TPDNet: A Tiny Pupil Detection Neural Network for Embedded Machine Learning Processor Arm Ethos-U55

Gernot Fiala[1,2], Zhenyu Ye[2], and Christian Steger[1]

[1] Graz University of Technology, Institute of Technical Informatics, Graz, Austria, {gernot.fiala, steger}@tugraz.at
[2] ams-OSRAM AG, INO Innovation Office Semiconductors, Premstaetten, Austria, {gernot.fiala, zhenyu.ye}@ams-osram.com

Abstract. Augmented reality and virtual reality (AR/VR) systems contain several different sensors including images sensors for gesture recognition, head pose tracking and pupil/eye tracking. The data of all these sensors must be processed by a host processor in real-time. For future AR/VR systems, new sensing technologies are required to fulfill the demands in power consumption and performance. Currently pupil detection is performed with images on resolutions around 300x300 pixels and above. Therefore, deep neural networks (DNN) need host platforms, which are capable to compute the DNNs with such input resolutions to process them in real-time. In this work, the image resolution for pupil detection is optimized to a resolution of 100x100 pixels. A tiny pupil detection neural network is introduced, which can be processed with the ARM Cortex-M55 and the Embedded Machine Learning (ML) processor Arm Ethos-U55 with a performance of 189 frames per second (FPS) with high detection rates. This allows to reduce the power consumption of the communication between image sensor and host for future AR/VR devices.

Keywords: Arm Ethos-U55, Arm M55, augmented reality, deep neural network, machine learning, neural network processor, pupil detection, virtual reality

1 Introduction

Pupil detection and Eye tracking is widely used for different kinds of applications like eye tracking-based illness detection [1, 2], in-cabin sensing of cars to detect the attention of the drivers and for augmented reality (AR) and virtual reality (VR) systems. In the field of AR/VR systems, there are smart glasses used for teaching, industry [3, 4], gaming [5, 6] and simulation systems [7]. Future devices will use more and more human machine interfaces (HMI) to interact with the environment through different sensors and displays. Hence, more and more processing is required. Some of these systems allow a higher power consumption but for other devices like smart glasses, power consumption is an important factor, which needs to be optimized. It is expected to use stacked configurations for image sensors for future AR/VR devices stated by Chiao Liu et al. in [8]. Furthermore, conventional image sensors, cannot fulfill the requirements [9] for future AR/VR systems. Probably next generation smart glasses will become the next smart phones as stated by Chiao Liu et al. in [9]. Therefore, new sensor technologies and system designs must be developed to lower the power consumption of such AR/VR devices.

One idea to lower the power consumption is, to send less data from the sensor towards the next processing levels as described by Chiao Liu et al. in [9]. For image sensors, the mobile industry processor interface (MIPI) [3] is used to transfer the image sensor data to a host platform. If only region-of-interest (ROI) data or images with lower resolutions are transferred from the image sensor to the host, power consumption of the communication can be saved significantly, by switching the MIPI interface into the sleep mode, shown by Gernot Fiala et al. in [12]. To lower the amount of data transferred to the host, region-of-interest (ROI) of the eye or even the processed data in form of pupil center coordinates can be transmitted to the host. Both ways need some processing steps directly on the image sensor. Machine learning algorithms for pupil detection outperform standard edge-based pupil detection algorithms [13], but the processing demand is higher. Sony developed in 2020 the first intelligent image sensor with integrated processing units in a stacked layer configuration [14, 15]. Mobilenet V1 can be processed directly on the sensor for object classification. Since image sensors for pupil detection or eye tracking in AR/VR systems are required to have a smaller form factor, the resources for processing units and memory are more limited due to less available chip area. Therefore, optimizations of the image processing system are required.

In this paper, the focus is on optimizing pupil detection with an image resolution of 100x100 pixels and a neural network, which can be processed on embedded processors Arm M55 [16] in cooperation with the neural network processing unit (NPU) Arm Ethos-U55 [17] and evaluate the pupil detection rate and execution time. The main contributions of this paper are:

– Introduction of a tiny neural network for pupil detection with an input resolution of 100x100 pixels. A training process to improve the detection rate. The trained neural network is quantized with TensorFlow Lite and can be processed in real-time on embedded processors Arm M55 and Arm Ethos-U55 with 189 FPS and high detection rates.
– An extension of an existing pupil detection dataset, with images generated from an image sensor software model with different illumination powers with sensor specific artifacts and noise levels for a resolution of 100x100 and 200x200 pixels.

This paper is structured as follows: Section II shows related work with pupil detection and optimization methods. Section III introduces the extension of the

[3]https://www.mipi.org/

dataset. Section IV describes the neural network architecture, the training process and quantization. Section V shows the pupil detection results and execution times and Section VI concludes the paper.

2 Related Work

There is a lot of research in the area of pupil detection and eye tracking for AR/VR applications. Several years ago, standard edge-based pupil detection algorithm were introduced by Lech Swirski et al. [19], Wolfgang Fuhl et al. [22], Amir-Homayoun Javadi et al. [24] and many more. More recently neural networks and machine learning algorithms got more popular for pupil detection and eye tracking. With synthetic eye data, the generation of datasets with ground truth data is easier compared to a complete hardware setup. Some of the 3D eye models use Blender [10,20,21,27], Unity3D [25,26] or even the Unreal gaming engine to render images with ground truth data containing either eye segmentation, pupil center coordinates, gaze vector or other eye information. Joohwan Kim et al. highlights in [13], that machine learning algorithms outperform standard edge-based pupil detection algorithms, but these neural networks use different kernel sizes for convolutions and are deep neural networks (DNN). They require more processing power than standard edge-based pupil detection algorithms and the evaluation is mostly done with general purpose processors [22, 23, 28] or graphic processing units (GPUs) [13]. With these hardware platforms, high frame rates can be achieved, but for small AR/VR devices like future smart glasses not suitable.

There is also research to improve the execution time for neural networks as described by Wolfgang Fuhl et al. in [23]. They split the input image into sections and use 2 neural networks to find the pupil center position. One neural network is used for a rough estimation and the other one for a more accurate estimation. It can be processed in 7 ms with a single core and in 2 ms with multi core on an Intel i5-4570. Another algorithm was developed by Thiago Santini et al. described in [28]. With an Intel i5-4590, 120 FPS were reached.

Since the demand for smart or intelligent image sensors is growing, new methods are required to optimize the processing of such neural networks. The most well known framework nowadays is TensorFlow [29] with TensorFlow Lite [30]. It allows a quantization of 32 bit float weights to 8 bit integer values. It lowers the memory footprint of the whole neural network model. Another framework is Apache's TVM [31], a machine learning compiler to optimize the code for a specific hardware architecture. Lately, the research in RISC-V processors and code optimization with instruction set extensions is growing. Such extensions can be single instruction multiple data (SIMD) or even hardware (HW) accelerators for speeding up the processing of convolutions or other neural network related operations.

Conventional eye tracking hardware uses mainly embedded GPUs to process the image frames from the image sensors, like Qualcom Snapdragon [32, 33]. They can process more than 60 frames per second (FPS) with higher resolutions. To make the AR/VR devices even smaller or even go to smart glasses, the focus is more on embedded NPUs or image signal processors (ISP). The research to develop embedded NPUs or ISPs is growing. There are new products on the commercial market, which are optimized with vector instructions and HW-accelerators for neural network processing, like Arm M55 [16], Arm Ethos-U55 [17] or Quadric Chimera [34].

In this work, pupil detection is optimized to an image resolution of 100x100 pixels. A tiny neural network for pupil detection was developed, trained and tested with an extended pupil detection dataset with different illumination powers and noise levels. The trained neural network was quantized with TensorFlow Lite and processed with the embedded processors Arm M55 and Arm Ethos-U55. High frame rates of more than 90 FPS can be achieved by using images with 100x100 pixel resolution. The NPU accelerates the processing of the neural network. This shows, that smaller embedded hardware can be used to process neural networks with good performance. An image resolution of 100x100 pixel is sufficient for pupil detection to get high detection rates.

3 Pupil Detection Dataset

For training the neural networks, large eye datasets are required. Therefore, we used the modified 3D eye model from [10], which is based on [20, 21] to render a dataset of 11000 images for each of the resolutions, 100x100, 200x200 and 500x500 pixels with the pupil center positions as ground truth information. Furthermore, an image sensor model (ISM) was used to generate additional images. The image sensor model contains proprietary information and processes. Therefore, this paper does not explain internals of the ISM. However, the image sensor model can be tuned with different illumination powers and different resolutions for the output image. The generated output image contain camera artifacts and noise. As input for the ISM, the rendered 500x500 pixel dataset was used to generate output images with a resolution of 100x100 and 200x200 pixels shown in Fig. 1. The illumination power is tuned with values of 1, $\frac{1}{4}$, $\frac{1}{16}$ and $\frac{1}{64}$. That means, the illumination power was reduced each time by one quarter of the previous illumination power. Due to the output behaviour of the ISM, the generated images with an illumination power of $\frac{1}{16}$ look brighter compared to images generated with an illumination power of $\frac{1}{64}$. In total, the rendered eye dataset was expanded by additional 44000 images with the previously mentioned illumination powers to a total image number of 55000 images for each of the resolutions 100x100 and 200x200 pixels. These dataset was used to train the neural network.

Example images of the dataset are shown in Fig. 2. On the left are the rendered images and on the right the images generated by the ISM with different illumination powers of 1, $\frac{1}{4}$, $\frac{1}{16}$ and $\frac{1}{64}$. The images with higher illumination power are darker compared to the rendered images. Lower illumination power increases the noise level. For an illumination power of $\frac{1}{16}$, the images are brighter and noisy due to a specific bit readout strategy. A further reduction of the illumination power adds even more noise to the images.

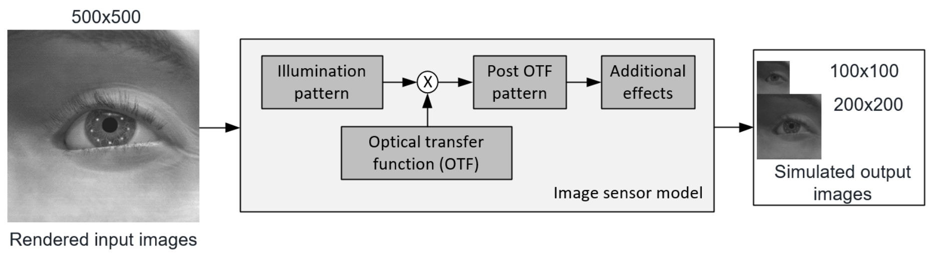

Fig. 1: Generation of images with image sensor software model.

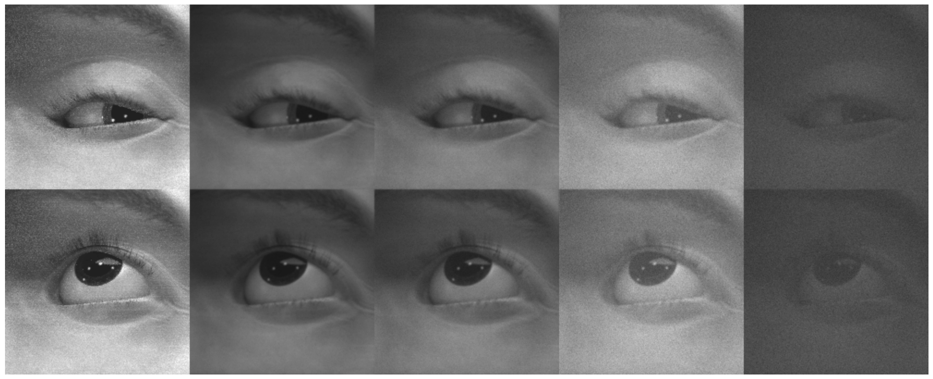

Fig. 2: Example images of the extended dataset. Left is the rendered image, then images with illumination powers of 1, $\frac{1}{4}$, $\frac{1}{16}$ and $\frac{1}{64}$ from left to right.

As separate test data, we got the small dataset generated by Gernot Fiala et al. introduced in [11]. Here, two different image sensor models were used to tune the pixel pitch value, which influences the brightness of the output images. We used again the 500x500 image resolution of the rendered images as input for our image sensor model and generated test images with the different illumination powers in addition to the provided test images from [11]. A detailed overview of the entire dataset with the subsets is shown in Table 1.

4 Tiny Pupil Detection Neural Network

Machine learning with neural networks for pupil detection is state-of-the-art and neural networks outperform standard edge-based computer vision (CV) algorithms. Therefore, the focus is on the development of a tiny pupil detection neural network, which can be processed on Arm M55 and Arm Ethos-U55.

Table 1: Overview of the pupil detection dataset for training and testing for each resolution of 100x100 and 200x200 pixels with subsets.

Train Dataset	Test Dataset from [11]	Extended Test Dataset
11000x rendered	130x rendered	130x illPower_1
11000x illPower_1	130x ISM1_Set1	130x illPower_$\frac{1}{4}$
11000x illPower_$\frac{1}{4}$	130x ISM1_Set2	130x illPower_$\frac{1}{16}$
11000x illPower_$\frac{1}{16}$	130x ISM1_Set3	130x illPower_$\frac{1}{64}$
11000x illPower_$\frac{1}{64}$	130x ISM2_Set1	
	130x ISM2_Set2	
	130x ISM2_Set3	

4.1 Architecture of Neural Network

Since input image resolution impacts the processing time of layers and the number of parameters of the neural network, we started with an input resolution of 100x100 pixels. The architecture of the model (model_1) is shown in Table 2. We used Python 3.8.10 and TensorFlow version 2.10.0. All convolution layers have a kernel size of 3, except for the last convolution layer, which has a kernel size 1. The max pooling layers have size 2. The max pooling layers are used after the first, third and fourth convolution layer. The activation function is 'relu' for all convolution layers and the first layer uses a random uniform kernel initializer. In total this model has 363758 parameters. The parameter number is important due to resource requirements for the processing on the Arm M55 and Arm Ethos-U55.

We also slightly varied some layers to see the change in performance and number of parameters. The changes described below were performed on model_1:

- model_2: The layers conv_3 and conv_4 were changed. The number of kernels were increased from 24 to 32 and from 64 to 96. The number of parameters increased to 452502.
- model_3: The max pooling layers were removed. For conv_1 and conv_3 a stride of 2 were used and the kernel size of conv_5 was changed from 3 to 1. This model has 618734 parameters.
- model_4: The kernel size of the first 2 convolution layer were changed to 5. The number of kernels of conv_5 were changed from 256 to 128 and after the convolution layer an additional max pooling layer with size 2 was added. The kernel size of conv_6 was changed from 1 to 3. This model has 689934 parameters.

In addition to our own models, we used the DNN for pupil detection from Nvidia described by Joohwan Kim in [13]. To process it for a 200x200 input resolution, the stride of the first convolution layer had to be removed. The architecture is shown in Table 3. For a comparison of the models with different input resolutions, the pixel error between the ground truth values and the predicted pupil locations were normalized to an image resolution of 500x500 pixels and the pupil detection rate calculated.

Table 2: Architecture of tiny neural network for pupil detection (model_1)

Layer	kernel size	output size
data		100x100x1
conv_1	3	100x100x10
max_pool_1	2	50x50x10
conv_2	3	48x48x16
conv_3	3	46x46x24
max_pool_2	2	23x23x24
conv_4	3	21x21x64
max_pool_3	2	10x10x64
conv_5	3	8x8x256
conv_6	1	8x8x512
flatten		32768
dense		2

Table 3: Architecture of Nvidia pupil detection DNN with 200x200 input

Layer	kernel size	stride	output size
data			200x200x1
conv_1	9		200x200x24
conv_2	7	2	97x97x36
conv_3	5	2	47x47x52
conv_4	5	2	22x22x80
conv_5	3	2	10x10x124
conv_6	3	2	4x4x256
conv_7	3	2	1x1x512
flatten			512
dense			2

4.2 Training Process

For the training of the DNNs, the training dataset was used. In total, there are 55000 images per resolution available. The dataset was split into 80% train data and 20% test data. From the train data again 10% where used as validation dataset. Since, we require a very high accuracy of the pupil center prediction, iterative training was used. An overview of this process is shown in Fig. 3. First, the DNNs are trained with a learning rate of 0.0005 with a batch size of 32. The model is trained, saved and tested. Then, for the next iteration, the already trained model is loaded and retrained on the same dataset and again saved and tested. We used 10 iterations to train the model and after half the iterations, the learning rate was reduced to 0.0001. This allows the model to predict the pupil center position with higher accuracy.

For all predictions, the pixel error is calculated and normalized to an image resolution of 500x500 pixels. Then, the pupil detection rate is calculated, which gives information of the correct detected pupil center position for a given pixel error. Typically, a pixel error up to 5 is considered as correct detection. This allows a good comparison between the models with different input resolutions. Due to the normalization to a resolution of 500x500 pixels, the models should be very accurate in the prediction because for a 100x100 pixel resolution, only a pixel error of maximum 1 would be allowed to count as correct detection. The Nvidia model was trained with the 200x200 dataset and also the results were normalized to a resolution of 500x500 pixels.

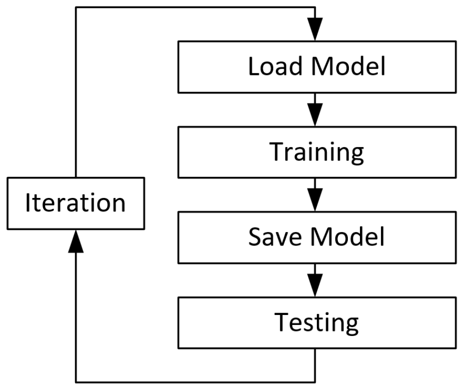

Fig. 3: Overview of the iterative training process for the DNNs.

4.3 Quantization of Neural Network and Simulation

After the training, all models were tested on the additional test set and the extended test set with the different illumination powers, shown in Table 1. Afterwards, the models were quantized with the TensorFlow Lite converter from 32 bit float weights to 8 bit integer weights. The quantization process slightly decreases the pupil detection rate.

The workflow of the model quantization and simulation on Arm M55 and Arm Ethos-U55 is shown in Fig. 4. The trained model with the 32 bit float weights is converted into the 8 bit integer model. Representative data are used to create a data binary file with data information. These files are used together with the quantized model as an input for the Arm Vela compiler, which generates two header files, one for the data and one for the model. Both header files are used in the Arm Development Studio (Arm-DS) [4] version 2021.1.

To simulate the Arm M55 and Arm Ethos-U55, Corstone SSE300 [18] reference package was used. The Arm-DS project was built and the run-time was simulated with the fast cycle simulator. The cycle simulator counts the cycles for processing the DNNs on Arm M55 and Arm Ethos-U55. With a given frequency, the processing time can be calculated. In this work, the run-time was calculated based on a frequency of 100 MHz. The results are presented in the next section.

5 Results

This section shows the results of the tiny pupil detection neural network and compares them with each other. Furthermore, a detailed analysis of the detection rates is done based on the separate test sets introduced in Table 1.

[4]https://developer.arm.com/Tools%20and%20Software/Arm%20Development%20Studio

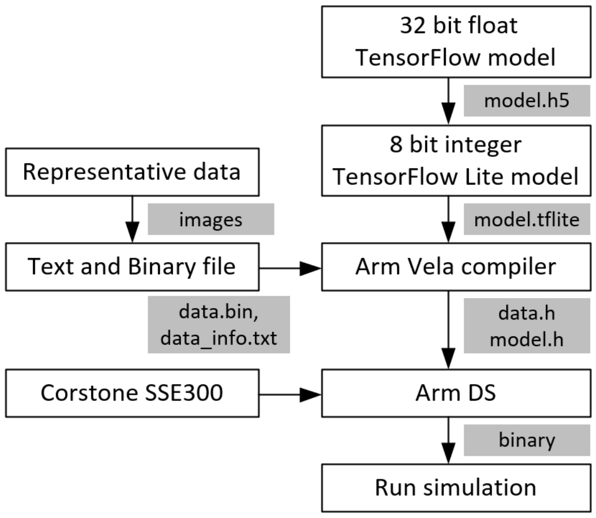

Fig. 4: Overview of model quantization with TensorFlow Lite and simulation on Arm M55 and Arm Ethos-U55.

5.1 Detection Rate of Tiny Pupil Detection Neural Network

Each of the neural networks was tested separately on the different test sets. A detailed overview of the detection rates for a pixel error of 5 is shown in Table 4 and for the quantized model in Table 5. All detection rates, as already mentioned, are normalized to a resolution of 500x500 pixels. For a 100x100 image resolution this would equal to a pixel error of 1. As shown in Table 4 and Table 5, all neural network models achieve similar normalized detection rates. However, there are 2 test sets, where some models perform slightly worse compared to others. For model_1 and model_3, the normalized detection rate is lower for the test dataset ISM1_Set1 from [11]. This test set contains bright images that are slightly overexposed. However, the normalized detection rate drops a little bit for model_4 and the Nvidia model at the test data with illumination power being set to $\frac{1}{64}$. This low illumination power generates a lot of noise, shown in Fig. 2 on the right side. The same results can be seen with the quantized models. In some cases the detection rate drops, but it can also be a little bit higher due to this quantization step. The overall performance of all models is very good and all of the models are used for the run-time evaluation.

The edge-based CV algorithms Swiski [19] and ElSe [22] perform worse than the neural networks, shown in Table 4. The highest detection rates with the algorithms Swirski and ElSe are achieved for the rendered test dataset with nearly 20% and 51.5%. For the other test sets, the detection rate drops. The worst detection rate can be observed for the test set with illumination power being set to $\frac{1}{64}$, with lower than 5%. The noise influences the edge detection steps of the algorithms. Examples of a graphical comparison between the normalized pupil detection rates of the algorithms from Table 4 are shown in Fig. 5 for the test dataset with illumination power being set to 1 and in Fig. 6 for the rendered test dataset. All of the neural networks have a detection rate around 90%.

Table 4: Normalized pupil detection rates in % of the DNNs for the different test sets at a pixel error of 5.

Test dataset	model_1	model_2	model_3	model_4	Nvidia [13]	Swirski [19]	ElSe [22]
130x rendered	92.3	92.3	93.0	92.3	92.3	19.6	51.5
130x ISM1_Set1	87.6	91.5	82.3	94.6	92.3	16.6	38.4
130x ISM1_Set2	91.5	91.5	92.3	95.3	92.3	16.8	40.0
130x ISM1_Set3	91.5	91.5	91.5	92.3	92.3	16.6	43.0
130x ISM2_Set1	90.7	91.5	91.5	95.3	92.3	16.6	43.0
130x ISM2_Set2	91.5	92.3	91.5	93.8	92.3	15.7	44.6
130x ISM2_Set3	92.3	91.5	92.3	91.5	92.3	17.6	47.6
130x illPower_$\frac{1}{64}$	88.4	91.5	90.0	86.1	89.2	0.5	3.0
130x illPower_$\frac{1}{16}$	92.3	91.5	90.7	92.3	91.5	15.0	32.3
130x illPower_$\frac{1}{4}$	92.3	91.5	92.3	92.3	92.3	17.9	33.8
130x illPower_1	92.3	89.2	91.5	91.5	92.3	16.4	40.0
130 overall	91.1	91.4	90.8	92.5	91.9	15.39	37.9
20% of trainset	99.0	99.4	94.8	97.2	99.6	n.a.	n.a.

Representative results of the pupil center detection of the TensorFlow Lite converted model_1 are shown in Fig. 7. The white dots represent the ground truth value and the gray X is the predicted pupil center position. The shown images include different illumination powers, noise levels and brightness values.

5.2 Run-Time of Quantized Tiny Pupil Detection Neural Network

This section shows the run-time/cycle count simulation results with Arm M55 and Arm Ethos-U55 of the models simulated with the Arm-DS. The cycle counts with the associated frame rates are shown in Table 6. Model_1 is the smallest model, and therefore the fastest to be processed with 189 FPS. Nearly all processing is done on the Arm Ethos-U55. This is the same for all other models except model_3. Model_3 is has less parameters compared to model_4 but it takes much longer to process. This is due to the big flatten layer with 225792 parameters of model_3, which is too much for the U55 internal memory. Therefore, the M55 takes over the processing. The advantages of the NPU are not fully used in this case and the frame rate is much lower compared to the similar sized model_4. The difference of the processing time is 48 FPS. The M55 in this case needs more than 374000 cycles. All of these models have an input resolution of 100x100 pixel. The Nvidia model with an input resolution of 200x200 pixels takes almost 11 million cycles on Arm Ethos-U55. Therefore, this model can only

Table 5: Normalized pupil detection rates in % of the quantized DNNs for the different test sets at pixel error of 5. Weights converted to 8 bit integer values.

Test dataset	model_1	model_2	model_3	model_4	Nvidia [13]
130x rendered	92.3	92.3	93.0	91.5	92.3
130x ISM1_Set1	86.9	91.5	82.3	93.0	91.5
130x ISM1_Set2	92.3	92.3	91.5	94.6	92.3
130x ISM1_Set3	90.7	90.7	91.5	92.3	92.3
130x ISM2_Set1	90.7	91.5	91.5	95.3	92.3
130x ISM2_Set2	92.3	92.3	91.5	93.0	92.3
130x ISM2_Set3	90.7	90.7	90.7	92.3	92.3
130x illPower_$\frac{1}{64}$	90.0	90.7	90.7	84.6	90.0
130x illPower_$\frac{1}{16}$	92.3	91.5	92.3	92.3	91.5
130x illPower_$\frac{1}{4}$	92.3	91.5	92.3	92.3	92.3
130x illPower_1	91.5	89.2	91.5	91.5	92.3
130 overall	91.1	91.2	90.7	92.5	92.0

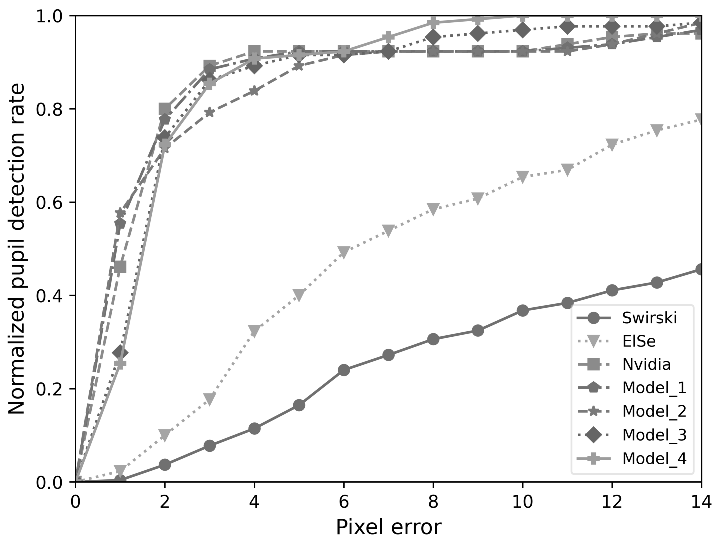

Fig. 5: Comparison of the normalized pupil detection rate on the test dataset with illumiantion power being set to 1 for the edge-based algorithms Swirski [19], ElSe [22], the neural network Nvidia [13] and your own neural networks model_1, model_2, model_3 and model_4.

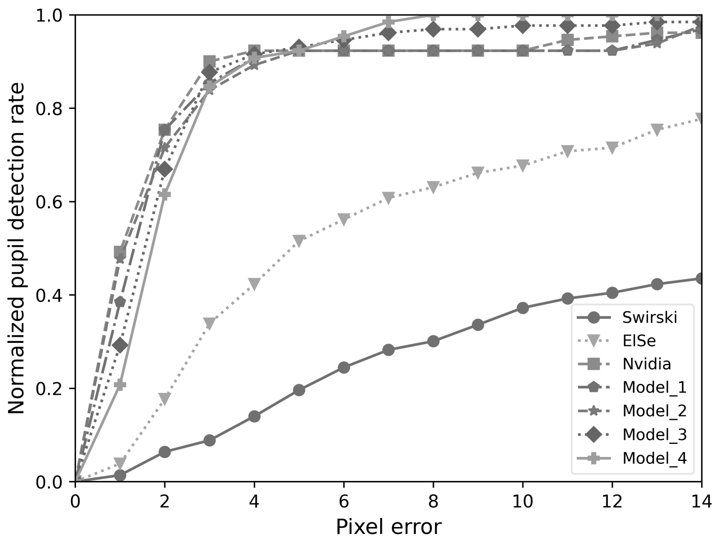

Fig. 6: Comparison of the normalized pupil detection rate on the rendered test dataset for the edge-based algorithms Swirski [19], ElSe [22], the neural network Nvidia [13] and your own neural networks model_1, model_2, model_3 and model_4.

be processed with 9 FPS and is the slowest. The original Nvidia model uses an input resolution of 293x293 pixels and was processed with Titan V and Jetson TX2, shown in [13].

A comparison of the run-time with other algorithms is shown in Table 7. The execution time for the Arm M55 and Arm Ethos-U55 are based on a frequency of 100 MHz. The values of the other algorithms are taken from the associated papers that used different evaluation hardware. With 90–189 FPS, our models are in a good range for pupil detection even if they were processed on an embedded hardware.

6 Conclusion and Future Work

This paper introduces a tiny pupil detection neural network, which can be processed on the embedded processors Arm M55 and Arm Ethos-U55 with 189 FPS with a normalized pupil detection rate of around 90%. This was achieved with the iterative training of neural networks and lowering the learning rate during the iterations. The hardware capability of Arm Ethos-U55 limits the number of parameters that can be used in the flatten layer. If the flatten layer has more

Table 6: Arm M55 and Arm Ethos-U55 simulation results of the different pupil detection neural networks. The used frequency is 100 MHz.

	model_1	model_2	model_3	model_4	Nvidia [13]
Parameters	363758	452502	618734	689934	1751814
cycle count M55	205	205	374242	205	206
cycle count U55	527073	623073	712073	713073	10846073
active cycles	526474	622604	711365	712170	10845500
idle cycles	599	469	708	903	573
total cycles	527278	623278	1086315	713278	10846279
exec. time [ms]	5.273	6.233	10.863	7.133	108.463
FPS	189.64	160.43	92.05	140.19	9.21

Table 7: Arm M55 and Arm Ethos-U55 simulation results with a used frequency of 100 MHz compared with execution times of other algorithms on different hardware platforms.

model	HW	exec. time [ms]	FPS
model_1	Arm M55/U55	5.273	189
model_2	Arm M55/U55	6.233	160
model_3	Arm M55/U55	10.863	92
model_4	Arm M55/U55	7.133	140
Nvidia [13]	Arm M55/U55	108.463	9
Nvidia [13]	Titan V	0.914	1694
Nvidia [13]	Jetson TX2	3.781	264
ElSe [22]	Intel i5-4570	7	142
PupilNet v2.0 [23]	Intel i5-4570	2	500
PuRe [28]	Intel i5-4590	8.333	120

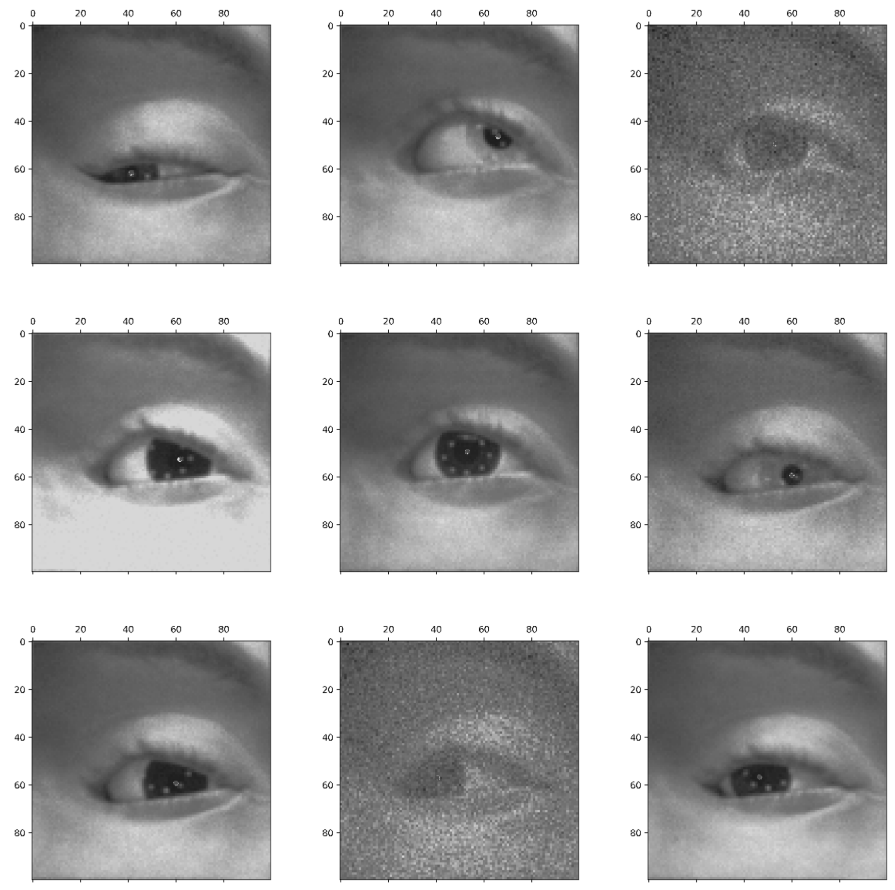

Fig. 7: Example images with the predicted pupil center position with the quantized model_1 for different test subsets. The ground truth are highlighted with a white dot and the prediction is a gray X.

than 65000 parameters, the Arm Ethos-U55 hands the processing back to the Arm M55, due to memory limitations. Therefore, the advantages of the NPU cannot be fully used, which significantly increases the processing time. Furthermore, an existing dataset was extended with images, which were generated by an image sensor model. For that, different illumination powers were simulated, which adds noise to the images and also changes the brightness. Using the proposed tiny pupil detection neural networks, which achieve both good detection rates and short processing times, future AR/VR image sensors can make use of in-sensor processing to lower the power consumption of the communication between the image sensor and the host.

Future work is to extend the dataset with more challenging eye images (occlusion, reflections). Additionally, the neural networks should be optimized with Apache TVM and evaluate the performance on a RISC-V processor with an instruction set extensions or HW-accelerators.

Acknowledgment

A big thank you goes to my colleagues and ams-OSRAM AG, Premstaetten, Austria, which supported this research.

References

1. T. Akter, M. H. Ali, M. I. Khan, M. S. Satu and M. A. Moni, ”Machine Learning Model To Predict Autism Investigating Eye-Tracking Dataset,” 2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST), DHAKA, Bangladesh, 2021, pp. 383–387, doi: https://doi.org/10.1109/ICREST51555.2021.9331152.
2. K. Mengoudi, D. Ravi, K. X. X. Yong, S. Primativo, I. M. Pavisic, E. Brotherhood, K. Lu, J. M. Schott, S. J. Crutch, and D. C. Alexander, ”Augmenting Dementia Cognitive Assessment With Instruction-Less Eye-Tracking Tests,” in IEEE Journal of Biomedical and Health Informatics, vol. 24, no. 11, pp. 3066–3075, Nov. 2020, doi: https://doi.org/10.1109/JBHI.2020.3004686.
3. L. Angreisani, ”A Wearable Brain–Computer Interface Instrument for Augmented Reality-Based Inspection in Industry 4.0”, 2020.
4. P. K. Katkuri, A. Mantri and S. Anireddy, ”Innovations in Tourism Industry & Development Using Augmented Reality (AR), Virtual Reality (VR),” TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), Kochi, India, 2019, pp. 2578–2581, doi: https://doi.org/10.1109/TENCON.2019.8929478.
5. Microsoft, ”HoloLens 2”, https://www.microsoft.com/en-us/hololens, 2022.
6. Meta, ”Meta Quest”, https://www.meta.com/at/en/quest/, 2022.
7. FLAIM Solutions, ”FLAIM”, https://flaimsystems.com/.
8. C. Liu, M. Hall, R. De Nardi, N. Trail, R. Newcombe, ”Sensors for Future VR Applications”, 2017 International Image Sensor Workshop (IISW), pp. 250–253, 2017.
9. C. Liu, A. Berkovich, S. Chen, H. Reyserhove, S.S. Sarwar, T.-H. Tsai, ”Intelligent Vision Systems – Bringing Human-Machine Interface to AR/VR”, 2019 IEEE International Electron Devices Meeting (IEDM), pp. 10.5.1–10.5.4, 2019, doi: https://doi.org/10.1109/IEDM19573.2019.8993566.
10. G. Fiala, Z. Ye and C. Steger, ”Pupil Detection for Augmented and Virtual Reality based on Images with Reduced Bit Depths,” 2022 IEEE Sensors Applications Symposium (SAS), 2022, pp. 1–5, doi: https://doi.org/10.1109/SAS54819.2022.9881378.
11. G. Fiala, Z. Ye and C. Steger, ”Framework for Image Sensor Design Parameter Optimization for Pupil Detection,” 2022 8th International Conference on Systems and Informatics (ICSAI), Kunming, China, 2022, pp. 1–6, doi: https://doi.org/10.1109/ICSAI57119.2022.10005532.
12. G. Fiala, J. Loinig, C. Steger, ”Impact of Image Sensor Output Data on Power Consumption of the Image Processing System”, In: Arai, K. (eds) Intelligent Systems and Applications, IntelliSys 2022, Lecture Notes in Networks and Systems, vol 542, pp. 618–636, Springer, 2023, doi: https://doi.org/10.1007/978-3-031-16072-1_45.

13. J. Kim, M. Stengel, A. Majercik, S. De Mello, D. Dunn, S. Laine, M. McGuire, D. Luebke, "NVGaze: An Anatomically-Informed Dataset for Low-Latency, Near-Eye Gaze Estimation", In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, pp. 1–12, 2019, doi: 10.1145/3290605.3300780.
14. Sony Corporation, "Sony to Release World's First Intelligent Vision Sensors with AI Processing Functionality", https://www.sony.com/en/SonyInfo/News/Press/202005/20-037E/, 14 May 2020, [Online; accessed 4 January 2023].
15. Sony Group Corporation, "Sony's Latest Image Sensors and the Technologies that Lies Behind Them", https://www.sony.com/en/SonyInfo/technology/stories/imagesensor7tech/, 15 October 2020, [Online; accessed 4 January 2023].
16. Arm, "Arm Cortex-M55", https://www.arm.com/products/silicon-ip-cpu/cortex-m/cortex-m55.
17. Arm, "Arm Ethos-U55", https://www.arm.com/products/silicon-ip-cpu/ethos/ethos-u55.
18. Arm, "Arm Corstone-300", https://developer.arm.com/Processors/Corstone-300
19. L. Swirski, A. Bulling and N. Dodgson, "Robust real-time pupil tracking in highly off-axis images", In Proceedings of Eye Tracking Research and Applications Symposium (ETRA), pp. 173–176, March 2012, doi: 10.1145/2168556.2168585.
20. L. Swirski and N. Dodgson, "Rendering synthetic ground truth images for eye tracker evaluation", In Proceedings of Eye Tracking Research and Applications Symposium (ETRA), pp. 219–222, March 2014, doi: https://doi.org/10.1145/2578153.2578188.
21. L. Swirski, "Eyemodel", https://github.com/LeszekSwirski/eyemodel.
22. W. Fuhl, T. Santini, T. Kübler, E. Kasneci, "ElSe: Ellipse Selection for Robust Pupil Detection in Real-World Environments", Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications (ETRA), pp. 123–130, March 2016, doi: https://doi.org/10.1145/2857491.2857505.
23. W. Fuhl, T. Santini, G. Kasneci, W. Rosenstiel, E. Kasneci, "PupilNet v2.0: Convolutional Neural Networks for CPU based real time Robust Pupil Detection", 2017, doi: https://doi.org/10.48550/arXiv.1601.04902.
24. A.-H. Javadi, Z. Hakimi, M. Barati, V. Walsh, L. Tcheang, "SET: A Pupil Detection Method Using Sinusoidal Approximation", Frontiers in Neuroengineering, 2015, doi: https://doi.org/10.3389/fneng.2015.00004.
25. E. Wood, T. Baltrušaitis, X. Zhang, Y. Sugano, P. Robinson, A. Bulling, "Rendering of Eyes for Eye-Shape Registration and Gaze Estimation", in Proc. of the IEEE International Conference on Computer Vision (ICCV 2015), 2015.
26. E. Wood, T. Baltrušaitis, L.-P. Morency, P. Robinson, A. Bulling, "Learning an Appearance-Based Gaze Estimator from One Million Synthesised Images", in Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications, pp. 131–138, 2016.
27. S. Porta, B. Bossavit, R. Cabeza, A. Larumbe-Bergera, G. Garde and A. Villanueva, "U2Eyes: A Binocular Dataset for Eye Tracking and Gaze Estimation," 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), Seoul, Korea (South), 2019, pp. 3660–3664, doi: https://doi.org/10.1109/ICCVW.2019.00451.
28. T. Santini and W. Fuhl and E. Kasneci, "PuRe: Robust pupil detection for real-time pervasive eye tracking", Journal of Computer Vision and Image Understanding, vol. 170, pp. 40–50, 2018, doi: https://doi.org/10.1016/j.cviu.2018.02.002.
29. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving,

M. Isard, R. Jozefowicz, Y. Jia, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, M. Schuster, Ra. Monga, S. Moore, D. Murray, C. Olah, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, ”TensorFlow: Large-scale machine learning on heterogeneous systems”, 2015. Software available from https://www.tensorflow.org/.

30. Google Brain Team, ”TensorFlow Lite”, https://www.tensorflow.org/lite.
31. Apache Software Foundation, ”Apache TVM”, https://tvm.apache.org/download.
32. Qualcom, ”Qualcom Snapdragon”, https://www.qualcomm.com/snapdragon.
33. Qualcom, ”Qualcom Snapdragon Wear”, https://www.qualcomm.com/products/application/wearables/snapdragon-wear-3100-platform
34. Quadric, ”Quadric Chimera”, https://quadric.io/products/.