Mechanics Transport Communications

Academic journal http://www.mtc-aj.com

ISSN 1312-3823 (print)
ISSN 2367-6620 (online)
volume 21, issue 1, 2023
article № 2343

EXPERIMENTAL STUDY OF OPTIMIZED FACE RECOGNITION ALGORITHMS FOR RESOURCE – CONSTRAINED

Volodymyr Lysechko[1]
volodymyr.lysechko@gmail.com
Olena Zorina[1]
olzor@ukr.net
Borys Sadovnykov[1]
borys.sadovnykov@gmail.com
Galina Cherneva[2]
cherneva@vtu.bg
Volodymyr Pastushenko[3]
vpast255@gmail.com

[1]Ukrainian State University of Railway Transport, Kharkiv, Feuerbach Square 7, UKRAINE
[2]Todor Kableshkov University of Transport, Sofia, Geo Milev Str. 158, BULGARIA
[3]Kruty Heroes Military Institute of Telecommunications and Information Technologies, UKRAINE

Key words: *face recognition, Viola-Jones algorithm, convolutional neural network MTCNN, system with limited resources, static quantization, model optimization*

Abstract: *The existing face recognition algorithms were studied, and the effectiveness of the best of them was substantiated according to a set of criteria, namely: checking accuracy, speed and reliability when working on devices with limited resources, such as embedded devices. Mathematical modeling of the face recognition algorithm designed to work in systems with limited resources was carried out. A study of facial recognition methods was conducted with the aim of choosing the most effective ones for further optimization. It is substantiated that the most effective method of face recognition is mixed convolutional neural networks, namely the method using the FaceNet neural network. A series of experiments was conducted to determine the difference between optimized versions deployed on an embedded device and non-optimized versions.*

According to the results of experiments, it has been proven that the static quantization model has the best results. Its advantages are: no need to additionally train the model or train it from the beginning, the possibility of optimizing any model according to this principle. It is justified that static quantization eliminates the need to select the loss function and training parameters, as in the case of knowledge distillation or network pruning. It has been proven that from a practical point of view, quantization is the simplest method of optimization, and in terms of accuracy, the experiment proved only minor losses that do not affect the final result.

INTRODUCTION

Biometric methods of authentication, such as recognition by face, fingerprint, voice, are gaining wide popularity in today's world. Because they often become a vital necessity. First, the Covid, and now the war in Ukraine complicates the ways of authentication. Including, there is a transition to non-contact methods. When recognizing a face, first of all it is necessary to get a photo with a face. For this, an external video module on an IoT device or a built-in camera is used. Next, you need to process the image and highlight the face itself. For this, the Viola-Jones algorithm [1] or the convolutional neural network MTCNN [2] are most often used. In fig. 1 presents the difference between the algorithms.

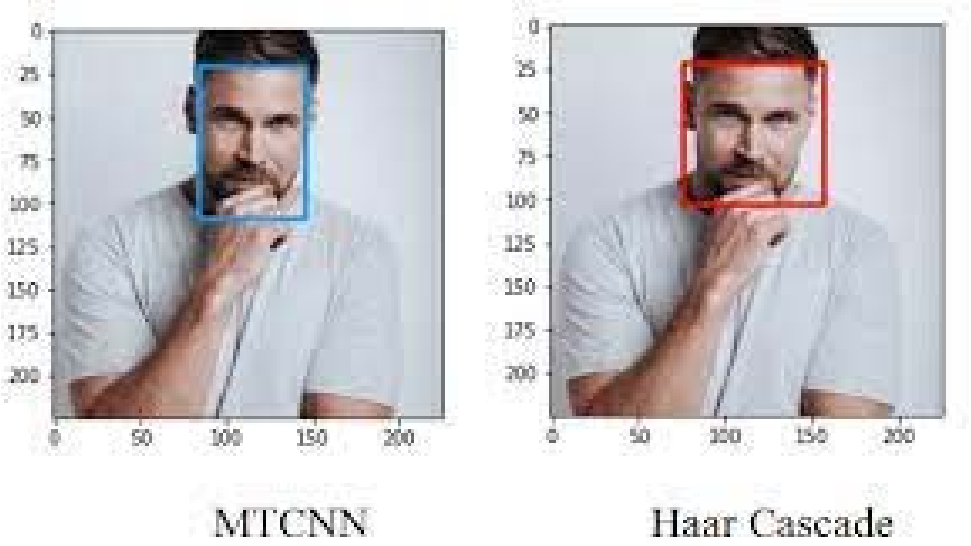

Fig. 1. Difference between MTCNN and Viola-Jones algorithm

The difference in the shape of the face region can change the final accuracy of the algorithm, so it makes sense to measure the accuracy using both algorithms. And it should be taken into account that MTCNN uses more system resources and works longer than the Viola-Jones algorithm. After receiving an image with a face, an optional transformation [3] is resizing. Its necessity is determined by the requirements of the further steps of the algorithm. Next, the resulting photo is normalized, reduced to shades of gray to simplify further calculations, optimize the use of CPU and memory. And the stage of direct recognition of the face in the photo begins. Today, the most effective is a mixed approach. In which part of the features of the face is calculated at the local level, and the other part is based on statistical data about the entire face. The most relevant mixed methods of face recognition are convolutional neurons [4] of the network, which demonstrate the best results in tasks related to image analysis. And one of the most effective methods for face recognition in this class is the method using the FaceNet neural network.

THE MAIN PART

Let's consider the general steps required to recognize a person in a photo using the FaceNet neural network. As output data, that is, at the input, we have a processed image: normalized, of the required size, containing only the face that needs to be recognized. Depending on the input image, the result of the function changes, and for one argument there must always be the same result. It follows that the model is deterministic. After receiving the image, features (for example, a vector representation of the face) are highlighted, by which we will determine who is in the photo. There are a number of algorithms that allow you to do

this, but let's consider a simplified version of the mathematical model without detailed elements, with a generalized algorithm for calculating numerical values of facial features.

The last step of the algorithm is to compare the received numerical information about the face with the data of those face images known to the system and, accordingly, to search for a match. For this, a comparison algorithm is used, a good example of which is the Euclidean distance. According to this metric, the smaller its value, the more two faces are similar to each other. The algorithm must work in an operating system with a permanent hardware configuration that will not change during the operation of the algorithm, hence the stationary mode of operation.

Having considered the main steps of the algorithm, you can build equations for the mathematical model. Let E be a function for transforming a face image into a numerical form, C be a function for finding the distance between two numerical representations of a face. Then we have the following equation:

$$f(x) = min(C(E(x), m1), C(E(x), m2), \dots, C(E(x), mn), p) \quad (1)$$

where mn is the numerical form of a known face;

n – number of known faces;

p is a threshold value that allows you to determine that the face is unknown.

It follows from this equation that the speed of the algorithm depends on the number of known faces in the database [5] of the device. That is, the smaller the number of known faces, the faster the algorithm will work, since there are fewer distance calculations and comparisons. It should be noted that each pair of methods for calculating the numerical representation of a face and the distance between two faces has its own ratio.

In fig. 2, the subject of research with the environment is highlighted and their points of interaction are investigated.

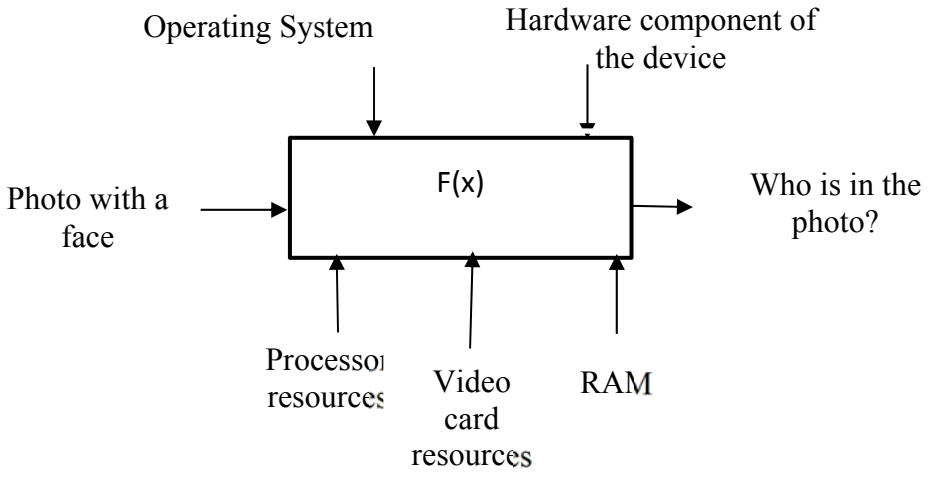

Fig. 2. Model of the subject of research in the form of a «black box»

Let's consider the resources of the external environment, which are necessary for the model to function correctly. First of all, it is the computing power of the central processing unit. It should be noted that the more powerful the processor, the faster the steps of the algorithm will be performed, since some stages depend only on the CPU.

The next factor is access to video card resources. Some of the face-to-numeric algorithms can be run on the GPU, which can greatly increase performance. Here, similarly to the processor, we will depend on the characteristics of the equipment. Unlike the processor,

the video card is mandatory, it can only speed up the operation of the algorithm in some cases.

The last system resource is RAM. A large amount of RAM allows you to perform complex calculations and, for example, use larger and more powerful neural networks as components of our algorithm.

It is also important to note that the data used in the algorithm must be in RAM. Therefore, due to RAM limitations, there are also restrictions on the maximum number of faces that the device can recognize, and the size of neural network weights is limited. Another possible option is to reduce the size of the model weights through optimization, which will allow the use of more complex networks.

According to the results of the simulation, the system configuration factors that affect the possibility and quality of using the FaceNet model on the device are highlighted. The main factor is the RAM of the device. In case of insufficient amount of RAM, it will not be possible to load the model. In addition, if there is memory only for the model, then there will be no possibility to work with images, since their transformations also require a certain amount of RAM. It should be noted that there is a swap file that expands the RAM at the expense of the permanent memory, but if it is used, the speed is much lower than that of the normal RAM, which will affect the recognition execution time. One of the optimization goals should be to reduce the model's RAM requirements.

Another factor is the speed of execution, which depends on the processor and determines how effective the use of the selected algorithm is in real time. If the recognition is too long, it is useless from a practical point of view, so it makes sense to optimize the execution time of one recognition. This can be done by reducing the number of operations, or by speeding them up, for example, by changing the type of weighing data to one that is faster on the selected hardware configuration.

The main problem of using neural networks on embedded devices is the high requirements for RAM and processor. If a less powerful processor will allow the model to work, but at a lower speed, then with a lack of memory there is no way to even start the network. Let's consider methods that will reduce both the use of the processor and make the model smaller in size.

As you know, the size of the model depends on the architecture underlying it. The more complex, larger, deeper the network architecture, the more complex tasks it can solve and the more accurate it is. Along with this, the size of the model grows due to the large number of weights that adjust to the training data.

For use on resource-constrained devices such as mobile phones, there are special network architectures that have a much lower footprint, such as the MobileNet family of architectures, which can weigh between 16 and 60 MB. The existence of such types of architecture does not make it possible to use them to solve any problem on an embedded device, because their power is limited and in such complex tasks as face recognition, they will not be able to demonstrate a satisfactory result. The reason for this is insufficient depth to find complex dependencies.

The knowledge distillation method overcomes these limitations by using a more powerful model to calculate the weights. The bottom line is that there is some powerful model teacher that already has weights ready to use to solve the problem at hand. Due to its large size, this model cannot be applied to an IoT device, so an additional smaller student model is introduced, which can already be used on a target device. Next, the student model is trained with the help of the teacher model, that is, the initial dataset for the problem was processed by a powerful model that found the necessary dependencies and formed valid weights, after which a smaller model based on these weights formed its weights. Thus, we get a smaller

model with corresponding efficiency. Please note that this approach will not work equally well in all tasks and there may be losses in accuracy compared to the teacher model.

The next possible method to reduce the size of the model is quantization. The quantization process of the model consists of reduced precision numbers that preserve the weights. The main idea is that it is not necessary for the weights of the model to be displayed as a 32-bit number, it is possible to reduce this size to 16 or 8 bits while maintaining the functionality of the model. As a result of the operation, the size of the model decreases, and the accuracy may decrease. It is impossible to predict how much the accuracy will drop, but usually the difference is quite small and does not affect the performance of the model. A special case of quantization is binarization, where the size of the weights is reduced to 2 bits per value. This is the most effective optimization method, as it gives a significant increase in the speed of operation along with a decrease in the used memory, but it significantly reduces the accuracy of the model, which can make it impossible to use it further. There is also a dynamic range quantization option that quantizes the model weights, but all operations remain in floating point numbers.

Therefore, the quantized model tends to spike in memory usage due to the partial conversion of weights to float representation during computation, but provides minimal loss in accuracy.

There are three quantization methods: dynamic, static, and quantization-aware training.

Dynamic quantization consists in the fact that quantization occurs during model launch and such a scale is selected to preserve accuracy as much as possible. This type of optimization does not always work and depends on the network architecture.

Unlike dynamic quantization, static quantization is used once after completion of training, after quantization, the calibration process takes place on a subset of the training dataset. In practice, using this method, it is possible to reduce the size of the model by 2-3 times and speed up work by about 1.5 times, depending on the network architecture.

During quantization training, the model immediately rounds the weights to the required accuracy. All weight calculations are performed as floating-point numbers, and only the result is rounded. This method provides the model with the highest accuracy, but requires training a new model from scratch. That is, it is impossible to optimize an already existing, trained model in this way.

Typically, in neural networks, each neuron, channel, and layer has a different effect on the final result. A situation is possible when a neuron has very little influence and in fact, the network can work without it with the same result. In such cases, it is advisable to use the network pruning technique, during which neurons or layers that do not have a sufficient impact on the result are removed. In this way, it is possible to simultaneously reduce the size of the model and speed up its operation, since the fewer neurons, the fewer computational operations per recognition operation.

Each neuron in the network has its own weight value that it uses during recognition. In the case of convolutional neural networks, filters are used that pass over the image. As a rule, the size of the filter is smaller than the size of the image, so the same filter is used on the same image several times and each time creates new weights for each of its positions. To save the space used by the model, one weight can be used for each of the possible positions of the filter. This technique is called the combined use of weights and allows to reduce both the size of the network and the time of its training.

In research, it makes sense to use the following optimizations: static quantization after training, weight trimming, and knowledge distillation. Dynamic quantization is not suitable because of the need to constantly repeat it, the compatible use of weights requires

changes to the architecture of the model, which can affect the speed, accuracy and ability of the model to work in general.

To compare the efficiency of resource use and recognition accuracy of optimized and conventional methods, a series of experiments was conducted with the following prerequisites:

- the optimized and the normal variant of the algorithm were compared with each other;
- the same computing device was used for the comparison of memory consumption, since some differences in memory usage are possible due to differences in architecture;
- to compare the speed of operation, it was permissible to use different computing devices, since some optimizations can lead to a decrease in the speed of operation due to a change in the target architecture;
- different devices were used to compare accuracy;
- all measurements used the same dataset and algorithm for searching and extracting faces from the image;
- the experiments were conducted using the Viola-Jones algorithm and the MTCNN network;
- calculation of the face vector was performed using the FaceNet neural network;
- both absolute and relative differences are obtained for memory consumption and execution time.

The test sample of the Labeled Faces in the Wild dataset [6], which consists of 1000 pairs of images, was chosen for the experiments. At present, this dataset is quite often used to determine the accuracy of the model due to the fact that it is not large in size and the images on it have a realistic environment that simulates the use of the algorithm in real conditions. In addition, the dataset contains already formed training pairs of images that should be used when checking the accuracy of recognition.

The purpose of the experiments is to obtain data:

- memory usage, speed and accuracy of the following versions of the trained FaceNet model:

1) ordinary;
2) statically quantized with a dynamic range;
3) statically quantized to float16;
4) statically quantized to uint8;
5) with truncated scales;

- memory usage, speed, and accuracy of a smaller model that was trained by knowledge distillation from a trained FaceNet model as a teacher;
- analysis of the possibility of using such models on built-in devices.

Regarding the hardware configuration of the computing devices on which the research was conducted, it was taken into account that the processor architecture of embedded devices differs from desktop computers. As a rule, IoT devices have ARM or ARM64 processor architectures, so in the tests it is necessary to use a device with a similar architecture.

Since optimization frameworks are configured to work on ARM-like architectures, when executing optimized models, for example, on a processor with x64 architecture, the execution time increases. At first glance, this looks like pessimism, but the situation may be different on a device with the architecture for which it was optimized.

A computer with x64 architecture has the following characteristics:

- Windows 10 operating system;
- processor frequency – 2.5 GHz;
- 8 cores and 16 threads, 30 MB cache;
-12 GB of RAM with a frequency of 2400 MHz;

- 8 GB of video memory with a bus bit rate of 256 bits.

A computer with ARM architecture has the following characteristics:

- MacOS 11 operating system;
- processor frequency – 3.2 GHz;
- 8 cores and 8 threads, 320 KB of first-level cache per core, and 12 MB of second-level cache for all cores;
-16 GB of RAM;
- RAM is used as video memory, the theoretical maximum is 16 GB, but due to the use of RAM by other processes and the system, the exact current value cannot be predicted.

It should be emphasized that the current configuration of the ARM device is much more powerful than the vast majority of built-in devices, therefore, not the absolute values of the execution time, but the relative values were considered. It also made it possible to investigate the performance and efficiency of complex and large models on ARM architectures.

To obtain results regarding the effectiveness of the used optimizations, it is necessary to have information about the base version of the model. We will conduct a series of experiments on the basic version of the model (table 1).

Table 1. FaceNet metrics without optimization on an ARM processor

	Precision	Size in memory, МБ	Average calculation time, мс	Min calculation time, мс	Max calculation time, мс	Size on disk, МБ
MTCNN	96.1%	196	30.1	29.8	792.3	92.5
Viola-Jones method	92.5%	196	31.9	29.8	809.6	92.5

Similar experiments on the x64 architecture are shown in Table 2.

Table 2. FaceNet metrics without optimization on x64 processor

	Precision	Size in memory, МБ	Average calculation time, мс	Min calculation time, мс	Max calculation time, мс	Size on disk, МБ
MTCNN	96.1%	196	91.5	83.2	1964.4	92.5
Viola-Jones method	92.5%	196	91.3	84.1	1970	92.5

The results show that when using MTCNN, we have higher accuracy compared to the Viola-Jones method.

It is important to note that during the research, the execution time of the experiment using MTCNN is noticeably higher, so let's measure the time used by each algorithm to extract a face from an image. This information will give us the opportunity to better understand how much time the recognition procedure takes and what part of it is spent on face search and what part is on direct recognition. The results of the study are presented in Table 3.

Table 3 shows that the Viola-Jones method is approximately 68 times faster than the current MTCNN implementation. But you have to keep in mind that both algorithms run on the CPU in the current execution and environment.

Table 3. Comparison of running time of face extraction algorithms on ARM architecture

	Average calculation time, мс	Min calculation time, мс	Max calculation time, мс
MTCNN	259.26	149.7	520.1
Viola-Jones method	3.8	2.5	45.5

To optimize the model, static quantization was performed on the already trained model. A total of three quantized models are considered, which differ in accuracy: dynamic range, up to float16, up to int8. It is important to note that the quantization was done using the TensorFlow Lite library, and the model was transformed for more efficient use on mobile architectures. Thus, it is possible to deteriorate the execution time on the x64 architecture. The measurement results for the dynamic range of quantization are presented in Table 4.

Table 4. FaceNet metrics with dynamic range of quantization on an ARM processor

	Precision	Size in memory, МБ	Average calculation time, мс	Min calculation time, мс	Max calculation time, мс	Size on disk, МБ
MTCNN	96.1%	196	91.5	83.2	1964.4	92.5
Viola-Jones method	92.5%	196	91.3	84.1	1970	92.5

Analogous experiments on the x64 architecture are shown in the table. 5.

It is proven that the size of the model is reduced by a factor of about four, since the weights themselves are quantized to int8, while the weights of the activation functions remain unchanged.

Regarding the average recognition time, it decreased by 13.5-14% on ARM architecture. While we can observe significant pessimism on the x64 architecture. This can be explained by the fact that the models executed using TensorFlow Lite have optimizations for computing on ARM processors and are significantly lower on server solutions. Another possible case is the lack of processor support for some commands. From this, we can conclude that further measurements of models that are optimized and performed with the help of TensorFlow Lite do not make sense to do on the current configuration of the machine based on the x64 architecture.

Table 5. FaceNet metrics with dynamic quantization range on x64 processor

	Precision	Size in memory, МБ	Average calculation time, мс	Min calculation time, мс	Max calculation time, мс	Size on disk, МБ
MTCNN	96.0%	68	13156	12815	13600	23.7
Viola-Jones method	92.7%	68	13149	12823.5	13590.65	23.7

Changes in accuracy lie within 0.1-0.2%, which is quite insignificant.

Experiments were conducted with the model quantized to float16 (Table 6).

Table 6. FaceNet metrics with quantization up to float16 on an ARM processor

	Precision	Size in memory, МБ	Average calculation time, мс	Min calculation time, мс	Max calculation time, мс	Size on disk, МБ
MTCNN	96.1%	98	25.1	21.3	150.7	45.7
Viola-Jones method	92.0%	98	24.8	22.8	78	45.7

Since initially the model kept its weights in float32, when quantizing to float16 a reduction of exactly two times was obtained. Unlike the previous experiment, all weights were quantized. The average recognition time decreased by 24-25%, which is quite a good result. The accuracy in the case of MTCNN did not change, but when using the Viola-Jones method it became lower by 0.5%.

Consider a model whose weights are quantized to a range of values of type int8. The following results were obtained (Table 7).

Table 7. FaceNet metrics with quantization up to int8 on an ARM processor

	Precision	Size in memory, МБ	Average calculation time, мс	Min calculation time, мс	Max calculation time, мс	Size on disk, МБ
MTCNN	96.0%	49	27.6	26.8	38.2	23.7
Viola-Jones method	92.3%	49	27.6	26.9	32.41	23.7

The main difference consists in the degraded accuracy when using the Viola-Jones algorithm by 0.5%. Such a difference can be explained by the fact that due to the quantization of the weights of the activation functions, the model became less sensitive, which led to such differences.

Of all the investigated quantization options, the greatest acceleration, by 24-25%, is provided by quantization to float16 precision. This can be explained by the fact that performing operations with this type of data is the most efficient of the proposed ones. Also, such quantization leads to minimal loss of accuracy. While quantization to int8 reduces the memory requirements as much as possible, namely by a factor of 4. But at the same time, it worsens the accuracy in the range of 0.2-0.5%. Quantization with dynamic range reduces the weight of the model less than quantization to int8, but the loss of accuracy is less - 0.1-0.2%. The execution speed of these models is similar.

It was also experimentally found that models optimized and converted to the TensorFlow Lite format run slower on the current test configuration with an x64 architecture processor.

An experimental study of the FaceNet model optimized by weight pruning was also conducted.

To cut the weights, the data was selected on which the model will be further trained and tested in order to remove the weights that have the least impact on the final result. In addition, a loss calculation function is required. Center Loss can be such a function, since the optimizing FaceNet model was trained on it.

The TensorFlow Lite library provides a toolkit for pruning the network weights, so it will be appropriate to transform the model to this view along with the optimization. Thus, it will be appropriate to conduct the experiment only on a computing device with ARM architecture.

The results of the experiments are shown in Table 8.

Table 8. FaceNet metrics after weight trimming on an ARM processor

	Precision	Size in memory, МБ	Average calculation time, мс	Min calculation time, мс	Max calculation time, мс	Size on disk, МБ
MTCNN	66.1%	55	22.21	19.3	86	25.7
Viola-Jones method	62.0%	55	23.8	20.8	82.5	25.7

The results show that the accuracy of the model significantly worsened as a result of the optimization. The important point is that 50% of the tests are negative. If the model does not work at all, it may have 50% accuracy or close to it. That is, it can be concluded that the accuracy of the model is not satisfactory.

The main factor that could affect accuracy is removing too many weights. The reason for this behavior could be insufficient training data used when trimming the weights, an inappropriate optimizer, or loss calculation function. The issue of the loss calculation function is particularly relevant, since the training process is key to the correct operation of FaceNet.

First, let's check how the MobileNet v2 model trained on the ImageNet dataset performs in the face recognition task. Because a model without weights will require much more data to train. In the current case, the process of retraining and adjusting model weights will be carried out, which can also be considered a distillation of knowledge.

Metrics are given in table. 9.

Table 9. MobileNet v2 Metrics for Knowledge Distillation

	Precision	Size in memory, МБ	Average calculation time, мс	Min calculation time, мс	Max calculation time, мс	Size on disk, МБ
MTCNN	65.4%	52.2	21.5	19.2	332	16
Viola-Jones method	64.5%	52.2	20.7	19.1	387	16

The network provides rather low accuracy, as in the previous case. In terms of calculation time, this model also has an advantage due to its simplicity. After the distillation of knowledge, these characteristics should remain at approximately the same level. Changes in size are possible due to weight correction.

To distill knowledge, we will use two loss calculation functions - Softmax, Center Loss, and the result of Center Loss will be multiplied by 0.5 to reduce the influence of this function on the weights. This choice is due to the fact that the teacher model was trained using the same set and coefficients.

For additional training with the teacher model, we will use 5400 photos from the LFW dataset, in batches of 70 images over 40 epochs.

Regarding the parameters of knowledge distillation, we will use alpha - the coefficient of how strongly the teacher's predictions affect the obtained joint weights, as 3.

A function that calculates the loss based on the losses of the student model and the teacher model will be the Kullback-Leibler Divergence.

Table 10 shows measurements after knowledge distillation.

Table 10. MobileNet v2 metrics after knowledge distillation

	Precision	Size in memory, МБ	Average calculation time, мс	Min calculation time, мс	Max calculation time, мс	Size on disk, МБ
MTCNN	77.3%	54.2	21.1	19.3	360	16
Viola-Jones method	73.5%	54.2	20.8	19.5	358.7	16

First of all, the accuracy of the model has increased significantly compared to the initial state, namely by 9-12%. An interesting observation is the disproportionate increase in accuracy when using MTCNN. A possible reason is that this particular algorithm was used during additional training, and the teacher model was trained on faces processed by this algorithm.

The resulting model works 40-45% faster than the teacher model and uses 3.5-3.6 times less RAM.

There are options to improve the results. First of all, using a larger dataset in combination with a different set of training parameters, such as learning rate. The default values were used in the current experiment. Also, a different set of loss calculation functions may lead to improved accuracy, or the development of some new loss calculation function specifically for this task.

CONCLUSIONS

The results of the experiment proved that the static quantization model has the best results. The main advantage is the absence of the need to additionally train the model or to train it from scratch. This means that it is possible to optimize any existing model in this way, without having a suitable dataset for its training. It also eliminates the need to select the loss calculation function and training parameters, as in the case of knowledge distillation or network pruning. From a practical point of view, quantization is the simplest optimization method. From an accuracy perspective, the losses are negligible and cannot greatly affect the final result. But it should be borne in mind that such a result varies from model to model, so it does not make sense to perform quantization of each model without first investigating the impact on accuracy.

Regarding the reduction of RAM after quantization, a clear dependence was found - the model is reduced exactly as many times as the current data type is larger than the future one. This rule is not entirely correct in relation to models quantized with dynamic range, since the weights of the activation functions remain the same, which leads to an increase in the size of the quantized model. In terms of accuracy, dynamic range quantization showed better accuracy than simple quantization to int8, with slightly more memory usage.

Thus, for the optimal use of static quantization, it is necessary to quantize the model taking into account a high level of accuracy, check their characteristics and choose the optimal accuracy for the current task. Also, the face extraction algorithm can affect the final accuracy, according to the study, when using MTCNN, higher accuracy is obtained, but the execution speed is significantly lower than Viola-Jones algorithm.

LITERATURE

[1]. Szeliski, R. Computer Vision: Algorithms and Applications. — Berlin: Springer, 2010. — 812 p.

[2]. Zhang K., Zhang Z., Li Z., Qiao Y., Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks, Arxiv, 2016, URL: https://arxiv.org/abs/1604.02878

[3]. Simon J. D. Prince, Computer Vision: Models, Learning, and Inference. – Cambridge: Cambridge University Press, 2012. – 581 p.

[4]. Smelyakov, K., Chupryna, A., Bohomolov, O., Hunko, N. The Neural Network Models Effectiveness for Face Detection and Face Recognition 2021 IEEE Open Conference of Electrical, Electronic and Information Sciences, eStream 2021 - Proceedings, 2021, 9431476 DOI 10.1109/eStream53087.2021.9431476

[5]. Hernandez, M. Database Design For Mere Mortals – Boston: Addison-Wesley Professiona, 2013 – 614 c.

[6]. LFW Face Database: Main . - URL: http://vis-www.cs.umass.edu/lfw