Limited Resource Optimization for Face Recognition Convolutional Neural Networks

Sub-byte quantization of MobileFaceNet using QKeras

Sebastian Bunda s1701290

University of Twente, Enschede
May 10, 2022

***Abstract*—Face recognition is one of the most populair biometric identification systems and as such is widely used. With the growing need for digital personal data security, it is crucial to seek solutions to work on personal devices. To stimulate these developments, the computational and memory footprint of these face recognition systems should be reduced to fit on edge devices. Based on the populair MobileNetV2, MobileFaceNet is a very efficient face recognition neural network with 99.15% accuracy on the LFW dataset with a model size of only 4MB using a 32-bit representation. This work presents a method to reduce the bit length of MobileFaceNet in the form of QMobileFaceNet using sub-byte quantization. This is achieved by first identifying the most strategic use of the QKeras library enabling sub-byte dynamic fixed-point quantization. This work shows that 8-bit and 4-bit versions of QMobileFaceNet can be obtained with 98.68% and 98.63% accuracy on the LFW dataset which reduces footprint to 25% and 12.5% of the original weight respectively. Both show an accuracy loss similar to the performance described by other quantization methods applied on MobileNetV2. Using mixed-precision, an accuracy of 98.17% can be achieved whilst requiring only 10% of the original weight footprint.**

***Index Terms*—Resource Limited Face Recognition, Deep Neural Networks, QKeras, Sub-byte Quantization**

I. Introduction

As of late 2021, the Dutch government proposed a new coalition agreement which focuses, amongst many other things, on the enhancement and enforcement of the security of digital personal information[41]. One of the points discussed in this agreement is the security of the 'individual "online" identity' and the assurance of the abstinence of using Face Recognition systems without strict lawful delineation and control. This is in line with the proposal of the EU artificial intelligence act that was presented in November 2021[22], which can be considered as a first step to regulating the use of artificial intelligence and the regulation of face recognition[23].

With an increasing need for personal digital security, the question may be asked if there is a way to adapt a face recognition system (FRS) such that the user is in charge of his own biometric template. A biometric template can be defined as a digital feature vector that can be used to compare with a new feature vector to identify a person if they are similar enough. In order to comply with this need for personal data processing, ideally, this conversion from a face to a biometric feature vector should be done locally on a personal (mobile) device. A possible scenario (as in Figure 1) could be access control where a secured door is accompanied by a security camera that can detect spoofing and suspicious behaviour. Then when a person requests access, a live photo of the face can then be transferred to the personal device where it can verify the person with a digital handshake.

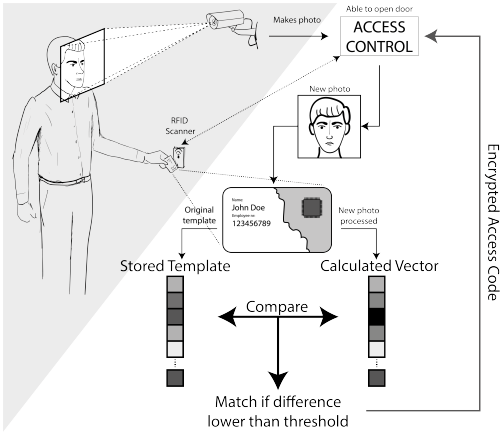


Figure 1: Visualisation of a possible scenario with local biometric verification

In the meantime, machine learning has already found its way to mobile devices due to the exponential growth of available mobile computing power. However, the mobile resources (compared to computers) are still quite limited for the deployment of neural networks (due to a huge number of parameters, calculations and internal representation) which stimulates the search for more efficient neural network architectures and implementations. The current state-of-the-art (SOTA) in mobile image classification networks can be found with e.g. MobileNetV2[32] and ShuffleNet[20] which achieve competitive accuracies whilst requiring only a fraction of the computing power of their much larger counterparts. For example MobileNetV2 achieves 72% Top-1 accuracy compared to ResNet-101[12] 76.4% whilst requiring just 10% of the total number of parameters.

The field of network optimization is dedicated to improving the trade-off between limited resource factors (i.e. size and latency) versus accuracy. We have split the research into two scopes: architecture optimization and network compression. While the first scope is dedicated to improving the efficiency of network architectures, the second is focused on reducing the computation complexity and footprint of existing network architectures through the use of e.g. quantization (minimizing the value representation) and pruning (minimizing the number of unnecessary calculations). Besides the fact that these methods reduce the memory footprint, integer-only inference has shown to also improve the throughput on hardware optimized (for mobile devices) fixed point operations[18][39].

Popular quantization methods for neural networks can be found for both Tensorflow Lite (TFLite)[39][38] and Brevitas for PyTorch[29], although popular edge devices (e.g. ESP32 and Arduino) only support TFLite for Microcontrollers[46] (uTFLite). uTFLite is a split-off of the regular TFLite library optimizing machine learning inference on microcontrollers. Unfortunately, the official implementations of Tensorflow's quantization methods only support 8-bit quantization up until now. This can mainly be attributed to the fact that most of the current microcontroller instruction sets only support the handling of bytes (8 bits). The expectation is, however, that these instruction sets will get support for mixed-precision words as research shows some great progress in the last few years on both (RISC-V) hardware[9, 28] as on a software level[3, 31].

Another implementation that works with Tensorflow and does support quantization that is not limited to only 8-bits (lower than 8-bit is known as sub-byte quantization) is called QKeras[5]. Although QKeras is still in development, the library does show promising results[1]. By utilizing the sub-byte quantization support, the effect of sub-byte quantization on the accuracy of a (face recognition) neural network can be investigated.

Since face recognition can be seen as a subset of image classification, the same principles can be applied here as well. The SOTA in mobile face recognition can be found in networks such as ShuffleFaceNet[24] and MobileFaceNet[4], which have the same basis as the networks described previously. Unfortunately, the quantization of an efficient neural network is a relatively unexplored area in the field of face recognition. It is thus interesting to explore this by applying network optimization on architectures such as MobileFaceNet and identify possible problems such as the discriminability of the feature vector due to the limited solution space.

This work focuses on limiting the resources required to have a face recognition system by quantizing a mobile face recognition neural network. The purpose of this work is to be used when a system has to be designed given hardware constraints. Only when hardware constraints are available a good trade-off can be made between the accuracy loss of the network and the footprint reduction. The main contributions of this work are listed below:

- A Quantization Framework Tooling based on QKeras that can deploy several quantization methods layer-wise for a given neural network model.
- Evaluation of several quantization methods provided by QKeras on different network-layer configurations.
- Evaluation of several quantization strategies by visualizing a bit length vs. accuracy trade-off.
- Exploration of the effects of quantization on the biometric template.

A. Research Questions

In order to approach this problem properly, a research question has been formulated as follows: *What are methodologies that reduce the footprint of a mobile face recognition network architecture using sub-byte quantization and what is their impact on the accuracy of the network?*

As examined earlier, the footprint of a neural network can be drastically reduced using (sub-byte) quantization. On a fundamental level, it is thus interesting to explore how much this affects the performance compared to the original network. In order to mitigate the degradation of the accuracy, an analysis has to be done to identify what could be a good method to reduce this footprint. As quantization is relatively unexplored in face recognition systems, the effect on the discriminability between the feature vectors also has to be investigated.

So to answer the main research question, the following sub-questions will be investigated:

1) What methods are described in the literature to minimize the computational and memory footprints of a convolutional neural network?
2) How is the accuracy of the individual convolutional layer configurations in a mobile face recognition network affected by sub-byte quantization?
3) How are the footprint and accuracy of a mobile face recognition network affected using uniform and individual layer quantization?
4) How is discriminability between biometric face templates affected by sub-byte quantization of a mobile face recognition network?

First, a literature study will explore related work that can help answer the first research question and present SOTA solutions. The literature section will be followed by a preliminary study to analyse the effect of quantization on a layer level using the QKeras quantization framework.

This analysis will be used to answer the second research question and to reduce the quantization parameter search space for MobileFaceNet in the method section. Section IV will explore several quantization strategies and be the foundation to answer the last two questions. The results of the quantization strategies are shown in Section V and discussed in Section VI. Lastly, a conclusion will be drawn from the results to answer the main research question.

II. Literature

In this section, an overview of neural network optimization options will be discussed. First populair mobile deep neural network (DNN) face recognition architectures will be described and compared. This will be followed by the current SOTA of network compression such as quantization and pruning.

A. Mobile Deep Neural Networks

Due to competitions like ImageNet, networks like AlexNet (2012) and ResNet (2015) were able to show the sheer power of DNNs if sufficient data and GPU computing power were available[8]. In 2017, Howard et al. [14] were the first to show with MobileNets that with efficient deep neural network architectures a good trade-off can be made between latency and accuracy. MobileNets replace conventional convolutional layers with depthwise separable convolutions (DSC) to drastically reduce the number of multiply additions without sacrificing the performance.

MobileNetV2[32] quickly followed by replacing the core DSC blocks with inverted residual blocks that are connected at the thin bottleneck layers, gaining 1.4 %p[1] on the Top-1 accuracy of Imagenet whilst reducing the number of parameters and Multiply-Additions with 20% and 48% respectively.

An alternative network ShuffleNet[50], and its successor ShuffleNetV2[20], aimed to increase the computational speed using pointwise group convolutions and channel shuffles. ShuffleNetV2 improved the design by implementing bottleneck structures and a new channel split operation. The difference between the network blocks can be seen in Figure 2.

B. Mobile DNN Face Recognition Systems

The LFW evaluation dataset is one of the most populair face datasets containing 6000 pairs of faces based on 13,233 images and 5749 identities. The AgeDB-30 evaluation dataset also has 6000 face pairs but contains faces with a 30 year age gap. Commonly, these datasets are evaluated using 10-K folding after which the best threshold is found using the most optimal point on the Receiver Operating Characteristic (ROC)-curve. The accuracy of the model is then defined as the accuracy of the system using this threshold.

[1]Percent point, or %p, is the arithmetic difference between two percentages.

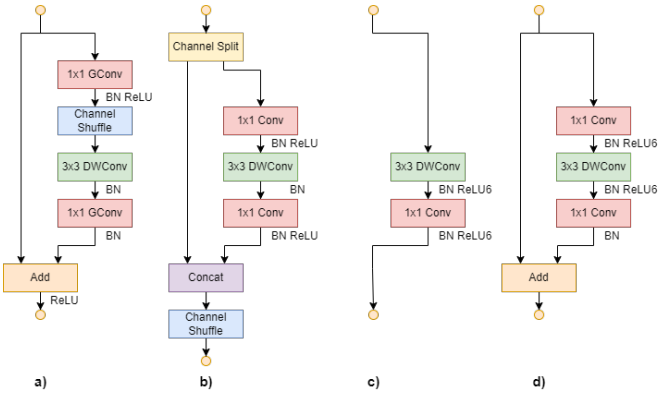


Figure 2: The difference in mobile network architectures. a) the basic ShuffleNet[50] unit; b) the improved ShuffleNetV2[20] unit; c) the basic MobileNet[14] unit; d) the improved MobileNetV2[32] unit. DWConv is a depthwise convolution, GConv is a group convolution and BN is batch normalization.

Populair SOTA mobile DNN face recognition systems are MobileFaceNet[4], ShuffleFaceNet[20] and MixFaceNets[2]. The performance is compared in Table I. The table shows for each model the number of Floating Point Operations (FLOPs) it requires and the number of parameters. The accuracies are shown for the Labeled Faces in the Wild[15] and the AgeDB-30 dataset[26].

MobileFaceNet is based on MobileNetV2[32] and utilizes the same building blocks as MobileNetV2, but reduces the expansion factors in the inverted residual blocks effectively reducing the size of the network. The network also implements a Global Depthwise Convolutional layer as an alternative to a Global Average Pooling layer and PReLU as an improved non-linear activation function. The paper shows a 0.70 %p improvement on the LFW dataset[15] compared to MobileNetV2 using 50% of the total number of parameters.

ShuffleFaceNet is based on ShuffleNetV2[20]. This network also implements a Global Depthwise Convolutional layer and PReLU but uses the ShuffleNet architecture.

MixFaceNet is a relatively new network architecture which is based on MixNets[37] which uses Neural Architecture Search (NAS) to find the optimum network structure. The network uses several kernel sizes in parallel to benefit from the different kernels whilst limiting the computational complexity. For the detection of larger patterns, one needs a larger kernel and a small kernel for the low-resolution patterns. NAS is used to aid the finding of the best parameters for the network. Also implements channel shuffle operations to improve the discriminative ability of the network in the form of ShuffleMixFaceNets.

In this work, we have mainly focused on the architecture of MobileFaceNet as MobileNet is widely used in literature in relation to network compression.

Table I: Comparison of a few very lightweight deep neural network face recognition models. The models in this table have been trained with the MS1M-v2 dataset[6] using the ArcFace loss function[6]

Method	MFLOPs	#Mparams	LFW (%)	AgeDB-30 (%)
ResNet100[25]	24211	65.2	99.83	98.40
ShuffleFaceNet[24]	275.8	1.4	99.45	96.33
ShuffleFaceNet 0.5×[24]	**66.9**	**0.5**	99.23	93.22
MobileFaceNet[4]	439.8[2][2]	0.99	99.55	96.07
MixFaceNet-XS[2]	161.9	1.04	**99.60**	95.85
MixFaceNet-S[2]	451.7	3.07	99.60	96.63
ShuffleMixFaceNet-S[2]	451.7	3.07	99.58	**97.05**

C. Network Compression: Precision Reduction

Another category of network optimisation can be seen as a form of network compression. Network compression aims to minimize the footprint and memory consumption and increase the throughput but can degrade the accuracy compared to the original network. Its goal is therefore also to minimize any deterioration of accuracy.

The first approach in network compression is the reduction of precision (also known as quantization). Most of the SOTA networks are built on the 32-bit floating-point number representation (FP-32). Reducing the bit-precision to e.g. 8-bits can significantly reduce the size of the network model, as an 8-bit word only requires 1 byte compared to the 4 bytes required for a 32-bit word.

A FP-32 number has the following (IEEE-754) representation: $(-1)^s \times m \times 2^{e-127}$ using the sign bit (s), 23 bits for the mantissa (m) and 8 bits for the exponent (e). The absolute number range becomes 10^{-38} – 10^{38}.

Usually, during quantization, this floating-point number is converted to a N-bit fixed-point representation. As such the fixed-point number gets the following representation: $(-1)^s \times m \times 2^{-f}$ using the sign bit (s), (N-1) bits for the mantissa (m) and the scaling factor 2^f. The decimal point is defined using the f parameter. If f is zero, the fixed point representation of an integer is obtained. If f is $N-1$, the number is normalized between [-1, 1].

Dynamic fixed-point representations, having different values for f for e.g. weights and activations, opens up new possibilities compared to an integer only representations. Dynamic fixed-point representations have been shown to work with 8-bit weights and 8-bit activations without significantly affecting the accuracy[10] with some fine-tuning.

The easiest form of quantizing a network is by direct quantizing the weights and activation functions. This is known as Post Training Quantization (PTQ). This results in several quantization losses degrading the quality of the network. If the network is allowed to continue the training by fine-tuning the weights, the losses can be mitigated. This is known as Quantize Aware Training (QAT) and shows in general better results[27].

As an alternative to dynamic fixed-point representations, some research is also dedicated to quantization to enable integer-arithmetic-only inference. Jacob et al. [18] and Zhao, Liu, and Li [51] present a quantization scheme to improve the inference speed on fixed-point hardware (such as ARM processors). Jacob et al. show that integer only arithmetic inference improves the latency vs accuracy trade-off compared to a 32 floating-point network. This work is also incorporated into the Tensorflow optimisation tooling where it only shows a 1 %p drop in accuracy for 8-bit MobileNetV2 using QAT[39].

Zhao, Liu, and Li [51] propose an improvement over the work of Jacob et al. in the form of a Bounded Rectified Linear Unit (BReLU). The goal of this activation function is to scale the upper bound of the ReLU6 dynamically with the maximum value of the input. Instead of the scale being affected by relatively large numbers, it tries to minimize the quantization loss by limiting the effect of those large outliers.

Understandably, the current implementations of the tooling are still limited to 8-bit integers (the smallest width in common instruction sets) and lack the support for flexible quantization to find the optimal quantization method for each layer. Although integer-only arithmetic can speed up inference on mobile devices, the support on x86 devices is limited resulting in a significant increase in inference time[3].

An alternative quantization scheme is provided by QKeras[5]. The library works as an extension on the Keras framework of Tensorflow and allows drop-in Qlayer replacements for each layer. It is able to quantize to a dynamic fixed point, exponent (in powers of two), and sub-byte representations. Furthermore, it also allows for custom quantization settings for each layer. For the sake of customizability and sub-byte quantization support, this paper will investigate the possibilities of quantizing MobileFaceNet using QKeras.

D. Network Compression: Reduction of Operations

Another approach in network compression is reducing the number of operations within the network architecture. Due to over-parameterization in large networks, it can be beneficial to consider removing elements in

[2]The original MobileFaceNets paper states 221 million Multiply Addition operations which is roughly twice the number of Floating Point Operations (FLOPs)

[3]https://github.com/tensorflow/tensorflow/issues/40183#issuecomment-641754983

the network that don't contribute to the output of the network (weights that are zero or near zero). This process, also known as pruning, can significantly reduce the size and computational load of the network. Han et al. [11] shows that AlexNet and VGG-16 can be compressed with a factor of 9 and 13, respectively without affecting the accuracy (with some fine-tuning). More modern methods can be applied to take the energy consumption into account[44] and reverse pruning which starts the pruning process from the output layer back to the input[49]. There is some work aiming to prune MobileNet, but it shows a significant degradation of the performance[40].

Proposed by Hinton, Vinyals, and Dean [13], knowledge distillation is another method to reduce the number of operations. By training a significantly smaller model using the responses of the output neurons of a larger reference model, similar accuracy can be achieved. This idea is also applied in the domain of face recognition with e.g. ProxylessKD[35]. The authors optimize a small face recognition network by using a large model's classifier to guide the smaller model's classifier to learn discriminative embeddings which directly optimizes the model accuracy.

Methods such as pruning and knowledge distillation will not be considered in this work as it is likely not necessarily conducive to the performance, as it assumes redundancy within the network. Given the fact that mobile network architectures, such as MobileFaceNet, do already employ efficient architectures, it may be assumed that optimizations such as those documented by Han et al. will not be achieved compared to the quantization prospects. Nevertheless, optimizations such as pruning might be able to work on the quantized network, and it could be considered for future work in further compressing the network.

E. Summary

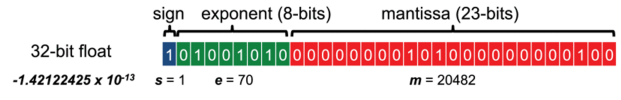
To summarize this literature study and to answer the first research question, several methods exist to reduce the computational and memory footprint of a convolutional neural network. The first method is by implementing convolutional computation more efficiently, as can be seen with a DSC in the case of MobileNet, or group convolutions and channel shuffle as seen in ShuffleNet. Given an efficient network architecture such as MobileNet, there are also other methods like pruning, knowledge distillation and quantization. Quantization seems to have a great potential to significantly reduce the footprint of a CNN, as it has shown to reduce the footprint of MobileNetV2 by $3.8\times$ (including some extra overhead, such as a scaling factor) with an int-8 representation with only a 1%p drop of accuracy. In order to minimize a face recognition system such as MobileFaceNet, it seems evident to start with quantizing the network. To also be able to investigate sub-byte quantization capabilities, QKeras will be used.

III. Preliminary Study: A QKeras Quantization Loss Analysis

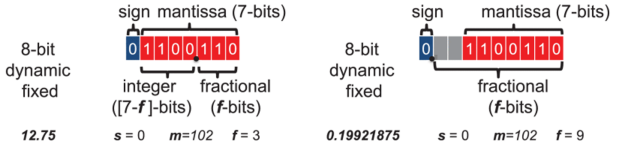
In this section, a preliminary study will be done to aid the core research of this paper. The QKeras library will be used to obtain QMobileFaceNet. However, this library supports several different implementation parameters and within the scope of this work, it is not possible to evaluate all of them on the face recognition network. Therefore, the study will try to answer the second research question: *How is the accuracy of the individual convolutional layer configurations in a mobile face recognition network affected by sub-byte quantization?*

To answer this question, the sub-byte quantization method will be elaborated upon first. Secondly, several small experiments will be done on the individual layer blocks found in MobileFaceNet. To reduce the training time and to explore multiple parameters, these layers will be placed in small networks and trained for the MNIST[7] and MNIST-Fashion[43] dataset. The MNIST dataset contains handwritten digits and the more complex Fashion-MNIST dataset[43] contains 10 different clothing categories. This will stress the system for two levels of difficulty without much variability. Both datasets are in black and white, same dimensions and are equal in size. Finally, based on these experiments an analysis will be done to see which parameters are likely to benefit the quantization of MobileFaceNet in the method.

A. QKeras Quantization Methods



(a) 32-bit floating point example



(b) 8-bit dynamic fixed point examples

Figure 3: The difference in the 32-bit floating point representation and an 8-bit dynamic fixed point representation[36].

As described in the related work section, QKeras[5][1] uses dynamic fixed point quantization. This means that the quantization method can specify how many bits (of the total N) are reserved for the integer representation i:

$$(-1)^s \times m \times 2^{-f}\,, \tag{1}$$

where $m = N - 1$ and $f = m - i$. A quantized version q of a real value x can be converted using the following method:

$$q = \text{clamp}(s \cdot \left[\frac{x}{s}\right], -2^i + s, 2^i - s) \tag{2}$$

"clamp" and s are defined as follows:

$$\begin{aligned} \text{clamp}(x; a, b) &:= \begin{cases} a, & x \leq a \\ x, & a < x \leq b \\ b, & x > b \end{cases}, \\ s(i, n) &:= 2^{i-n+1}, \end{aligned} \tag{3}$$

where $[\cdot]$ represents the rounding operation, i is the number of bits representing the integer left of the decimal and n is the total number of bits. An example of the difference in representation can be found in Figure 3. QKeras has the option to disable symmetric quantization adding one option from the lower bound resulting in the clamp becoming: $\text{clamp}(x; -2^i + s, 2^i - s)$. This also shifts the zero point off-centre and is not used in this work. Finally, the PReLU activation function of MobileFaceNet is not supported, so quantized ReLU will be implemented instead.

Below is a list of options available for the scaling factor of the quantization method using QKeras:

- **None or '1' (default):** No specific scaling is defined and uses the whole range.
- **Automatic:** Computes scale based on each output channel
- **Auto po2:** Computes the scale to be a power of 2 (can be beneficial for multiplications but creates logarithmic steps between the available values options).

These effects of the default and the first automatic option will be analysed in the rest of this section as logarithmic quantization is not considered in this work.

B. Quantization of Convolutional Layers

The QKeras library is built to work on top of the Keras framework of Tensorflow[5]. As such, QKeras provides Qlayer equivalents that can be interchanged with the normal layers but act as a shell to handle all the quantization tasks.

The network architecture of MobileFaceNet implements alternatives amongst the standard convolutional layer such as the DSC. In order to obtain the best QMobileFaceNet network, we have to find the best configuration for each type of layer. The concept of DSC and other layer optimizations used by MobileFaceNet are elaborated in Section IV-B.

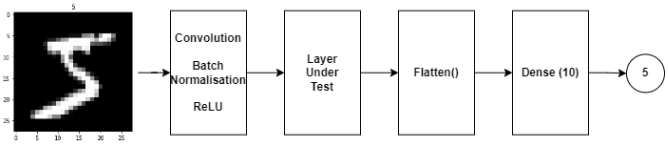


Figure 4: The MNIST network to test the quantization performance of each type of layer.

In order to figure out the best quantization configuration for each layer configuration, some experiments will be performed for each type of layer. The network that will be trained for each layer can be seen in Figure 4. The first layer (CBA[4]) is a standard convolutional layer using batch normalization and activation function and is needed to provide the necessary channels for the layer block configurations under test. The kernel of each layer block configuration is always 5×5 pixels, except for the linear global depthwise convolutional layer which is the same size as the input width (28×28). The output dimension (depth) is always 3.

The floating-point network is trained for 20 epochs and is compared with the same network is which quantized after 10 epochs and trained using QAT for an additional 10 epochs. Each training is repeated 5 times with different seeds. Each network will be tested for 2 – 8-bit levels on the MNIST and MNISTFashion datasets.

Experiment 1: *Batch Normalization Folding*

For the first experiment, the effect of the folding of the batch normalization for the convolutional layer will be investigated. As explored by Hubens [16], batch normalization can be folded (or fused) with the convolutional layer. Hubens shows that the folding of the batch normalization with the convolutional layer results in a speedup in training with less learnable parameters without a significant impact on the accuracy. The folding of batch normalization is also implemented in QKeras, unfortunately, it is no longer supported by Tensorflow 2.0+ so the quantized folded network will be compared to the QKeras implementation without a quantizer. An additional advantage is that the folded layers can be unfolded after training, removing the batch normalization, resulting in a smaller inference model. The layers that support folding (in QKeras) are the normal convolutional layer and the depthwise convolutional layer. The results can be seen in Figure 5. The figure shows the mean of 5 different runs, with the shaded areas representing the confidence interval.

Experiment 2: *Fixed Point Location*

For the second experiment, the effect of the fixed point location will be explored. The position of the fixed point specifies the range in which the weight can be

[4]Convolution – Batch normalisation – Activation function

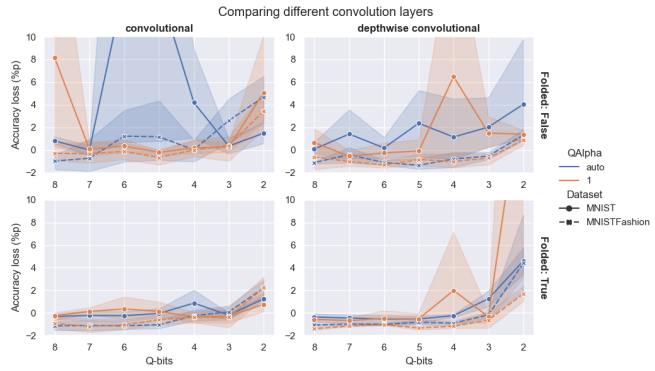


Figure 5: The percent point loss defined as the loss in accuracy due to folding the batch normalization layer on the convolutional layers (lower is better). The first row is the convolutional layer without folding and the second row is with the batch normalization folded into the convolutional layer.

expressed. An integer representation of e.g. 2 or 3 bits results in a range of (-4, 4) and (-8, 8) respectively. The results for both scaling options can be found in Figure 6. The percent point accuracy loss for each fixed point position with a certain bit-length is shown for the inverted residual blocks.

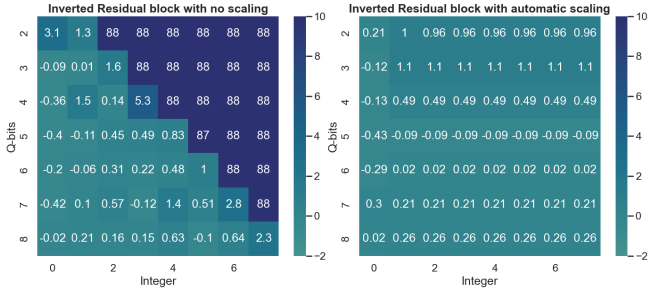


Figure 6: The effect of the decimal point position (Integer) for different bit lengths (Q-bits) and scaling on the inverted residual blocks. The values are the percent point accuracy loss (lower is better). Using the MNIST dataset.

Experiment 3: *Scaling Factor for different types of convolution*

For the last experiment, the effects of the absence and automatic scaling factor will be investigated for each layer type found in MobileFaceNet. Each configuration uses a folded convolutional (CBA) layer and zero bits dedicated to the integer representation. The results can be seen in Figure 7. The translucent areas represent the confidence interval from 5 different runs, the line itself is the mean of those runs.

Experiment 4: *Non-linear activation in the Depthwise Separable Convolution*

In the original network architectures of MobileNets the DSC is implemented by applying both batch normalization and the ReLU activation function after the depthwise convolution and after the pointwise convolution. However, research shows that this implementation can have a significant effect on the accuracy of the quantized version of MobileNetV1. Sheng et al. [34] show that by removing the batch normalization and the ReLU after the depthwise convolution. This would increase the accuracy from 1.8% to 68.03%. Yun and Wong [47] come to the same conclusion. On the other hand, at the Tensorflow website[39] it is shown that their method of quantization did not have a significant effect on the performance as they achieve 70% accuracy with MobileNetV1 and 70.9% with MobileNetV2 compared with the original score of 71.9% with MobileNetV2 without quantization.

Assuming the Tensorflow implementation does not change the network architecture, this shows that the quantization method can heavily impact the quantization performance. Therefore, another experiment has been conducted to estimate the effect of this non-linear activation function on the quantization performance of the DSC.

Figure 8 shows the effect of the quantization on the DSC (as in MobileNetV1) and the inverted residual (as in MobileNetV2). The orange line shows the original (split) implementation and the blue line takes the suggestion of[34] to remove the split of the DSC by removing the batch normalization and ReLU after the depthwise convolution.

C. Analysis

From Figure 5 it can be clearly seen that the convolutional and the depthwise convolutional layer benefit from the folding of the batch normalization. Both the automatic scaling and the no scaling have significant lower spikes, with the exception of the 2-bit depthwise convolutional layer. The runs with the MNISTFashion, however, show much smaller deviations. This would still suggest that the folded networks are more likely to be easier to quantize with a lower quantization error than the standard Keras implementation without folding. Nonetheless, the MNISTFashion runs show that separate batch normalization layers do not necessarily yield bad results.

When we consider the position of the decimal point, as seen in Figure 6, it can be concluded that the quantization without scaling only works if the bit reserved for the integer number representation is lower than the total bit length. When automatic scaling is applied, this is then compensated for resulting in the same performance for all integer bit-lengths larger than 0. Interestingly enough, the full fractional length (integer=0) often yields a (slightly) better performance in the case of the inverted residual blocks. From this, we can conclude that in general, the network doesn't seem to benefit from being able to represent the integer bit.

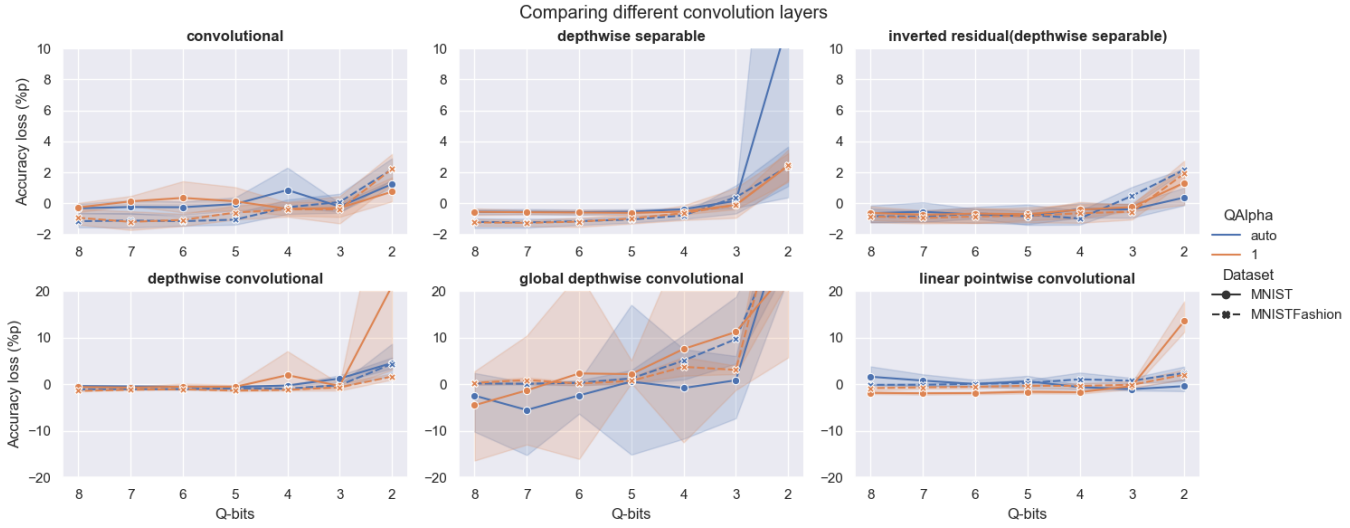


Figure 7: The effect of the scaling factor for different convolutional layer configurations for different bit lengths.

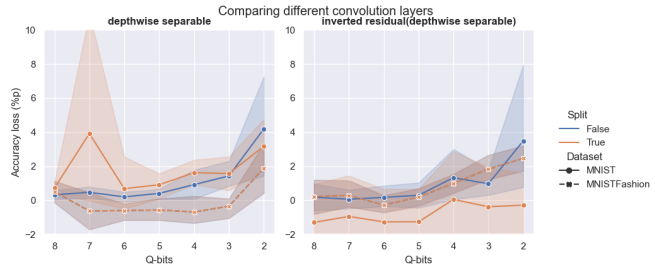


Figure 8: The effect of the implementation of the separable convolution for different bit lengths. The split signifies whether the depthwise and pointwise convolution are separated by a batch normalization and activation function. The scaling factor is determined using the 'auto' parameter.

From the third experiment in figure 7, we mostly see similar performance between the two scaling options for all the types of convolutional layer configurations. The only exception is the linear global depthwise convolutional layer, which sees a great variance in the performance over the course of several runs. This could be mainly attributed to the fact that this layer acts as a global average pooling layer and sees a great dimensionality reduction. This discarding of information is thus likely to lead to great variance in performance and quantization accuracy.

The last experiment in Figure 8 estimates the effect of different implementations for the DSC. The first observation that can be made is that both implementations don't differ much from each other, except for three points. The first point is the fact that the DSC with the DSC-split has a large mean accuracy loss at 7-bits. However, this is likely due to an outlier as the confidence interval is also large and the rest of the bit lengths of the network seem to behave similarly to the rest of the experiments. The other point of interest is the same orange mean line at the inverted residual implementation seems to behave better than the rest of the MNIST dataset. On the other hand, for MNISTFashion, the implementation seems to behave on par with the blue lines. The last observation is the fact that the DSC implementation without the split has the potential to behave slightly worse using a 2-bit quantization. Nevertheless, this experiment does not show major differences between the two implementations and can be considered inconclusive.

Considering this analysis, it can be said that the folding of the batch normalization and the use of the full bit range for the fractional number are likely to be the best quantization parameters to quantize MobileFaceNet. The analysis did not rule out one of the scaling options, as both seem to have a similar performance. The same holds for the implementation of the DSC.

IV. Methods

In this section, several experiments will be designed to explore the effect of sub-byte quantization on the performance of MobileFaceNet. We will make use of the conclusions drawn from Section III concerning the best quantization parameters for MobileFaceNet. At the start of this section, an explanation of the model architecture of MobileFaceNet will be provided.

The first set of experiments will focus on the research question concerning the effect of the level of quantization on the accuracy of evaluating both uniform and individual layer quantization. The experiments will explore the performance of QMobileFaceNet for different quantization methods using 8-bit, 4-bit and 2-bit quantization. 1-bit quantization will not be considered in this work as this requires adaptations to the network and can be considered a work in itself.

The last experiments will focus on the remaining research question that delves into the question of whether

the discriminability of the face recognition system is affected by the (sub-byte) quantization. This will be done by using the 8, 4 and 2-bit uniformly quantized networks.

A. Datasets and evaluation metrices

The CASIA-Webface[45] dataset is used to train all the (Q)MobileFaceNet networks. The CASIA-Webface dataset consists of 435.779 images of 10.575 different identities build from images found on the internet using a semi-automatic method.

Chen et al. mention the MTCNN[48] alignment method is used for MobileFaceNet, however, when applied to the CASIA-Webface dataset the accuracy could not be matched and many mistakes were found after further inspection of the aligned images. This resulted in the use of the alignment protocol StyleGAN2[19], which yielded better-aligned faces but uses an inherently different alignment method as the evaluation dataset. This could result in a minor decrease in accuracy. The performance is evaluated using the Labeled Faces in the Wild[15] and the AgeDB-30 dataset[26]. The evaluation datasets both contain 6000 frontal face pairs, of which half is a face pair of the same identity. The network is evaluated (as[6]) by comparing both feature vectors that are calculated for each face by calculating the angle between the vectors. Then the False Positive Rate and False Negative Rates are calculated for several thresholds. Finally using 10-K folding, the best threshold is found, and for this threshold, the accuracy is determined.

B. QMobileFaceNet

The network architecture of MobileFaceNet is visualized in Table II. The bottlenecks are the same type used in MobileNetV2 and consist of inverse residual bottlenecks with an expansion (t), channel depth (c), repetition (n) and a stride (s). An official implementation of MobileFaceNet using the TensorFlow 2+ framework does not exist. An attempt has been made to recreate a truthful implementation in Python using TensorFlow 2.4 following the training algorithm described by Chen et al. [4]. As mentioned earlier, QKeras does not support a quantizable version of PReLU (the activation function used by MobileFaceNet), our version of QMobileFaceNet will use the ReLU activation function instead (with 3 bits for the integer representation).

Chen et al. mention only using ArcFace as a loss function, however, we were unable to recreate the performance shown in their paper. Better accuracies were achieved by first training with a softmax classification header and then swapping it with an ArcFace classification header before being retrained again.

For both the softmax and ArcFace header, the models were trained using the Stochastic Gradient Descent (SGD) optimizer function with a momentum of 0.9 and a batch size of 512. The learning rate started with a value of 0.1 for the softmax and 0.01 for the ArcFace header. After 36K, 52K and 58K batches the learning rate was divided by 10. Each training session ended after 60K batches. For the QAT it was found that the Adam optimizer with a learning rate of 0.01 worked better than the SGD optimizer.

Interestingly enough, as it differs from the original implementation, it was found that removing the batch normalization with a folded network after the depthwise convolution of the second and ninth layers improved both the float-32 and quantized network accuracy. This is implemented by replacing the folded depthwise convolution block with a normal depthwise convolution block and retraining the network for 10 more epochs.

Table II: The network architecture of MobileFaceNet. t is the expansion factor, c is the channel depth, n is the number of repetitions and s is the stride (of the first layer if bottleneck)

Architecture MobileFaceNet[4]						
Layer	**Input**	**Operator**	**t**	**c**	**n**	**s**
1	$112^2 \times 3$	conv3×3		64	1	2
2	$56^2 \times 64$	depthwise conv3×3		64	1	1
3	$56^2 \times 64$	bottleneck	2	64	5	2
4	$28^2 \times 64$	bottleneck	4	128	1	2
5	$14^2 \times 128$	bottleneck	2	128	6	1
6	$14^2 \times 128$	bottleneck	4	128	1	2
7	$7^2 \times 128$	bottleneck	2	128	2	1
8	$7^2 \times 128$	conv1×1		512	1	1
9	$7^2 \times 512$	linear GDC7×7		512	1	1
10	$1^2 \times 512$	linear conv1×1		128	1	1

C. Post-Training Quantization

The simplest form of quantization is by quantizing the trained float-32 weights and activation functions of MobileFaceNet. Using this experiment QMobileFaceNet will compare 8, 4, and 2-bit quantization for both scaling and DSC split implementation options.

D. Quantize Aware Training

QKeras is designed such that the weights and activations can take specific quantization schemes into account while training. In order to test this capability on a larger scale than in Section III, several experiments will be done to find the best method to quantize the whole QMobileFaceNet network. The quantization analysis concluded that the networks should be folded and have only fractional bits.

1) Individual Layer Quantization

An initial experiment will quantize each layer of MobileFaceNet individually as shown in Table II. The goal is to find the location where the convolution is the most affected by the quantization. The expectation is

that the first layers will affect the accuracy the most as errors made in the beginning will propagate the most towards the end. We also expect that the last layers will not affect the accuracy in a significant way for the same reason.

2) Uniform bit-length quantization
The second strategy to fully quantize MobileFaceNet is by continuing the training of the post-training quantized network. The network is trained for 20 epochs with a learning rate of 0.001 using the Adam optimizer. This strategy will compare the 8, 4 and 2-bit uniform quantized network with both weights and activations quantized using the specified bit length. To show the consistency, the mean and standard deviation will be calculated for 3 separate runs for each bit length.

3) Mixed precision quantization
The third strategy will involve different mixed-precision networks for each layer configuration. The goal is to obtain a graph to show a trade-off between footprint and accuracy. The estimation of the footprint will be made using the number of bits required for the representation of the weights and thus does not include any overhead required for the implementation. For this experiment, the first two layers will remain 8-bits and the rest of the layers are either 4-bit or 2-bit. A detailed table with the models and the corresponding total weight bits can be found in Table A.2.

E. Analysis on the quantization of the Face Feature Vector

The feature vector output of the face recognition system is used to compare the features between two faces to determine whether the features are similar enough to belong to the same identity. Quantization significantly reduces the total solution space by reducing the number of representation spaces per feature. In order to investigate the effect of (sub-byte) quantization on this decision making, some small experiments will be done.

First, the feature vectors of 10 identities from the MS1M-V2 dataset[6] are calculated. For these identities, a similarity matrix is constructed to show the difference in angles between different identities. For the same set of angles, a t-distributed stochastic neighbour embedding (t-SNE) analysis[21] will be performed to show the clustering ability between the feature vectors. The high dimensional data points are modelled in a two-dimensional map, where similar points are clustered together using their probability distribution. The resulting figures of the t-SNE will thus not give any numerical results.

The second experiment will generate the ROC curves for the LFW and AgeDB-30 datasets. These should show the trade-off for several thresholds and show whether the curve is flattened at some point limiting the performance. From this ROC curve, the area under the curve can be obtained and compared.

V. Results

In this section the several QKeras quantization methods will be evaluated on their performance compared to the float-32 implementation.

A. Post-Training Quantization

The results of the post-training quantization can be found in Table III. If the accuracy is 50% it means that by comparing the two faces, the threshold is placed at 0 degrees and the result of the comparison is always negative. This will give an accuracy of 50% as half of the face pairs is a face pair of the same identity.

Table III: The results of Post-Training Quantizing MobileFaceNets. The best method is highlighted.

QMobileFaceNet		**LFW**				**AgeDB-30**			
Split	Scaling	32-bit	8-bit	4-bit	2-bit	32-bit	8-bit	4-bit	2-bit
True	Auto	98.85	84.27	59.33	50.82	90.33	61.60	53.00	49.97
True	1	98.85	56.70	50.00	50.00	90.33	52.15	50.00	50.00
False	**Auto**	**98.85**	**94.65**	**63.15**	**51.55**	**90.38**	**73.42**	**54.98**	**50.55**
False	1	98.85	58.48	50.00	50.00	90.38	52.85	50.00	50.00

B. Quantize Aware Training

In the methodology section, we have proposed several quantization strategies to obtain QMobileFaceNet. In this section, QMobileFaceNet is implemented using automatic scaling and no split of the DSC as this was the best method using PTQ. The results can be seen below.

Table IV: The mean results (standard deviation) of Quantize Aware Training of MobileFaceNets

QMobileFaceNet				
LFW	**32-bit**	**8-bit**	**4-bit**	**2-bit**
Post Training Quantization	98.85	94.65	63.15	51.55
Quantize Aware Training	98.85	98.68 (0.15)	98.63 (0.18)	93.45 (0.66)
AgeDB-30	**32-bit**	**8-bit**	**4-bit**	**2-bit**
Post Training Quantization	90.38	73.42	54.98	50.55
Quantize Aware Training	90.38	88.79 (0.10)	88.20 (0.50)	75.62 (2.83)

1) Individual Layer Quantization
The results of the individual layer quantization can be observed in Figure 9. This plot also visualizes the number of parameters that are quantized in each layer. This plot shows that the most accuracy degradation can be attributed to the quantization of the layers 3–5.

2) Uniform bit-length quantization
The mean results for the uniform bit lengths quantization can be found in Table IV. The QAT results are based on the mean (and standard deviation) of three different quantization training sessions. This table shows that quantize aware training can significantly improve the quality of the quantized network. It also shows that the network could be quantized to 4-bits with only an

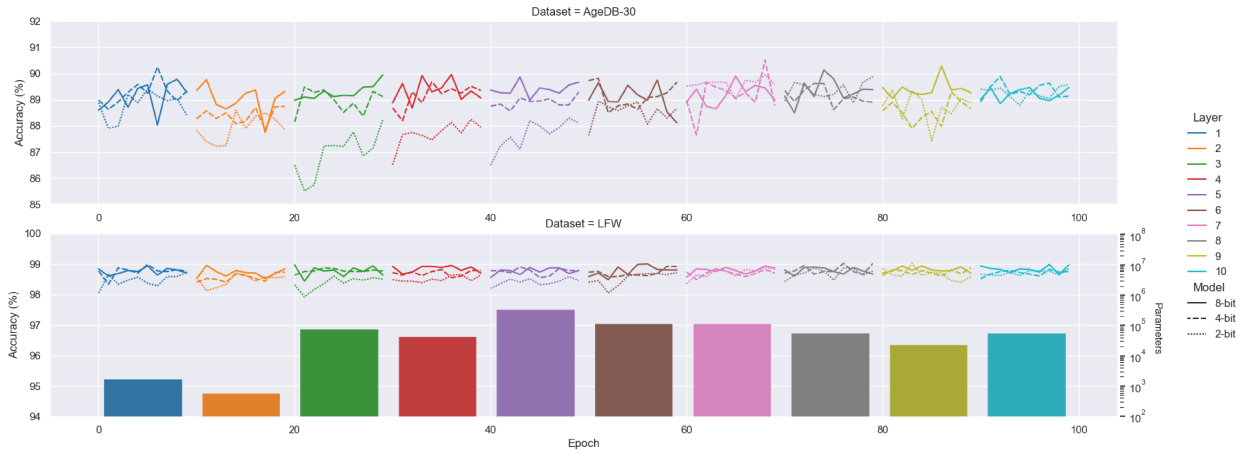


Figure 9: The accuracy of 8, 4 and 2-bit QMobileFaceNet with folding by quantizing each layer individually for both the LFW and the AgeDB-30 dataset. In bottom plot also shows the number of parameters per layer on a logarithmic scale

accuracy loss of 0.22 %p using the LFW dataset and 2.18 %p using the AgeDB-30 dataset.

3) Mixed precision quantization

The results for the mixed-precision quantization can be found in Figure 10. This figure visualizes the trade-off between the number of bits required for the weight representation versus the accuracy of the LFW and AgeDB-30 datasets. These results are compared with (all) the uniform QAT results from Table IV.

Figure 10: The total weight bits versus accuracy trade-off for the LFW and AgeDB-30 datasets.

C. Relative Performance

The performance of the quantization strategies is summarized in Table V. To place the performance within relevant literature, results from other quantization methods applied on MobileNetV2 are listed.

D. Analysis on the quantization of the Face Feature Vector

To analyse the effect of the quantization of the feature vector on the performance of the face recognition system two small experiments have been done. The first experiment was based on the discriminability between different faces. For this experiment, the angle is calculated between two feature vectors from faces found in the MS1M-V2 dataset.

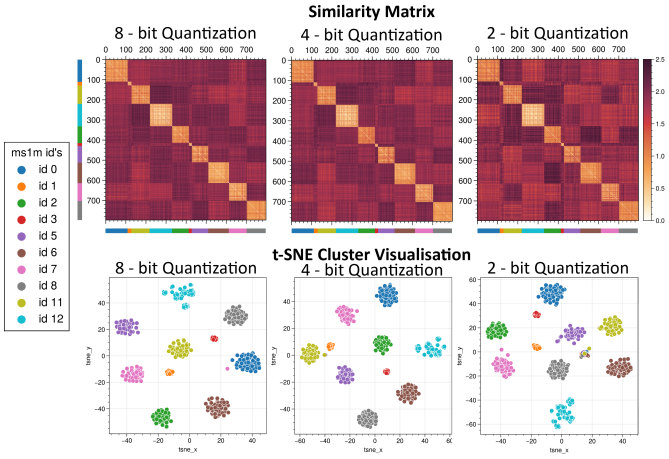


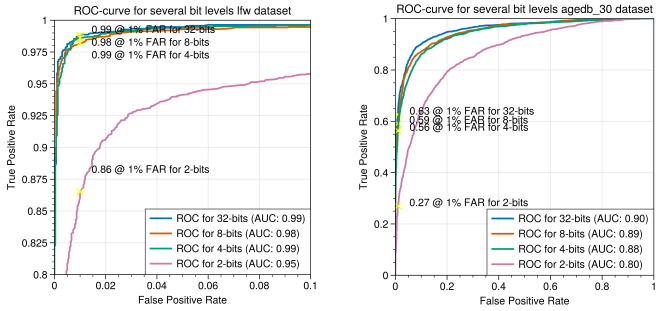
Figure 11: (top-row) The resulting similarity matrices of the 8, 4 and 2-bit quantized last layers. The similarity is defined as the angle between the vectors with a lower value the better. (bottom-row) The visualisation of the ability to cluster the identities using the t-SNE algorithm.

The similarity matrices of the angle between the feature vectors for 10 subjects can be seen in the top row of Figure 11. Between the similarity matrices of 8 and 4-bits, there are not many differences other than it seems that the 4-bit matrix appears to be slightly less red in the non-mated regions compared to the 8-bit equivalent. The t-SNE clustering (in the bottom row of the figure) shows that the 8-bit and 4-bit quantized networks generate vectors that can easily be clustered and that there is good segregation between the mated and non-mated vectors.

The 2-bit similarity matrix, however, looks visibly different. It contains patches of non-mated vectors that have a much larger angle between each other, resulting in a deep red colour. In the mated vectors, there are a lot more vectors that also have a larger angle, which could result in mismatches. This chance for mismatches is visible in the t-SNE plot for the 2-bit network around (18, 0). Here a cluster of non-mated vectors can be observed as shown by many dots of different colours.

Table V: The results of quantizing MobileFaceNets compared to other works on MobileNetV2. The accuracy loss is described in percent point (%p) and relative percentage loss (%)

		Performance			
Architecture	**Representation**	**LFW**	**Accuracy Loss**	**AgeDB-30**	**Accuracy Loss**
MobileFaceNet (ReLU)[4]	Float-32	99.15%		92.93%	
MobileFaceNet (Self)	Float-32	98.85%		90.38%	
QMobileFaceNet (PTQ)	Fixed point-8	94.65%	4.20 %p (4.25%)	73.42%	16.96 %p (18.77%)
QMobileFaceNet (QAT)	Fixed point-8	98.68%	0.17 %p (0.17%)	88.79%	1.59 %p (1.76%)
QMobileFaceNet (QAT)	Fixed point-4	98.63%	0.22 %p (0.22%)	88.20%	2.18 %p (2.41%)
QMobileFaceNet (QAT)	Fixed point-2	93.45%	5.40 %p (5.46%)	75.62%	14.76 %p (16.33%)
		Top-1	**Accuracy Loss**		
Mobilenet-v2-1-224[39]	Float-32	71.90%			
Mobilenet-v2-1-224[39] (TFLite) (PTQ)	Fixed point-8	63.70%	8.20 %p (11.40%)		
Mobilenet-v2-1-224[39] (TFLite) (QAT)	Fixed point-8	71.26%	0.64 %p (0.89%)		
MobileNetV2[42]	Fixed point-16	71.40%	0.50 %p (0.70%)		
MobileNetV2[42]	Fixed point-8	70.18%	1.72 %p (2.39%)		
MobileNetV2[27]	Fixed point-8	71.10%	0.80 %p (1.11%)		
MobileNetV2[27]	Fixed point-4	64.80%	7.10 %p (9.87%)		
MobileNetV2+[30]	Float-32	71.94%			
MobileNetV2+[30]	Fixed point-8	72.35%	-0.41 %p (-0.57%)		
MobileNetV2+[30]	Fixed point-4	71.56%	0.38 %p (0.53%)		



(a) ROC LFW Dataset **(b)** ROC AgeDB-30 Dataset

Figure 12: The ROC curves for several bit lengths. For each curve the location of the False Acceptance Rate (same as FPR) at 1% is highlighted. In the legend also each area under the curve can be found.

The second experiment shows the ROC curve for the LFW and AgeDB-30 datasets that determine the accuracy scores of the QAT networks found in Table IV. The ROC curve of the LFW dataset has been zoomed in for an FPR of max 10% for visibility. The full graph can be found in the Figure A.2.

VI. Discussion

The preliminary study in Section III-A already demonstrated the potential for quantization using the QKeras framework. This section will discuss the results of the quantization of MobileFaceNet that can be found in Section V.

In order to enable the quantization of MobileFaceNet, a pre-trained version of MobileFaceNet is required. Unfortunately, an official implementation using Tensorflow 2.X was not available and had to be built and trained from scratch. This did result in some accuracy differences. The MobileFaceNet network itself was built using standard Tensorflow layers, however, the ArcLoss loss function had to be implemented using a custom function. Additionally, the dataset used for training (CASIA-WebFace) also had to be aligned and cropped. This was first done using the MTCNN algorithm but the newly aligned dataset contained relatively many miss-aligned images after manual inspection. For example, most eyes were not correctly aligned when compared to the images from the (properly) aligned evaluation dataset. Finally, the StyleGAN2 alignment algorithm was applied, which yielded better results but the trained network could still not match the original performance.

The fact that the performance of our system and the original version of MobileFaceNet (using ReLU) differs suggests that the implementation used in this work is likely not optimal and resulted in the use of percent points to show the relative performance. The fact that the accuracy using the AgeDB-30 dataset is clearly worse (92.93% versus 90.38%) than on the LFW dataset (99.15% versus 98.85 %) would suggest the network has more difficulty with comparing faces with a 30-year time difference. This would hint that either the network hasn't seen enough different faces or the ArcLoss implementation did not cluster the identities enough. Several implementations of the ArcLoss loss function were tested but did not result in better accuracy.

A. Analysis QMobileFaceNet

The results in Table III show the performance of MobileFaceNet when the weights and activation functions are directly quantized after training. From these results, it can be seen that the automatic scaling for the quantization seems to enable the post-training quantization of 8-bit networks. The PTQ is also improved by removing the split in the DSC. The difference is less significant as described by Sheng et al. [34], which seems to suggest that QKeras is a more quantization-friendly method for DSC layers.

In order to understand the loss in accuracy for the different bit lengths within the network, the results in Figure 9 can be investigated. These show that the second layer and the first four sets of inverted residual blocks

have the most impact on the accuracy of the AgeDB-30 dataset for the 2-bit quantization. What can also be noted is that after some quantize aware training this can be somewhat mitigated. The expectation was that the quantization loss had the most impact at the beginning of the network. The results seem to be in line with the expectation, although it has to be noted that the inverted residual blocks also account for most of the parameters in the network which could also have led to the quantization loss. Another interesting observation is that the 4-bit quantization has a very similar performance as the 8-bit equivalent, whilst in theory only requires half of the bits to describe the weights. This, and the fact that the last four 2-bit quantized layers behave similar to the other bit lengths, is an indication that mixed-precision networks could reduce the size of the network significantly without having to sacrifice the accuracy too much.

Following the previous results, Table IV shows the performance of QMobileFaceNet when the uniformly quantized network is allowed to train for 20 more epochs. This table shows a significant improvement in the 8-bit and 4-bit networks, only a few percent points lower than the original 32-bit floating-point network. Especially the results of the 4-bit network are noteworthy, as it is only 12.5% of the original 32-bit weight footprint. The 2-bit network also sees an improvement after QAT, but clearly has to sacrifice accuracy for the smaller weight representation.

The third quantization experiment investigates the effect on performance if the network is not limited to one bit-length. These results can be found in Figure 10 and show a clear drop-off around the 3 million bits. The network just before this drop-off with 3.1 million weight bits achieved 98.17% and 87.37% on the LWF and AgeDB-30 dataset respectively. This network had a configuration where the first two layers were quantized using 8-bit quantization, layer 3–5 using 4-bit quantization and layer 6–10 using 2-bit quantization. This is 10% of the footprint of the original 32-bit equivalent with an accuracy loss of only 0.68 %p on LFW and 3.01 %p on the AgeDB-30 dataset. These results indicate new possibilities for implementations on edge devices that only have a limited amount of storage available. A major downside is, however, that the system has to support mixed-precision architectures, which could bring a toll on calculation complexity. For future research, it is interesting to investigate the effect of implementing similar mixed-precision networks on the latency in mobile and edge devices.

Another interesting observation is the fact that the accuracy drops significantly (for both datasets) when the fifth layer is quantized using 2-bits, which is also in line with the observation during the individual layer quantization. The fifth layer is also the largest layer in the entire network, and it might be interesting to investigate the effect of dividing this layer into smaller blocks that are quantized separately. This was not considered in this work as it significantly increased the training time, but is recommended for further work.

The performance of this work is compared to other work in Table V. To overcome the fact that no other work on the quantization of MobileFaceNet was found, the results are compared to other quantization research using MobileNetV2. An attempt was made to also quantize MobileNetV2 using QKeras, but was aborted due to some problems. One of these problems was the fact that continuing training with pre-trained weights (found on the Tensorflow website[39]) only decreased the accuracy. Another problem was that the default implementation did not seem to quantize out of the box as PTQ, whole model QAT and layer for layer QAT resulted in sub 1% accuracies. For both training options, the accuracy seem to drop to almost 0% halfway during training such that it could not be recovered. Several attempts with different learning rates were made but did not yield any improvements. The implementation changes made to MobileFaceNet have the potential to improve MobileNetV2 but would have required a new training session of the whole model from scratch and were considered out of the scope of this work. The network is also larger than MobileFaceNet, as seen in Table A.1 and would have required more resources to train. It was decided to abort the attempt of quantization of MobileNetV2 and focus on improving the performance of QMobileFaceNet.

The comparison between the quantization results of QMobileFaceNet and other quantization methods on MobileNetV2 shows that the relative performance is very similar. The relative accuracy loss of the 8-bit and 4-bit network of QMobileFaceNet on the LFW dataset seems to be less than documented by TensorFlow [39], Wu and Huang [42] and Nagel et al. [27] for MobileNetV2. On the other hand, the relative accuracy loss for the AgeDB-30 dataset is a bit higher than documented by TensorFlow [39]. Unfortunately, this comparison cannot be done quantitatively as the network is not the same and MobileNetV2 is a classifier and MobileFaceNet is a face recognition network. What can be observed, however, is that the quantized versions of MobileNetV2 by Nagel et al. [27] show quite some degradation in accuracy for the 4-bit quantized network which is larger than the relative performance loss of our 4-bit version. The PROFIT quantization method by Park and Yoo [30] seems to have the best relative performance but implemented a more elaborate quantization algorithm (compared to our method) that could be considered for further work. The conclusion that can be drawn from this comparison is that the 8-bit quantization method described in this paper shows no noteworthy differences

in performance degradation compared to similar quantization methods. The performance degradation of the 4-bit QMobileFaceNet does seem to be less severe than described by Nagel et al. [27].

B. Analysis Face Feature Vector

The feature vector experiments on the 8, 4 and 2-bit quantization of QMobileFaceNet evaluated the similarity between feature vectors of ten identities, the ability to cluster the vectors of these identities and how the quantization affected the threshold selection for different datasets. The first experiment was designed to visualize the discriminability between mated and non-mated feature vectors. As shown in Figure 11, this qualitative approach shows only minor differences between 8 and 4-bit quantization. The only observation is the fact that the angles between the non-mated identities of the 4-bit network seem less extreme than for the 8-bit network, which might result in a lower threshold and thus less room for variability within one identity. As this difference is only trivial, it suggests that the 4-bit network is likely similar in performance compared to the 8-bit network and did not require notable changes in the network during the training. The 2-bit quantized network, however, shows different results. The fact that the similarity matrix looks visibly different, suggests that this network did undergo considerable changes during training to support the reduction in the solution space. Also, the 2-bit similarity matrix of Figure 11 shows dark red lines within the mated identity squares. This suggests that the 2-bit network does encounter problems with the discriminability between the vectors and is not a recommended network for a face recognition system.

To show the effect of the quantization on the trade-off between the True Positive Rate and the False Positive Rate, the ROC curves can be found in Figure 12. Similar to the results in the first experiment, the curves between 32-bit, 8-bit and 4-bit networks are very similar. Interestingly enough, for the LFW dataset, the 4-bit network seems to work better than the 8-bit network with an FPR larger than 0.5%. This is not the case for the AgeDB-30 dataset and seems to suggest that this is a marginal effect and does not mean that the 4-bit network outperforms the 8-bit network. The 2-bit network performs visibly worse than the other networks. The ROC curve for the LFW dataset still has a comparable AUC to the other networks, which would suggest still a moderate performance. For the AgeDB-30 dataset, however, the curve only converges to almost 100% with an FPR higher than 90%. This is also in line with findings of the first experiment on the 2-bit network that it clearly underperforms and should not be considered for a face recognition system.

C. Future Work

QKeras simulates the network as if it was a fixed-point number but in reality, it is still implemented as a 32-bit floating-point number. For further work, it can be strongly suggested to implement the proposed networks as it was a fixed point integer and also test inference on a mobile device to obtain the inference. Also, as discussed earlier in this section, the performance of MobileFaceNet was not the same as shown in the original paper. For further work, we would suggest trying to investigate this issue. Implementations using PyTorch have shown great accuracy and could be a good start.

Another point that can be investigated is by using a larger face dataset to train the network. In this work the CASIA-WebFace dataset is used, however, the performance of this work can not match the results in the original MobileFaceNet paper[4]. That paper also shows that using the MS1M-V2 dataset[6] can also improve the accuracy of the system (3%p on the AgeDB-30 dataset). The dataset is almost $10\times$ larger so will also increase the training time. This work did not consider binary and logarithmic quantization, but are worth investigating to also optimize computational complexity.

VII. Conclusion

Several quantization methods are proposed to quantize a Face Recognition System in order to give an answer to the original research question: *What are methodologies that reduce the footprint of a mobile face recognition network architecture using sub-byte quantization and what is their impact on the accuracy of the network?*. In order to answer this question, the problem was divided into multiple smaller sub-questions.

First, this research was placed within related work which primarily shows that although literature discusses mobile neural network architectures for face recognition systems, no research on quantization can be found for face recognition systems. Fortunately, this subject is populair for other neural networks such as MobileNet. The mobile face recognition network MobileFaceNet is based on MobileNet and is the most suitable to compare quantization performance results with. A study of network compression for neural networks shows that integer-quantization can produce an int-8 version of MobileNet with a 0.64 percent point drop in accuracy whilst producing a model which is almost $4\times$ smaller.

The next question pondered the effect of sub-byte quantization for the individual layers that could be found in MobileFaceNet. To enable the experimentation with sub-byte quantization, the QKeras framework is used. However, there were several parameters that could be chosen, such as the folding of the batch normalization layer, the position of the decimal point and the scaling factor. So in order to answer this question, a preliminary

study has been done to analyse the QKeras quantization loss. This study then investigated the effect of several parameters for each of these layers. Here the conclusion was drawn that the folding and full fractional bit representation were the best options to be used for the sub-byte quantization of MobileFaceNet. The experiments using the scaling factor and the split of the DSC yielded no significant conclusion and were tested again with the quantization of the whole model.

The third question delved into the problem of the quantization strategy. This problem was dealt with by defining several options to quantize the network. First, Post Training Quantization showed that by removing the batch normalization and ReLU after the depthwise convolution in the DSC and choosing an automatic scaling factor the quantization resulted in the highest quantization scores. Secondly, this version of QMobileFaceNet was then fine-tuned using Quantize Aware Training which showed an impressive improvement for the 8-bit and 4-bit networks, effectively showing that a network with 12.5% of the original model footprint (concerning the weights) can score a mean accuracy of 98.63% for the LFW dataset and 88.20% for the AgeDB-30 dataset, which is only 0.22 and 2.18 %p lower than the floating-point version. Using the mixed-precision experiment is also shown that, in theory, the network can reduce the footprint down to 10% of the floating-point equivalent with an accuracy loss of only 0.68 %p and 3.01 %p on the LFW and AgeDB-30 dataset respectively.

The fact that there is no literature found on the quantization of face recognition systems, this work formed the last research question investigating the effect of uniform quantization on the discriminability between several identities. These experiments showed that 8-bit and 4-bit quantization both had a minimal toll on the discriminability of different identities. 2-bit quantization, however, did show problems identifying mated feature vectors and should not be considered for a small face recognition system.

Considering all, to answer the main research question, this work shows several methods to quantize MobileFaceNet using the QKeras framework in Tensorflow. This works shows that it is possible to quantize a face recognition system using 8 and 4 bits for the weight representation and activation function with a 0.17 and 0.22 %p loss in accuracy using the LFW evaluation dataset. The research also shows that using mixed precision the size of the footprint can be reduced further. The results of this work suggest that sub-byte and mixed-precision is a good method to reduce the size of a face recognition system without losing performance and can thus be used as a basis for further work that has hardware requirements for actual implementation.

ACKNOWLEDGEMENTS

I would like to show my appreciation to all of my peers who have supported me throughout my thesis. I would like to thank them for all the daily update meetings and useful sparring sessions. I would also like to thank my daily supervisor for the interesting discussions that kept me sharp and motivated for this work.

REFERENCES

[1] Claudionor N. Coelho Jr. au2 et al. *Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors*. 2021. eprint: 2006.10159.

[2] Fadi Boutros et al. “MixFaceNets: Extremely Efficient Face Recognition Networks”. In: *2021 IEEE International Joint Conference on Biometrics (IJCB)*. 2021, pp. 1–8. DOI: 10.1109/IJCB52358.2021.9484374.

[3] Alessandro Capotondi et al. “CMix-NN: Mixed Low-Precision CNN Library for Memory-Constrained Edge Devices”. In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 67.5 (2020), pp. 871–875. DOI: 10.1109/TCSII.2020.2983648.

[4] Sheng Chen et al. “MobileFaceNets: Efficient CNNs for Accurate Real-Time Face Verification on Mobile Devices”. In: *Biometric Recognition*. 2018, pp. 428–438. DOI: 10.1007/978-3-319-97909-0_46.

[5] Claudionor Coelho. *Google/QKeras*. 2019. URL: https://github.com/google/qkeras (visited on 03/02/2022).

[6] Jiankang Deng, Jia Guo, and Stefanos Zafeiriou. “ArcFace: Additive Angular Margin Loss for Deep Face Recognition”. In: *CoRR* abs/1801.07698 (2018). arXiv: 1801.07698. URL: http://arxiv.org/abs/1801.07698.

[7] Li Deng. “The mnist database of handwritten digit images for machine learning research”. In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[8] The Economist. *From not working to neural networking The artificial-intelligence boom is based on an old idea, but with a modern twist*. Accessed on 21-12-2021. URL: https://www.economist.com/special-report/2016/06/23/from-not-working-to-neural-networking.

[9] Angelo Garofalo et al. “XpulpNN: Accelerating Quantized Neural Networks on RISC-V Processors Through ISA Extensions”. In: *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*. 2020, pp. 186–191. DOI: 10.23919/DATE48585.2020.9116529.

[10] Philipp Gysel, Mohammad Motamedi, and Soheil Ghiasi. "Hardware-oriented Approximation of Convolutional Neural Networks". In: *CoRR* abs/1604.03168 (2016). arXiv: 1604.03168. URL: http://arxiv.org/abs/1604.03168.

[11] Song Han et al. "Learning both Weights and Connections for Efficient Neural Networks". In: *CoRR* abs/1506.02626 (2015). arXiv: 1506.02626. URL: http://arxiv.org/abs/1506.02626.

[12] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[13] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. *Distilling the Knowledge in a Neural Network*. 2015. arXiv: 1503.02531 [stat.ML].

[14] Andrew G. Howard et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". In: *CoRR* abs/1704.04861 (2017). arXiv: 1704.04861. URL: http://arxiv.org/abs/1704.04861.

[15] Gary Huang et al. "Labeled Faces in the Wild: A Database forStudying Face Recognition in Unconstrained Environments". In: *Tech. rep.* (Oct. 2008).

[16] Nathan Hubens. *Speed-up inference with Batch Normalization Folding*. 2020. URL: https://towardsdatascience.com/speed-up-inference-with-batch-normalization-folding-8a45a83a89d8 (visited on 03/07/2022).

[17] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *CoRR* abs/1502.03167 (2015). arXiv: 1502.03167. URL: http://arxiv.org/abs/1502.03167.

[18] Benoit Jacob et al. "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2704–2713. DOI: 10.1109/CVPR.2018.00286.

[19] Tero Karras et al. "Analyzing and Improving the Image Quality of StyleGAN". In: *Proc. CVPR*. 2020.

[20] Ningning Ma et al. "ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design". In: *Computer Vision – ECCV 2018*. Ed. by Vittorio Ferrari et al. Cham: Springer International Publishing, 2018, pp. 122–138. ISBN: 978-3-030-01264-9.

[21] Laurens van der Maaten and Geoffrey Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9.86 (2008), pp. 2579–2605. URL: http://jmlr.org/papers/v9/vandermaaten08a.html.

[22] Tambiama Madiega. *Artificial intelligence act [EU Legislation in Progress][Policy Podcast]*. Accessed on 16-12-2021. URL: https://epthinktank.eu/2021/11/18/artificial-intelligence-act-eu-legislation-in-progress/.

[23] Tambiama Madiega and Hendrik Mildebrath. "Regulating f acial recognition in the EU". In: *Members' Research Service* (2021). Accessed on 16-12-2021. DOI: 10.2861/140928. URL: https://www.europarl.europa.eu/RegData/etudes/IDAN/2021/698021/EPRS_IDA(2021)698021_EN.pdf.

[24] Yoanna Martindez-Díaz et al. "ShuffleFaceNet: A Lightweight Face Architecture for Efficient and Highly-Accurate Face Recognition". In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. 2019, pp. 2721–2728. DOI: 10.1109/ICCVW.2019.00333.

[25] Yoanna Martínez-Díaz et al. "Benchmarking lightweight face architectures on specific face recognition scenarios". In: *Artificial Intelligence Review* (Feb. 2021). DOI: 10.1007/s10462-021-09974-2.

[26] Stylianos Moschoglou et al. "AgeDB: The First Manually Collected, In-the-Wild Age Database". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2017, pp. 1997–2005. DOI: 10.1109/CVPRW.2017.250.

[27] Markus Nagel et al. "A White Paper on Neural Network Quantization". In: *CoRR* abs/2106.08295 (2021). arXiv: 2106.08295. URL: https://arxiv.org/abs/2106.08295.

[28] Gianmarco Ottavi et al. "A Mixed-Precision RISC-V Processor for Extreme-Edge DNN Inference". In: *2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. 2020, pp. 512–517. DOI: 10.1109/ISVLSI49217.2020.000-5.

[29] Alessandro Pappalardo. *Xilinx/brevitas*. 2021. DOI: 10.5281/zenodo.3333552. URL: https://doi.org/10.5281/zenodo.3333552 (visited on 03/02/2022).

[30] Eunhyeok Park and Sungjoo Yoo. "PROFIT: A Novel Training Method for sub-4-bit MobileNet Models". In: *CoRR* abs/2008.04693 (2020). arXiv: 2008.04693. URL: https://arxiv.org/abs/2008.04693.

[31] Manuele Rusci, Alessandro Capotondi, and Luca Benini. "Memory-Driven Mixed Low Precision Quantization For Enabling Deep Network Inference On Microcontrollers". In: *CoRR* abs/1905.13082 (2019). arXiv: 1905.13082. URL: http://arxiv.org/abs/1905.13082.

[32] Mark Sandler et al. "MobileNetV2: Inverted Residuals and Linear Bottlenecks". In: *2018 IEEE/CVF Conference on Computer Vision and*

Pattern Recognition. 2018, pp. 4510–4520. DOI: 10.1109/CVPR.2018.00474.

[33] Shibani Santurkar et al. *How Does Batch Normalization Help Optimization?* 2019. arXiv: 1805.11604 [stat.ML].

[34] Tao Sheng et al. "A Quantization-Friendly Separable Convolution for MobileNets". In: *CoRR* abs/1803.08607 (2018). arXiv: 1803.08607. URL: http://arxiv.org/abs/1803.08607.

[35] Weidong Shi et al. "ProxylessKD: Direct Knowledge Distillation with Inherited Classifier for Face Recognition". In: *CoRR* abs/2011.00265 (2020). arXiv: 2011.00265. URL: https://arxiv.org/abs/2011.00265.

[36] Vivienne Sze et al. "Efficient Processing of Deep Neural Networks: A Tutorial and Survey". In: *Proceedings of the IEEE* 105.12 (2017), pp. 2295–2329. DOI: 10.1109/JPROC.2017.2761740.

[37] Mingxing Tan and Quoc V. Le. "MixConv: Mixed Depthwise Convolutional Kernels". In: *CoRR* abs/1907.09595 (2019). arXiv: 1907.09595. URL: http://arxiv.org/abs/1907.09595.

[38] TensorFlow. *TensorFlow Lite*. 2021. URL: https://www.tensorflow.org/lite/guide (visited on 03/02/2022).

[39] TensorFlow. *TensorFlow model optimization*. 2021. URL: https://www.tensorflow.org/model_optimization/guide (visited on 03/02/2022).

[40] Cheng-Hao Tu et al. "Pruning Depthwise Separable Convolutions for MobileNet Compression". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. 2020, pp. 1–8. DOI: 10.1109/IJCNN48605.2020.9207259.

[41] CDA en ChristenUnie VVD D66. *Coalitieakkoord 2021 – 2025*. Accessed on 16-12-2021. URL: https://www.kabinetsformatie2021.nl/binaries/kabinetsformatie/documenten/publicaties/2021/12/15/coalitieakkoord-omzien-naar-elkaar-vooruitkijken-naar-de-toekomst/coalitieakkoord-2021-2025.pdf.

[42] Yueh-Chi Wu and Chih- Tsun Huang. "Efficient Dynamic Fixed-Point Quantization of CNN Inference Accelerators for Edge Devices". In: *2019 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*. 2019, pp. 1–4. DOI: 10.1109/VLSI-DAT.2019.8742040.

[43] Han Xiao, Kashif Rasul, and Roland Vollgraf. "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms". In: *CoRR* abs/1708.07747 (2017). arXiv: 1708.07747. URL: http://arxiv.org/abs/1708.07747.

[44] Tien-Ju Yang, Yu-Hsin Chen, and Vivienne Sze. "Designing Energy-Efficient Convolutional Neural Networks Using Energy-Aware Pruning". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6071–6079. DOI: 10.1109/CVPR.2017.643.

[45] Dong Yi et al. "Learning Face Representation from Scratch". In: *CoRR* abs/1411.7923 (2014). arXiv: 1411.7923. URL: http://arxiv.org/abs/1411.7923.

[46] Arm Ltd Yuan Tang Google LLC. *tensorflow/tflite-micro*. 2021. URL: https://github.com/tensorflow/tflite-micro (visited on 03/14/2022).

[47] Stone Yun and Alexander Wong. "Do All MobileNets Quantize Poorly? Gaining Insights into the Effect of Quantization on Depthwise Separable Convolutional Networks Through the Eyes of Multi-scale Distributional Dynamics". In: *CoRR* abs/2104.11849 (2021). arXiv: 2104.11849. URL: https://arxiv.org/abs/2104.11849.

[48] Kaipeng Zhang et al. "Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks". In: *CoRR* abs/1604.02878 (2016). arXiv: 1604.02878. URL: http://arxiv.org/abs/1604.02878.

[49] Min Zhang et al. "Optimized Compression for Implementing Convolutional Neural Networks on FPGA". In: *Electronics* 8.3 (2019). ISSN: 2079-9292. DOI: 10.3390/electronics8030295. URL: https://www.mdpi.com/2079-9292/8/3/295.

[50] Xiangyu Zhang et al. *ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices*. 2017. arXiv: 1707.01083 [cs.CV].

[51] Hengrui Zhao, Dong Liu, and Houqiang Li. "Efficient Integer-Arithmetic-Only Convolutional Networks with Bounded ReLU". In: *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2021, pp. 1–5. DOI: 10.1109/ISCAS51556.2021.9401448.

APPENDIX

A. MobileNetV2

MobileNetV2[32] is the basis for MobileFaceNet. The network architecture is shown in table A.1.

B. MobileFaceNet Network Elements

1) Depthwise-Separable convolution

The basis for MobileFaceNet (and its predecessor MobileNet) is the introduction of the notion of a Depthwise Separable Convolutional layer (DSC-layer). Whereas conventional convolutional layers calculate an output based on the whole input feature map, DSC aims to separate the process to reduce the amount of multiplications without having the need for linearly separable filter kernels. Conventional convolutional layers calculate the output pixel value by multiplying the $k^2 \cdot C$, where k^2 represents the number of filter (with size k) multiplications and C is the number of input channels. Usually

Table A.1: The network architecture of MobileNetV2.

Architecture MobileNetV2[32]						
Layer	**Input**	**Operator**	**t**	**c**	**n**	**s**
1	$224^2 \times 3$	conv3x3		32	1	2
2	$112^2 \times 32$	bottleneck	1	16	1	1
3	$112^2 \times 16$	bottleneck	6	24	2	2
4	$56^2 \times 24$	bottleneck	6	32	3	2
5	$28^2 \times 32$	bottleneck	6	64	4	2
6	$14^2 \times 64$	bottleneck	6	96	3	1
7	$14^2 \times 96$	bottleneck	6	160	3	2
8	$7^2 \times 160$	bottleneck	6	320	1	1
9	$7^2 \times 320$	conv1x1		1280	1	1
10	$7^2 \times 1280$	avgpool7x7			1	
11	$1 \times 1 \times 1280$	conv 1x1		k	1	1

a convolutional layer contains an N number of filter kernels resulting in the following number of required multiplications given the input H height and W width of an input feature map:

$$MUL_{conv2D} = H \cdot W \cdot C \cdot N \cdot k^2 \tag{4}$$

A DSC-layer separates the image convolution from the channel-wise convolution. These are named Depthwise and Pointwise convolutions accordingly. Depthwise convolutions apply a convolutional filter only over the Height and Width of the input, resulting in C number of output feature map. Pointwise convolutions apply an N number amount of 1x1 convolutional operations over the input channels. The combination of both operations results in the following number of required multiplications, which can be significantly less than a conventional convolutional layer:

$$MUL_{DSC} = H \cdot W \cdot C \cdot (N + k^2) \tag{5}$$

2) Inverted Residual Bottleneck

MobileNetV2 introduced us to Inverted Residual Bottlenecks[32], a concept that combines DSC-layers with a residual connection to improve the ability of a gradient to propagate through multiple layers according to the authors. The inverted bottleneck refers to the fact that the dimension of the internal convolution block is first expanded with an expansion ratio t, opposed to reducing the dimensions inside the bottleneck (hence the name). The residual connection means that the input is added to the output given the dimensions match. The diagram of this block can be found in figure A.1.

3) Global Depthwise Convolution

The authors of MobileFaceNet[4] claim that due to the fact that the network is trained on a set aligned faces, the network does not benefit from the use of conventional Global Average Pooling that treats every feature map the same. The reasoning behind this is the fact that often the corners of the image (should) have a significant smaller impact on classification than the center of the image.

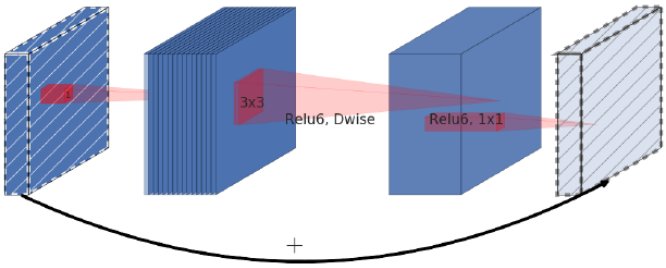


Figure A.1: Inverted Residual Bottleneck Block as described in MobileNetV2[32]

Instead they propose the use of a Global Depthwise Convolutional (GDC) layer that sets the kernel size equal to the input feature map. This method increased the accuracy with 0.3% (LFW) and 1.86% (AgeDB-30), whilst only adding 25k parameters. Finally, the authors also implement a pointwise convolution to reduce the output feature vector to 128 parameters as an alternative to a fully connected layer, which reduces the total number with 8 million. The final MobileFaceNet network only requires 0.99 million parameters.

4) Batch Normalisation

It is common knowledge that batch normalization[17] improves the performances of deep neural networks, at as such widely used (also in e.g. MobileNet and MobileFaceNets). However the reasons why are still highly debated. The original authors[17] claim that the effectiveness is due to the reduction of internal covariate shift (the shift of weights during training). However this seems to be disproven by Santurkar et al. [33], which they claim that the effectiveness comes due to the impact of batch normalization on optimization landscape smoothness (resulting in smoother gradients and thus faster and better convergence). Santurkar et al. even claim these properties are not even limited to batch normalization and can even be achieved using other normalization methods. Nevertheless, to understand the Mobile(Face)Net architecture, the batch normalisation should not be glanced over. The algorithm is described below:

Algorithm 1 Algorithm for Batch Normalization[17]

Input: Values x in batch: $B = \{x_1..m\}$

1. $\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$
2. $\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$
3. $\hat{x_i} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$ (ϵ is a constant for numerical stability)
4. $y_i = \gamma\hat{x_i} + \beta$

Output: $\mathrm{BN}_{\gamma,\beta}(x_i)$

C. Mixed Precision parameters

D. Full ROC curve of the LFW dataset

Table A.2: The mixed precision models and the number of bits for the weights they require.

	Layer 1	**Layer 2**	**Layer 3**	**Layer 4**	**Layer 5**	**Layer 6**	**Layer 7**	**Layer 8**	**Layer 9**	**Layer 10**	**Parameters**
Parameters	*1985*	*576*	*90245*	*52481*	*413190*	*137729*	*137730*	*67585*	*25088*	*66049*	992658
32-bit	32	32	32	32	32	32	32	32	32	32	**Total bits**
Weight bits	*63520*	*18432*	*2887840*	*1679392*	*13222080*	*4407328*	*4407360*	*2162720*	*802816*	*2113568*	31765056
8-bit	8	8	8	8	8	8	8	8	8	8	
Weight bits	*15880*	*4608*	*721960*	*419848*	*3305520*	*1101832*	*1101840*	*540680*	*200704*	*528392*	7941264
4-bit	4	4	4	4	4	4	4	4	4	4	
Weight bits	*7940*	*2304*	*360980*	*209924*	*1652760*	*550916*	*550920*	*270340*	*100352*	*264196*	3970632
2-bit	8	2	2	2	2	2	2	2	2	2	
Weight bits	*15880*	*1152*	*180490*	*104962*	*826380*	*275458*	*275460*	*135170*	*50176*	*132098*	1997226
Mixed 1	8	8	4	4	4	4	4	2	2	2	
Weight bits	*15880*	*4608*	*360980*	*209924*	*1652760*	*550916*	*550920*	*135170*	*50176*	*132098*	3663432
Mixed 2	8	8	4	4	4	4	2	2	2	2	
Weight bits	*15880*	*4608*	*360980*	*209924*	*1652760*	*550916*	*275460*	*135170*	*50176*	*132098*	3387972
Mixed 3	8	8	4	4	4	2	2	2	2	2	
Weight bits	*15880*	*4608*	*360980*	*209924*	*1652760*	*275458*	*275460*	*135170*	*50176*	*132098*	3112514
Mixed 4	8	8	4	4	2	2	2	2	2	2	
Weight bits	*15880*	*4608*	*360980*	*209924*	*826380*	*275458*	*275460*	*135170*	*50176*	*132098*	2286134
Mixed 5	8	8	4	2	2	2	2	2	2	2	
Weight bits	*15880*	*4608*	*360980*	*104962*	*826380*	*275458*	*275460*	*135170*	*50176*	*132098*	2181172
Mixed 6	8	8	2	2	2	2	2	2	2	2	
Weight bits	*15880*	*4608*	*180490*	*104962*	*826380*	*275458*	*275460*	*135170*	*50176*	*132098*	2000682

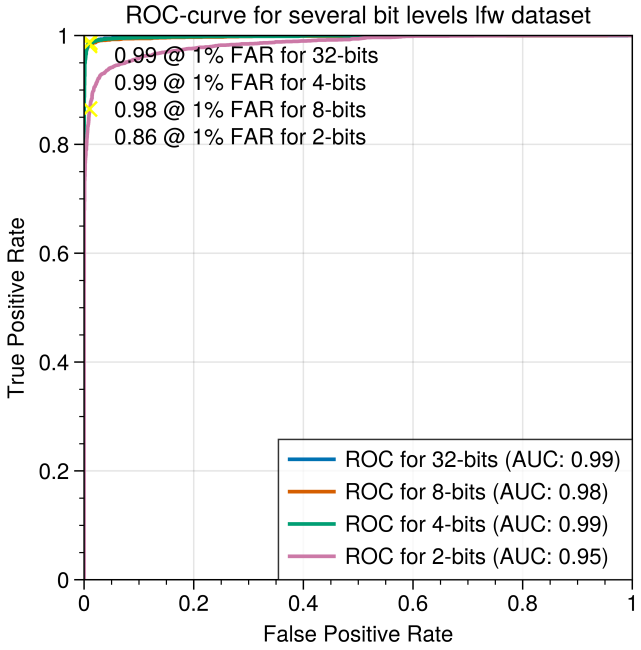


Figure A.2: The zoomed out graph of the ROC curve of the LFW Dataset