Deep Learning for Facial Recognition

Karol Marso[1]

Abstract: In biometrics face recognition is used to identify an individual. It can find use in many different applications ranging from crime solving, identifying suspects to authenticating an individual when unlocking a smartphone or other device. Nowadays even embedded devices such as mobile phones use face recognition to authenticate an individual. It is important for a face recognition algorithm not only to be successful in ideal conditions. Many times these conditions cannot be met and therefore we need to design an algorithm that can adapt to these conditions. Rise of deep learning provides us with a strong mean to solve changing conditions and allows a more precise face recognition. Another important aspect is efficiency of such algorithms. However, apart from efficiency in terms of complexity of the algorithm we also have to deal with another problem nowadays which is a power consumption. Since face recognition is trained on thousands, millions or even hundreds of millions of pictures, we have to do a lot of computations and because of this power consumption can rise greatly. In our paper we are examining various different approaches towards face recognition using deep learning and also for publicly available frameworks. We are comparing these solutions in various different terms such as success rate. The final Section of the paper also discusses possible use of FPGAs as hardware accelerators for face recognition with lower power consumption.

Keywords: face recognition, deep learning, survey, FPGA

1 Introduction

In recent years machine-learning found its use in many applications. Machine-learning is used in applications such as databases and data mining applications, genetical research, or biometrics. The greatest advantage of machine learning is the fact that its algorithms can efficiently adapt to new situations and can improve. However, the main problem of a conventional machine-learning is that we always need to process data in their raw form and thus we need to discover their representation. Because of this a new approach, called deep learning was developed. Deep learning is capable of developing new representations by layering the data across multiple layers and transforming their representation [LBH15]. It found its use in many modern applications ranging from text-to-speech recognition to face recognition while beating most of the approaches using traditional machine learning.

Deep learning takes machine learning one step further by using artificial neural networks. These networks resemble a structure of the brain. Each has set of neurons. These neurons extract some part of the data from the layer and use weights to produce output. Adjusting weights is based on gradients and in practice the most common method is to use stochastic gradient descent. In the first step the neural network is trained and then is filled with new unknown data set which is called test set. This test set is then used to test if the network is

[1] Technical University of Denmark, Department of Applied Mathematics and Computer Science, s172747@student.dtu.dk

capable of producing any sensible data. After training the network we can use it to process the data and obtained output can be further used in another applications, for example face recognition.

In our paper we examine various techniques of face recognition using deep learning. We also examine different available frameworks and projects that are doing face recognition using deep learning. We are also proposing a novel approach towards face recognition by using FPGAs as hardware accelerators and that can achieve not only better performance than standard CPUs, but also lower power consumption while still preserving reconfigurability capabilities of CPUs.

The paper is structured into 6 sections. In Section 1 we are introducing a topic of our paper, in Section 2 we examine state-of-the-art research works on the field of deep learning for face recognition. Section 3 discusses benchmarks of the face recognition algorithms and the results of some of the state-of-the art algorithms on these benchmarks. In Section 4 we present some of the available frameworks for face recognition. Section 5 discusses possibility of using FPGA for face recognition with lower power consumption. Section 6 concludes the paper.

2 Deep learning approaches for face recognition

Face recognition dates all the way to early 1960s when the first semi-automated methods were used [Vi13]. Since these times the face recognition have improved and many different techniques were developed e.g. Karhuen-Loeve expansions (Eigenfaces as one of the most used method is based on this), correleation based methods and others [Vi13]. Due to rise in computing power and improvements on the field of machine learning we are now capable to use deep learning for face recognition with high efficiency. There are many works that propose different approaches towards the face recognition with use of deep learning. In this section we are reviewing used approaches and are discussing their results.

2.1 FaceNet

In [SKP15] authors created a face recognition system called FaceNet which achieved a record breaking success rate. Their approach is based upon embedding function f(x) of an image x into feature space of $\mathbb{R}^d$. The squared distances between similar faces should be smaller than those of different faces. Function f(x) is embedded upon triplets that consist of anchor image, of a specific person. This image is much closer to positive image of the same person than to negative image of a different person. From these authors created a set of triplets. The triplets are then used for the training. Proper selection of mentioned triplets is then crucial for efficiency of the algorithm. Authors are selecting triplets based on mini-batches which consist of 40 faces per batch. Convolutional neural network is with Stochastic Gradient Descent with standard back-propagation. Overall architecture can be seen in Figure 1. Convolutional neural network leads to L2 normalization which then produces face embedding. Triplet loss is the next step in training which minimizes

the distance between anchor image and positive image while distance between anchor and negative image is maximized. Learning rate of the network is 0.05 and training time is between 1000 to 2000 hours. Architecture is based on [ZF14] with 22 layers with 140 million parmeters and requires approximatelly 1.6 billion FLOPS.

Fig. 1: Architecture of the network [SKP15].

Authors of the FaceNet [SKP15] composed architecture of their neural network on two works [ZF14] and [Sz15]. In [ZF14] authors state the main problem of neural networks in past was the fact that many neural networks were developed mostly by trial-and-error method which is from scientifical perspective highly unsatisfactory. Because of this authors propose a new way to visualize these convolutional networks. These visualizations are then used to build better architectures for neural networks since they show how each layer extracts features. In [Sz15] authors propose a deep convolutional neural network with improved utilization of computing resources inside of the network. Authors created network called GoogLeNet. The architecture of the network is based on so called ”Inception” modules. These modules are doing convolutions on smaller patches with sizes 1 x 1, 3 x 3 and 5 x 5. Figure 2

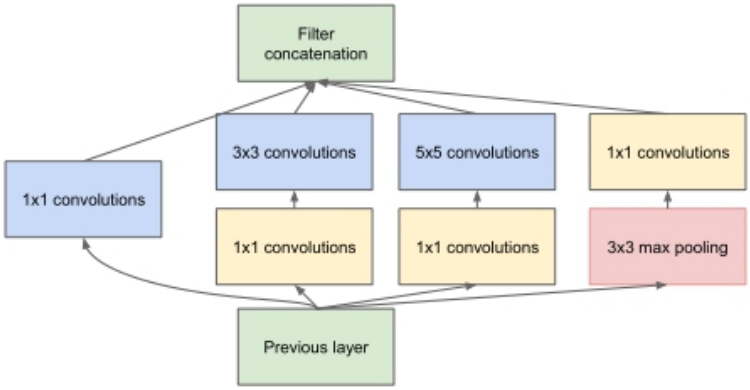

Fig. 2: Stacking inception modules onto each other [Sz15]

Modules are then stacked on top of each other with occasional max pooling. It allows for increasing the depth of the network without increasing the computational complexity too much and thus creating more complex networks. Another alternative is to create inferior networks with lower utilization of computational resources. Overally authors were capable of creating optimal sparse structure with dense building blocks while still leaving the network competitive. FaceNet then improved all of this models to build their face recognition system.

FaceNet [SKP15] was experimenting with trade-offs between FLOPS and accuracy. The results show correlation between FLOPS and accuracy. On the other hand, amount of parameters did not have such impact on the accuracy. Another important parameters which increased accuracy of the system were quality of image, number of training images and number of pixels. Embedding dimensionality showed that dimensions of 128 had the best results and higher dimensions (e.g. 256 and 512) cause a drop in the accuracy. This does not necessarily mean that higher amount of dimensions is bad, it can also mean that with growing amount of dimensions there is more training needed. When it comes to amount of data, tens of millions of training images showed to reduce the error most effectively. With hundreds of millions of images, the accuracy was slightly improved, but the accuracy was not improved too much.

2.2 DeepFace

Direct competitor of FaceNet which was created by Google is DeepFace [Ta14] developed by Facebook. DeepFace authors created a deep neural network for face recognition and face alignment system which explictly models a 3D face. Face is at the beginning aligned and frontalized. For alignement DeepFace is using fiducial points. Face alignment starts with a 2D alignment of the face. This is using 6 fiducial points which are centered at the center of the eye, tip of the nose and mouth location. Through this, DeepFace obtains anchor locations. At this point DeepFace creates 2D aligned crop. Next step is then 3D alignment. In this step 2D aligned crop is transformed into 3D shape. For this DeepFace localizes 67 additoinal fiducial points with second support vector regression. After this, face frontalization is performed. The whole process is described by Figure 3. Deep neural network consists of 8 layers. C1 and C3 are responsible for performing the convolutions. Layer M2 is performing max pooling. These layers are mostly performing pre-processing and are responsible for most of the computation. Layers L4, L5 and L6 are locally connected and are using a lot more parameters than previous layers. Figure 4 shows the architecture of the neural network.

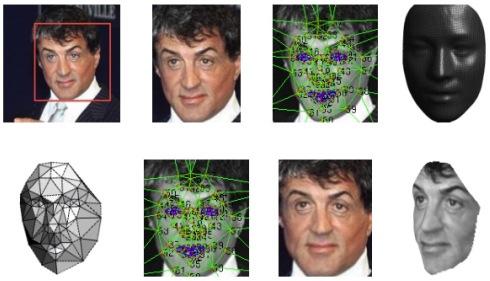

Fig. 3: Face recognition process [Ta14]

The last two layers are F7 and F8. These layers are fully connected which means that each output is connected with each input. Layer F7 produces output which is used as a raw face representation feature vector. Output of F8 is used in K-way softmax. This produces dis-

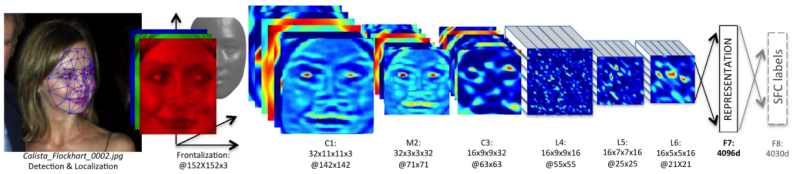

Fig. 4: DeepFace neural network architecture [Ta14]

tribution across class labels. The very last stage is normalization. In this stage, features are normalized between zero and one to reduce sensitivity to illumination changes. Another approach DeepFace tested was Siamese network. This approach replicates face recognition network for each image twice and then uses features to directly predict if the face belongs to the same person. This approach however, requires twice as much computations, but uses the same amount of parameters.

The network was tested on various datasets. In Labeled Faces in the Wild the network was capable to achieve accuracy of over 97.5 % which is comparable with human performance. Comparison of different approaches are showing that on the Labeled Faces in the Wild, Siamese network performs best, but accuracy is comparable with 2D aligned approach or with gradient based approach. On the other dataset called YouTube Faces, DeepFace was also compared with other state-of-the-art approaches of its time (2014).

2.3 OpenFace

OpenFace [ALS16] is a work based on FaceNet. In this work, authors developed an open-source library for face recognition. Their main aim was for mobile systems which require real-time face recognition. From the architectural point of view, neural network is based on the FaceNet. FaceNet is capable of detecting faces even in non-ideal conditions all due to large dataset which can be used to run heuristics upon the input. On the other hand OpenFace is working on much smaller dataset and thus requires more normalization procedures to transform face. This transformation is based upon discovering face landmarks and makes the key features of the face appear at the same locations. To train the network, OpenFace is using smaller amount of samples (approximately 500K photos). OpenFace is using embedding on the hypersphere, for this the network is using FaceNet‘s triplet loss function and Euclidean distance to represent similarity. Since OpenFace framework is open-source we will also talk about it in Section 4 where we examine open-source frameworks for face recognition. Testing of the framework showed that on Labeled Faces in the Wilderness, OpenFace had accuracy rate of around 92% which is close to the other state-of-the-art techniques such as DeepFace and FaceNet.

2.4 Regional-based neural network face recognition

The work [Re15] authors are proposing fast region proposal network. It creates rectangular areas with objectness scores. These are created by moving a sliding window which slides over regions. This method allows for fast detection of objects with speed between 5-17 fps. This approach is used and improved in [Zh16]. It aims to improve face recognition in non-ideal conditions such as low resolutions, bad angles or bad illumination conditions. By extending the framework presented in [Re15] authors use the idea of R-CNN to span the network into multiple deep feature maps. Authors use two types of convolutional networks to create regions of interest. In order to improve facial recognition authors are combining local and global features together. They also perform regions of interest pooling from shallower layers of the network to a deeper layer which contains less information and without it might not contain enough information to detect the face. Architecture of the network is 13 layers deep and after convolution layer there is a ReLU layer. Network contains 5 major layers each with 2-3 convolutional layers. Last three layers are then put to region of interest (RoI) pooling and also concatenated for L2 normalization. They are then fed to another layers of the network. Architecture of the proposed network is shown in Figure 5. Layers conv3, conv4 and conv5 are concatenated as can be seen in the figure. Normalizations are done in each layer independently.

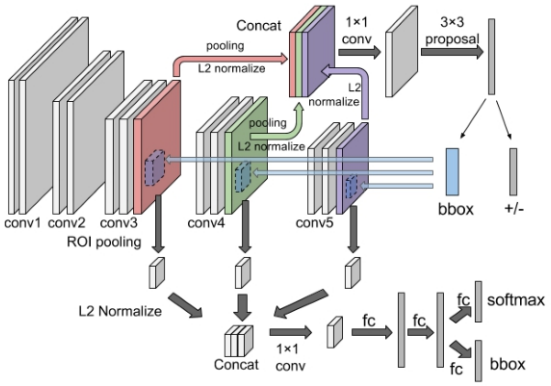

Fig. 5: Architecture of the R-CNN used for face recognition [Zh16]

The main problem of the network was that features could not be concatenated, but they needed to be first processed. System took multiple features and applies L2 normalization upon these features. Another step was to re-weight all these features across the layers. These features are then concatenated to one single tensor. This way features are more stabilized.

2.5 Multimodal biometrics

More advanced approach towards face recognition is presented in [MAMA17] which deals with multimodal biometrics. The goal was to create unconstrained video face recognition

which meant that different modalities might appear in different and rather random angles and features might have been more difficult to obtain. In this work authors are extracting multiple modalities which helps to increase the quality of recognition. The novelty of their algorithm is the fact that they use deep learning to train neural network automatically recognize and extract the features (e.g. left ear, right ear etc.) even though the record can be random and thus these features might be more difficult to obtain. Features are first extracted, then they are put through Gabor filters which help to detect non-redundant features. In the next part deep neural network is used to learn the features and then uses supervised learning to learn to detect the features. This whole process is explained in Figure 6

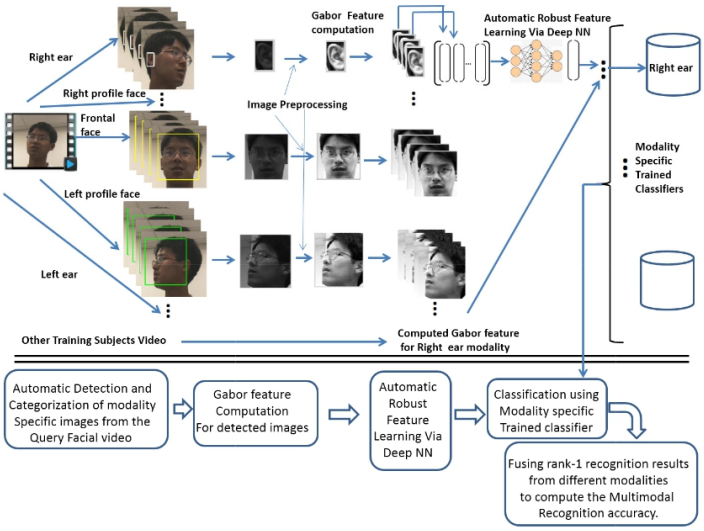

Fig. 6: Block diagram of the system for multimodal biometrics[MAMA17]

When it comes to modalities, authors are trying to recognize 5 different modalities - left ear, left profile, front face, right profile, right ear. Dictionaries of the modalities are built from deep learning network. There are 5 dictionaries for each training sequence. For the score normalization is used function Tanh. The results of the modalities are fused together after normalization using weighted sum technique. Weights were obtained in two different ways, either by exhaustive search or were based on the individual performance of the classifiers.

During tests, network had amount of features of 9600 for frontal and profile face and 4160 for each of the ears. This was however, too complex and authors reduced this length by using principal component analysis. Through this, it was possible to reduce amount of features to 1000 and to 500 features. Multimodal accuracy from tests was then highest for 1000 features and also 500 features performed better than results without reduction.

3 Deep learning performance benchmarks

In previous Section, we were discussing multiple different approaches. If we want to compare two different approaches we need to have some method which can objectively be used to determine which approach had a better accuracy. In [SKP15] [Ta14] and [ALS16] tests were performed on Labeled Faces in the Wild and YouTube Faces datasets. This section is a small introduction to thse two widely used benchmarks and we also compare the results of the three face recognition approaches on these benchmarks

3.1 Labeled Faces in the Wild

Labeled Faces in the Wild [Laa] were introduced in 2007 to help with research in the face recognition [Le16]. It tests both verification and identification and is used as a benchmark to test accuracy of the new approaches. In short, Labeled Faces in the Wild or LFW is a database containing more than 13 thousand images of more than 5 thousand people. There are multiple sets which are used either for training or for benchmarking. A deeper survey of many approaches towards face recognition (also approaches that use different methods than deep learning) and of the whole benchmark are presented in [Le16]

The first three reviewed approaches were all using LFW as their benchmark. In 2015, the best performing approach was FaceNet [SKP15] with approximately 99.6% accuracy. DeepFace by Facebook [Ta14] scored approximately 97.35% accuracy. The third presented approach scored accuracy of approximately 92% however, their aim was to create more open-source approach with much less resources required. Even though their accuracy was not as good as those of FaceNet and DeepFace, they used much smaller amount of training data. just 500 thousand images compared to 100 to 200 million that was by FaceNet and 4.4 million used by DeepFace from SFC dataset. Since the goal was to achieve high accuracy with smaller amount of data, it is clear that authors of OpenFace succeeded in their goal.

Even though Facenet and DeepFace had high scores (FaceNet was the best performer of its time) the research went on and nowadays the best performer is YI+AI by Yitu tech. Since this work was done by a Chinese company it was difficult to obtain their method because author is not familiar with Chinese language and there was no technical report released in English on this work as far as goes our knowledge. The best performers can be found at [Lab].

3.2 YouTube Faces

Another benchmark which both FaceNet and DeepFace used is called YouTube Faces. Its idea is based on Labeled Faces in the Wild and contains more than 3 thousand videos of 1.5 thousand different people from YouTube which can be used for benchmarks [Yo].

For this benchmark FaceNet was using average similarity for first few hundreds of frames of the video and also for first few thousand faces. The accuracy was in both cases around

95%. DeepFace had accuracy of around 91% at first, but authors mention that there were around 100 wrong labels and after fixing these, the accuracy was increased to 92%. The lower accuracy of recognition in these benchmark resulted from the fact that quality of images in the YouTube Faces database was worse than of those in LFW.

Apart from the above mentioned benchmarks there are also many other databases. In [Le16] authors examined three other notable databases - namely IJB-A database and benchmark which consists of videos and images and encourages to explore using both videos and images for face recognition. Another databases are FaceScrub and CASIA. These databases do not contain many individuals however, there are around 500 images per individual person and thus these databases are rather deep than broad. There are two implications for these databases - images can be rejected if they are considered outliers, which means the picture for example has too much occlusion and therefore these outliers are not suitable for training. Another implication is the fact that there is no guarantee that all the images are correctly labeled. The last database authors of the survey point out to is called MegaFace. This database can be used to identify exactly one individual from large set of individuals in a gallery. It is meant to be used in conjuction with other different databases.

4 Available frameworks

There are many publicly available frameworks and libraries on the Internet which are using deep learning for face recognition. Our main source for searching was GitHub where we found many available libraries.s

First framework that we tested out is OpenFace [ALS16]. This work is described in Section 2. OpenFace [Gib] framework is compatible with Ubuntu and OSX. It can be installed either through Docker or natively. This offers a variability to developers that would like to use the framework in their applications. The framework is available under Apache 2.0 license. We tested out multiple provided demos. Especially interesting was demo which allowed for real-time face recognition. Framework however, suffers from rather smaller documentation and also contains only few examples which show capabilities of matching faces and training new classifiers, but sources of the demos are not well documented and because of that it was difficult to create any example application. As mentioned the most interesting demo was the real-time face recognition. This demo nicely shows capabilities of the library. It also allows for learning new faces which can be recognized. We tested it out with multiple faces however, it was capable to recognize only one face at a time. Another demos show comparison of the images where we compared pictures of the author with some celebrities. We also tried image classification. We again used photos of the author and of the celebrities to train the network and then used some other photos for comparison and classification. OpenFace correctly identified all of the samples. The last demo required older version of OpenCV library and we were not able to run it because of this. Overally the library has good demos that show capabilities of it, on the other hand it lacks any kind of tutorial, or better explanation of what each function does.

Another interesting open source solution that can be found at GitHub [Gic] and it is without a name. However, the description states it is the simplest facial recognition API for

Python. And from all of the tested solutions it truly was the simplest one available. Authors state that they achieved accuracy of 99.38% on the Labeled Faces in the Wild. This library offers many features that are beyond just face recognition. It allows also manipulation of facial features. It can be also combined with other libraries to build real-time face recognition and there is also example of this. Similarily with OpenFace library, also this library is compatible with Linux and Mac OSX. Compared to the OpenFace library, this face recognition library is more simple to use. We tested multiple features of this library using Python and Jupyter Notebook which is a very useful prototyping tool. We first tested face recognition where we put multiple different images with persons and then exctracted the individual faces. In another test we downloaded pictures of a person and compared them with two pictures, one of the same person and another of a different and then used the framework to compare them. We used multiple images under different conditions and in most cases this comparison was correct. Out of all the libraries we tested this one was the easiest and most most comfortable library to use.

Another FaceNet [SKP15] inspired library is called Face Recognition using TensorFlow [Gia]. As the name suggests, this library implements FaceNet style face recognition system using TensorFlow. This library is compatible with Linux (Ubuntu) and Mac OSX. This library was pre-trained on two datasets, namely CASIA-WEBFACE and VGGFace2. For Casia system scored success rate of around 99.05% accuracy on LFW and for VGGFace2 it scored 99.65% accuracy on LFW. The source code is also inspired by OpenFace [ALS16] library. We did not run a training since it takes around 30 hours on NVidia Titan X GPU which is a high end GPU not available to us. We wanted to run some of the tests, but we were not able to find any proper documentation on how to run the tests, or which datasets should be used. Because of this we can only rely on the published results. Overally library seems as a powerful network that should perform much better than OpenFace, but we could not prove it, since we could not run it. On the other hand use of tensorflow and inspiration by FaceNet make this library still an interesting solution worth mentioning. If the authors develop some tutorials, examples and will provide a proper documentation this library could be an interesting framework.

The last reviewed framework was developed for Android mobile operating system and is called Android Face Recognition with Deep Learning Test Framework [MS]. It is based on Android NDK which wraps around C++. Similarily to other libraries also this one is using OpenCV, TensorFlow, Caffe and other libraries that are usually used for face recognition and image processing. From the C++ libraries some of the functionalities were moved into Android SDK. Authors created two repositories. The first repository contains an Android app which can be found at Google Play. The second library contains the library itself and can be implemented with other Android applications. We decided to test the app for face recognition Android application provided on Google Play. The application can be found under a name Face Recognition [Fa]. The application has a simple UI, it allows for training on new faces and can run a real-time face recognition. The main problem of the application was that after removing faces and trying to relearn them, the application started crashing and even after reinstalling it, we were not able to make the training work again.

To sum up, there are many available frameworks that allow face recognition using deep learning. Most of them are rather difficult to use and usually lack any useful documentation which is a great problem. On the other hand the research in this field grows and there are going to be many

5 Towards low power deep neural networks with FPGAs

With growing amount of data there is a need for more power efficient approaches. Even though GPUs show efficient and fast way to run the neural networks, they also consume a lot of power. Because of this we decided to devote a section to another alternative which are FPGAs.

Nature of convolutional networks allows for use of a massive parallelism. Because of this fact, GPUs are widely used as a hardware accelerator to run these neural networks. However, in recent years there were many proposals to use massive parallelism capabilities of the FPGAs. FPGAs have many features which can be used such as runtime reconfigurations which however might take a longer time, but on the other hand offer larger amount of processing elements than CPUs [Fa11]. In [Fa11] authors examined basic use of the convolutional networks on the FPGAs. In their work, authors proposed a data-flow based processor which is using 2D grids of processing tiles (in modern FPGAs, these tiles can have different forms from simple logical blocks capable of performing any kind of logical operation to DSP blocks which can perform more complex operations e.g. floating point multiplication and division) from which FPGA is composed. Performance tests compared overall performance of FPGAs, to CPU, GPU and ASICs. While ASIC was proved to be most efficient, development of such device requires more resources. On the other hand FPGAs performed much better than CPUs and even GPUs with 75% lower power consumption.

Microsoft created its FPGA-CPU architecture called Catapult which can be used to accelerate neural networks [Pu14]. The tests in [Ov15] showed that Catapult was capable of recognizing 134 images per second on ImageNet classification test [KSH12] while architecture with GPUs was capable of identifying between 500 to 800 images per second. On the other hand power consumption of the FPGA was just 25W compared to 235W of GPU.

The main problem of the FPGAs is that they require specialist knowledge of the technology and because of this are many times more difficult to use for development. However, nowadays there are possibilities also for high-level synthesis and even frameworks such as OpenCL can be used to create neural networks on the FPGA [LTA16].

All of the above mentioned works prove us that FPGAs are going to be used more and more in future for performing deep learning. In [Zh18] authors proposed a CNN used for face recognition. It uses a hybrid approach combining multiple algorithms, namely Winograd‘s convolution which is efficient for not too large kernel sizes and FFT-based algorithm for larger kernels. Face recognition is inspired by FaceNet [SKP15] and Inception modules [Sz15]. This work shows how powerful are parallel processing capabilities of the FPGA and its reconfigurability. It allows for great variety of designing possibilities. However,

it comes at a price and it is still a difficult task to achieve maximal possible efficiency and careful analysis is important. To create a neural network authors are using Inception modules inside of which are implemented Winograd's convolution and FFT-based convolutions. Implementation was then evaluated against Inception work and also compared to other works that dealt with face/image recognition on the FPGA. Results show that latency of the implementation is better than that of the GPU (NVidia GTX 1080) - 23.7 ms per picture on the FPGA compared to 89 ms per picture on the GPU. Morover, energy efficiency per picture is 0.251 J/pic/s (joule per picture per second) on the FPGA and 9.780 J/pic/s on the GPU.

6 Conclusion

Our review shows that deep learning proves as the most efficient and accurate method for face recognition up to date. Growth of the computational power helped to spark the use of this technique. Especially improvements in hardware accelerators such as GPGPUs (general purpose GPUs) allows for faster training and recognition. In our paper we reviewed state-of-the-art techniques and their benchmarks. The best results had those solutions that could use most of the data and that is the reason why solutions by huge companies such as Facebook and Google which have large databases and huge amount of training data available to them, were able to achieve almost human performance.

We also examined available frameworks. Most of them can be easily found on GitHub and many of them can be used as libraries. Most of them were compatible with Python programming language which is currently one of the most popular languages used for machine learning and deep learning. These libraries still lack a better documentation and better tutorials.

The main problem of deep learning is the huge computational power it requires. With this the power consumption becomes a huge problem. Because of this we also reviewed face recognition using deep learning in FPGAs. FPGAs allow us to build easily reconfigurable and low power specialized hardware that can be used to accelerate face recognition. This solution is already used in practice by Microsoft which built its FPGA-CPU system called Catapultin order to accelerate deep learning applications.

References

[ALS16] Amos, Brandon; Ludwiczuk, Bartosz; Satyanarayanan, Mahadev: OpenFace: A general-purpose face recognition library with mobile applications. Technical report, CMU-CS-16-118, CMU School of Computer Science, 2016.

[Fa] Face Recognition Android. https://play.google.com/store/apps/details?id=ch.zhaw.facerecognition. Accessed: 2018-06-21.

[Fa11] Farabet, Clément; LeCun, Yann; Kavukcuoglu, Koray; Culurciello, Eugenio; Martini, Berin; Akselrod, Polina; Talay, Selcuk: Large-scale FPGA-based convolutional networks. Scaling up Machine Learning: Parallel and Distributed Approaches, pp. 399–419, 2011.

[Gia] GitHub Face recognition using Tensorflow. https://github.com/davidsandberg/facenet. Accessed: 2018-06-20.

[Gib] GitHub OpenFace. https://github.com/cmusatyalab/openface. Accessed: 2018-06-15.

[Gic] GitHub The world's simplest facial recognition api for Python and the command line. https://github.com/ageitgey/face$_r ecognition. Accessed : 2018 – 06 – 20.$

[KSH12] Krizhevsky, Alex; Sutskever, Ilya; Hinton, Geoffrey E: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105, 2012.

[Laa] Labeled Faces in the Wild. http://vis-www.cs.umass.edu/lfw/. Accessed: 2018-06-14.

[Lab] Labeled Faces in the Wild Results. http://vis-www.cs.umass.edu/lfw/results.htmlnotes. Accessed: 2018-06-14.

[LBH15] LeCun, Yann; Bengio, Yoshua; Hinton, Geoffrey: Deep learning. nature, 521(7553):436, 2015.

[Le16] Learned-Miller, Erik; Huang, Gary B; RoyChowdhury, Aruni; Li, Haoxiang; Hua, Gang: Labeled faces in the wild: A survey. In: Advances in face detection and facial image analysis, pp. 189–248. Springer, 2016.

[LTA16] Lacey, Griffin; Taylor, Graham W; Areibi, Shawki: Deep learning on fpgas: Past, present, and future. arXiv preprint arXiv:1602.04283, 2016.

[MAMA17] Maity, Sayan; Abdel-Mottaleb, Mohamed; Asfour, Shihab S: Multimodal Biometrics Recognition From Facial Video via Deep Learning. Signal & Image Processing: An International Journal, 8(1):1–9, 2017.

[MS] Michael Sladoje, Mike Schlchli: , Android Face Recognition with Deep Learning - Test Framework. https://github.com/Qualeams/Android-Face-Recognition-with-Deep-Learning-Test-Framework. Accessed: 2018-06-21.

[Ov15] Ovtcharov, Kalin; Ruwase, Olatunji; Kim, Joo-Young; Fowers, Jeremy; Strauss, Karin; Chung, Eric S: Accelerating deep convolutional neural networks using specialized hardware. Microsoft Research Whitepaper, 2(11), 2015.

[Pu14] Putnam, Andrew; Caulfield, Adrian M; Chung, Eric S; Chiou, Derek; Constantinides, Kypros; Demme, John; Esmaeilzadeh, Hadi; Fowers, Jeremy; Gopal, Gopi Prashanth; Gray, Jan et al.: A reconfigurable fabric for accelerating large-scale datacenter services. ACM SIGARCH Computer Architecture News, 42(3):13–24, 2014.

[Re15] Ren, Shaoqing; He, Kaiming; Girshick, Ross; Sun, Jian: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99, 2015.

[SKP15] Schroff, Florian; Kalenichenko, Dmitry; Philbin, James: Facenet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 815–823, 2015.

[Sz15] Szegedy, Christian; Liu, Wei; Jia, Yangqing; Sermanet, Pierre; Reed, Scott; Anguelov, Dragomir; Erhan, Dumitru; Vanhoucke, Vincent; Rabinovich, Andrew et al.: Going deeper with convolutions. Cvpr, 2015.

[Ta14] Taigman, Yaniv; Yang, Ming; Ranzato, Marc'Aurelio; Wolf, Lior: Deepface: Closing the gap to human-level performance in face verification. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1701–1708, 2014.

[Vi13] Vijayakumari, V: Face recognition techniques: A survey. World journal of computer application and technology, 1(2):41–50, 2013.

[Yo] Youtube Faces. https://www.cs.tau.ac.il/ wolf/ytfaces/. Accessed: 2018-06-14.

[ZF14] Zeiler, Matthew D; Fergus, Rob: Visualizing and understanding convolutional networks. In: European conference on computer vision. Springer, pp. 818–833, 2014.

[Zh16] Zheng, Yutong; Zhu, Chenchen; Luu, Khoa; Bhagavatula, Chandrasekhar; Le, T Hoang Ngan; Savvides, Marios: Towards a deep learning framework for unconstrained face detection. In: Biometrics Theory, Applications and Systems (BTAS), 2016 IEEE 8th International Conference on. IEEE, pp. 1–8, 2016.

[Zh18] Zhuge, Chuanhao; Liu, Xinheng; Zhang, Xiaofan; Gummadi, Sudeep; Xiong, Jinjun; Chen, Deming: Face Recognition with Hybrid Efficient Convolution Algorithms on FPGAs. In: Proceedings of the 2018 on Great Lakes Symposium on VLSI. ACM, pp. 123–128, 2018.