has one or both of his hands on the steering wheel or whose hands are not visible in the image. Notably also includes drivers that are probably using their phone, but are doing so without it being visible by the camera, by for example keeping the phone behind the dashboard.

Some classes overlap. In such a case, the one first from the top is chosen. For example, one could argue that a driver that is handheld calling could be classified as both *Handheld calling* as well as *Hand close to head* or others. In this case, it should be classified as *Handheld calling* because it is the first applicable from the top.

When completely finished, the software should be able to distinguish between risky behavior (a driver visibly using his phone) and non-risky behavior (the rest). The classes above are only used as input to see where the different types of driver behaviors end up. This can for example make it possible to determine what behavior often leads to false positives.

Challenges

The chosen camera position above the highway makes it so that people holding their phone behind their dashboard will not be detected. There is not much that can be done about this, but it can be seen as a challenge for this project.

Quite a lot of research has been conducted that is focused on detecting handheld calling while driving. Detecting both calling and texting is a more challenging and less tried problem. Drivers that are handheld calling will almost always have their phone next to their head, whereas drivers that are texting do not have a similar distinct pose. The space of possible different configurations of 'hand holding phone next to head' is smaller than the space for 'drivers using cell phones in their car'. So to adequately train classifiers for both sets, the amount of training data needed for the latter is larger than for the former. Since a very small percentage of collected images actually contain drivers using their phone, collecting sufficient training data can be a challenge. This will be mitigated by collecting as much data as possible and by using smart features that reduce the amount of data that is necessary.

Phones are very small, detecting them is impractical. Therefore it might be necessary to supplement phone detection by detecting specific poses of the driver or other clues that imply phone usage. Detecting something that implies phone usage instead of actually detecting phone usage itself will lead to false positives. By tuning parameters and using state-of-the-art algorithms, the amount of false positives will be kept to a minimum.

Finally, there is no control over the lighting conditions. Reflections on the windshield and low brightness during the evening or at night can render the driver almost invisible to the camera. Bad lighting is a common challenge in computer vision and it can make it difficult to classify the behaviour of drivers. Usage of a polarizing filter together with image processing techniques will mitigate as much of the influence of bad lighting as possible.

1.2 Relevance of the thesis

For society

The impact on society is intended to be sizable. Fully working phone detection software would lead to a dramatic time reduction for the police force when issuing fines for phone usage. This leads to more people being fined for their dangerous behavior and hopefully to more awareness among the Dutch population and safer roads altogether. Society would thus benefit in the form of more safety and less traffic jams due to a reduction in car accidents.

For science

A better understanding of phone detection from video would help increase insight in how to tackle such problems in the future. Because there are many different ways to do this detection, every research project that attempts this in a different way helps to guide future research towards higher performing methods. Even if performance turns out to be lacking, it would help researchers steer away from this method.

For technology

The Dutch police force has not found any published software that meets their needs in this regard. If the project software works well, it could be a stepping stone for other technology that aims to improve road safety. It could also be shown as a proof of concept for the development of similar solutions in other countries.

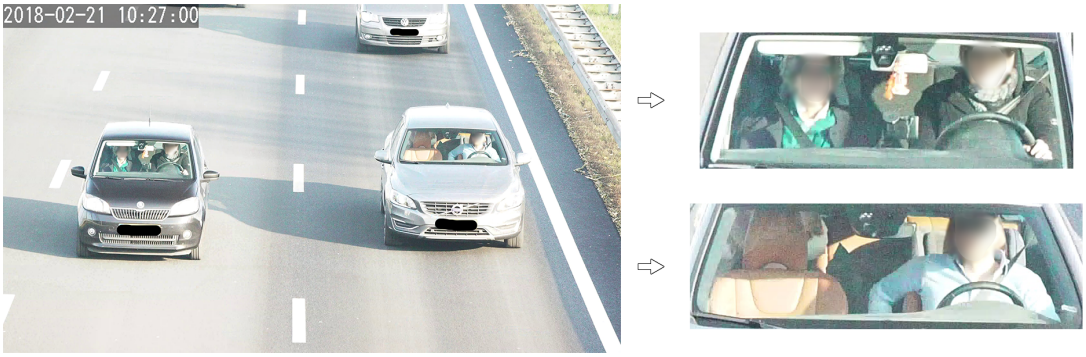


Figure 2: Detecting windshields in video frames.

2 Related work

The detection of phone usage in cars can be approached in multiple ways. For this project, the input will be an image or video and the output will be a classification and a location. The steps in-between can vary, however. The next sections will give a high-level overview of the different possible steps in-between and the related research.

2.1 Locating the driver

Since the input is going to be a video or image of the highway, the regions containing the driver will be small parts of the entire frame. The position of the driver within the frame is variable in time and there are times when there are no or multiple drivers visible in the same frame. For these reasons, it is necessary to first find where the drivers are located.

The task of classifying if an object is present and locating it in an image is called object detection. The first step of the final software needs to be some form of object detection to find the driver regions. It could just be person detection, but also a combination of person, car or windshield detection and image cropping. For example, Artan, Bulan, Loce and Paul (2) first detect windshields and then detect faces within them. Groot and Jiang (22) detect cars and then detect the rightmost person in the car. There is not one best way, the most important aspect here is robustness. Because since the classification will only be applied to the driver regions, any errors in this step will propagate to the next steps.

There are many object detection algorithms that achieve high accuracy on simple objects like cars and windshields. The Dutch police has already developed a simple tool based on Local Binary Patterns (36), a method that can determine the similarity of patterns. Their tool can detect windshields based on number plates. It works by looking for number plates in a set region of the video frame. When a number plate is detected in this region, the windshield of the corresponding car is detected and saved, together with

the number plate number and a timestamp. The tool works with reasonable accuracy.

The project will not be focused on this part of the complete handheld phone usage detection pipeline. Since the tool developed by the police works reasonably well, it will be used for this purpose.

2.2 Classification-only approach

Once the driver region has been located, the next procedure could be to classify immediately. By simply taking all of the pixels of the driver region, generating image features from them (using for example a Convolutional Neural Network or a Deformable Part Model as discussed in Section 2.3.1) and classifying them using any classification algorithm (using for example a Support Vector Machine or a MultiLayer Perceptron). This way, no prior knowledge about the contents of the images is used and no assumptions are made.

A positive aspect of just generating features and classifying is that it is relatively simple. The algorithms mentioned above are algorithms with open-source implementations that can do this quick and robust. Not having to develop a custom algorithm also saves time developing, testing and debugging, which is especially important when research time is limited.

The simplicity also leads to the biggest drawback of this way of handling things. The context contains much information that the algorithm could benefit from that is not used optimally. For example, a human would know that the images will always feature drivers, and that drivers have a face and hands. If one locates the hands of the driver, it is possible to see if they are both on the steering wheel. Knowing if the hands of the driver are on the steering wheel is very informative for classification. Humans can immediately grasp this, but the classification algorithm would have to learn this on its own. By adding positions of meaningful object parts (such as hands) as features, the classification could be steered in the right direction. If the notion of meaningful object parts is missing, like in these algorithms, this information is lacking and classification is thus solely dependent on the raw pixel values.

A consequence of letting the algorithm figure it out is that more training data is necessary than if it were given additional contextual information. It simply needs more examples for it to learn the desired relationships on its own. Like mentioned in Section 1.1, getting enough training data is already a challenge for this project. Requiring more training data is therefore a significant drawback of this approach.

2.3 Feature extraction and alignment

From the straightforward approach it becomes clear that looking at contextual information and meaningful object parts could be beneficial for the accuracy of the classifier.

Aside from adding information to the classifier, having as little variation in the data as possible can also benefit classification accuracy. In the perfect scenario, the only variation is the exact difference in classes that is being captured. To reduce the variation in unwanted areas, the data can be aligned.

This section will focus on possibilities and accompanying research for these two procedures.

2.3.1 Object detection

Many alignment approaches require getting object position information from the image. Important objects in the driver region could for instance be the drivers' face, the drivers' hands or the steering wheel.

In this section, the general methods for detecting objects will be discussed. There are more specialized algorithms for person, face and pose detection, which will be discussed in upcoming sections.

Custom features

Object detection algorithms make use of features. Features are numerical values that are extracted from an image and contain information that describes the image or parts of it. Feature values can be used to analyze the image and decide what is on it. Since pixel values are numerical, features are often (small) images themselves. These image features can for example be edges, points, shapes or textures. Although features can be regular numbers or vectors as well. An example of this would be the positions of the different body parts of the driver.

Creating features by hand can be beneficial as a starting point. Algorithms that generate their own features often do so in an incremental way. By starting with handcrafted features, these algorithms get a head start and will therefore converge more quickly. Alternatively, additional hand crafted features can be added to an algorithm just before the classification step to improve accuracy.

Convolutional Neural Networks

An example of an algorithm that generates its own features from pixels is a Convolutional Neural Network (CNN) (31), a biologically inspired type of Artificial Neural Network (ANN). Where hand crafted features are limited by human imagination, features created by CNNs can be of arbitrary patterns and are therefore more flexible. CNNs have gained a lot of popularity in the field of Computer Vision in the past years due to their excellent performance on various visual classification tasks (37). A CNN consists of multiple convolution layers, alternated with pooling (sometimes called sub-sampling) layers.

The first layer of a CNN is a convolution layer. In a convolution layer, a matrix called a

kernel (also called filter) is applied to the input pixels using a sliding window approach. The output becomes a transformed and slightly smaller version of the input image, called an activation map. In image processing, kernels are often used for edge detection, blurring or sharpening. That is when they are manually created. The CNN learns them through back-propagation. So it creates kernels that capture a pattern, but do not have to be understandable by humans.

Pooling means combining groups of input values into a single output value. The activation maps that are outputted by the previous layer are split into rectangular groups of values. The pooling layer combines the groups, leading to a smaller output activation map. Note that pooling works on values, not on pixels. So for instance pixels represented as RGB values will just be treated as if they are an activation map with a depth of three. The main goal of pooling is to help achieve local and spatial invariance. There are different types of pooling layers. Most common is the max pooling layer, where the max of the input values is used as output values.

The convolutional neural network consists of combinations of convolution and pooling (and sometimes other types of) layers, repeated multiple times at different scales. The kernels in the lower level layers measure more generic features, like edges and patterns. The kernels in the higher level layers combine the lower level features into larger, more specific structures, like parts of objects or even complete objects.

The activation maps that are outputted by the last convolution or pooling layer are then often fed as input into one or more fully connected layers to do the actual classification. The fully connected layers are like a regular Multi-Layered Perceptron (MLP). The convolution and pooling layers can also be used without the fully connected layers for just feature generation and combined with other classifiers like an SVM.

Instead of training a CNN from scratch, which requires vast amounts of time and computing power, CNNs are usually trained with the use of transfer learning: using the information from a different trained network to give the new network a head start. By acquiring a CNN pretrained on a different dataset and substituting the fully connected layers with new ones, the features are reused and only the classification has to be trained for the new dataset. Alternatively, while training the classifier, the non-fully connected layers can also be tuned to the new dataset by continuing the back propagation. This can be done for all layers or just for the higher level ones by keeping the lower levels fixed.

The fact that the CNN generates its own features is a major advantage, as it reduces the amount of manual preprocessing needed to use it. On top of this, the CNN learns which of the features are most important so there cannot be any human bias in this regard.

A disadvantage of the CNN is that its size and complexity make it difficult to customize and interpret the model.

CNNs work on the entire input given. So without modification, a CNN works on the entire image. This is fine for classification. For object detection, however, localization

is necessary. Therefore, regions of the image have to be selected to use the CNN on. Deciding on regions to apply and at what stage in the algorithm has been a hot topic in object detection. Many variants have been proposed. Some of the ones that are discussed in papers about driver phone recognition include:

Regions with CNN features (R-CNN) Originally proposed by Girshick, Donahue, Darrell and Malik (20), the R-CNN was designed specifically for object detection. The main improvement of R-CNN over regular CNNs is the use of region proposals. Regions within the image are proposed based on similar texture, color or intensity. Intuitively, this can be seen as trying to find objects first and recognizing them second. After the regions are proposed, a regular CNN tries to classify the object in the region. The found regions are then turned into bounding boxes. Finally, the locations and sizes of the bounding boxes are tuned using a class-specific bounding box regressor.

The regular R-CNN proved to be quite slow due to it requiring many passes through the CNN per image and because it requires training three different models simultaneously (the CNN, the classifier and the regression model). These drawbacks are addressed in the next proposed version, called *Fast R-CNN*, which was proposed by Girshick (19). The region proposal step is moved to after the feature extraction step, leading to one pass through the CNN per image instead of thousands. The other improvement is the replacement of the classifier and the bounding box regressor with their corresponding ANN layer counterparts, essentially integrating the two other models into the CNN.

Because the region proposal step is after the feature extraction step in Fast R-CNN, the extracted features can be used to generate the region proposals. This was the key insight of the third R-CNN version, *Faster R-CNN*, proposed by Ren, He, Girshick and Sun (42). By doing it this way, the region proposal is almost free in terms of computing time, leading to an even faster algorithm.

Single Shot Detector (SSD) The region proposal step of Faster R-CNN mentioned above proposes class-agnostic regions. In the next stage, these proposals are classified by a CNN. Instead of classifying in the second stage, a network could propose class-specific regions immediately: a single shot detector. The term SSD was coined by Liu et al. in regards to the object detection method they proposed (34). However, the method of directly predicting classes without requiring a second stage classification operation is not new. In this work, all methods that handle detection in this fashion are meant when talking about SSDs. Using this definition, DSSD (18), YOLO (39; 40; 41), Overfeat (44) and RetinaNet (32) can also be classified as such. Compared to Faster R-CNN, SSDs appear to be slightly faster and slightly less accurate in object detection tasks. Their speed can be attributed to the absence of the second stage classification. Although the RetinaNet paper claims to achieve comparable results to Faster R-CNN without sacrificing on speed.

Region-based Fully Convolutional Networks (R-FCN) Another method that tries to do what Faster R-CNN does, but faster, is called R-FCN. Proposed by Dai, Li, He and Sun (11), R-FCN is built on the insight that Faster R-CNN calculates features for

every bounding box proposal in an image to be able to classify that region. It does so by cropping the features from the output of a layer in the middle of the CNN. There are typically hundreds of these proposals per image. R-FCN moves the cropping of these features to the last layer before classification. So the box proposals do not have to go through a part of the network, leading to a speedup of the algorithm.

The above discussed CNN meta-architectures are examined further in an overview paper by Huang et al. (25). The trade-off between accuracy and speed for different parameter settings and feature extractors are also discussed. From their work, Faster R-CNN can be concluded to be the most accurate overall.

There are some things that have to be done before a CNN like Faster R-CNN can be incorporated into the project pipeline. The network has to be trained to detect the specific object of interest. Training is very computationally intensive and therefore requires a computer with a high-end graphics card to do somewhat quickly. Not only that, large amounts of annotated training data needs to be collected to get the highest performing detector.

Deformable Part Models

Another major group of object detection algorithms are based on Deformable Part Models (DPM), as introduced by Felzenszwalb, Girshick, McAllester and Ramanan (17). DPMs are a way to model objects as a combination of parts. They can also be used for object detection. Working with parts can lead to more robust classifiers than when working with one complete object because when the object is in different configurations, the classifier can still detect the individual parts. To give an example, a person detector trained on standing people will have problems detecting a sitting person, as it looks like a different object. This is because a human, as opposed to a rigid object, is deformable. The different body parts can occur at varying positions and still be part of the person. A parts-based detector that can handle deformation can detect that the body parts of the sitting person are the same as those of the standing person, except rearranged. Therefore it will have less trouble detecting it as a person. Aside from generalizing well to unseen configurations of parts, the algorithm can deal with partial obstructions of the object by hallucinating the invisible parts.

A DPM object model consists of a root filter, part filters and deformation models. The root filter is a Histogram of Oriented Gradients (HOG) of the entire object. The part filters are higher resolution HOGs of parts of the object. Each part filter has a deformation model: a function that represents the cost of placing the corresponding part filter relative to the root template.

When training, the DPM requires only bounding boxes of the complete object, not of the individual parts. The DPM can create the root filter, part filters and deformation models by using something called Latent SVM. The Latent SVM is a variation on the regular SVM. The difference is that the Latent SVM is specifically designed for DPM. It considers the DPM parts and deformation models as hidden variables that it tries to

optimize. Its aim is to find the parts that are most informative during object detection.

When detecting an object, the DPM will calculate a detection score, based on the position of the root filter, the part filters and the relative position of the part filters. The score that a relative position gets is dependent on the deformation models. To calculate this score, given an input image, the DPM creates a pyramid of HOG features of the image. It will then place the root filter in a HOG image of low resolution and the part filters at a higher resolution at their relative positions. The total detection score of the model at this position is the sum of all filter scores minus the displacement of the parts with respect to the root model. This means that if the parts look different, the score goes down. If the parts are in an uncommon position, the spatial deformation (displacement) goes up and the score goes down as well.

Interestingly, in a technical report, Girshick, Iandola, Darrell and Malik (21) show that DPMs and CNNs are not two completely distinct methods. A DPM can be formulated as a CNN by mapping each DPM step to an equivalent CNN layer. By showing that this CNN version achieves higher scores on the Pascal 2010 (16) object detection dataset than a regular HOG-based DPM, they claim that it learns better features. Savalle, Tsogkas, Papandreou and Kokkinos (43) have constructed models using a DPM combined with the features of a CNN, also leading to higher object detection (on the Pascal 2007 dataset (15)) performance than when using just a DPM. Yet it should be noted that regular CNN performed even better.

In general, CNNs appear to perform better than DPMs. However, there is an upside to using DPMs. Whereas CNNs are very dependent on large amounts of training data to learn meaningful features and not overfit, DPMs perform well using a relatively small amount. This is due to the explicit modelling of the parts based structure, which eliminates the need for the algorithm to learn it on its own.

2.3.2 Person detection

Using person detection is very similar to using object detection. Almost all algorithms capable of person detection are also usable for regular object detection. Which seems sensible, because a person can be seen as just another object. So the object detection algorithms discussed above are also viable person detectors. However, there have been some methods developed specifically for person detection.

Bourdev and Malik (4) proposed to detect people using poselets: 'Poselets are tightly clustered in both appearance space (and thus are easy to detect) as well as in configuration space (and thus are helpful for localization and segmentation).' Poselets are similar to the part filters in a DPM, they describe a small part of the object that is being detected, encoded in a HOG feature. A face looking forward could be a poselet, but crossed legs could also be a poselet. In the case of this project, one could imagine a hand next to a head as a poselet that could be interesting. The difference with a DPM is that the parts are not deformable. They have a fixed structure instead.

A regular person detector can be created by classifying poselet activations and positions. Poselets can also be utilized for pose estimation. A pose is a particular configuration of the body parts of a person. Each poselet can be labeled with its corresponding action. Detected poselets can then function as features of a classifier of the action or pose that is portrayed.

2.3.3 Pose estimation

Aside from poselets, there are other pose estimation algorithms that could be useful for this project. Estimating the pose of the driver can give a great amount of additional information. For example, when handheld calling, many drivers will keep the phone close to their head. Knowing that the driver is in a pose where the hands are close to the head will provide the software with more information to base its classification on. It will most likely not classify this as a driver with both hands on the wheel. So just knowing the pose can eliminate some classes as viable options.

The pose of the driver is very informative when classifying. Since phones are small and not always visible due to occlusion and other factors, adding driver pose information can benefit classification of phone usage.

A disadvantage of estimating poses is that it adds another computing step between input and output, meaning errors in this step will propagate to the classification step. Besides errors propagating, the resulting software becomes slower and more reliant on different applications.

Complete pose estimation

Following is a quick dive into research on detecting human poses.

Based on DPM, the flexible mixtures of parts method was proposed by Yang and Ramanan (56) as a method to estimate human poses. Parts are smaller and more in number than those of a regular DPM. The parts also have a fixed orientation. In regular part based models, the parts would be stretched and rotated to fit. In this model, a different part is taken altogether. The reasoning is that a part can look different at different locations. They claim that due to these properties, the dependency of global geometry on local appearance is better captured.

A method proposed by Chen and Yuille (9) stands out because of its high performance on upper body pose recognition. Since the images used in this project show only the upper body, this method might be interesting to experiment with. It approaches object detection similar to a DPM in that it sees the human body as a graph with body parts as nodes. In this case, the nodes are at the joints of the body. The first difference with DPM is that parts are not modelled using HOG descriptors, instead they opt to use CNN features. Another difference is that the image patches around the found joints are put through a CNN that predicts the spatial relationship with the joints that are connected

to it. The idea is that from an image of, for example, a bend elbow, the shoulder and wrist positions can be deducted. So the method combines a graph representation with a CNN.

Face detection

Instead of detecting the complete pose, specific body parts can be detected, like faces. Knowing where the face of the driver is can benefit the algorithm. For example, some of the phone detection research uses face detection to search for phones near the drivers' ears.

For the thesis project, the faces that would be detected are almost always going to be looking straight ahead at the road in front of them. This is fortunate, because under these conditions all capable face detection algorithms should work relatively well.

In 2004, Paul Viola and Michael J. Jones (53) proposed a real-time face detection method using Haar-like features (52). Nowadays, the Viola-Jones face detector is a proven method of detecting faces. It can be used for detecting other objects, but is mostly used for faces. The main idea comes from the insight that faces almost always have darker and lighter colors in the same regions. For example, the eyes of a person are almost always darker in color than the cheeks. When using Haar-like features, relations between darker and lighter regions like these are saved in the form of a feature.

The training of this detector starts by constructing features in all possible positions and scales of rectangles consisting of a darker and a lighter area. These features are then placed on the training images and a threshold is calculated for the feature that results in the best classification of faces and non-faces. All features are then applied to all images. Misclassified images are given a higher weight and all features are again applied to all images. This procedure is repeated until a chosen accuracy or other stopping condition is met.

The Viola-Jones detector is most often combined with Adaboost, a method of creating a strong classifier based on weak classifiers. The weak classifiers are the features in this case. During the detection phase, selected features are placed on the image using a sliding window approach. the best performing features are applied first. If they fail, the window is discarded. Otherwise, the next few features are tried. If all features succeed, the window is classified as a face.

A Viola-Jones detector works well with frontal faces, but not as well with rotated faces. The light and dark areas in a face can be completely different when the face is rotated. There are extensions that can do this, however.

Zhu and Ramanan (58) propose a DPM based method to localize faces. The method is specifically geared towards use 'in the wild', meaning it also works on rotated faces and under differing lighting conditions. An important aspect of this algorithm is that it arranges the parts in a tree structure, which leads to increased computing speed. This method is used by Artan et al. (2) in their phone detection research for both windshield

and face detection. From small tests, it appears to achieve higher accuracy scores than Viola-Jones, but it executes slower. This is what was expected, looking at the complexity of the two models.

A few of the found methods for phone detection use *MTCNN* of Zhang, K., Zhang, Z., Li and Qiao (57). Their approach consists of a series of steps. First, the image gets resized multiple times at multiple scales. Then, a CNN (called P-Net) is used to obtain proposal bounding boxes. The bounding boxes go through two more CNNs (R-Net and O-Net) in succession to remove false positives and detect landmarks. This algorithm achieves state-of-the-art performance on face detection datasets FDDB (26) and WIDER FACE (55).

Hand detection

Other very informative parts of the driver pose are the positions of his hands. All but one of the chosen categories for classification are based on hand visibility. As mentioned before, hands can be seen as just another object and thus they can be detected by object detection algorithms. There are also specific hand detection algorithms that claim to be better suited to the variable shapes of hands.

One of these algorithms was proposed by Mittal, Zisserman and Torr (35). They combine three different hand detectors, each proposing hand bounding boxes. The hand detectors are each based on a different detection aspect, one for hand shape, one for context and one for skin color. Then, the final confidence of the bounding boxes is decided by an SVM classifier. The best performing hand detector scores a precision of 48.2% and a recall of 85.3%. The researchers also published an annotated hand dataset that could be useful for this project.

Another method of hand detection specifically tested on drivers was proposed by Siddharth, Rangesh, Ohn-Bar and Trivedi (46). Their method relies on YOLO (39) pre-trained on the Pascal VOC dataset (14) and then trained on the VIVA dataset (12), which is a dataset consisting of videos taken from inside the car with the drivers' hands annotated. YOLO proposes hand regions of which HSV histograms are generated. The histograms are then clustered and later classified using a random forest classifier. This procedure helps remove false positives. From the remaining bounding boxes, masks of only the hands are created using the HSV histograms. Both the bounding boxes and the masked images are used to generate HOG and VGG (a type of CNN) features. The hand detection accuracy is not reported, but the precision and recall are 74.1 and 47.2 for the part of the dataset with hands larger than 70 pixels and from only a single view. Interesting about this research is that they also try to detect phones in the hands. Their best performing phone detector achieves an accuracy of only 53%, showing the difficulty of this task. They also find that masking the hand significantly improves the classification accuracy.

2.4 Computer vision approaches for phone detection

Now that most of the underlying methods have been introduced, the research that is focused specifically on using computer vision to detect phone usage of car drivers can be discussed. Now following is an overview of recent vision based research with this goal.

Berri, Silva, Parpinelli, Girardi and Arthur (3) propose an SVM-based model to detect drivers using their cellphone to make calls. They use Haar-like features and Adaboost as a classifier to detect faces. Once the drivers' face is detected, the skin pixels are isolated and a region around the face is checked for skin colored pixels that could be the driver's hand and arm. The percentage of skin pixels found, combined with Hu's Moments (24) are used as features. An SVM is used for classification. Over five videos they seem to reach reasonable accuracy, especially in the detection of non-risky behaviour. However, they removed some data of videos taken on very sunny days to get these results. The relative simple approach of this paper shows that under the right circumstances, detecting handheld calling is possible with decent accuracy. The difference in performance when the sun is shining or not appears to be a significant drawback, however.

Seshadri, Juefei-Xu, Pal, Savvides and Thor (45) make use of the space next to detected faces to look for hands and cell phones. The input videos come from the Strategic Highway Research Program (SHRP-2) dataset, a dataset that is used in other related research as well. The dataset features drivers driving and performing a series of tasks, like making a phone call, reporting the vehicle's speed and turning the radio on and off. It is filmed by multiple cameras at various positions in the car. It has recordings of a face view, lap view, hand view and a forward view. Their approach is to first detect the drivers' face using Supervised Descent Method (54). Then, two regions on either side of the detected face is extracted. These regions are used to create two feature representations, one using the raw pixels and one using HOG descriptors. The features are then classified using Real Adaboost, an SVM and a random forest classifier. The highest accuracy feature-classifier combination turned out to be HOG-Adaboost. The highest AUC of the ROC curve was achieved by the HOG-SVM combination. The researchers note that they did filter harsh lighting conditions, large face occlusions and face pose variatons from the data. Interesting about this research is that the different classification techniques do not differ a lot in performance, when measured in AUC of the ROC curve. A big difference between this research and the project is that only handheld calling is detected in this research, whereas for the project, texting also has to be detected. The regions of interest can thus simply be cropped from around the detected face. On the one hand, this makes the research less useful for the current use-case. On the other hand, the research shows that if the hand regions can be detected consistently, high performing classifiers can be trained.

A method that boasts higher accuracy on the SHRP-2 dataset is proposed by Le, Zheng, Zhu, Luu and Savvides (29). They claim that regular Faster R-CNN has trouble with detecting small objects like hands and faces in this dataset. For this reason, they use a method they call Multiple Scale Faster R-CNN (MS-FRCNN). MS-FRCNN is a different

layout of the neural network that trains object proposals at various scales. It is shown that this network achieves slightly higher accuracy than regular Faster R-CNN. The network is used to detect steering wheels, hands and faces. When the hands are close to the face, the driver is classified as using his phone. In the paper it is mentioned that their method is faster than the one by Seshadri et al. (45), because it does not do the costly facial landmarking step. The tables in their paper display a different result, however. In Table 1, the FPS of Seshadri et al. is higher than that of Le et al. themselves. It is unclear what causes this discrepancy. The achieved phone detection accuracy is very decent. The precision and recall (or similar measures) are not mentioned and it is therefore unknown whether the results could be skewed in either direction. From this paper, the MS-FRCNN algorithm could be quite interesting for this project, because it will feature small objects as well. The only issue is that from the results in this paper it can be concluded that it performs only very slightly better than regular Faster R-CNN. Therefore, it does not warrant the time that would be invested in implementing it and it will not be used. Detecting whether the drivers' hand(s) are close to his/her face could be interesting to implement in this project.

In another paper, published a year later, Le, Zhu, Zheng, Luu and Savvides (30) propose a framework that detects unsafe driving behaviour in general. Part of the framework is a hand on phone detector. The first step of their approach is to train a CNN on driver body parts (face, torso and seat belt). The CNN features are used to create a probability map that contains the confidence score of a pixel belonging to the object of interest. These maps are used as initial seeds for a Semi-Supervised Normalized Cuts segmentation algorithm (10) that creates object proposals. Then, R-CNN extracts features from the bounding box of the region and from the foreground at different scales. An SVM gives a score for each class to each candidate. Finally, Pictorial Structures (PS) (1), a technique similar to DPM, is used to decide the final candidates. By using CNN features, as opposed to HOG or SIFT features that are commonly used with PS, they claim to have better performance than other research that uses PS. This seems like a reasonable claim because other research (21; 43) has shown that CNN features work better in combination with DPMs as well. The method is tested on the VIVA (12) dataset mentioned before in Section 2.3.3. Their achieved performance is quite high, although it should be noted that the phone detection receives already detected hands as input. And the hand detection error is not given. Nevertheless, the techniques used seem promising and they could be interesting to try out during this project.

Where all of the above research focuses on videos taken by cameras from inside the car, Artan et al. (2) have used video from a near infrared (NIR) camera hanging above the road. This adds the challenge of having to find the position of the driver in a larger than usual search space behind a windshield that could be reflective. They solve this by first finding the location of the windshield using a DPM based method proposed by Zhu et al. (58) (see Section 2.3.3 for more on this method). This not only reduces the search space, it also guarantees that there is a face in that region. The face detection algorithm can therefore assume that the region that is most face-like is a face. Even when a face is partially occluded, it can still be found this way. This is essential because many of

the faces are occluded by sun visors. The faces are detected using the same method as the windshield detection. The found face region is combined with regions to the left and right of it and this region is fed into the classification pipeline. First, SIFT features are generated. Then, three vector signatures are created for each image using Bag of Visual Words (BoVW), Vector of Locally Aggregated Descriptors (VLAD) and Fisher Vectors (FV). The resulting vectors are classified using an SVM. Their results improve when classifying the right and left side of the head separately instead of classifying the head as a whole. The FV image signatures were classified with the highest accuracy. Interesting for this project is the use of vector signatures. This could be something to experiment with. The insight that only the most face-like region has to be found from the right side of windshields could also prove useful. Use of a NIR camera is something that could be considered by the Dutch police in a later stage of the development of this technology.

Groot and Jiang have worked on this phone detection task in the same setting (22). They have used the same videos as the ones that will be used in this project with the same intended outcome. Therefore their work is very informative for this project. They approach the problem in the straightforward way (as described in Section 2.2). YOLO is first used to detect cars. The driver is subsequently detected by taking the person detections from YOLO and choosing the bounding boxes with their center to the right of the center of the detected cars. From these bounding boxes, the one with the highest certainty is chosen. This driver bounding box is cut out of the image and the cutout is classified as risky or non-risky using a CNN. Notable here is that all drivers with one or both hands visibly off the wheel are classified as risky. The CNN has been trained on the right side of windscreen images taken from videos from a different video set. Due to the low number of risky training images, combined with an apparent focus on accuracy over other measures that balance precision and recall, leads them to preferring models with high accuracy, but low recall. Even though they did find a model with high recall in their parameter optimization searches. This might be the largest drawback of this research. The parameter tuning of the CNN has been mostly focused on achieving a low Mean Squared Error (MSE). Whereas it could have been focused on maximizing a measure like F-score or area under the ROC-curve. Parameters leading to a high F-score would probably include more false positives than a low MSE counterpart, but it would find more drivers that are actually behaving risky. Besides, since all risky classifications will be checked by a human to filter out false positives, they are less of a problem. From this paper, the idea of classifying hands off the wheel as risky might be useful for this project. If the amount of training data is too small, this could boost the number of positive samples. Another important lesson from this paper is that accuracy should not be considered as the be-all and end-all metric to optimize.

Authors	**Camera positioning**	**Objects that are detected**	**Features**	**Classification**
Artan et al. (2)	Outside of vehicle	Windshields, faces	SIFT (BoW, VLAD, FV)	SVM
Berri et al. (3)	Inside of vehicle	Faces	Haar + adaboost	SVM
Groot and Jiang (22)	Outside of vehicle	Vehicles, people	CNN	CNN
Le et al. (29)	Inside of vehicle	Faces, hands and steering wheel	CNN	CNN
Le et al. (30)	Inside of vehicle	Faces, facial components, hands, seatbelts and upper bodies	CNN	SVM
Seshadri et al. (45)	Inside of vehicle	Faces	Haar + SDM	SVM

Table 1: Overview of other research

3 About the data

The data that will be used consists of videos of cars driving on the highway. The videos have a resolution of 3840×2160 pixels. The frame shows two lanes of traffic. See Figure 1 for an example of a video frame.

From the videos, images of the windshields of the cars are extracted using the tool developed by the Dutch police, as mentioned in Section 2.1. The right side of these images (where the driver is located) is cropped. Our dataset consists of 2038 of these images of driver regions. The average size of a driver region is 370×296 pixels.

The 2038 images were taken from a set containing 61391 windshield images, see Figure 3. The 61391 images were manually categorized into the input classes mentioned in Section 1.1 by a few members of the IT department of the Dutch police using an in-house developed tool. The distribution among the different input classes can be found in Table 2. From the resulting classified images all phone usage images were taken for the new dataset. With 206 *Handheld calling* and 232 *Handheld phone usage* images, these input classes had the fewest items. The other input classes each had 400 images taken from them at random. More would lead to a large class imbalance, which makes training CNNs more difficult. Fewer would lead to an even smaller dataset and therefore worse expected results.

The original data was collected during 38 hours of filming above the same highway, near Utrecht, The Netherlands. Although it cannot be assumed that other roads will have the same percentage of drivers using their phones, the ratio can be expected to be very much skewed towards more *Non-risky behavior* on other roads as well. This is important to note, as the dataset consisting of 2038 images has a different class balance. This means that any images in the *Non-risky behavior* class, that are wrongly classified as phone usage in the dataset consisting of 2038 images, will lead to a larger amount of wrongly classified phone usage in real-life data.

Class type	**Input class**	**Total nr of images**	**Train images (75%)**	**Validation images (15%)**	**Test images (10%)**
Risky classes	Handheld calling	206	154	30	22
	Handheld phone usage	232	174	34	24
Non-risky classes	Holding something else	400	300	60	40
	Hand close to head	400	300	60	40
	Hand off the wheel	400	300	60	40
	Non-risky behavior	400	300	60	40
Total		2038	1528	304	206

Table 2: Overview of the dataset.

	Nr of images	Percentage of total
Non-risky behavior	55457	90.33%
Hand off the wheel	4131	6.73%
Hand close to head	600	0.98%
Holding something else	765	1.25%
Handheld phone usage	232	0.38%
Handheld calling	206	0.34%
Total	61391	100%

Table 3: The original dataset that the images were taken from. Note the skewed ratio between *Non-risky behavior* and the two risky classes.

3.1 Data statistics

The next section will discuss the possibilities of using off-the-shelf applications for pose estimation and face detection as these could be informative when determining driver behavior.

The pose and face applications were tested on the 438 images of drivers using their phone. The reason for choosing these classes is that the techniques have to work on images from these classes to work well when determining risky behavior.

Pose

To see if pose estimation would work with the input that will be used, OpenPose: a pose estimation library that produces state-of-the-art real time pose estimations for multiple persons was tried. OpenPose is based on a paper by Cao et al.(6). The videos they provide show impressive results. However, one cannot assume that the technique works as well in this setting, as the data that is used is quite different. The thesis dataset consists of images of relatively low quality, showing only the upper body of the driver and the drivers are partly occluded by the steering wheel. A quick test with the 438 images of drivers using their phone did not work very well. See Figures 3a and 3b. Even when it estimated correctly, it often did not find the lower-arm holding the cell phone. Of course this was just a quick look without own training and even without parameter tuning. Nevertheless, it is indicative of the difficulty of pose estimation on this dataset.

Face

Two face detection algorithms were tested to see how they perform on the data. The algorithms are the last ones discussed in Section 2.3.3, Face detection in the wild (58) and MTCNN (57). Both were tested on the 438 images of phone usage in the dataset.

(a) Correct estimation example (b) Incorrect estimation example

Figure 3: Pose estimation using OpenPose.

MTCNN turned out to be both faster and more accurate and is therefore an interesting option for face detection.

Face, hand and phone position distributions

To learn more about the data and to see if there was any truth to the idea that the pose of the drivers can indicate phone usage, the faces, hands and phone positions of the images in the dataset have been annotated. This was done by the author of this paper using the application LabelImg (49).

See Figures 4 and 5 for the results. Analyzing these distributions lead to a few insights:

- Phones are somewhat common in non-risky classes. These are phones inside a mount. Just detecting a phone would therefore not be enough to determine risky behavior. It does appear that mounts are positioned primarily near the center of the windshield.
- There is a lot of variation in phone positioning for drivers that have their phone in their hand, but not close to their ear. Although there appears to be a slight bias towards right-handed phone usage, the overall distribution is fairly even. This contradicts the idea that there is a single pose or a few poses that are indicative of phone usage.
- Drivers that are handheld calling do show a distinct pattern in hand position. However, as expected, hand positions of the *Hand close to head* and *Handheld calling* classes are very similar. Distinguishing between the two solely based on hand positions is not going to be possible.
- The non-risky class has far fewer visible hands than any other. All hands in this class were resting on the steering wheel.

From these diagrams it appears that there are some patterns, but due to the variation in the dataset there is no clear-cut pattern that enables distinguishing between risky and non-risky behavior based on position of hands and faces alone.

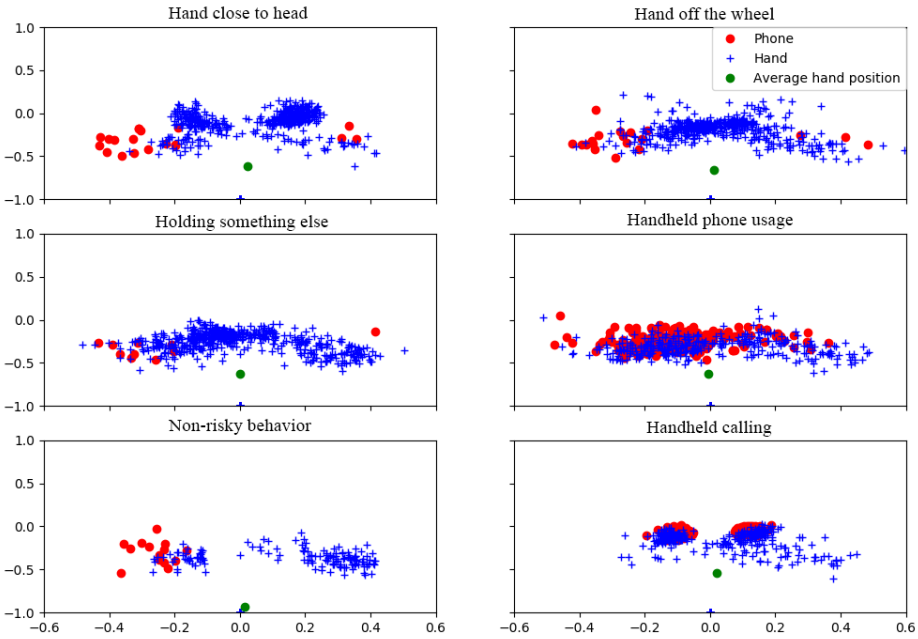


Figure 4: The distribution of the positions of hands and phones, relative to the position of the face of the driver. The face position is (0, 0) in all figures. The center of the windshield is on the left and the side of the car is on the right. The hand and phone positions are normalized for image size. When there were one or two hands not visible they were plotted at (0, -1) to make the average hand position more insightful. Phones in classes that are not the 'risky' classes are in a phone mount.

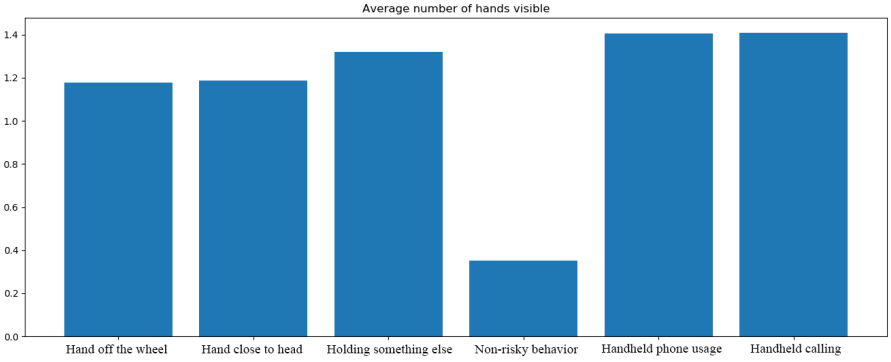


Figure 5: The average number of hands that were visible in the images for each class.

4 Research questions

First of all, the main goal of the thesis project will be to develop software that adequately classifies whether drivers are using their handheld phone or not. The developed software will be used as a tool to answer the questions presented below.

1. **Does a pipeline based algorithm outperform a straightforward single CNN based method?**
   The straightforward method was experimented with by researchers before (22). They concluded that there was room for improvement. It would be interesting to find out if a more intricate pipeline of detectors and classifiers for different objects in the image could outperform it.
2. **Does data augmentation improve classification scores?**
   Augmenting the data can improve performance of classification. Especially in settings with small amounts of data. The augmentation steps that will be implemented are:
   - Random rotations between (-6, 6) degrees.
   - Random shearing with a shearing angle between (-6, 6) degrees.
   - Random translations between (-0.1, 0.1) times the image size in both x and y directions.
   - Random zooming between (0.9, 1.1) times the image size.
   - Horizontal flipping with a probability of 50%.
3. **What input class can be classified the best?**
   There are a few different types of driver behavior that have been categorized into input classes. It would be interesting to see what class could be classified the best when evaluating the algorithm. The input classes can be found in Section 1.1.

Performance of methods will be measured by comparing their highest found F-scores.

5 The pipeline approach

This section will outline the design of the main software used to answer the research questions. The software is based on a pipeline of different detection and classification algorithms combined, and will be called the pipeline approach. For a complete visual overview, see Figure 6.

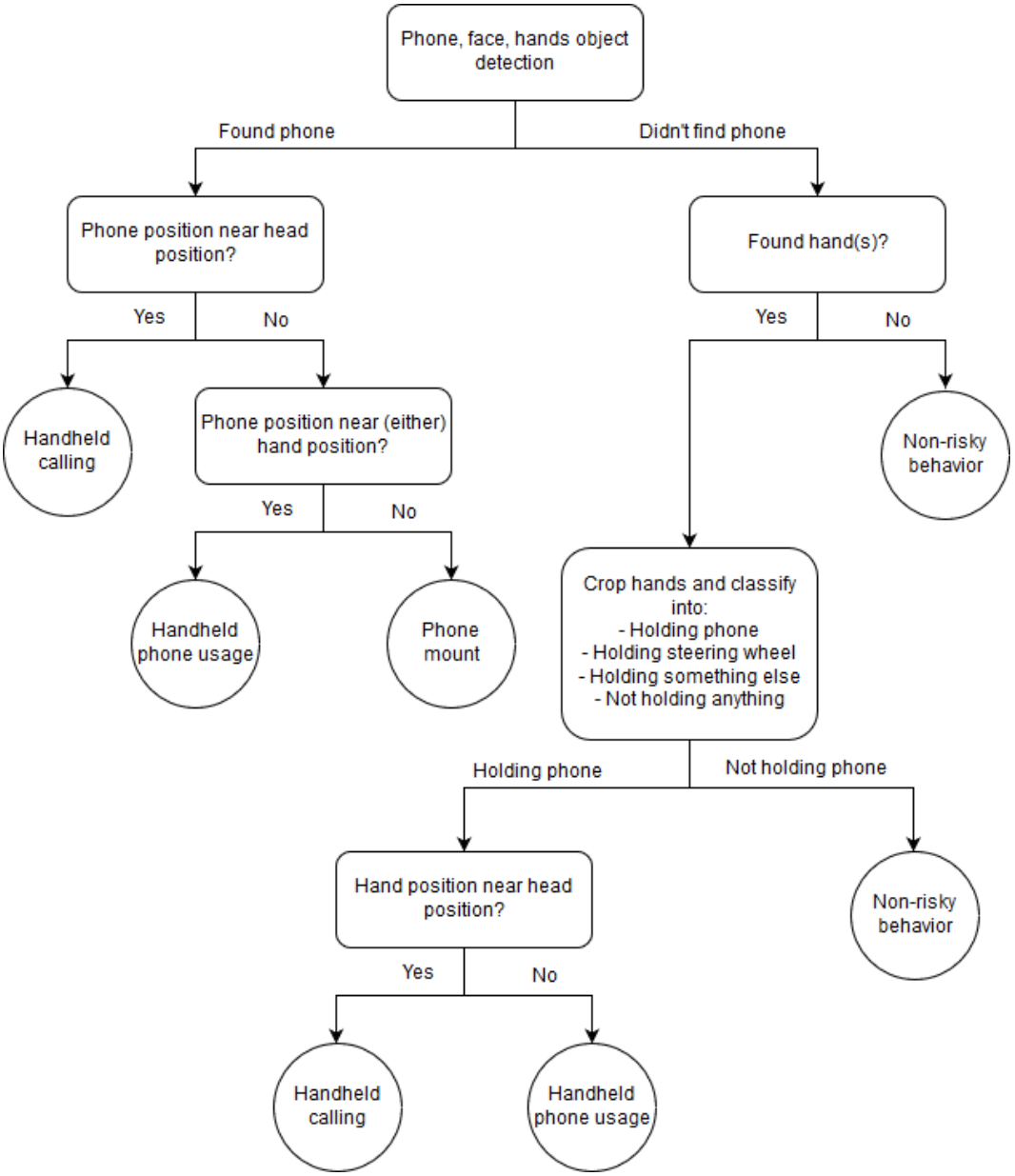


Figure 6: An overview of the pipeline.

An issue that a straightforward classification approach might suffer from is a lack of focus for the neural network. The network receives the full driver region image as input, but the relevant pattern (phone usage) is only visible in a small part. The poor signal-to-noise ratio, combined with the small amount of data makes for an environment where classification is difficult. The aim of the pipeline approach is to be less affected by the issues.

5.1 Input

As discussed before, the developed software will not concern itself with the detection of the driver region. It instead relies on images of driver regions as input. These images have the right side of the windscreen of a car on them. For training, as well as validation and testing, these are one image per driver (as opposed to sequences of multiple images per driver).

The experiments will thus be working with images instead of video. The reason for this is that the expected performance gain from better frame classification is a lot higher than that of adding video. Besides, adding support for sequences of frames later is relatively trivial.

Before using the driver region images, contrast limited adaptive histogram equalization is applied to improve contrast. For the experiments with augmented images, the images were augmented using the augmentations found in Section 4.

5.2 Hand and phone detection

The first step of the pipeline approach is detecting the hands and phones of the driver using RetinaNet (32). This algorithm was briefly discussed in Section 2.3.1 and chosen because of its impressive performance on the COCO dataset (33). The backbone used with RetinaNet is ResNet50 (23). ResNet50 was chosen here based on research by Huang et al. (25). According to their experiments, a ResNet50 layout strikes a nice balance between accuracy and speed. To train the model, the hands and phones in the dataset of 2038 images were annotated. The images were resized to have the shortest side be 800 pixels while keeping the aspect ratio. This is done to match the RetinaNet paper. The hand detection works quite well, with AP scores of around 90, whereas the phone detection, with APs of around 50, does not. From a look at the classified images, a lot of phones are not detected, but among the detected phones are few false positives.

Analyzing the output revealed that clearly visible phones that were oriented with their back towards the camera were most often found. False negatives seemed to mostly occur when the phone was occluded by a hand. See Figure 7 for two examples.

Figure 7: On the left a situation in which the phone and hand detector works well. Everything is correctly detected. On the right only a hand is detected, but the partly occluded phone is not.

5.3 Face detection

Next, the drivers' faces are located using an existing (trained) MTCNN (57) solution, called FaceNet. FaceNet is a face recognition suite, of which only the face alignment code is integrated into the pipeline. MTCNN was chosen because of its use in other research, because of its good performance and because a trained model was readily available. The face detection algorithm was not customized aside from picking only the highest scoring detection per image. This way, it is certain that there is going to be one and only one face detection in the driver region. This is done to simplify the next steps. A drawback of doing it this way could be that the algorithm detects faces of people in the back seat. In practice, this happened only very rarely.

5.4 Follow-up steps

As shown in Figure 6, based on whether a phone is detected, an image goes through different follow-up steps. If a phone is detected, the class can be determined by looking at the positioning of the different objects. If the phone is near the head, it is classified as *Handheld calling*, if it is near a hand, it is classified as *Handheld phone usage*, if it is nowhere near either a hand or face, it is classified as being in a *Phone mount*. This was done because of the findings documented in Section 3.1. It was found that there are a number of phones in phone mounts in input classes different from the handheld phone usage classes. To keep these from becoming false positives, the phones far away from both hands and faces are not classified as one of the handheld phone usage classes. A detected phone is deemed to be too far away from a hand or a face if its center is 0.2 times the image size away from the center of the hand or the head. This value was empirically chosen.

5.5 Hand classification

Since the phone detector has a low recall, often phones are not detected by it. From looking at the false negatives, it was determined to mostly be the case when the phone was occluded by a hand. Therefore, by looking at just the hand and learning the way a hand looks when it holds a phone without seeing the phone clearly could result in finding more of these partly occluded phones.

Based on this realization, if no phone is found, the hands that were found in the hand and phone detection step will be classified by a trained CNN. The CNN is trained using images that were acquired by cropping the images according to the bounding boxes attained from the hand detector, with an added margin of 20 pixels in all directions. The resulting images were resized to be of size 100 $\times$ 100 pixels. From a few quick experiments, larger images lead to a worse performing algorithm. The on Imagenet trained ResNet50 CNN layout is only compatible with images of size 197$\times$197 and up. It was therefore replaced with VGG-16 (47).

The classes of hands are:

- Hand holding phone.
- Hand holding steering wheel.
- Hand holding something else.
- Hand not holding anything.

Any images with hands that are classified as *Hand holding a phone* will be classified as either *Handheld calling* if it is near a head, or *Handheld phone usage* otherwise.

Images with hands that were classified as any of the other classes are classified to be *Non-risky behavior*. The same goes for images where no hands or phones are found at all.

6 Experiments

This section discusses the setup of the different experiments.

6.1 General

All networks used in these experiments were instantiated with Imagenet trained weights and trained for 25 epochs on the training data. The learning rate used was 0.00001. This was done for the RetinaNet implementation, which was reported to be less stable at higher learning rates. The other models also use this learning rate for consistency. Unless otherwise noted, data augmentation was used, using the settings mentioned in Section 4. Regular CNN classifiers like the baseline approach and the pipeline hand classifier were trained 10 times per experiment. The hand and phone detector was trained 5 times per experiment. It would have been preferable to train 10 networks so that it is more in line with the classifiers, however training this network is very time consuming, therefore training was repeated fewer times. The best performing network, based on F-Score (for classifiers) or mAP (for the detector) on the validation set, is chosen and its value on the test set is reported. This is done to negate overfitting. All experiments were implemented in Keras and OpenCV and executed on an Nvidia GTX 1060 6Gb GPU.

6.2 Baseline approach

From the images, a CNN is trained to immediately classify them, like described in Section 2.2. It is expected that this straightforward model will still have room for improvement, as Groot and Jiang (22) have shown in their work. However, the resulting model would lead to a fitting baseline model with which others can be compared.

The size of the images used to train the baseline approach was decided by looking at the average width:height ratio. The average image size of an image in the dataset was calculated to be 370×296 pixels, which is a ratio of about 1.25:1. Therefore, the input size used was sized to be that ratio as well, at 250×200 pixels.

Class distribution

Multiple variations of the baseline model are implemented, to see which performs best:

- A multiple class version that classifies all classes.
- A binary version with the classes *Binary risky* and *Binary non-risky*.
- A second binary version with different classes that may be easier to distinguish between.

For the first binary CNN, the dataset was split into two classes: the *Binary risky* class, which consists of all *Handheld calling* and *Handheld phone usage* images, and the *Binary non-risky* class, which consists of the other images.

For the second binary CNN, the *Binary non-risky* class only contains images from the original *Non-risky behavior* class and the *Binary risky* class contains all the other images. This was done based on the experiments documented in Section 3. The hand counts and the distributions of hand positions illustrated that the *Non-risky behavior* class was different than the other classes in this regard. The class distribution of the second CNN was chosen to capitalize on this.

6.3 Pipeline individual components

Before looking at the pipeline as a whole, the individual components will be evaluated. The performances of the components are measured with different configurations based on smaller data sizes, data augmentation on or off, and different neural network structures.

6.3.1 Data augmentation

The detector and classifier of the pipeline approach were trained with and without data augmentation.

6.3.2 Neural network layout

Hand and phone detection

For the hand and phone detection, RetinaNet was used in combination with a ResNet50 neural network layout. A VGG-16 layout is also tested to see if it will perform better.

Hand classification

For the hand classification, a VGG-16 network was utilized. This was because the minimum size of images used in an Imagenet trained ResNet50 was 197×197 pixels, whereas the minimum size for VGG-16 was 48×48 pixels. When testing the VGG-16 network, image sizes of around 100×100 pixels empirically performed the best. However, since ResNet50 performed better in the detection step of the pipeline, it was also tested for the hand classification step. To do this, the images were scaled up to the 197×197 pixels necessary. All other factors were kept the same.

6.3.3 Data size

It is common knowledge that training using a larger dataset in general leads to a better performing neural network. Also, the performance gain for extra data is expected to

gradually decrease, the more data is added. For the final software resulting from this thesis, it would be useful to get a sense of how much the performance would improve when training with more data. To analyze this, the pipeline approach hand and phone detector and hand classifier were retrained using smaller portions of the dataset. The portions are subsets of each other with 75%, 50% and 25% of all images respectively. The dataset of 75% of images was created by random sampling without replacement from each input class of the dataset. That way, the ratios of images between the different input classes stays the same. The 50% dataset images are taken from the set of 75% to make it a strict subset. The same goes for the 25% set with respect to the 50% set. The hand and phone detector was trained only once per smaller dataset for this experiment, because of the time necessary and because the other experiments were given priority.

7 Results and discussion

7.1 Baseline approach

The results of application of this baseline model on the test set can be found in Table 4. The method appears to perform poorly. This was to be expected, as a method like this is usually trained on images which feature the object very prominently, which is not the case here. This, combined with the relatively small dataset, makes for an environment where learning the relevant pattern is difficult.

A note about the F-Score: The F-Score can be defined in multiple ways for multi-class problems. To keep the comparison of this metric fair with respect to the other experiments, the multi-class results were converted and used to calculate a binary F-Score. This was done by combining the two risky input classes into one risky class and the four non-risky input classes into one non-risky class. The F-Score is then calculated with the risky class as the primary class. Unless otherwise noted, future experiments will use the same technique to calculate the F-Score.

Run	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Accuracy (%)	44.66	42.23	40.29	39.32	39.81	42.72	43.2	42.23	38.83	41.75
F-Score	0.451	0.405	0.339	0.418	0.371	0.222	0.512	0.387	0.593	0.355

Table 4: Results of the baseline multi-class approach.

Run	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Accuracy (%)	59.71	56.8	35.44	74.76	48.06	71.84	78.16	55.83	39.81	74.76
F-Score	0.385	0.360	0.387	0.333	0.352	0.293	0.286	0.372	0.386	0.257

Table 6: Results of the first binary CNN.

At first glance, looking at Table 6, the binary CNN might appear to perform better than the multi-class CNN, boasting higher accuracies. However, looking at the F-Scores shows that the multi-class CNN actually performs better than the binary CNN at differentiating between the risky classes and the non-risky classes.

Another disadvantage of the binary CNN is that it is less specific than its multi-class counterpart. It is unable to classify the image into the classes that have been chosen. All in all, the multi-class CNN should be preferred over the binary CNN.

The second binary neural network performs better than the first one. The improved performance is expected to be the result of the larger differences between the two classes compared to the previous two classes. All in all, it still does not perform very well.

Ground truth \ Prediction	Handheld calling	Handheld phone usage	Holding something else	Hand close to head	Hand off the wheel	Non-risky behavior
Handheld calling	2	5	1	3	11	0
Handheld phone usage	0	17	1	0	3	3
Holding something else	0	5	8	0	22	5
Hand close to head	0	3	1	1	29	6
Hand off the wheel	0	1	2	1	30	6
Non-risky behavior	0	2	0	0	16	22

Table 5: Confusion matrix of the best performing baseline multi-class CNN model, based on F-Score (run 9).

Note that the change in class contents means that the images that are outputted as risky will often not be risky, but simply drivers with their hands off the steering wheel. This is acceptable because when looking at all the video footage, around 90% of all drivers show *Non-risky behavior*. So if those can be automatically detected and ignored, that would already lead to a huge reduction in the time necessary to check for actual phone usage by a human.

Another result of the changed class contents is that the F-Score is higher than it would be in the other experiments. This is because the (binary) F-Score depends on the true positives, false positives and false negatives, but not the true negatives. When the ratio between true positives and true negatives changes, the F-Score changes. For example, it could also be looked at from the reverse perspective, with the non-risky class as positive. So the *Non-risky behavior* that is detected as such would be a true positive. The resulting ratio between positive and negative images (40:166) would be more in line with the other experiments (46:160). The F-Score would then be 0.513, only very slightly better than the F-Score of the multiple classes CNN. For this reason, the F-Score of this CNN cannot really be compared to the F-Scores of the other experiments.

		Prediction	
		Binary risky	**Binary non-risky**
Ground truth	**Binary risky**	42	4
	Binary non-risky	129	31

Table 7: Confusion matrix of the best performing first binary CNN model (run 3), based on F-Score.

Run	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Accuracy (%)	73.30	81.55	73.79	76.21	80.58	66.50	73.79	80.10	65.53	62.14
F-Score	0.816	0.886	0.819	0.844	0.874	0.760	0.812	0.869	0.746	0.702

Table 8: Results of the second binary CNN.

		Prediction	
		Binary risky	**Binary non-risky**
Ground truth	**Binary risky**	148	18
	Binary non-risky	20	20

Table 9: Confusion matrix of the best performing second binary CNN model (run 2), based on F-Score. Note that the number of images per class have shifted due to the new class contents. This also heavily influences the F-Score.

7.2 Pipeline individual components

7.2.1 Data augmentation

The results of the networks that performed best can be found in Table 10.

	DA	No DA
Hand AP	0.910	0.908
Phone AP	0.505	0.585
mAP	0.707	0.746

	DA	No DA
Accuracy	75.00%	69.74%
F-Score	0.652	0.578

Table 10: The results of the hand and phone detector (left) and the hand classifier (right) with and without data augmentation (DA).

The results show that the data augmentation step has a different influence on the performance of detector than it has on the performance of the classifier. The detector becomes worse whereas the classifier improves. This could be explained by the difference in types of images. The hand classifier improves because rotating hands, flipping them, etc. creates a new image of a hand that could exist in real life. It is still representative of the population of hand images. The phone and hand detector works with images of right sides of windshields. Rotating, flipping and other transformation methods change these images into new images that would never occur in real life. They become unrepresentative of the entire population of right sides of windshields.

7.2.2 Neural network layout

Hand and phone detection

The results of the best performing networks can be found in Table 11.

	ResNet50	VGG-16
Hand AP	0.910	0.912
Phone AP	0.505	0.490
mAP	0.707	0.701

Table 11: The AP scores of the two detectors for a ResNet50 and a VGG-16 neural network backbone.

The VGG-16 layout performs very similar to the ResNet50 layout and thus no change to the pipeline seems necessary, the ResNet50 layout is kept.

Hand classification

The results of the best performing networks are shown in Table 12.

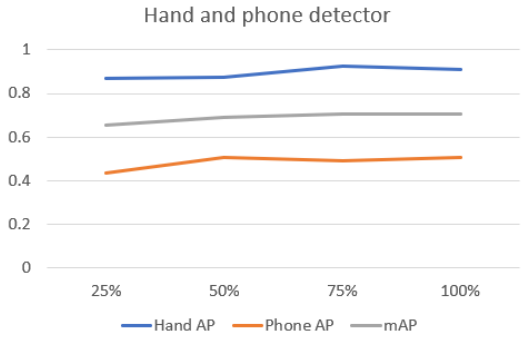
	ResNet50$_{197\times197}$	**VGG-16**$_{100\times100}$
Accuracy	54.39%	75.00%
F-Score	0.395	0.652

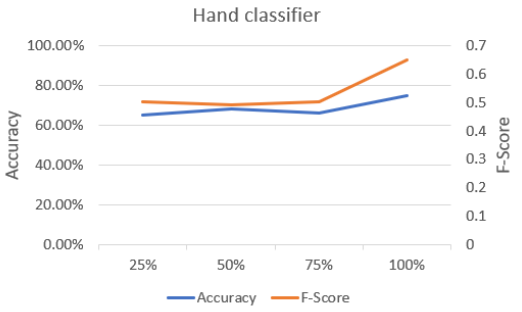
Table 12: The results of the two hand classifiers with different neural network structures.

These results show a clear superior performance of the VGG-16 classifier. This is probably due to an image size of 197×197 being too large and the network overfitting on small patterns in the image where the 100×100 images appear to be a more ideal size for the current patterns.

7.2.3 Data size

The results can be found in Figure 8.





	25%	**50%**	**75%**	**100%**
Hand AP	0.868	0.875	0.924	0.910
Phone AP	0.437	0.506	0.492	0.505
mAP	0.653	0.690	0.708	0.707

	25%	**50%**	**75%**	**100%**
Accuracy	65.35%	68.42%	66.23%	75.00%
F-Score	0.505	0.494	0.505	0.652

Figure 8: The results of the hand and phone detector (left) and the hand classifier (right) when using smaller datasets. The values are those of the best performing network.

The performance of the CNNs trained on less data degrades, but not very much.

Since these were all trained using data augmentation, this could be a reason that a

network trained on a smaller dataset could rival that of a network trained on a larger dataset. Yet, this was unexpected. Overall, this indicates that it might be wise to not put too many resources into collecting more data.

7.3 Complete pipeline

So far, the individual components have been evaluated. The next section will concern the complete pipeline.

See Figure 13 for the confusion matrix of the pipeline approach.

The precision, recall and F-Score of the pipeline approach are all 0.696. These results are very positive. It is a significant improvement over the baseline model.

One of the research questions can now be answered. The confusion matrix reveals that the *Non-risky behavior* input class can be classified the best, with 1 classification error for 39 correct classifications. This was somewhat expected, based on the analysis of the difference between input classes in Section 3.1.

Real life data

		Prediction	
		Risky	**Non-risky**
Ground truth	**Risky**	16	4
	Non-risky	243	4118

Table 14: Confusion matrix of the Pipeline approach on real data. The phone usage classes have been combined into the risky class. The *Phone mount* and *Non-risky behavior* classes have been combined into the non-risky class.

Table 14 shows how the pipeline performs on real life data. This was a dataset of 4381 windshields, collected after the original dataset. The pipeline performs well: 16 out of 20 (80%) drivers using their phone are detected. The set of images that has to be checked by a human (the true and false positives) consists of 259 images, $1/17^{th}$ of the original set. A significant reduction in manual time necessary to detect phone usage.

7.3.1 Quantitative analysis

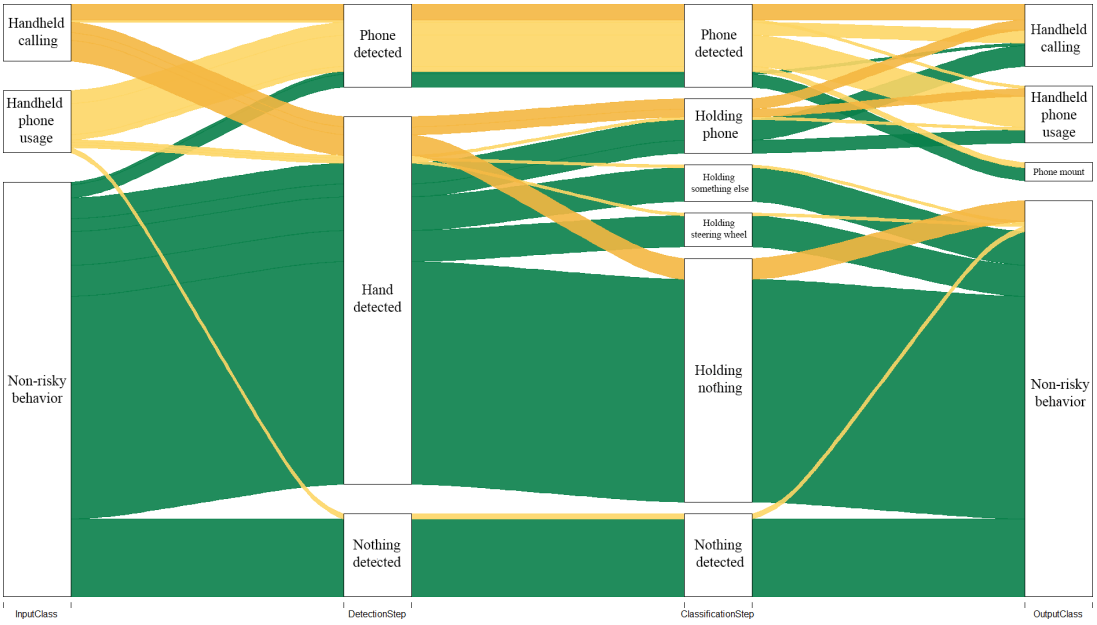


Figure 9: An alluvial diagram representing the paths the images took through the pipeline. Thickness of the strokes stands for the number of images. The non-risky input classes have been combined to better observe the behavior of the risky images. The two risky classes are orange and yellow and the non-risky classes are green.

The above alluvial diagram (Figure 9) illustrates the flow of images through the pipeline algorithms and highlights the risky input classes. The figure illustrates that phones in the *Handheld calling* input class are more difficult for the phone detector to detect than phones from the *Handheld phone usage* class. A significant part of the missed phones are subsequently identified in the classification step.

One could imagine that the hand classifier would classify hands holding phones as hands holding something other than phones quite often. This does not appear to be the case. It more often classifies them as not holding anything.

The other way around, where non-risky hands are classified as holding phones happens quite often. Over half of the hands that were classified as holding a phone were false positives. The question becomes if the hand classifier is really necessary, since it leads to this relatively large number of errors. Removing the entire classifier from the pipeline would reduce the number of false positives, but it would also reduce percentage of risky behavior detected in the *Handheld calling* input class significantly (from 63.6% to 31.8%). Therefore, instead of abolishing this classifier altogether, tuning the classifier's threshold might be a better solution to decrease the number of false positives.

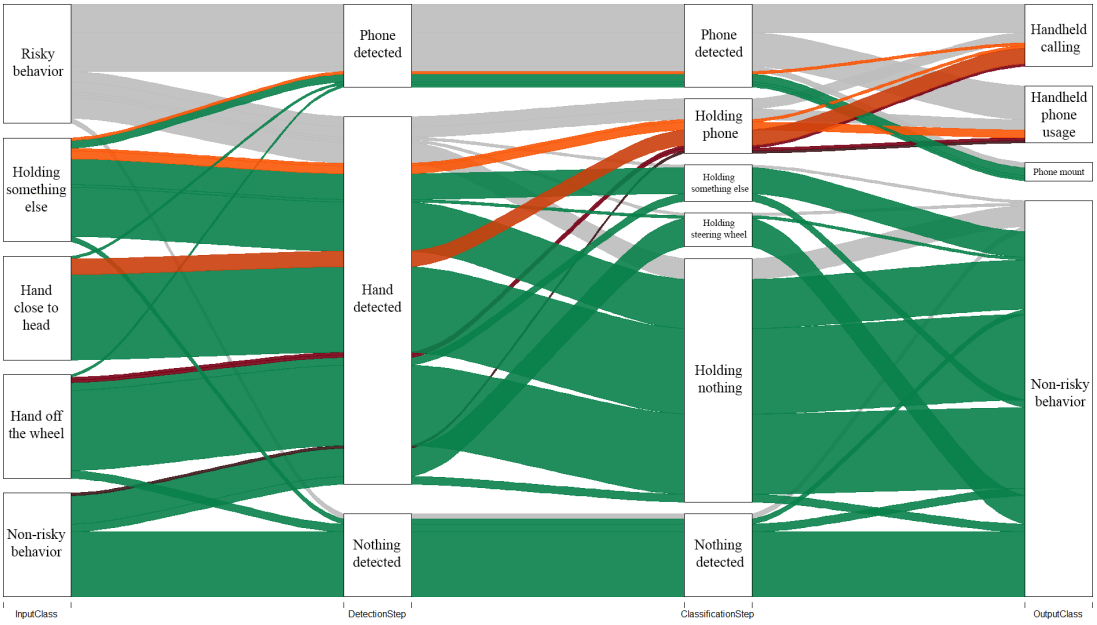


Figure 10: An alluvial diagram representing the paths the images took through the pipeline. The risky input classes have been combined to better observe the behavior of the non-risky images. True negatives are colored green, false positives are shades of red, true positives and false negatives are greyed out.

The above alluvial diagram (Figure 10) highlights the non-risky input classes. In particular, the false positives. The main takeaway from this diagram is that false positives are mostly introduced in the hand classification step and they originate from all non-risky input classes.

A positive aspect of the pipeline that can be seen here is that the fewest false positives come from the *Non-risky behavior* class. Since this is by far the largest class in a realistic dataset (around 90% of drivers), this is promising for performance 'in the wild'.

7.3.2 Qualitative analysis

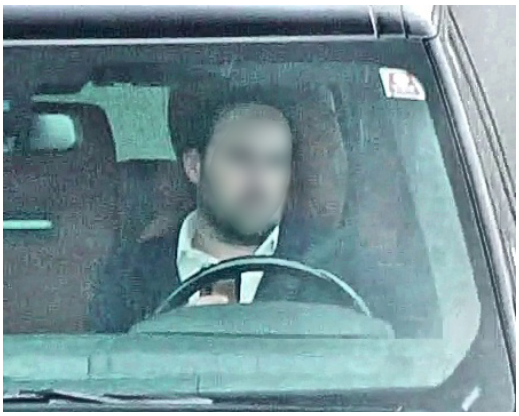
From the quantitative analysis, the hand classification step seemed somewhat effective. This step was introduced after a qualitative analysis of the images featuring phones that were not detected by the phone detector. These were mostly phones that were partly occluded by hands. This is also why the detector finds a lot more phones in the *Handheld phone usage* class than the *Handheld calling* class. Because that class has more phones that are directly facing the camera, clearly visible or only partly occluded by a hand.

Looking at the mistakes the algorithm makes, it becomes clear that the false negatives mostly consist of two varieties.

Firstly, there are drivers that are handheld calling and holding their phone in slightly abnormal ways, while partially occluding the phone. They should be picked up by the hand classifier, but that appears to be too difficult. For an example, see Figure 11a.

Secondly, there are phones in the *Handheld phone usage* class that are very close to the dashboard, where sometimes only a very small part of the hand holding the phone is visible and sometimes the hand is not visible at all. This makes it so the hand may not be detected and thus the hand classification step is not utilized. In such a situation, most often no phone is detected and the image is classified as *Non-risky behavior*. But even when the phone is detected, the phone is far away from the face and the image will be wrongly classified as featuring a phone in a phone mount.

(a) The first type of false negative. The phone is a small part of the image, due to its orientation. It is also partly occluded and the hand holding the phone is positioned and oriented slightly nonstandard.



(b) The second type of false negative. The phone is clearly visible, but the hand holding it is only very slightly visible. If no hand is detected in this image, the phone will be classified as being in a phone mount due to its distance from hands or faces.

Figure 11: The two types of false negatives that are most common.

The false positives come mostly from the hand classifier misclassifying. Although the patterns in the false positives are less obvious than the patterns in the false negatives, a grasping pose of the hand seems to be the most common false positive. See Figure 12 below.

(a) The hand in a grasping pose, together with the rectangular shape of the cup is probably why this was misclassified as a hand holding a phone.

(b) The hand of this driver is also in a pose that looks similar to a hand holding a phone. The hand being close to the side of the head might also add to this.

Figure 12: The types of false positives that seem the most common.

Ground truth \ Prediction	Handheld calling	Handheld phone usage	Phone mount	Non-risky behavior
Handheld calling	10	4	0	8
Handheld phone usage	5	13	2	4
Holding something else	2	3	3	32
Hand close to head	6	0	1	33
Hand off the wheel	1	1	1	37
Non-risky behavior	0	1	0	39

Table 13: Confusion matrix of the pipeline approach. Note that the output classes are split into *Handheld calling* and *Handheld phone usage* instead of combining them into *Risky behavior*, to aid the evaluation, because it allows for better tracking of where false positives end up.

8 Conclusion

A software solution was built to detect handheld cell phone usage among drivers using computer vision. The software can dramatically decrease the manual time necessary to detect phone usage by filtering out a large portion of non-risky behaving drivers.

The created pipeline approach achieved an F-Score of 0.70. It outperforms the best baseline CNN classification approach, which reaches an F-Score of 0.59. This could be because it is more focused on the important parts of the image instead of looking at the image as a whole.

The weakest part of the software appears to be the hand classifier, which introduces relatively many false positives compared to the detector part of the pipeline.

Data augmentation appeared to work well for the classifier that was used in the pipeline, but it led to worse performance for the detector. This might be due to the augmentations being too disruptive to the images for the windshields leading them to be unrepresentative of regular windshields seen in real life.

The input class that ended up being classified the best was the *Non-risky behavior* class. From analyzing the images it became clear that images in this class had far fewer hands on them than images in any of the other classes. By utilizing a hand detector as a central part of the pipeline, this fact was taken advantage of. This could be the reason why this class ended up being classified the best.

8.1 Limitations and future work

The software was not tested with video taken from different viewpoints. This is a limitation because a suspected large number of drivers is using their phone behind the dashboard, out of view of the current camera. One of the potential ways of detecting more phone usage would be to use a camera at a steeper angle. That way, phones cannot be hidden behind the dashboard anymore. A camera with a high frame rate would be necessary, but this could potentially increase the number of detected phones significantly.

The software is dependent on the police developed windshield detector. The windshield detector currently only works for Dutch cars, since it first recognizes the license plate and only when it is a valid Dutch license plate will it look for a windshield. This limitation is propagated to the pipeline software. In the future, this could be solved using different windshield detection software.

The hand classifier used in the pipeline approach appeared to introduce unwanted false positives. Removing the hand classifier altogether would reduce the number of detected phones by a large margin. The solution could be to tune the classifier's threshold. Increasing the amount of data could also work. These options could be explored in future research.

Some *Handheld phone usage* was classified as a *Phone mount* because the hand holding the phone was not detected. Looking at it the other way around might be interesting as well, by using a detection algorithm to detect the presence of a phone mount instead of the absence of a hand. This might reduce the number of false negatives.

The current pipeline approach uses images of windshields as input. Since the original data is in video format, part of the information is unused. Further research could be done to see how much impact using video has. For example, multiple frames of the same driver could be classified and aggregated into a single classification.

For the data augmentation experiments that were done in this work, the augmentations were either all on or all off. It would be interesting to see a more thorough analysis of the individual data augmentation types and their effects on performance.

Considering that there is quite a bit of work put into acquiring windshield images to detect phone usage of drivers, the other uses of these images could also be investigated. For example, the images could be used to detect if the driver is wearing a seat belt. Or they could be used to detect when drivers are distracted for other reasons than phone usage.

References

[1] Andriluka, M., Roth, S., and Schiele, B. Pictorial structures revisited: People detection and articulated pose estimation. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (2009), IEEE, pp. 1014–1021.

[2] Artan, Y., Bulan, O., Loce, R. P., and Paul, P. Driver cell phone usage detection from hov/hot nir images. In *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops* (June 2014), pp. 225–230.

[3] Berri, R., Silva, A., Parpinelli, R., Girardi, E., and Arthur, R. A pattern recognition system for detecting use of mobile phones while driving. In *VISAPP 2014 - Proceedings of the 9th International Conference on Computer Vision Theory and Applications* (08 2014), vol. 2.

[4] Bourdev, L., and Malik, J. Poselets: Body part detectors trained using 3d human pose annotations. In *International Conference on Computer Vision (ICCV)* (2009).

[5] Caird, J. K., Willness, C. R., Steel, P., and Scialfa, C. A meta-analysis of the effects of cell phones on driver performance. *Accident Analysis Prevention 40*, 4 (2008), 1282 – 1293.

[6] Cao, Z., Simon, T., Wei, S., and Sheikh, Y. Realtime multi-person 2d pose estimation using part affinity fields. *CoRR abs/1611.08050* (2016).

[7] CBS. Overledenen; doden door verkeersongeval in nederland, wijze van deelname. `http://statline.cbs.nl/Statweb/publication/?DM=SLNL&PA=71936NED`, 2006-2016. Accessed: 2018-02-06.

[8] CBS. Maatwerk - verkeersveiligheid. `https://www.cbs.nl/nl-nl/maatwerk/2017/20/1-op-de-8-bestuurders-chat-wel-eens-tijdens-het-rijden`, 2016. Accessed: 2018-03-06.

[9] Chen, X., and Yuille, A. L. Articulated pose estimation by a graphical model with image dependent pairwise relations. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 1736–1744.

[10] Chew, S. E., and Cahill, N. D. Semi-supervised normalized cuts for image segmentation. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)* (Washington, DC, USA, 2015), ICCV '15, IEEE Computer Society, pp. 1716–1723.

[11] Dai, J., Li, Y., He, K., and Sun, J. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 379–387.

[12] Das, N., Ohn-Bar, E., and Trivedi, M. M. On performance evaluation of driver hand detection algorithms: Challenges, dataset, and metrics. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. Sept 2015, pp. 2953–2958.

[13] Dingus, T. A., Guo, F., Lee, S., Antin, J. F., Perez, M., Buchanan-King, M., and Hankey, J. Driver crash risk factors and prevalence evaluation using naturalistic driving data. *Proceedings of the National Academy of Sciences 113*, 10 (2016), 2636–2641.

[14] Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision 111*, 1 (Jan 2015), 98–136.

[15] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[16] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results. http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html.

[17] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32*, 9 (Sept 2010), 1627–1645.

[18] Fu, C., Liu, W., Ranga, A., Tyagi, A., and Berg, A. C. DSSD : Deconvolutional single shot detector. *CoRR abs/1701.06659* (2017).

[19] Girshick, R. Fast r-cnn. In *2015 IEEE International Conference on Computer Vision (ICCV)* (Dec 2015), pp. 1440–1448.

[20] Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2014), pp. 580–587.

[21] Girshick, R. B., Iandola, F. N., Darrell, T., and Malik, J. Deformable part models are convolutional neural networks. *CoRR abs/1409.5403* (2014).

[22] Groot, R., and Jiang, S. C. Small Project Report - Detecting phone usage in cars using traffic cams, January 2018.

[23] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *CoRR abs/1512.03385* (2015).

[24] Hu, M.-K. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory 8*, 2 (February 1962), 179–187.

[25] Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *IEEE CVPR* (2017).

[26] Jain, V., and Learned-Miller, E. Fddb: A benchmark for face detection in unconstrained settings. Tech. Rep. UM-CS-2010-009, University of Massachusetts, Amherst, 2010.

[27] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093* (2014).

[28] Lamble, D., Kauranen, T., Laakso, M., and Summala, H. Cognitive load and detection thresholds in car following situations: safety implications for using mobile (cellular) telephones while driving. *Accident Analysis Prevention 31*, 6 (1999), 617 – 623.

[29] Le, T. H. N., Zheng, Y., Zhu, C., Luu, K., and Savvides, M. Multiple scale faster-rcnn approach to driver's cell-phone usage and hands on steering wheel detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (June 2016), pp. 46–53.

[30] Le, T. H. N., Zhu, C., Zheng, Y., Luu, K., and Savvides, M. Deepsafedrive: A grammar-aware driver parsing approach to driver behavioral situational awareness (db-saw). *Pattern Recognition 66* (2017), 229 – 238.

[31] LeCun, Y., and Bengio, Y. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks 3361*, 10 (1995), 1995.

[32] Lin, T., Goyal, P., Girshick, R. B., He, K., and Dollár, P. Focal loss for dense object detection. *CoRR abs/1708.02002* (2017).

[33] Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft COCO: common objects in context. *CoRR abs/1405.0312* (2014).

[34] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. Ssd: Single shot multibox detector. In *Computer Vision – ECCV 2016* (Cham, 2016), B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., Springer International Publishing, pp. 21–37.

[35] Mittal, A., Zisserman, A., and Torr, P. H. Hand detection using multiple proposals. In *BMVC* (2011), Citeseer, pp. 1–11.

[36] Ojala, T., Pietikainen, M., and Maenpaa, T. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 7 (Jul 2002), 971–987.

[37] Razavian, A. S., Azizpour, H., Sullivan, J., and Carlsson, S. Cnn features off-the-shelf: an astounding baseline for recognition. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on* (2014), IEEE, pp. 512–519.

[38] Redmon, J. Darknet: Open source neural networks in c. `http://pjreddie.com/darknet/`, 2013–2016.

[39] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 779–788.

[40] Redmon, J., and Farhadi, A. Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242* (2017).

[41] Redmon, J., and Farhadi, A. Yolov3: An incremental improvement. *arXiv* (2018).

[42] Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence 39*, 6 (June 2017), 1137–1149.

[43] Savalle, P.-A., Tsogkas, S., Papandreou, G., and Kokkinos, I. Deformable Part Models with CNN Features. In *European Conference on Computer Vision, Parts and Attributes Workshop* (Zurich, Switzerland, Sept. 2014).

[44] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR abs/1312.6229* (2013).

[45] Seshadri, K., Juefei-Xu, F., Pal, D. K., Savvides, M., and Thor, C. P. Driver cell phone usage detection on strategic highway research program (shrp2) face view videos. In *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (June 2015), pp. 35–43.

[46] Siddharth, Rangesh, A., Ohn-Bar, E., and Trivedi, M. M. Driver hand localization and grasp analysis: A vision-based real-time approach. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)* (Nov 2016), pp. 2545–2550.

[47] Simonyan, K., and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *CoRR abs/1409.1556* (2014).

[48] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints abs/1605.02688* (May 2016).

[49] Tzutalin. Labelimg. git code (2015. https://github.com/tzutalin/labelImg.

[50] Törnros, J., and Bolling, A. Mobile phone use – effects of conversation on mental workload and driving speed in rural and urban environments. *Transportation Research Part F: Traffic Psychology and Behaviour 9*, 4 (2006), 298 – 306.

[51] Vedaldi, A., and Fulkerson, B. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008.

[52] Viola, P., and Jones, M. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001* (2001), vol. 1, pp. I–511–I–518 vol.1.

[53] Viola, P., and Jones, M. J. Robust real-time face detection. *International journal of computer vision 57*, 2 (2004), 137–154.

[54] Xiong, X., and De la Torre, F. Supervised descent method and its applications to face alignment. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition* (Washington, DC, USA, 2013), CVPR '13, IEEE Computer Society, pp. 532–539.

[55] Yang, S., Luo, P., Loy, C. C., and Tang, X. Wider face: A face detection benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

[56] Yang, Y., and Ramanan, D. Articulated human detection with flexible mixtures of parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence 35*, 12 (Dec 2013), 2878–2890.

[57] Zhang, K., Zhang, Z., Li, Z., and Qiao, Y. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters 23*, 10 (Oct 2016), 1499–1503.

[58] Zhu, X., and Ramanan, D. Face detection, pose estimation, and landmark localization in the wild. In *2012 IEEE Conference on Computer Vision and Pattern Recognition* (June 2012), pp. 2879–2886.

A Software overview and selection

Some of the papers mentioned in this document use techniques that have not been implemented into any software library. These researchers have often published their own code, which will be used for this project if the license permits it. For most other techniques, it will be beneficial to use one or multiple software libraries, as it will greatly reduce the amount of time spent on coding general algorithms, leading to more time that can be spend on problem specific work. Luckily, most of the discussed techniques have been compiled into software libraries.

OpenCV With an estimated 14 million downloads, OpenCV is a computer vision library with a very large community. OpenCV is written in C and C++, with interfaces for Python and Java. It is the most extensive library in this list. It includes modules ranging from image processing, video I/O and a DPM module. As of recently, it also features its own deep neural network module, that can create neural networks from scratch, or import them from Caffe, Darknet and Tensorflow.

VLFeat (51) A computer vision library with a focus on visual features and clustering. Notably has built-in support for VLAD and FV (as used in the discussed paper by Artan et al. (2)). VLFeat can be used with MATLAB, Octave, C++ and from the command line. Development on VLFeat has been inconsistent. The most recent update is not from long ago, but the update before that was from three years ago.

BoofCV BoofCV is a computer vision library written with ease of use and high performance in mind. It is developed in Java, which is why it is expected to be slower than C/C++ libraries, at least on lower level computations.

Since the rise of deep neural networks, there have been many libraries developed specifically for the creation of neural networks.

Darknet (38) A framework for neural networks, written in C and CUDA. Interfacing is done using the command line. Also includes You Only Look Once (YOLO (39; 40; 41)), a real-time object detection system. The creator claims that Darknet is fast.

Tensorflow Mostly used for machine learning, but also capable of other numerical computations using data flow graphs, Tensorflow is a software library developed by Google. It can be interfaced with using Python, C++, Java and Go. Tensorflow is used a lot in academia. As of October 2017, it has the most citations on arXive out of all the neural network libraries.

Caffe (27) A framework specifically for deep learning, claims to be expressive, extensible and fast. Caffe is written in C++ and has command line, Python, and MATLAB interfaces.

Theano (48) Like Tensorflow, Theano is a software library that is described as a numerical computation library, but mainly used for neural networks. Written in Python.

Main developer MILA has stopped actively developing Theano as of the 15th of November 2017 because of the strong competition in this field.

Microsoft Cognitive Toolkit (CNTK) The Microsoft Cognitive Toolkit is also similar to Tensorflow and Theano. The toolkit can be developed with using Python, C++, C# and BrainScript.

Keras Keras offers a layer on top of other libraries like Tensorflow, Theano and CNTK. Its main feature is the ease of use for developers. Programming with Keras is done using Python.

All of the above libraries are open-source and they all have GPU implementations. To make a decision on what libraries will be used for the thesis project there are multiple factors to take into consideration. In order of importance: feature set, ease of use, previous experience, robustness and support. Based on these criteria Keras was chosen for neural networks and OpenCV for general image processing.