FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

Eetu Haapamäki

FACIAL LANDMARK DETECTION ON MOBILE DEVICES

Master's Thesis
Degree Programme in Computer Science and Engineering
December 2020

Haapamäki E. (2020) Facial Landmark Detection on Mobile Devices. University of Oulu, Degree Programme in Computer Science and Engineering, 44 p.

ABSTRACT

Human facial expressions are very important part of the way people communicate. Even though people are very capable of recognizing different faces and expressions, understanding facial expressions has been one of the common challenges of computer vision. One of the methods used to help in the recognition of the face is facial landmark detection where the human face is depicted as a set of keypoints.

The common problem with the facial landmark detection is the high amount of different possible variance that the human faces have. Just like with most of the other machine vision tasks, different types of deep learning based methods have started to become the standard solution for facial landmark detection. These methods have proven themselves to be very capable in handling the variations.

The high performance of modern mobile devices has offered possibilities to run different types of deep neural networks (DNNs) using the device's own hardware. Even though the computational power of the modern mobile devices is very high, it is still considered to be limiting factor when implementing DNNs.

In this thesis, three different deep learning based facial landmark detectors based were implemented. These networks were tested to see if it is possible to run DNN based facial landmark detector on mobile device in real-time with good enough prediction accuracy for practical use.

The results showed that the more lightweight DNNs are very capable of running even on mobile central processing units (CPUs). The digital signal processors (DSPs) of the mobile devices proved to be very efficient even when computing the larger DNNs. In some cases, the DSP nearly outperformed the desktop graphics processing unit (GPU) which further displays the efficiency of using DSP for computing DNNs.

Keywords: Convolutional neural network, Deep learning, deep neural network, embedded system

Haapamäki E. (2020) Kasvojen kiintopisteiden tunnistus mobiililaitteilla. Oulun yliopisto, Tietotekniikan tutkinto-ohjelma, 44 s.

TIIVISTELMÄ

Kasvot ovat tärkeä osa ihmisten välistä kommunikaatiota. Vaikka ihmiset pystyvät helposti tunnistamaan eri kasvot ja ihmiset, näiden tunnistaminen on ollut yksi konenäön yleisimmistä haasteista. Yksi kasvojen tunnistukseen käytettävistä menetelmistä on kasvojen kiintopisteiden tunnistus, jossa ihmisen kasvot esitetään kiintopisteiden sarjana.

Yksi kasvojen kiintopisteiden tunnistamisen yleisimmistä ongelmista on kasvoissa sekä ilmeissä esiintyvät suuret erot. Aivan kuten monissa muissakin konenäön sovelluksissa, syväoppimiseen perustuvista menetelmistä on tullut vakioratkaisu kasvojen kiintopisteiden tunnistamisessa. Nämä menetelmät ovat erittäin kykeneviä käsittelemään kasvoissa esiintyvät suuret erot.

Modernien mobiililaitteiden korkea suorituskyky on avannut uuden mahdollisuuden erilaisten syvien neuroverkkojen ajamisesta laitteiden omalla laitteistolla. Vaikka modernien mobiililaitteiden laitteisto on erittäin suorituskykyistä, tämä on silti rajoittavat tekijä syvien neuroverkkojen toteuttamiselle näille laitteille.

Tässä diplomityössä on toteutettu kolme erilaista syväoppimiseen perustuvaa kasvojen kiintopisteiden tunnistinta. Näiden verkkojen testaamisella on selvitetty, onko mobiililaitteella mahdollista ajaa syvään neuroverkkoon perustuvaa kasvojen kiintopisteiden tunnistinta reaaliajassa hyvällä tarkkuudella.

Tulokset osoittivat että kevyemmät syvät neuroverkot toimivat erittäin hyvin jopa mobiililaitteen prosessorilla. Mobiililaitteen digitaalinen signaaliprosessori osoittautui erittäin tehokkaaksi suurempien neuroverkkojen laskennassa. Jossain tapauksissa digitaalinen signaaliprosessori suoriutui melkein paremmin kuin pöytäkoneen grafiikkaprosessori.

Avainsanat: Konvoluutioneuroverkko, syväoppiminen, syvä neuroverkko, sulautettu järjestelmä

TABLE OF CONTENTS

FOREWORD

This thesis was written while working for Visidon Oy. The company provided great enthusiastic atmosphere for studying and learning new things. I am very grateful for this opportunity and all the help from my coworkers. Special thanks to my thesis supervisor Olli Silvén, technological advisor Juho-Petteri Lesonen and 2nd examiner Docent Jari Hannuksela, the R&D Director of Visidon for all the guidance and feedback. I would also like to thank my friends, family and my lovely fiancée for all the support during my whole studies and the process of writing this thesis.

Oulu, December 8th, 2020

Eetu Haapamäki

LIST OF ABBREVIATIONS AND SYMBOLS

ASIC	Application-specific Integrated Circuit
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CUDA	Compute unified device architecture
DAN	Deep Alignment Network
DSP	Digital Signal Processor
DNN	Deep Neural Network
Float32	Single precision 32-bit floating point representation
FPS	Frames per second
GPU	Graphics Processing Unit
HRNet	High-Resolution Network
NME	Normalized mean error
PFLD	Practical Facial Landmark Detector
RAM	Random-access-memory
SoC	System-on-chip
STD	Standard deviation
Uint8	unsigned 8-bit integer

1. INTRODUCTION

Facial landmark detection is the process of depicting the shape of the face with facial landmark points. Figure 1 depicts an example of the locations of 68 facial landmarks and shows how the points depict the characteristics of human face without the need for the full image. The information depicted by the facial landmarks can be used for plenty of different tasks related to face analysis like facial recognition, face alignment, pose-estimation or different classification problems [1].

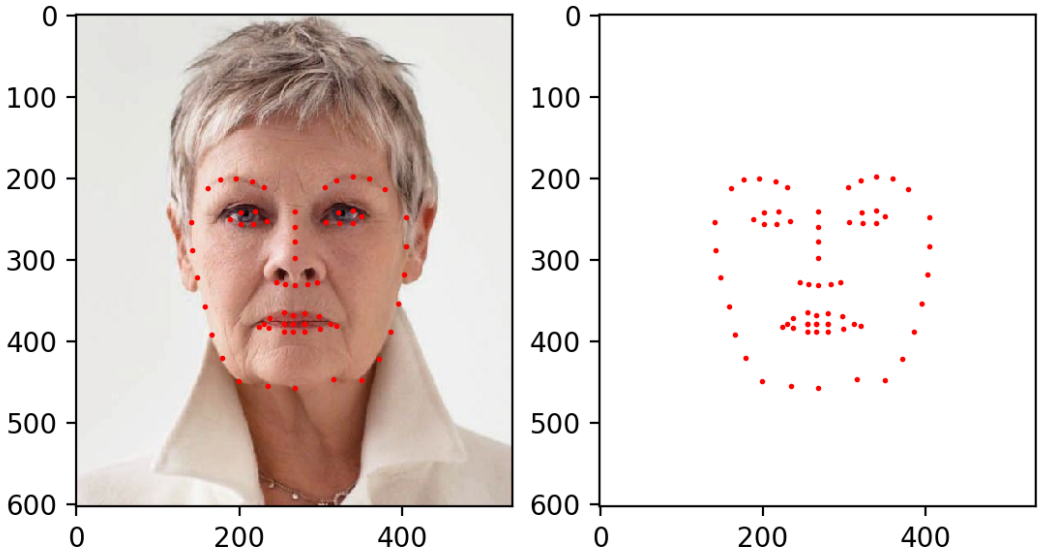

Figure 1. Example image with 68 point facial landmark annotated.

The challenges of facial landmark detection are mostly related to the high amount of possible variation in the images. The variation can be caused by the face in the image, such as the expression or pose or some kind of external effect like illumination or occlusion of part of the face. Figure 2 contains examples of the illumination, extreme pose and occlusion of the face. As seen in the images, these variations can alter or hide the point where the landmark should be positioned. The leftmost image has extreme illumination which hides parts of the face. The middle image has the face almost at 90 degree yaw angle which covers the other half of the face not facing the camera. The third image has the woman's hair covering over her whole right eye and other parts of the face around it.

The older methods used for facial landmark detection focused on easier situations with very little variation. Compared to the modern state of the art methods, the focus has shifted to more difficult situations with "in the wild" datasets that contain faces in much more unconstrained conditions than before [3]. [5, 6, 7]

The modern mobile devices offer plenty of different possibilities related to the different face analysis techniques. Pretty much all of the mobile phones have cameras in them that can produce high quality images and videos. Additionally the processors of these devices are constantly improving thus allowing the use of more computationally intensive methods.

Figure 2. Examples of different possible extreme variations from 300W dataset [2, 3, 4].

As of late, deep learning based methods have surpassed the more traditional methods in performance, especially on images with high variation [5]. The problem with these methods is that the most accurate networks are also computationally very heavy which makes them unsuitable for use in mobile devices [8, 9] . Trying to find the optimal balance between the accuracy and performance of deep networks has been studied extensively and the limited computational capabilities of mobile devices offer even more challenges to this subject [9].

Moving the calculations to cloud can be one way to get much more computational power without the need to use the devices own. However this has multiple downsides likes the constant need of connection, privacy issues and latency issues [10]. The cloud computing is out of the scope for this thesis as the focus is on running the models on-device so it will not be discussed in more depth.

The focus of this thesis is about testing whether the modern deep learning based facial landmark detection can function on mobile devices in real-time with acceptable accuracy. To test this, three different deep models are implemented and their performance and accuracy are benchmarked to see how well they are suitable for real-time use on mobile devices.

This thesis will first go through the facial landmark detection methods focusing on the DNN based ones. After that the mobile computing platforms are explained before focusing on the implementation of the facial landmark detectors on these platforms. Then the implemented networks accuracy and performance are evaluated on desktop PC and two different mobile devices. The results from this chapter are discussed in chapter five and finally, the thesis is concluded in the summary chapter.

2. FACIAL LANDMARK DETECTION METHODS

The human face is usually divided into coarse (global) and detailed (local) features that differ from person to person [11]. Wu and Ji [5] divide the facial landmark detection methods into three different categories: holistic methods, constrained local methods and regression-based methods. The holistic methods use holistic appearance and global shape of the face for the facial landmark detection. The most popular holistic method used is the active appearance model. Compared to the holistic methods, the constrained local methods also use the global shape of the face but instead of the holistic appearance, they use local appearance information around the facial landmarks. [5]

Finally, there are the regression based methods that do not generally build any kind of face shape model but instead simply learn the position from the images. The three different regression based method styles are direct regression, cascading regression and deep learning based regression. The difference between direct regression and cascading regression is that the cascading regression uses initial guess that it improves through multiple stages while the direct regression is more straightforward by simply just predicting the landmarks with one stage. Example on how cascading regression has initial guess and improves it in every step can be seen in Figure 3. [5]

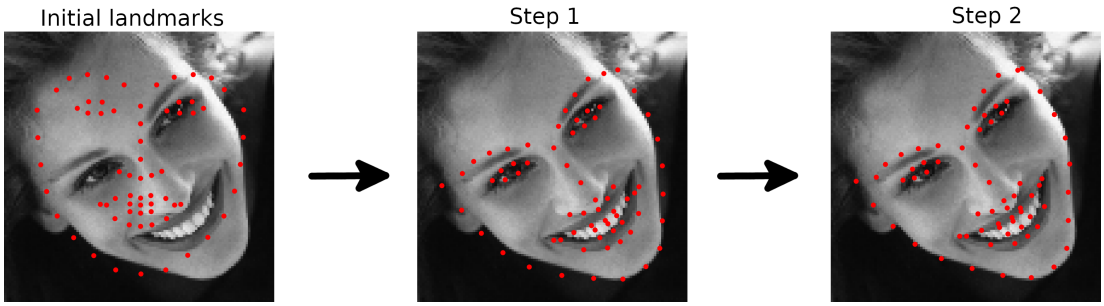

Figure 3. Cascading regression.

Most of the modern state of the state-of-the-art facial landmark detection methods are deep learning based methods which are considered regression based. The focus of this thesis are the deep learning methods so going more into depth into the holistic and constrained local methods will not be necessary.

2.1. Deep Learning Based Methods

As with most of the computer vision tasks, deep learning has become very popular technique to use for facial landmark detection. Deep learning can be split into two parts: training and inference [12]. The training process of DNNs is computationally much more expensive process which means that performing the training on mobile device is not reasonable [12]. This means that the networks are usually trained either on desktop computers or servers that are more capable of handling the complex calculations required in the training process. The inference is the process of getting output from the DNN and while it is much less computationally heavy process than the

training, the hardware of mobile devices is still limiting the use of the more complex networks [8].

Just like with almost all of the deep learning tasks, the facial landmark detection mostly uses convolutional neural network (CNN) models. The regression framework used for these methods can either be the more simple direct regression or cascaded regression. Wu & Ji divide the facial landmark detection methods using deep learning into pure-learning methods and hybrid deep methods. The difference between these methods is that the hybrid deep methods use different kinds of 3D vision strategies in conjunction with the deep learning models to estimate the position of the landmarks whereas the pure-learning models are simple CNNs that are used to directly predict the landmarks from the given images. [5]

Other major difference between different deep learning methods is the output of the model. Generally the output is either the direct coordinates or heatmaps for the landmarks [13]. Both of these output styles have their upsides and downsides.

According Teixeira et al. the heatmap regression is easier to train and the output is much more versatile as the heatmap generated could be used for other vision tasks whereas the coordinate regression outputs only the numerical coordinates [13]. The heatmaps used in facial landmark detection are most commonly Gaussian distribution centered probability values for each landmark [14]. The heatmap regression requires more functionalities to be implemented compared to the coordinate regression. As the ground truth values of the data are pretty much always given as numerical coordinates, the heatmap regression requires that the ground truth landmarks are encoded into heatmaps before using the data to train the model. Also the heatmap outputted by the model has to be decoded to get the numerical coordinates. Common problem with transforming coordinates to heatmaps is so called quantization error which happens because the common heatmap methods can only represent integer coordinates which usually leads to inaccuracies because the coordinates could be fractional numbers [15].

Overall, the heatmap regression based models are more accurate compared to coordinate regression based ones, but they lacking in robustness as the predictions are independent of each other [16]. Compared to heatmap regression, coordinate regression is usually much more lighter in terms of computing and memory usage but it comes with reduced local precision. Coordinate regression based methods are also usually more robust with dealing with global features whereas heatmap regression based models can completely fail in these situations and completely miss the correct landmark positions. [17]

2.1.1. Convolutional Neural Networks

The network model used for the deep learning strategies in facial landmark detection is pretty much always CNN [5]. Basic CNN model structure is depicted in Figure 4 and it is based on Task-Constrained Deep Convolutional Network by Zhang et al. [18]. The network takes and image as input and consists of four convolutional layers, three pooling layers and one fully connected layer. 100x1 feature vector outputted by the fully connected layer can be used for different tasks using logistic regression. In the case of the example image the output is ten values for five facial landmarks. Training the network is a process in which the parameters of the model are adjusted in order to

minimize the error in between the prediction outputted by the model and the ground truth of the data. The error is calculated according to predetermined loss function and it will be explained more thoroughly in Section 2.1.3. [19]

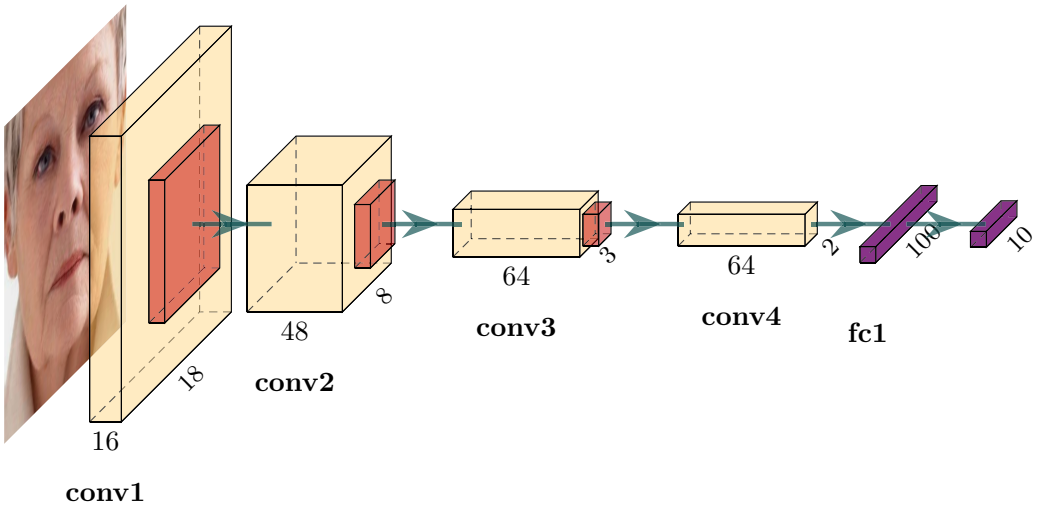

Figure 4. Convolutional neural network.

2.1.2. Datasets

To train and test different facial landmark methods, there needs to be data with images containing faces and annotations of the landmarks of the faces. The landmark annotation is mostly done manually which can cause problems. The process is very time consuming and human factors like fatigue can cause mistakes to the annotations [2]. According to Dong et al., the human made annotations vary according to the person creating the annotations and also the small inaccuracies in training data annotations can cause jittering in the detection process [20].

The number of landmarks annotated has increased in the datasets. Some of the older datasets contain only 4 to 5 landmarks which is usually enough for simpler tasks like face alignment [21]. Most of the current datasets provide 68-106 landmarks. The increased number of landmarks offer possibilities for more complex face analysis techniques. Most of the annotations are 2 dimensional, but nowadays there also exists annotations with three dimensions. It was decided that this thesis will focus on the 2D facial landmark detection so they will be left out of the discussion.

Table 1 shows some of the most important current in-the-wild datasets with facial landmark annotations. All of these datasets consists of images collected from multiple online sources like Google image search or Flickr. Making datasets this way is very common and easy practise and the high variety of different quality images make them very generalizable, but it also means that the data might not be comparable to the data that will be used in real-life situations. For example, extremely high quality images taken with high-end systems camera is not comparable to an image taken with middle

range smart phone camera. The 300-W dataset has been the most commonly used dataset for facial landmarks [21].

One of the limitations of most of the 2D facial landmark datasets is the ability to properly depict the faces that are in 90° angle. This is because when the face is fully in profile, certain landmark are practically impossible to annotate properly because they are not visible in the image. Menpo 2D dataset solves this problem by having different annotations for images from profile where the regular images have 68 landmarks annotated but the faces from profile are annotated with only the 39 landmarks visible from that angle.

The landmarks can be roughly split into two different categories: facial keypoints and interpolated landmarks. The facial key points are the main landmarks like the corners of the eyes or mouth whereas the interpolated landmarks connect the key points and/or describe the shape of the region of the face. Figure 5 shows an example annotation of a face with 68 landmarks. For example the corners of the right eye (points 37 and 40) are considered facial keypoints whereas the points that connect the two (points 38, 39, 41 and 42) are considered interpolated landmarks. [5]

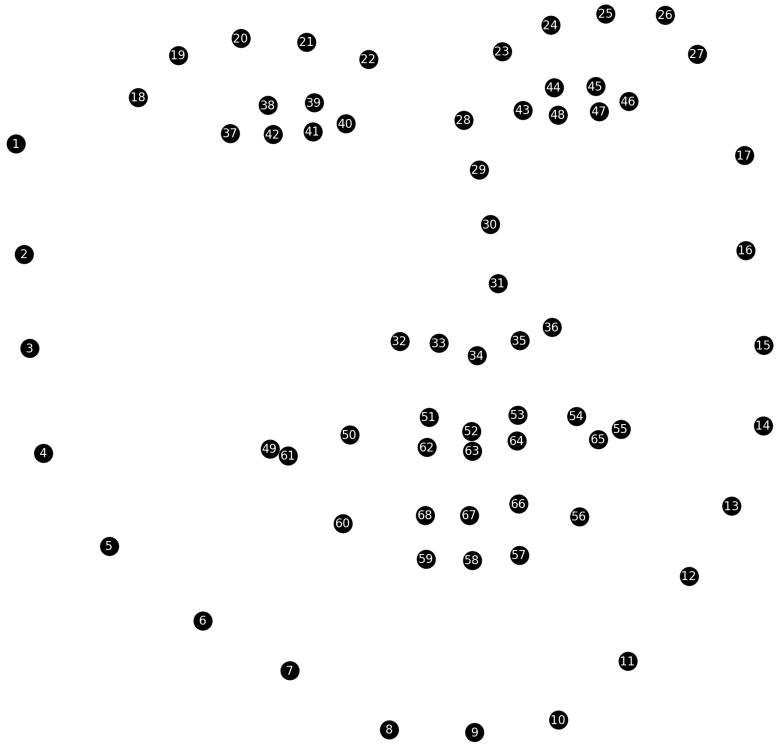

Figure 5. Example of a 68 point landmark annotation.

As the focus of the facial landmark detection research has shifted away from the controlled environment, the more modern databases contain much more different variations and such as more difficult for the detectors.

Table 1. Different datasets with facial landmark annotations

Dataset name	Year	Number of images	Landmark annotation
AFLW	2011	25993	21 facial landmarks
LFPW	2011	1432	29 facial landmarks
Helen	2012	2330	194 facial landmarks
COFW	2013	1852	29 facial landmarks
Ibug 300-W	2013	4000	68 facial landmarks
UTKFace	2017	20000	68 facial landmarks
Menpo 2D Benchmark	2018	14845	68/39 facial landmarks
WFLW	2018	10000	98 facial landmarks
JD-landmark	2019	15393	106 facial landmarks
LaPa	2019	22000	106 facial landmarks

2.1.3. Loss Functions

One of the most important parts of the deep neural networks is the loss function. The loss function is picked according to the given task as the different kinds of problems like classification problems and regression problems have completely different output values from where the error needs to be calculated [19]. Currently, the most popular loss function used for facial landmark detection is the L2 loss

$$\text{L2 loss} = \sum_{i=1}^{n} (y_i - \hat{y_i})^2 \tag{1}$$

with where n is the number of landmarks, y the ground truth value of the landmark and ŷ the predicted value of the landmark. Other widely used loss functions are L1 loss

$$\text{L1 loss} = \sum_{i=1}^{n} |(y_i - \hat{y_i})| \tag{2}$$

and Smooth L1 loss

$$\text{Smooth L1 loss} = \sum_{i=1}^{n} \begin{cases} 0{,}5(y_i - \hat{y_i})^2 & \text{if } |y_i - \hat{y_i}| < 1 \\ |y_i - \hat{y_i}| - 0{,}5 & \text{otherwise} \end{cases} \tag{3}$$

that are also used but less frequently than the L2 loss. [22]

When calculating the loss from the coordinates of the landmarks it is also common practise to normalize the calculated error. This is done to try to reduce the effect the size of the face has on the loss. Common methods of normalization are dividing the error with some value related to the shape of the face like the distance between the corners of the eyes or distance between pupils. [5]

In addition to these basic loss functions, there are also several different loss functions specifically designed for the facial landmark detection problem. Some of them simply expand the basic loss functions like Guo et al. expanded L2 loss to take into consideration the pose of the face and the possible variations like occlusion or extreme

lighting. The face pose estimation was done by having an auxilary network in the training process to predict the yaw, pitch and roll angles of the face and comparing these to the ground truth of these angles. The different variations were factored in to the loss by labeling the whole training dataset and weighting the loss in the images according to how rare they are, which should make it so that the network is trained more towards the situations with some kind of uncommon variation. [6]

2.1.4. DNN Optimization

Running DNNs is very resource intensive task, so trying to optimize them in an attempt to improve the performance has been very common area of different studies. The most obvious method trying to design the network in a way that it uses less resources like Google's MobileNets [23, 8]. One of the problems with this is that it restricts the optimization to be done before the network is trained.

There also exists post-training optimization techniques which are usually based on simplifying the models in a way that the performance improves in the expense of accuracy [8]. The most common of these methods are called quantization and pruning. Quantization is a process where the models weights are converted from the single precision 32-bit floating point representation (Float32) to unsigned 8-bit integer (Uint8) to improve the performance and reduce the size of the model [24]. This comes with a cost of accuracy as the Uint8 numbers can not represent the values with the same precision as Float32. Instead of simplifying the weights, pruning simply removes the weights that have little effect on the accuracy to reduce the calculation times and model sizes [8]. The networks used in this thesis are relatively small so pruning is not very good strategy as the smaller networks do not have as many nodes so removing them causes accuracy to drop really fast.

3. MOBILE PLATFORMS AND IMPLEMENTATIONS

Because the hardware is such a limiting factor when implementing DNNs on mobile devices it is important to consider the different possible methods on how to conduct the computing. Even though the focus of this thesis about running the models on mobile devices, it is also relevant to discuss the computing platforms with more general desktop computers in mind.

3.1. Mobile Computing Platforms

Computation-wise neural network calculations are mostly different types of matrix calculations. Figure 6 shows an example of a single artificial neuron, where the calculation done is

$$o_j = \sum_{i=1}^{n} x_i \cdot w_{ij} + \theta_j \tag{4}$$

with the x being the input, w the weight and θ is the scalar bias. The dot product calculation done here can also be depicted as dot product of matrices where the other contains the input values and the other corresponding weights. [25]

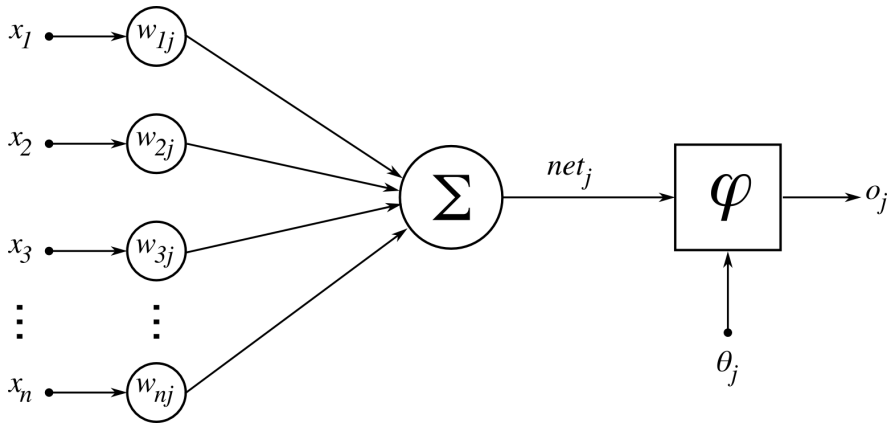

Figure 6. Example of a simplified neuron[1].

The Figure 7 depicts a simple neural network with two input layers, five hidden layers and one output layer. Each of the blue dots of the hidden layer can be thought of as neurons described earlier so the hidden layer can be depicted as a matrix multiplication of the inputs and weights [26].

As the deep CNNs can consist of tens of millions of neurons, the amount of calculations required are very high [27]. These matrix operations are computationally relatively slow operations and the high amount of these operations makes calculating these networks very demanding task [28].

[1]https://commons.wikimedia.org/wiki/Neural_network#/media/File:ArtificialNeuronModel.png, licensed under CC BY-SA 3.0, accessed 25.11.2020

[2]https://commons.wikimedia.org/wiki/Category:Artificial_neural_networks#/media/File:Neural_network.svg, licensed under CC BY 1.0, accessed 25.11.2020

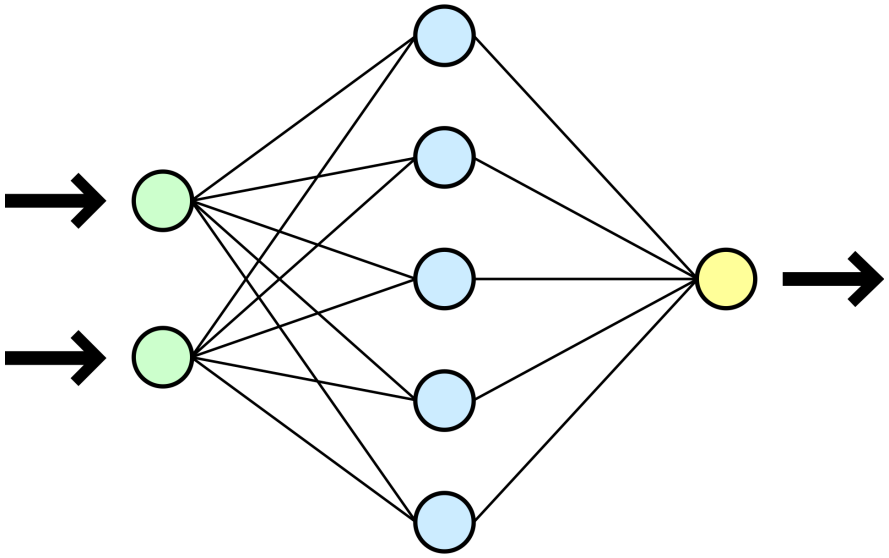

Figure 7. Example of a simplified neural network[2].

Even though the hardware of mobile devices has been substantially increasing in recent years, the computational power is still fairly limited to use for some of the more complex deep networks. Modern mobile phones consist of system-on-chip (SoC) that holds inside pretty much everything a regular desktop computer would have like processor, random-access-memory (RAM) and GPU [24]. In addition to these technologies, modern mobile SoCs can contain DSPs, application-specific integrated circuits (ASICs) and field-programmable gate arrays (FPGAs) [29].

3.1.1. CPU

The most common way to conduct computing on mobile devices is the CPU, because it is featured in virtually every mobile device. Even though modern mobile CPUs are really fast, they are still behind the common ones used in desktop computers and other non-mobile devices. CPUs have a very high clock rate and relatively low amount of threads [30]. This means that they are not the most optimal solution for computing neural networks, because they are more suitable for sequential tasks instead of parallel tasks. High utilization of mobile devices CPU will also lead into high energy consumption and possible issues with the thermal properties of the device [31].

3.1.2. GPU

Recently, GPUs have been the most suitable computing platform to use for different kinds of deep learning models. As the name suggests GPUs were originally designed for computing graphics, but tools such as NVIDIA's Compute Unified Device Architecture (CUDA) make is possible to create different kinds of GPU-accelerated applications [32, 33]. Utilizing GPUs for deep learning tasks was properly popularized in 2012 by AlexNet network [27]. Compared to CPUs, GPUs may have thousands of

smaller cores which makes them much more suitable for the specific types of matrix calculations required by the different deep learning strategies [31].

Compared to the cores of CPU, these cores have much lower clock speed making GPUs less suitable for sequential tasks [30]. The higher amount of cores and threads makes the GPU excel at parallel computing making them optimal for the high amount of computationally intensive tasks required by the neural networks. This performance comes with a downside of high power consumption which makes GPUs more suitable for the training process where the power usage is not an issue compared to the limited power of mobile devices [34]. Other big advantage of the GPUs over CPUs is the much higher memory bandwidth [35].

3.1.3. DSP

DSP is a type of specialized processor designed mainly for different signal processing tasks in mind. The usage of DSPs on mobile devices was originally mostly tasks related to voice and radio signals but when the devices started to gain more multimedia features such as video playback and image processing . The biggest advantage of a DSP over of CPU or GPU is the low power consumption of the DSP, which allows for running the DSP at higher frequency making it faster in real use [29]. One of the common DSPs used on modern mobile devices is the Qualcomm's Hexagon series [36]. [24]

3.1.4. ASIC

ASICs are processors designed to compute very specific task of application. Performance and energy efficiency wise, ASICs are generally the best solution because the hardware is designed purely for the task at hand. Because the ASICs are developed with specific task in mind, it limits their use compared to more general platforms like FPGA and GPU. There are modern ASICs designed purely with AI tasks in mind like tensor processing unit (TPU) by Google that has proven to be much faster and energy efficient than either CPUs or GPUs [37]. [29, 31]

3.1.5. FPGA

FPGAs are semiconductor devices that consist of programmable logic blocks. Overall the FPGAs are superior to regular CPUs and GPUs in performance and energy efficiency. The downside however is the fact that implementation process of FPGAs is much more complex and the architecture can limit the designs of the technologies implemented. Compared to ASICs, the FPGAs are not as efficient, but the FPGAs designs are much more flexible which makes them noteworthy solution to consider. [29, 31, 38]

3.2. Implementation

For the evaluation of different types of deep learning networks three networks were implemented: Practical Facial Landmark Detector (PFLD) by Guo et al., Deep Alignment Network (DAN) by Kowalski et al. and High-Resolution Network (HRNet) by Sun et al. [6, 39, 40]. The networks were picked for testing according to the results presented in the original papers and the different properties of the networks. The results were tested with the official code if it had been made available. The basic differences in the properties of the networks can be seen in Table 2. Overall the PFLD networks are so much more smaller than the other networks. For example PFLD0.25 has 3.3% of the parameters of the HRNet with 3.1% model size.

The networks were implemented in Google's TensorFlow machine learning library using Python programming language. For testing on mobile devices, the models were converted to TensorFlow Lite which is a framework designed for different types of mobile devices.

Table 2. Statistics of different networks tested

Name	Input size	Model size	Number of parameters
PFLD0.25	112x112x3	1.2 MB	322864
PFLD1.0	112x112x3	4.9 MB	1236544
DAN1	112x112x1	44.6 MB	11150664
DAN2	112x112x1	92.5 MB	23108444
HRNet	256x256x3	39.0 MB	9737688

3.2.1. PFLD

Out of all the models tested PFLD uses the simplest regression method: direct coordinate regression. PFLD was picked for the comparison because the original paper showed some really promising results by running the model on mobile device in real-time with state-of-the-art accuracy. Table 2 also shows that the model size of both of the PFLD models is much more smaller than the other networks. PFLD is constructed using MobilenetV2 bottleneck blocks and such it allows for the adjustable depth for the bottleneck blocks which in turn reduced the size of the network with a small loss in the accuracy of the predictions. Two PFLD models were implemented: PFLD1.0 that uses MobileNetV2 block width multiplier of 1.0 and PFLD0.25 that uses MobileNetV2 block width multiplier of 0.25. [6]

3.2.2. DAN

DAN in turn uses cascading coordinate regression, but it also uses heatmaps in connecting the cascading layers between each other. Like PFLD, the DAN is also scaleable, but unlike PFLD it is only scaleable upward by adding more stages. The original study shows that the accuracy barely increases past the second stage so only

two stages were implemented and trained for this test. Like with PFLD, two different models were implemented: DAN1 using only one stage and DAN2 with two stages. [39]

3.2.3. HRNet

Finally the HRNet uses direct heatmap regression. HRNet was picked because of this and also the fact that the original study results indicated that it would be more accurate than DAN or PFLD. Another interesting thing worth pointing out is that in comparisson to PFLD and DAN, the HRNet does not contain any fully connected layers so the network is fully convolutional network. Even though testing the original implementations implicated that the model would be computationally very heavy, it was taken into the comparison because the published accuracy results were so good. [40]

3.2.4. Dataset

The dataset picked for training was 300-W as the 68-landmarks it provides was considered sufficient and also all of the original papers used this dataset so the networks can be easily tested that they function close enough to the original results [2, 3, 4]. The 300-W datasets consists of re-annotated images of datasets LFPW, AFW, HELEN and XM2VTS [41, 42, 43, 44]. The 300-W dataset was split into training and testing data following the common used practices where the training set has 3148 images and consists of all of the images from AFW dataset and training datasets of HELEN and LFPW. The testing set has 689 images and consists of the full dataset of IBUG and testing datasets of HELEN and LFPW.

For the data augmentation, the original studies were followed. For all of the models, the training data was augmented by flipping all the samples, and rotating them from -30° to 30° with 5° between the angles. In addition to this, the PFLD also used small amounts of random Gaussian blur and also regularization technique called cutout where small part of the image (in this case 20%) is randomly masked with zeros to simulate random occlusions [45].

3.2.5. Devices

The performance of the models was tested on desktop computer and two different mobile phones: Huawei P30 Pro and LG Velvet 5G. The more specific features of the different devices is presented in Table 3. At the time of writing this thesis, the desktop PC that was used can be considered slightly more powerful than the average computers. The Huawei P30 Pro was released in March 2019 making it over a year older than the LG Velvet 5G released in September 2020. P30 Pro SoC is the HiSilicon Kirin980 whereas the Velvet 5G contains Dimensity 1000C produced by MediaTek.

The desktop processor used was AMD Ryzen 7 3700X containing 8 cores and 16 threads. Both of the mobile devices used have only 8 threads, which at least in theory

Table 3. Technical specifications of the devices used for testing

Device	CPU	GPU	RAM	DSP
Desktop PC	AMD Ryzen 7 3700X, 3.6GHz, 8-core	NVIDIA GeForce RTX 2060 Super 8GB	32GB	not applicable
Huawei P30 Pro	2.6Ghz Dual-Core ARM Cortex-A76 & 1.92Ghz Dual-Core ARM Cortex-A76 & 1.8Ghz Quad-Core ARM Cortex-A55	ARM Mali-G76 MP10 720Mhz	8GB	not applicable
LG Velvet 5G (G900TMY)	2Ghz Quad-Core ARM Cortex-A77 & 2Ghz Quad-Core ARM Cortex-A55	ARM Mali-G57 MC5 650MHz	6GB	4 Heterogeneous Core MediaTek APU 3.0

would mean that the desktop processor has an edge in parallel computing. In addition to this, the desktop CPU runs at much higher clock rate which means that it should be able to complete the instructions faster than the mobile CPUs. The GPU of the desktop has exactly 7979MB of dedicated memory and 2176 CUDA cores. The GPUs cores run at 1.695 GHz which is relatively low especially compared to the CPUs, but the high amount of cores and threads means that the GPU will perform well with parallel computing.

The LG Velvet 5G was released while this thesis was being worked on, so there did not exist proper benchmarks that compare the two mobile phones yet. Overall the CPUs on both of the devices are very similar with 8 cores in total. The architecture differs slightly as the Huawei P30 Pro has two dual-core processors and single quad-core whilst LG Velvet 5G has two quad-core processors. One of the processors is same for both of the phones (ARM Cortex-A55), but the one in the LG Velvet 5G has higher clock rate. The quad-core used by the LG Velvet 5G (ARM Cortex-A77) is slightly newer model than the two dual-core processors on the Huawei P30 Pro (ARM Cortex-A76). Even though the LG Velvet 5G is slightly newer than the Huawei P30 Pro, the ARM Mali-G76 of Huawei is slightly more powerful than the GPU used by LG.

4. EVALUATION

Evaluating facial landmark detectors are usually only about the accuracy of the predictions. When running the detectors on mobile devices, the limited computing power makes the evaluating process more of a balance between the prediction accuracy and the computational performance of the model. The heaviest models will most likely outperform the lighter ones in accuracy, but will be unusable in practical use because they are computationally too complex for mobile devices.

4.1. Accuracy Evaluation

The accuracy of the models is evaluated by predicting the landmarks of testing images and comparing the output to ground truth landmarks. The statistic picked for this is the normalized mean error (NME)

$$\text{NME} = \frac{\sum_{i=1}^{n} |(y_i - \hat{y}_i)|}{d} \tag{5}$$

where y_i is the predicted landmark value, $\hat{y}_i$ is the ground truth value of the landmark and d is the normalization distance. The distance picked for the normalization was the distance between the corners of the eyes, also known as the inter-ocular distance. Inter-ocular distance was picked because it is the most common distance used to normalize the error. Other normalization methods used before are inter-pupil distance which is the distance between the centers of the eyes and bounding box distance.

Figures 8 and 9 showcase how the normalized error typically looks like on general example faces. These images were taken from the 300-W dataset that was used for the accuracy evaluation. It is interesting to notice that even though there is noticable difference in the errors in Figure 8, the differences are surprisingly hard to notice in the actual images. The PFLD0.25 predictions show that some of the contours like the jawline and the eyebrows are slightly off compared to the other predictions. Even though the error seems to be pretty high, the PFLD0.25 predicted landmarks still represent the overall shape of the face fairly well.

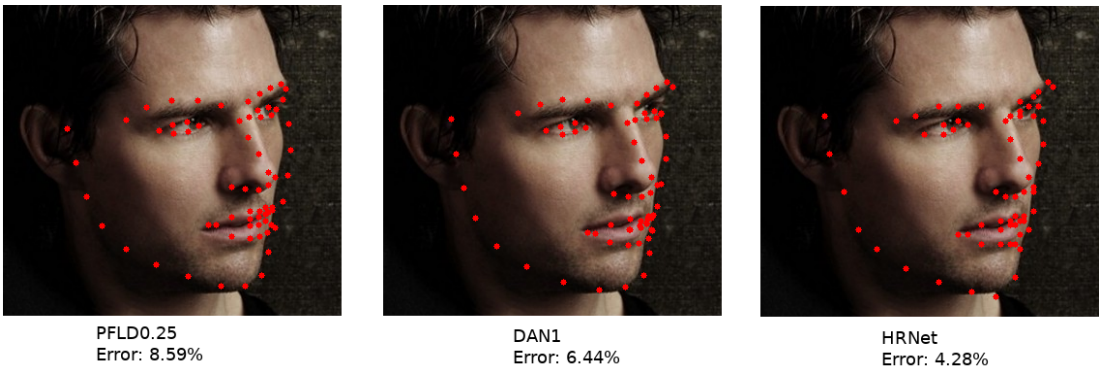

Figure 8. Example of normalized error on facial landmarks.

The errors of the images in Figure 9 are statistically higher and this time the errors are also much more visible in the actual images. The predictions by the PFLD0.25

model are very clearly not correct, as the whole jawline contour is misaligned and the other eye and eyebrow are completely misplaced. This makes it very difficult to correctly predict the position of the face from the landmarks as the predicted landmarks differ so much from the real ones. The error of the DAN1 model predictions is also pretty high at 8.30% but just like with the comparable error of 8.59% of the PFLD0.25 in Figure 8, the error is surprisingly hard to notice even when comparing the predictions to the best ones predicted by the HRNet.

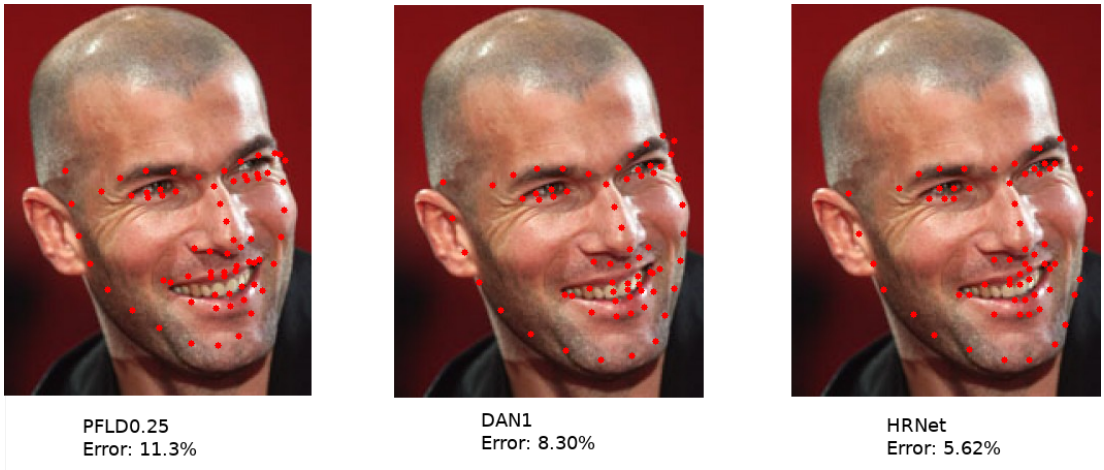

Figure 9. Example of normalized error on facial landmarks.

4.1.1. Initial Testing

Following the original studies, the 300W testing data was split into common and challenging datasets. The common dataset consists of 554 from LFPW and Helen testsets and the challenging datasets consists of 135 images from Ibug dataset. The models were first tested with the bounding boxes provided by the original dataset and the results are in Table 4.

Table 4. Error of the models on 300W testing data (interocular normalization, ground-truth bounding boxes)

Model	Common	Challenging	Full
PFLD0.25	4.0268	7.0156	4.6124
PFLD1.0	4.0149	6.8337	4.5672
DAN1	3.6889	6.3864	4.2174
DAN2	3.5802	6.1933	4.0922
HRNet	3.5413	6.3220	4.0861

Comparing the acquired results from Table 4 to the ones from the original studies presented in Table 5 we can see that the differences between the trained models and original models are biggest in both of the PFLD networks. This can be explained by the fact that the PFLD loss was not implemented completely because it would have required labeling the whole training dataset. It was decided that this would take too

much work and thus the results are not fully comparable. Other than that the results are satisfactory. It is also worth noting that the Table 5 is missing the results of DAN1 because the original study did not test that version of the model with 300W dataset.

Table 5. Error of the models on 300W testing data from the original studies (interocular normalization, ground-truth bounding boxes)

Model	Common	Challenging	Full
PFLD0.25 [6]	3.0300	5.1500	3.4500
PFLD1.0 [6]	3.0100	5.0800	3.4000
DAN2 [39]	3.1900	5.2400	3.5900
HRNet [40]	2.8700	5.1500	3.3200

After testing that the models work close enough to the accuracy of the original tests, the original bounding boxes were replaced with ones predicted with MTCNN [46]. This was done to make sure that the models work properly with the bounding boxes predicted by MTCNN as those will be used for testing the model on videos. At the same time, it will provide statistics on how well the models work on imperfect bounding boxes by by comparing the results to the ones using ground truth bounding boxes. The results can be seen in Table 6. The models still work pretty well with the imperfect bounding boxes. The prediction accuracy drop is pretty similar with all of the models. HRNet is still the most accurate of the models and the drop in accuracy is slightly smaller than the other models.

Table 6. Error of the models on 300W testing data (interocular normalization, MTCNN bounding boxes)

Model	Common	Challenging	Full
PFLD0.25	4.9899	8.1256	5.6043
PFLD1.0	4.7407	7.5147	5.2842
DAN1	4.7739	7.9278	5.3918
DAN2	4.7751	7.7262	5.3533
HRNet	4.0183	7.1191	4.6259

4.1.2. Testing Different Categories

Sometimes different facial landmark detectors function varies on different types of faces or situations, so it was important to test the models in a way that would reveal the possible strengths and weaknesses of each model. For testing the error of the models on different types of variations that occur in the images, it was decided that the 689 images of the 300W testing set is too small so more images were needed for the testing set. The added data was also part of the 300W challenge and it consists of 300 in-the-wild indoor images and 300 in-the-wild outdoor images. This data was used because it provided good in-the-wild faces with the same amount of landmarks as the other testing data and also the data was annotated using the same method so the results are more easily comparable. The resulting database was then categorized manually and the results can be seen in Table 7. The resulting dataset has pretty good balance between

males and females. It also contains good number of the different difficult variations like extreme poses.

Table 7. The categories and the amount of images belonging to that category from the dataset

Full	1289
Male	696
Female	593
Young	147
Old	110
Occlusion	220
Extreme expression	177
Extreme pose	195
Extreme lighting	164

Even with the added data it was decided that the resulting dataset did not contain enough faces with dark skin, so different testing method was thought of to compare how the different skin tone effects the accuracy of the models. The testing would be done on a completely different dataset with enough dark faces to compare to. The dataset that was picked was the Menpo2D [47, 48, 49]. Once again the dataset was picked because of it's good variation and the landmarks are also comparable to the data used in training. Out of the dataset 200 images of dark faces and 200 images of white faces were randomly picked for the testing set.

The prediction accuracy was tested on the full dataset and also the different categories. Once again the accuracy was measured in NME using interocular normalization and the results can be seen in Table 8.

Table 8. Error for the different dataset and it's categories

Model	Full	Female	Male	Young	Old	Oclusion	Extreme expression	Extreme pose	Extreme lighting
PFLD0.25	6.237	5.753	6.649	5.629	6.686	7.461	7.863	8.549	6.780
PFLD1.0	5.821	5.456	6.132	5.423	6.007	6.933	7.404	7.750	6.272
DAN1	5.984	5.593	6.316	5.538	6.042	7.242	7.577	8.268	6.433
DAN2	5.963	5.626	6.250	5.506	5.958	7.306	7.550	8.009	6.331
HRNet	5.227	4.960	5.455	4.677	5.089	6.706	6.727	7.383	5.910

Figures 10, 11 and 12 visualize the errors of the different categories normalized to the full dataset by subtracting the error of the full dataset from the error of the categories. This is done to easily visualize the possible differences in how the models handle different categories. The category "Synthetic occlusion" was done by picking 200 random images from the dataset and adding circles, rectangles and ellipses with random position, size and colour to occlude parts of the face in the images.

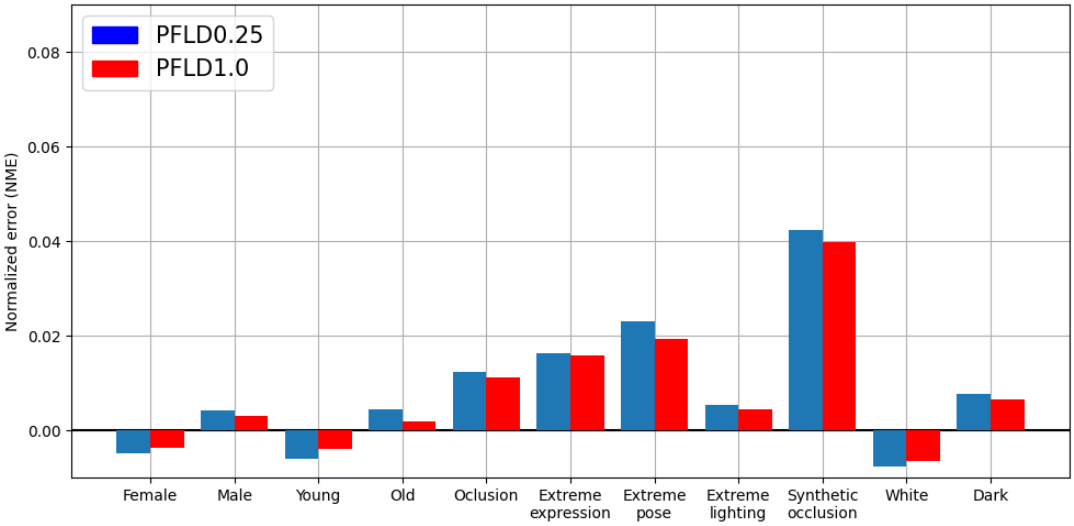

Figure 10. Normalized error for different categories with both PFLD models.

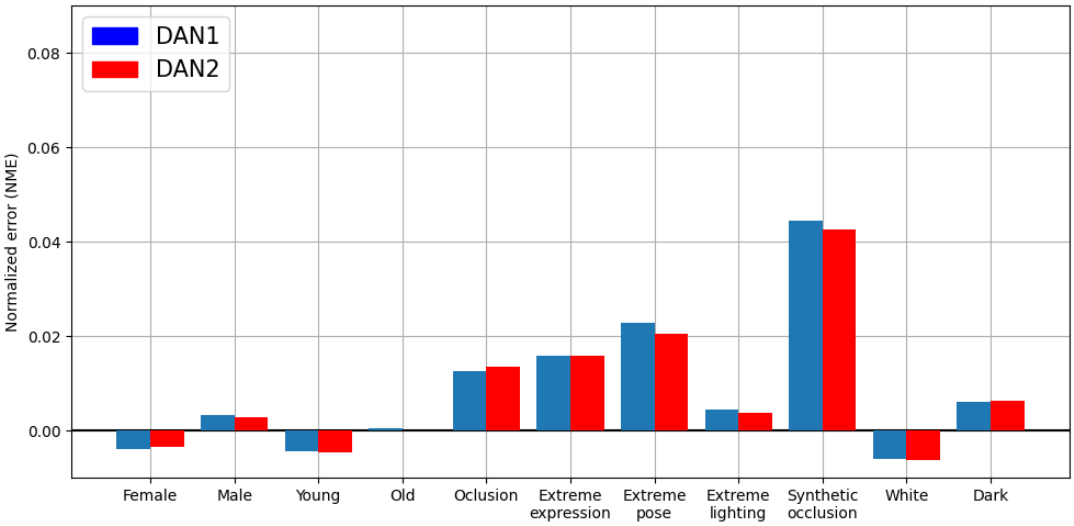

Figure 11. Normalized error for different categories with both DAN models.

Overall the results between the different models are very similar, with the synthetic occlusion, extreme pose and extreme expression being the three most difficult categories for the models. The only model showing real difference is HRNet that seems to have much bigger difficulties with synthetic occlusion with over 8 % normalized error compared to the approximately 4 % of the other models.

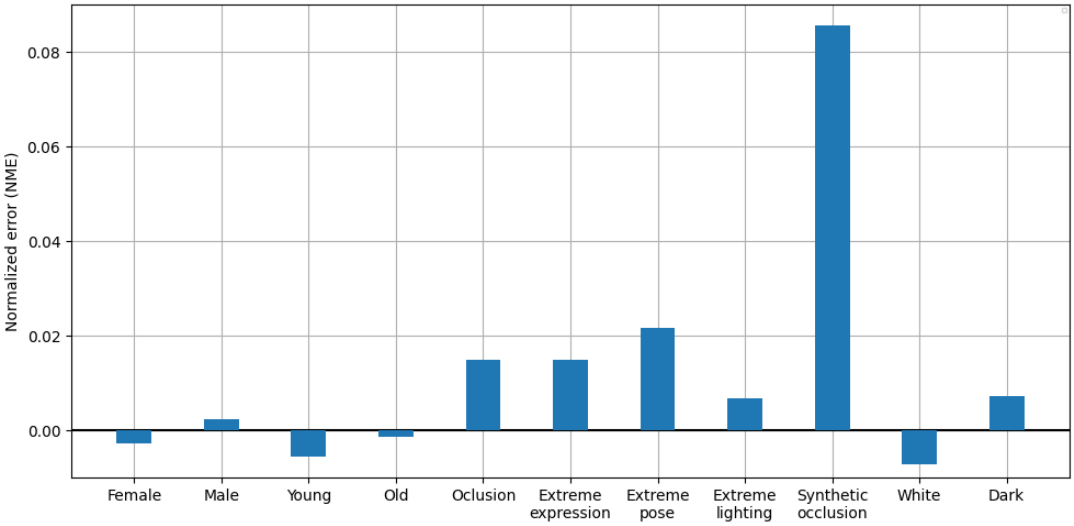

Figure 12. Normalized error for different categories with HRNet model.

Figure 13 shows the difference on how the different network handle heavy synthetic occlusion. The heavy occlusion causes the HRNet prediction to pretty much completely fail compared to PFLD and DAN which actually manage to output the landmarks in the shape of a face. This can be explained by the fact that HRNet uses heatmaps, which do not take into consideration the relationship between the landmarks, which makes it so that the prediction does not resemble the shape of a face like with DAN and PFLD.

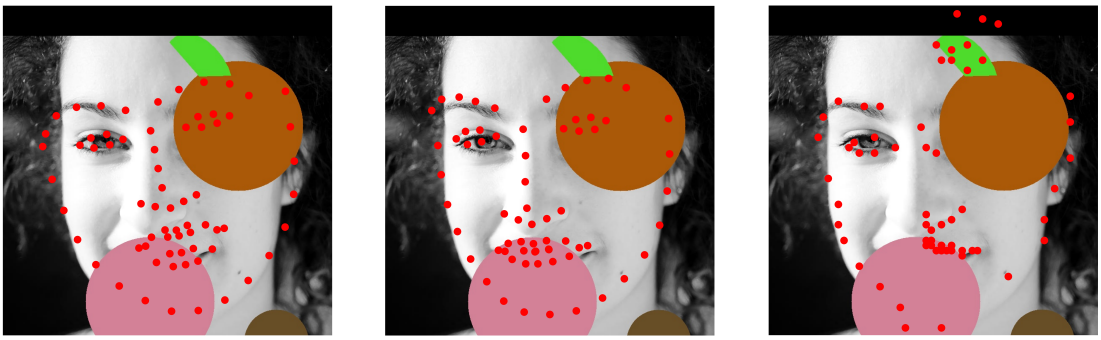

Figure 13. Effects of the synthetic occlusion on predictions on three of the models

4.1.3. Resolution Testing

Even though the cameras on modern mobile devices have improved greatly in recent years, the image quality still leaves room for improvement. To simulate the effects of lower quality images the testing data was simply scaled down for lower resolution and up for upscaling blur. To test how the resolution of the image effects the prediction accuracy of the models, the full dataset from earlier was downscaled by 2, 3 and 4. Figure 14 visualizes the effect of the scaling done on an image. As the image size gets

smaller more and more details are lost because as the amount of pixels gets smaller, the less information can be stored in one image.

Figure 14. Effect of the scaling done on image

Figure 15 visualizes the prediction errors of the models on the original full dataset and the dataset with different scaling factors. As expected, the error of the predictions increases as the resolution gets smaller, because the low resolution images contain less information than the full resolution ones. HRNet seems to be the worst one to deal with low resolution images as the error is the smallest with the full resolution images, but when the images are scaled to half, the error is same or worse than the other networks. The results on the 3x and 4x scaling factors further emphasise this as the HRNet is clearly the worst one of the models based on the error on these categories.

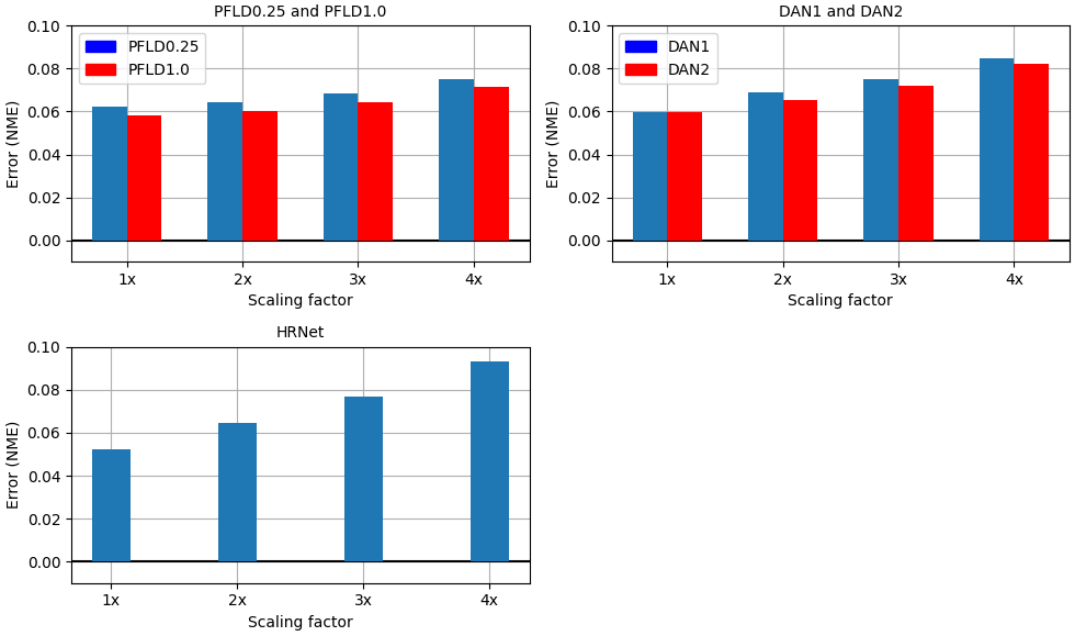

Figure 15. Error compared to downscaling of the image

The smaller changes in the resolution did not show that drastic decrease in the prediction accuracy, so the same test was done with much more extreme scaling (8, 16, 32 and 64) to see if the models behave differently. The results are shown in 16. Once again HRNet is the worst one, but overall the increase in error seems to follow the increase in the scaling factor almost linearly. Lowering the resolution does not seem to have any kind of sudden drop in performance as the accuracy seems to decrease steadily until it is completely unacceptable.

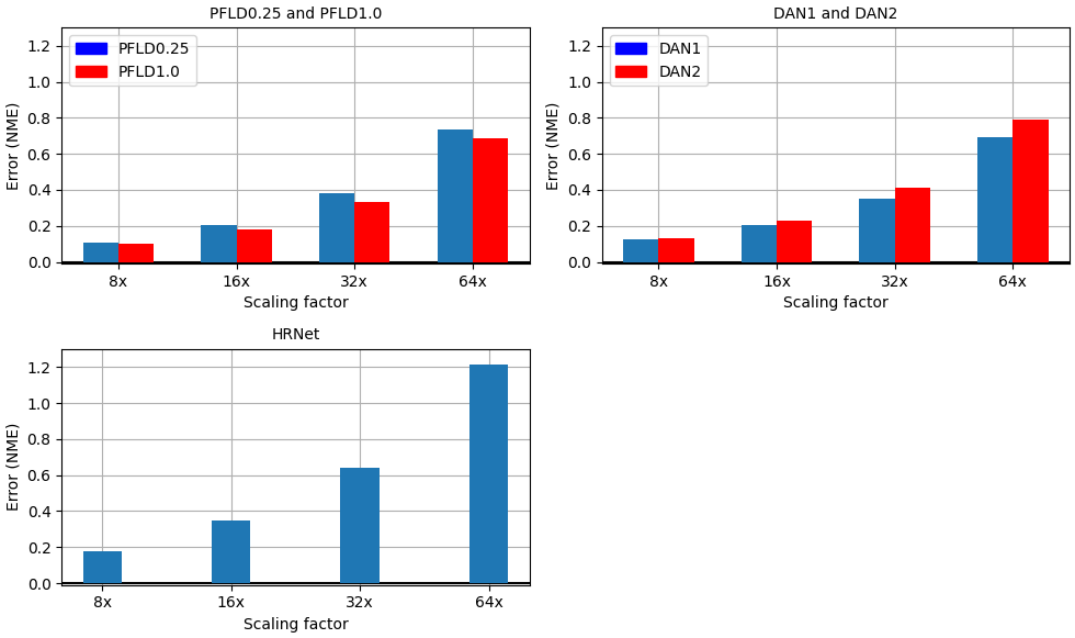

Figure 16. Error compared to extreme downscaling of the image

After testing the effects of the lowered resolution, the images were scaled back to the original size and the same tests were done again. This was done to test how the blur caused by upscaling the images effects the prediction accuracy of the models. The effect of the scaling blur can be seen in Figure 17 that show how even with at the lower scaling, the blur starts to hide some of the finer details like the wrinkles on the example image.

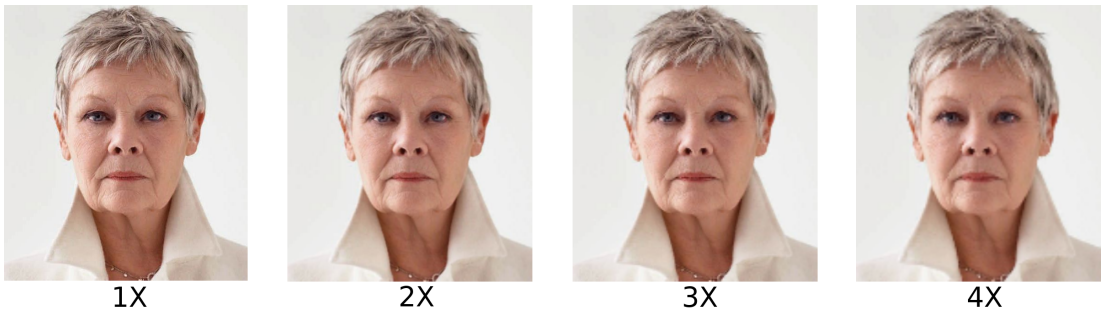

Figure 17. Effect of the scaling blur on image

Figure 18 shows the resulting errors and DAN1, DAN2 and HRNet work just as expected with the error increasing with the amount of scaling done to the images. Both of the PFLD models actually show improvement when the images are scaled to half and up and the further scaling seems to have no significant effect on the error. One of the explanations for this can be the fact that both of the PFLD models used small amounts of Gaussian filtering as augmentation during the training so the models are better at dealing with blur. Similarly to the results of the earlier tests with the resolutions, there is not that drastic drop in the accuracy so once again the tests were done with extreme values.

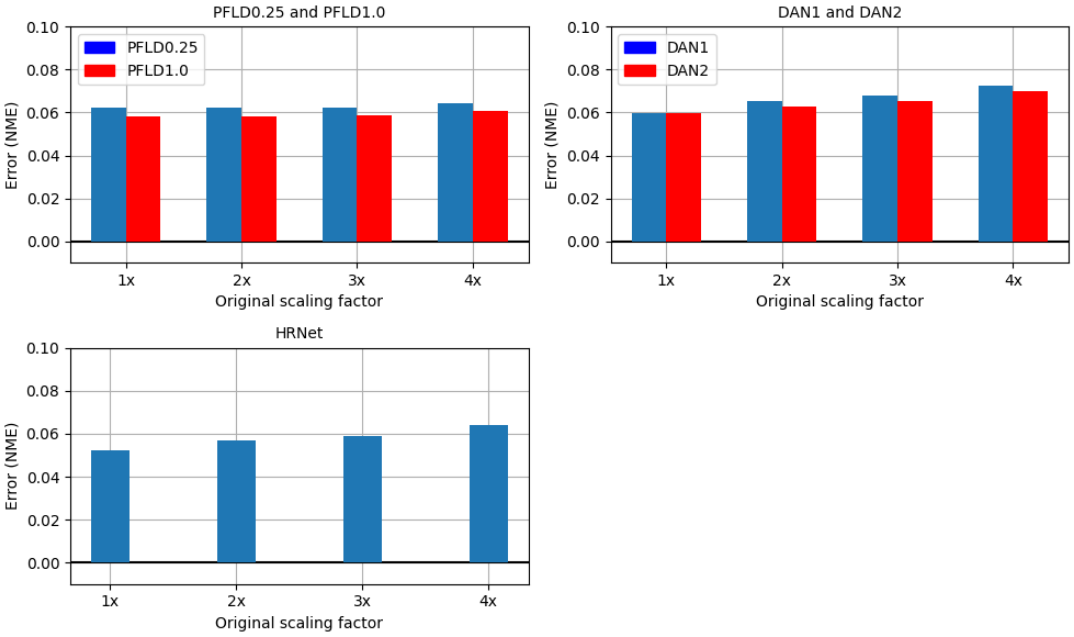

Figure 18. Error compared to scaling of the image

Figure 19 shows the effects of the scaling blur with extreme scaling factors. The extreme scaling blur seems to impact HRNet much more than the other models. With the 8x scaling, all of the models are pretty close together but after that HRNet seems to perform much worse. At the highest scaling (64x), the error of the HRNet is almost double that of the second worst model. Another interesting issue visible in the results is that while with lower scaling PFLDs error stayed very closely the same, this time there is very visible deterioration of the prediction accuracy. The DAN models surpass the PFLD models in accuracy with 32x and 64x scaling, which suggests that the PFLD models function better with low scaling blur whilst DAN is better with high scaling blur.

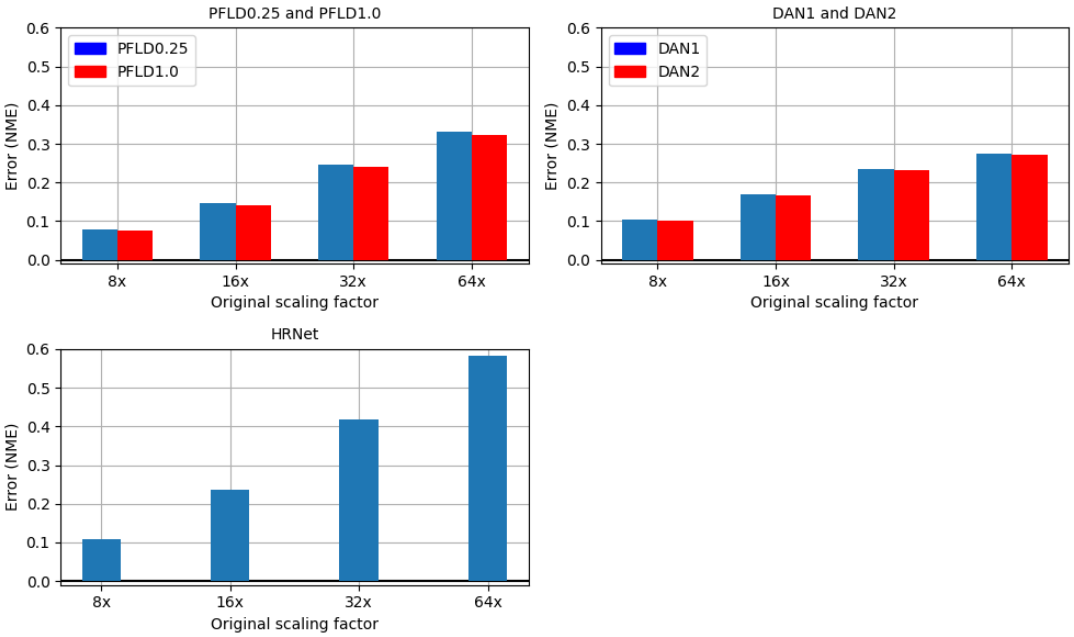

Figure 19. Error compared to extreme scaling of the image

4.1.4. Stability Testing

To see how well the facial landmark detectors work in real-time applications we followed Xiang et al. and added different kinds of noise to a single image and calculated standard deviation (STD)

$$\text{STD} = \frac{1}{d}\sqrt{\frac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{n-1}} \tag{6}$$

between the images to measure the stability between the images [50]. The testing was done with real noise and synthetic noise. The real noise test images was done by taking a video of a still image and splitting it up to multiple images. The random noise was added to a test image to create 200 new images. The noise added was Gaussian noise with mean of 0. For both of the imagesets, the bounding box was the same for all of the images to prevent any kind of jittering from variance in the bounding box. Three different values were tested for the standard deviation to see how the stability changes with the noise.

Table 9 shows the standard deviations of different networks with the testing video and three different kinds of Gaussian noise. The results show that the HRNet is more susceptible to the different types of noise than the other models. Comparing the test results qualitatively, HRNet seems to predict most of the landmarks with perfect stability, but few of the landmarks are constantly changing value with relatively big variation (approximately 10 pixels). Compared to the other models where the predicted landmarks change constantly but the change is so small that it is not that notable. The fact that HRNet outputs the landmarks as integers seems to help with the stability, but either the model itself or the heatmap decoding process is extremely poor at handling different kinds of noise which makes the model poor for real-time applications.

Table 9. Standard deviation of the different networks

Model	testvideo	$\sigma = 0{,}01$	$\sigma = 0{,}1$	$\sigma = 0{,}2$
PFLD0.25	0.3301	0.1542	0.9570	1.7665
PFLD1.0	0.3282	0.1615	1.0204	1.8179
DAN1	0.2928	0.1305	0.7876	2.2130
DAN2	0.2907	0.1242	0.7897	2.0419
HRNET	0.5774	0.2708	1.1084	2.3896

4.2. Performance Evaluation

To evaluate the performance of the models, all of them were converted to TensorFlow Lite models so that they can be tested properly on mobile devices. The models are benchmarked using TensorFlow's own benchmarking tool that includes all of the desired settings like number of threads used for inference and option on whether to use GPU or not.

Because one of the requirements for the model is that it is able to be run in real-time in mobile devices, it is important to define what that would possibly mean in practical implementation. Most of the cameras of modern mobile devices capture video at either 30 or 60 frames per second (FPS). In an optimal solution, the facial landmark detector should be able to process all of the frames of the video input. For a video with 30 FPS this means that for every image there is approximately 30 ms to process the image and 15 ms for the 60 FPS video. These values can be seen as the bare minimum of what could be called real time performance. However optimizing the performance even lower is desirable as in real life situation the landmark detector will not be the only process on the device like with the benchmark tools, but there will be plenty of other processes running like the camera or different types of image processing applications.

All of the models were benchmarked on both of the mobile phones to see how well they are suited to be used in real-time on mobile devices. The results show an interesting pattern on which the performance peaks at four threads and adding threads after that seems to only deteriorate the inference times. Compared to the desktop CPU results where the performance stayed the same after reaching the top performance, the mobile CPUs have some kind of bottleneck that reduces the maximum potential of parallelising the computations. As the mobile CPUs have cores with different speeds, increasing threads can cause them to use the slower cores in the calculations.

4.2.1. Quantization

To see how much more the performance of the models could be improved on mobile devices, the models were quantized. Then, the quantized models were benchmarked to see if there is any difference in how much the quantization effects the models performance and/or prediction accuracy. The same mobile benchmarks were conducted to have comparable data. All of the models quantized correctly except for the DAN2 model which caused it to function incorrectly with the DSP, and it was left out of that benchmark.

4.2.2. PFLD0.25 and PFLD1.0

Figure 20 shows the runtimes of the PFLD0.25 model on different platforms. It was the smallest of the models and the results show that it was also the fastest. Figure 21 has the runtimes of the PFLD1.0 model and once again the results are pretty good. They are not as good as the smaller PFLD0.25, but still promising.

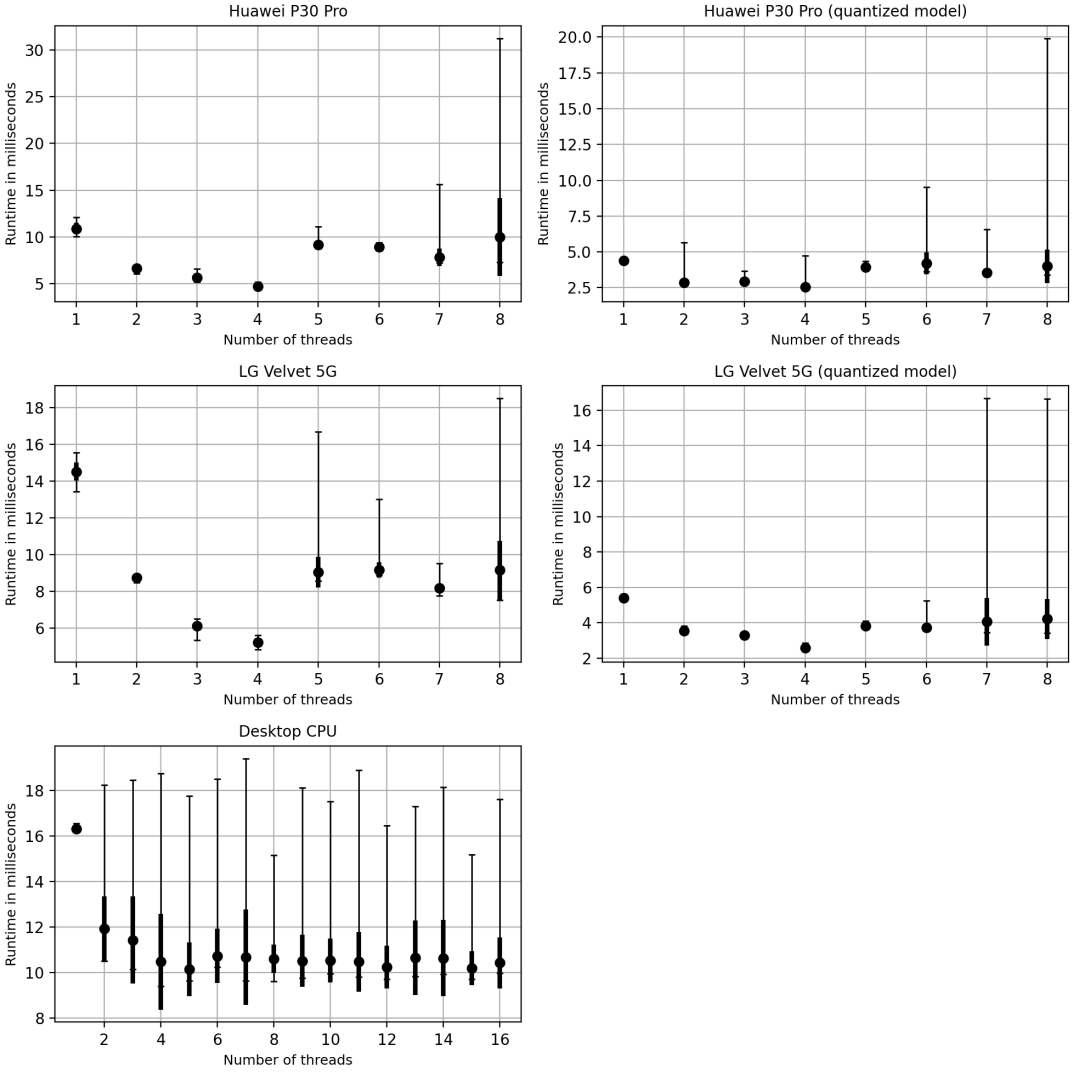

Figure 20. Inference times of the PFLD0.25 model on different platforms

All of the models do not seem to be able to fully utilize the maximum number of threads as the performance usually seems to stop improving after five or six threads. Both of the PFLD models seem to have some performance issues with desktop GPU, which further demonstrates that they were designed for mobile usage. One explanation for this can be that the mobile CPUs memory architecture can be more suitable for loading these small models faster.

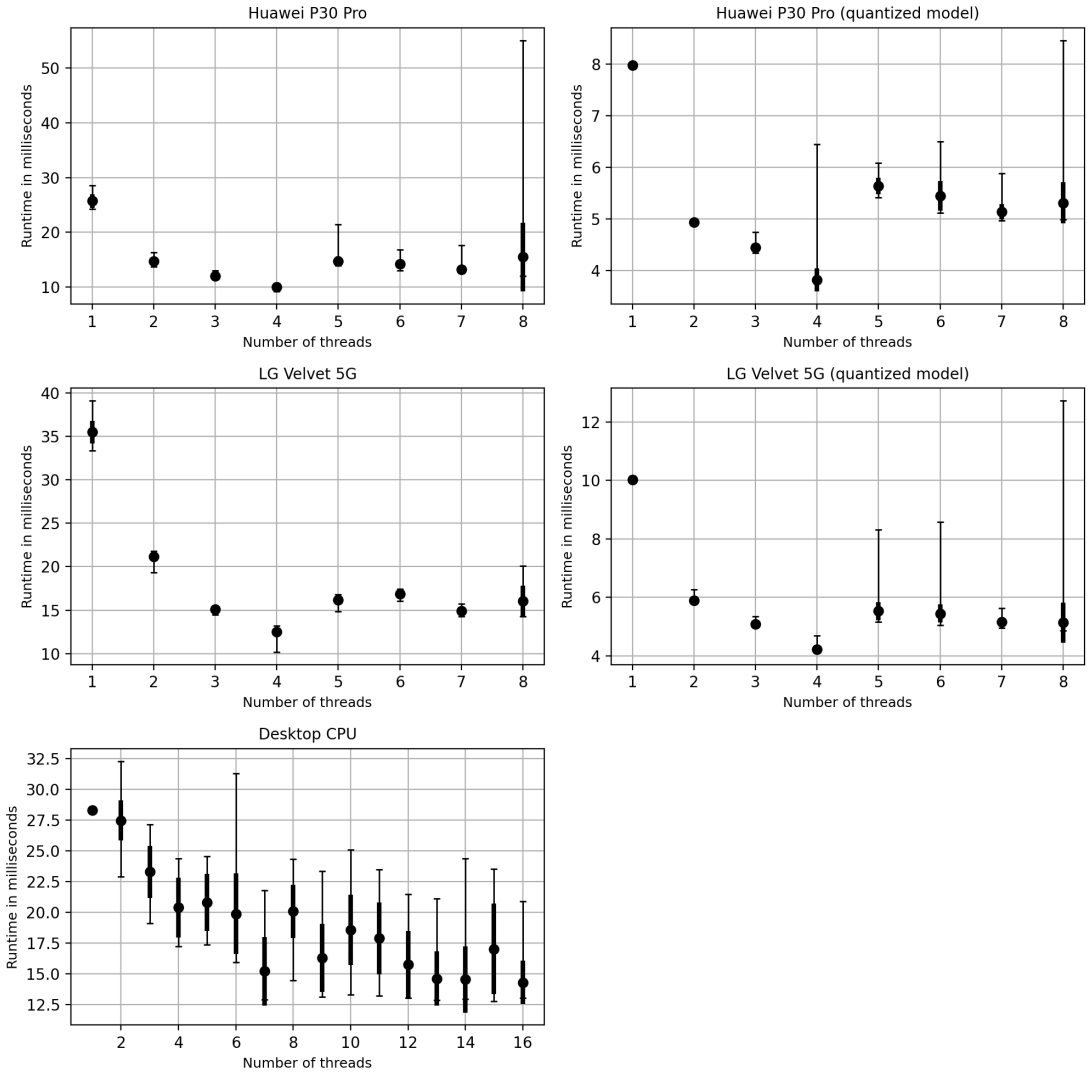

Figure 21. Inference times of the PFLD1.0 model on different platforms

PFLD shows very promising results even when running on the CPU without quantization. Even on the heavier PFLD1.0 model, the inference time is approximately 10 % of the second fastest model (DAN1).The PFLD models can be qualified as working in real-time with both of the PFLD models running under 15 ms on both devices. The runtimes from Figure 25 also show that the PFLD models do not even need to be run on the DSP as the quantized models run just as well on the mobile CPU.

4.2.3. DAN1 and DAN2

Compared to the PFLD models, the DAN models were much larger, but the input has one channel instead of three like in PFLD. The results for the DAN1 can be seen in Figure 22 and for the DAN2 in Figure 23.

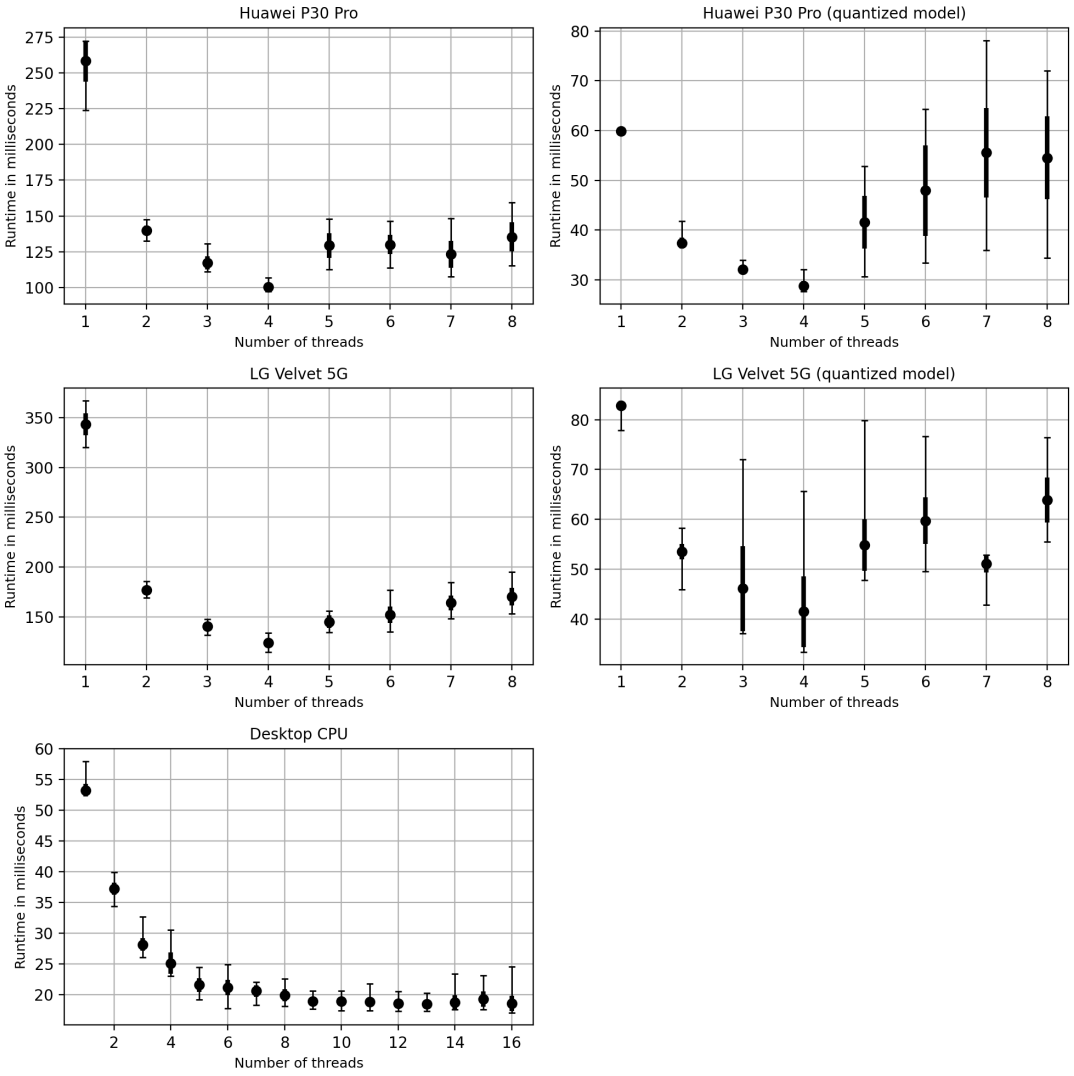

Figure 22. Inference times of the DAN1 model on different platforms

The results show that even the smaller DAN1 model is very slow compared to the lightweight PFLD models. The quantized results show that the DAN2 model is surprisingly slow. As there was issues with quantization of that particular model this can be caused by this, or the problem can be the architecture of the model.

The smaller DAN1 model runs well quantized, but not good enough to be used in real-time. The DAN models perform extremely well on desktop GPU and from Figure 25 we can see that the DAN1 model performs nearly as well as the PFLD models. This further demonstrates the PFLD models suitability for mobile computing, as they were faster in mobile CPU by a large margin.

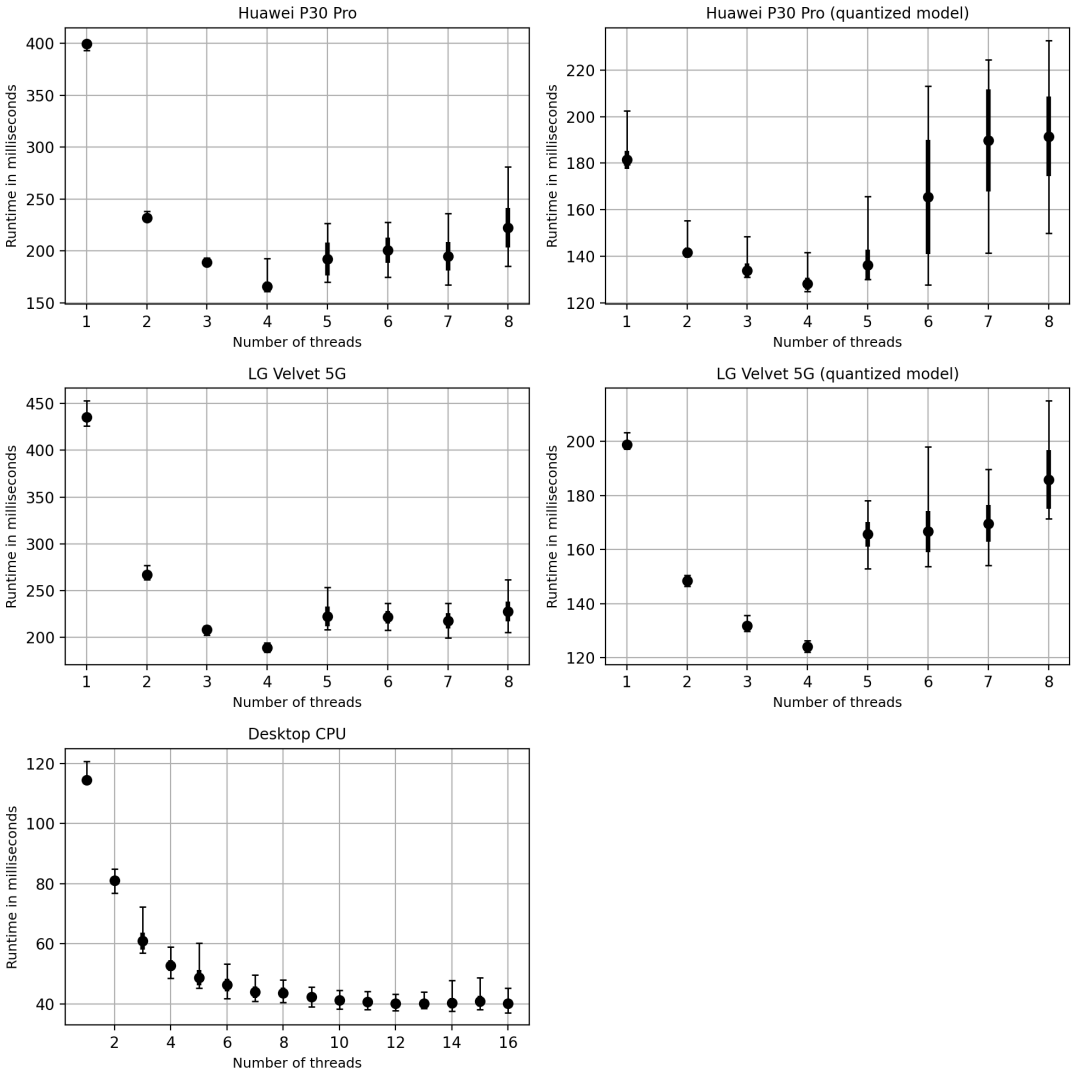

Figure 23. Inference times of the DAN2 model on different platforms

4.2.4. HRNet

HRNet was the heaviest model of the tested ones and the runtimes in Figure 24 display it. HRNet is extremely slow on every platform making it unusable in real-time applications and especially on mobile devices. Surprisingly, HRNet seems to quantize better than the DAN2 model. This can be explained by the fact that HRNet is fully convolutional network and does not contain fully connected layers like the DAN models. From Figure 25 we can see that even when running on desktop GPU, HRNet is still pretty slow at approximately 20ms inference time.

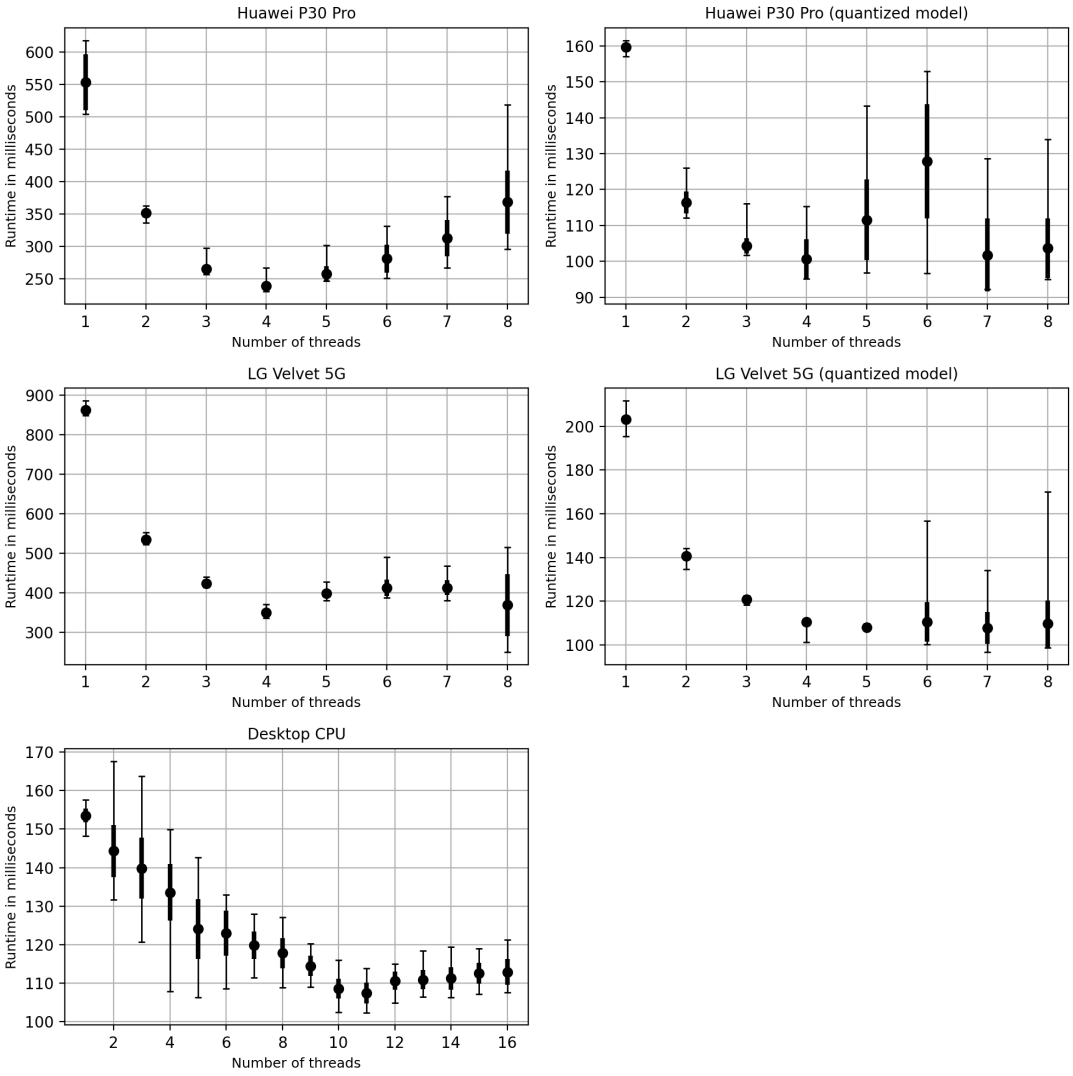

Figure 24. Inference times of the HRNet model on different platforms

4.2.5. Desktop GPU versus 5G DSP

Figure 25 compares how the different models perform on the desktop GPU and surprisingly, the DAN1 model beats both of the PFLD models in this case. Compared to the CPU performance, the DAN1 inference speed is more than four times faster when running on the GPU. One of the possible reasons for this is that the PFLD consists of MobileNetV2 blocks which are designed with mobile devices in mind and not desktop GPUs, so the architecture used in DAN could be more optimal to run on GPU. According to Orsic et al., the depthwise separable convolutions used in MobileNetV2 blocks that are used in the PFLD are not completely supported by the cuDNN library of the GPU which causes some slowness in the calculations [51].

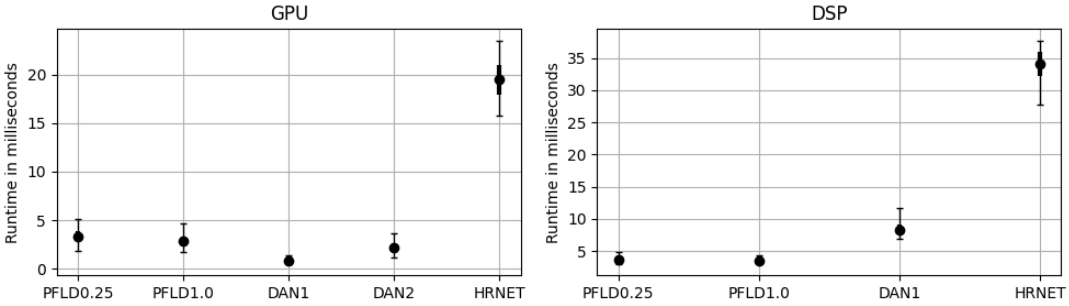

Figure 25. Inference times using desktop GPU and LG Velvet 5G DSP

5. DISCUSSION

The results from the tests show that the modern mobile devices are pretty capable for running deep learning based facial landmark detectors. The limitations of the mobile devices need to be taken into consideration when implementing the networks. This can be either done by adopting more lightweight network architecture, optimizing the existing model or picking more efficient computing platform. The improvements in the performance of the networks practically always leads to drop in the prediction accuracy, which means that the implementation should be balanced in a way that it can run properly on mobile device while having sufficient prediction accuracy.

When running the more lightweight models, the drop in accuracy is not as noticeable as the increased runtime of the heavier models. Even though the heatmap regression based HRNet was the most accurate network in many of the tests, the harder tests displayed some of the weaknesses of the network. The results of these tests suggest that even though heatmap regression based methods are statistically more accurate than coordinate regression, the weaknesses of them seem to outweight the strengths, especially when running the model in realtime on mobile devices.

Quantization did not offer that big of a improvement when running the models on the mobile CPU. Moving the computing to the device's DSP proved to be a very good way to further improve the runtimes, thus making running the quantized models on the CPU pretty much pointless if the device device used has DSP. Moving the computing to DSP offers plenty of possibilities in running even larger and heavier models on the mobile devices. Running the model on GPU is not always the most optimal solution such as with the PFLD model, where the desktop GPU is not the fastest platform. This is because of the architecture the PFLD is designed for mobile usage instead of desktop usage. For future work, the models could be tested to see how much the quantization effects the prediction accuracy.

The PFLD models proved to be very fast even when running on mobile CPU. This means that they can be deployed very easily on different types of devices and the accuracy of the model does not deteriorate because of the quantization. In the future, the mobile devices will very likely be running even more complex DNNs as the computing possibilities improve.

6. SUMMARY

The goal of this thesis was to study if it is possible to run deep learning based facial landmark detectors on mobile devices. This was tested by implementing three different deep learning based facial landmark detectors to see how accurate they are on in-the-wild applications and how well they run on modern mobile devices. The methods tested showed that they are very capable of handling different kinds of variations found in faces. Running the lighter networks on modern mobile devices was also found to be feasible, even when running them on CPUs.

7. REFERENCES

[1] Kopp P., Bradley D., Beeler T. & Gross M. (2019) Analysis and improvement of facial landmark detection. Tech. rep., Unpublished. URL: https://search.datacite.org/works/10.13140/rg.2.2.10980.42886.

[2] Sagonas C., Tzimiropoulos G., Zafeiriou S. & Pantic M. (2013) A semi-automatic methodology for facial landmark annotation. IEEE, pp. 896–903. URL: https://ieeexplore.ieee.org/document/6595977.

[3] Sagonas C., Tzimiropoulos G., Zafeiriou S. & Pantic M. (2013) 300 faces in-the-wild challenge: The first facial landmark localization challenge. IEEE, pp. 397–403. URL: https://ieeexplore.ieee.org/document/6755925.

[4] Sagonas C., Antonakos E., Tzimiropoulos G., Zafeiriou S. & Pantic M. (2016) 300 faces in-the-wild challenge: database and results. Image and vision computing 47, pp. 3–18. URL: https://search.datacite.org/works/10.1016/j.imavis.2016.01.002.

[5] Wu Y. & Ji Q. (2019) Facial landmark detection: A literature survey. International Journal of Computer Vision 127, pp. 115–142. URL: https://search.proquest.com/docview/2036355806.

[6] Guo X., Li S., Yu J., Zhang J., Ma J., Ma L., Liu W. & Ling H. (2019), Pfld: A practical facial landmark detector. URL: https://www.openaire.eu/search/publication?articleId=od_____18::39a3b56e838bf23171ea2d5b5ee06413.

[7] Bodini M. (2019) A review of facial landmark extraction in 2d images and videos using deep learning. Big data and cognitive computing 3, p. 14. URL: https://search.datacite.org/works/10.3390/bdcc3010014.

[8] Ogden S.S. & Guo T. (2019), Characterizing the deep neural networks inference performance of mobile applications. URL: https://explore.openaire.eu/search/publication?articleId=od_____18::91e7baf8d4d8b461b77adf16e3883b08.

[9] Sandler M., Howard A., Zhu M., Zhmoginov A. & Chen L.C. (2018) Mobilenetv2: Inverted residuals and linear bottlenecks. IEEE, pp. 4510–4520. URL: https://ieeexplore.ieee.org/document/8578572.

[10] Lee J., Chirkov N., Ignasheva E., Pisarchyk Y., Shieh M., Riccardi F., Sarokin R., Kulik A. & Grundmann M. (2019), On-device neural net inference with mobile gpus. URL: https://explore.openaire.eu/search/publication?articleId=od_____18::ce446ba92ed091c704e222a951a7ab6a.

[11] Mirza A.M., Hussain M., Almuzaini H., Muhammad G., Aboalsamh H. & Bebis G., Gender recognition using fusion of local and global facial features.

[12] Wang J., Cao B., Yu P., Sun L., Bao W. & Zhu X. (2018) Deep learning towards mobile applications. IEEE, pp. 1385–1393. URL: https://ieeexplore.ieee.org/document/8416402.

[13] Teixeira B., Tamersoy B., Singh V. & Kapoor A. (2019), Adaloss: Adaptive loss function for landmark localization. URL: https://www.openaire.eu/search/publication?articleId=od_____18::d2ed8afa8c25caed82edc33307b2c030.

[14] Wang X., Bo L. & Fuxin L. (2019) Adaptive wing loss for robust face alignment via heatmap regression. IEEE, pp. 6970–6980. URL: https://ieeexplore.ieee.org/document/9010657.

[15] Yu B. & Tao D. (2020) Heatmap regression via randomized rounding URL: https://arxiv.org/abs/2009.00225.

[16] Jin H., Liao S. & Shao L. (2020), Pixel-in-pixel net: Towards efficient facial landmark detection in the wild. URL: https://explore.openaire.eu/search/publication?articleId=od_____18::591d9fea07d3c2f643388a884af41b7f.

[17] Yan Y., Duffner S., Phutane P., Berthelier A., Blanc C., Garcia C. & Chateau T. (2019) Facial landmark correlation analysis URL: https://arxiv.org/abs/1911.10576.

[18] Zhangpeng Z., Ping L., Cheng C.L. & Xiaoou T. (2014) Facial landmark detection by deep multi-task learning. European Conference on Computer Vision , pp. 94–108.

[19] Yamashita R., Nishio M., Do R.K.G. & Togashi K. (2018) Convolutional neural networks: an overview and application in radiology. Insights into imaging 9, pp. 611–629. URL: https://search.datacite.org/works/10.1007/s13244-018-0639-9.

[20] Dong X., Yu S.I., Weng X., Wei S.E., Yang Y. & Sheikh Y. (2018) Supervision-by-registration: An unsupervised approach to improve the precision of facial landmark detectors. IEEE, pp. 360–368. URL: https://ieeexplore.ieee.org/document/8578143.

[21] Du H., Shi H., Zeng D. & Mei T. (2020) The elements of end-to-end deep face recognition: A survey of recent advances URL: https://arxiv.org/abs/2009.13290.

[22] Feng Z.H., Kittler J., Awais M., Huber P. & Wu X.J. (2018) Wing loss for robust facial landmark localisation with convolutional neural networks. IEEE, pp. 2235–2245. URL: https://ieeexplore.ieee.org/document/8578336.

[23] Howard A.G., Zhu M., Chen B., Kalenichenko D., Wang W., Weyand T., Andreetto M. & Adam H. (2017), Mobilenets: Efficient convolutional neural networks for mobile vision applications. URL: https://explore.

openaire.eu/search/publication?articleId=od_____
_18::9e7d8da172878604a58acdda0ca2ecfc.

[24] Ignatov A., Timofte R., Chou W., Wang K., Wu M., Hartley T. & Gool L. (2018), Ai benchmark: Running deep neural networks on android smartphones. URL: https://explore.openaire.eu/search/publication?articleId=od_____18::415fe760875ed7701622f77e41fae151.

[25] Parr T. & Howard J. (2018), The matrix calculus you need for deep learning. URL: https://explore.openaire.eu/search/publication?articleId=od_____18::74901043ea0a3a473e2e89815e3a8232.

[26] Vidal R., Bruna J., Giryes R. & Soatto S. (2017), Mathematics of deep learning. URL: https://explore.openaire.eu/search/publication?articleId=od_____18::683f6edef368dfd2154988b2a1ee5e14.

[27] Krizhevsky A., Sutskever I. & Hinton G. (2017), Imagenet classification with deep convolutional neural networks. URL: http://dl.acm.org/citation.cfm?id=3065386.

[28] Lind E. & Velasquez A.P. (2019), A performance comparison between cpu and gpu in tensorflow. URL: https://explore.openaire.eu/search/publication?articleId=od_____260::1953c289d3d48d33f791e4c314028df0.

[29] Gschwend D. (2020) Zynqnet: An fpga-accelerated embedded convolutional neural network URL: https://arxiv.org/abs/2005.06892.

[30] Naz N., Malik A.H., Khurshid A.B., Aziz F., Alouffi B., Uddin M.I. & AlGhamdi A. (2020) Efficient processing of image processing applications on cpu/gpu. Mathematical problems in engineering 2020. URL: https://dx.doi.org/10.1155/2020/4839876.

[31] Deng Y. (2019) Deep learning on mobile devices: a review. SPIE, vol. 10993, pp. 109930A–15. URL: http://www.dx.doi.org/10.1117/12.2518469.

[32] Cuda toolkit . URL: https://developer.nvidia.com/cuda-toolkit.

[33] Zhang C., Patras P. & Haddadi H. (2019) Deep learning in mobile and wireless networking: A survey. IEEE Communications surveys and tutorials 21, pp. 2224–2287. URL: https://search.datacite.org/works/10.1109/comst.2019.2904897.

[34] Boutros A., Yazdanshenas S. & Betz V. (2018) You cannot improve what you do not measure. ACM Transactions on Reconfigurable Technology and Systems

(TRETS) 11, pp. 1–23. URL: http://dl.acm.org/citation.cfm?id=3242898.

[35] Habermaier A. & Knapp A. (2012) On the Correctness of the SIMT Execution Model of GPUs. Springer Berlin Heidelberg, Berlin, Heidelberg, 316-335 p. URL: https://search.datacite.org/works/10.1007/978-3-642-28869-2_16.

[36] Codrescu L. (2013) Qualcomm hexagon dsp: An architecture optimized for mobile multimedia and communications. IEEE, pp. 1–23. URL: https://ieeexplore.ieee.org/document/7478317.

[37] Jouppi N.P., Young C., Patil N., Patterson D.A., Agrawal G., Bajwa R., Bates S., Bhatia S., Boden N., Borchers A., Boyle R., Cantin P., Chao C., Clark C., Coriell J., Daley M., Dau M., Dean J., Gelb B., Ghaemmaghami T.V., Gottipati R., Gulland W., Hagmann R., Ho R.C., Hogberg D., Hu J., Hundt R., Hurt D., Ibarz J., Jaffey A., Jaworski A., Kaplan A., Khaitan H., Koch A., Kumar N., Lacy S., Laudon J., Law J., Le D., Leary C., Liu Z., Lucke K., Lundin A., MacKean G., Maggiore A., Mahony M., Miller K., Nagarajan R., Narayanaswami R., Ni R., Nix K., Norrie T., Omernick M., Penukonda N., Phelps A., Ross J., Salek A., Samadiani E., Severn C., Sizikov G., Snelham M., Souter J., Steinberg D., Swing A., Tan M., Thorson G., Tian B., Toma H., Tuttle E., Vasudevan V., Walter R., Wang W., Wilcox E. & Yoon D.H. (2017), In-datacenter performance analysis of a tensor processing unit. URL: http://arxiv.org/abs/1704.04760.

[38] Nurvitadhi E., Sheffield D., Sim J., Mishra A., Venkatesh G. & Marr D. (2016) Accelerating binarized neural networks: Comparison of fpga, cpu, gpu, and asic. Institute of Electrical and Electronics Engineers (IEEE), pp. 77–84. URL: https://search.datacite.org/works/10.1109/fpt.2016.7929192.

[39] Kowalski M., Naruniec J. & Trzcinski T. (2017) Deep alignment network: A convolutional neural network for robust face alignment. IEEE, pp. 2034–2043. URL: https://ieeexplore.ieee.org/document/8014988.

[40] Sun K., Zhao Y., Jiang B., Cheng T., Xiao B., Liu D., Mu Y., Wang X., Liu W. & Wang J. (2019), High-resolution representations for labeling pixels and regions. URL: https://www.openaire.eu/search/publication?articleId=od_____18::5d36f67126045f9bec077e95ff128f0a.

[41] Belhumeur P.N., Jacobs D.W., Kriegman D.J. & Kumar N. (2011) Localizing parts of faces using a consensus of exemplars. Institute of Electrical and Electronics Engineers (IEEE), pp. 545–552. URL: https://search.datacite.org/works/10.1109/cvpr.2011.5995602.

[42] Zhu X. & Ramanan D. (2012) Face detection, pose estimation, and landmark localization in the wild. Institute of Electrical and Electronics Engineers (IEEE), pp. 2879–2886. URL: https://search.datacite.org/works/10.1109/cvpr.2012.6248014.

[43] Le V., Brandt J., Lin Z., Bourdev L. & Huang T.S. (2012) Interactive Facial Feature Localization, vol. 7574. Springer Berlin Heidelberg, Berlin, Heidelberg, 679-692 p. URL: https://search.datacite.org/works/10.1007/978-3-642-33712-3_49.

[44] Messer K., Matas J., Kittler J. & Jonsson K. (1999) Xm2vtsdb: The extended m2vts database. In: In Second International Conference on Audio and Video-based Biometric Person Authentication, pp. 72–77.

[45] Devries T. & Taylor G.W. (2017), Improved regularization of convolutional neural networks with cutout. URL: https://explore.openaire.eu/search/publication?articleId=od_____18::d00234ef6029aa2e90ad819951077ec8.

[46] Zhang K., Zhang Z., Li Z. & Qiao Y. (2016) Joint face detection and alignment using multitask cascaded convolutional networks. IEEE signal processing letters 23, pp. 1499–1503. URL: https://search.datacite.org/works/10.1109/lsp.2016.2603342.

[47] Zafeiriou S., Trigeorgis G., Chrysos G., Deng J. & Shen J. (2017), The menpo facial landmark localisation challenge:a step towards the solution. URL: https://www.openaire.eu/search/publication?articleId=od_____2423::4b0cbe310deafe584ee6c7fa12f005ad.

[48] Zafeiriou S., Chrysos G.G., Roussos A., Ververas E., Deng J. & Trigeorgis G. (2017) The 3d menpo facial landmark tracking challenge. IEEE. URL: http://hdl.handle.net/10044/1/71262.

[49] Deng J., Roussos A., Chrysos G., Ververas E., Kotsia I., Shen J. & Zafeiriou S. (2018) The menpo benchmark for multi-pose 2d and 3d facial landmark localisation and tracking. International journal of computer vision 127, pp. 599–624. URL: https://search.datacite.org/works/10.1007/s11263-018-1134-y.

[50] Xiang X., Cheng Y., Xu S., Lin Q. & Allebach J. (2020) The blessing and the curse of the noise behind facial landmark annotations URL: https://arxiv.org/abs/2007.15269.

[51] Orsic M., Kreso I., Bevandic P. & Segvic S. (2019) In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images. IEEE, pp. 12599–12608. URL: https://ieeexplore.ieee.org/document/8954097.