The Application of Neural Networks for Facial Landmarking on Mobile Devices

Connah Kendrick[1], Kevin Tan[1], Kevin Walker[2] and Moi Hoon Yap[1]
[1]*School of Computing, Mathematics and Digital Technology, Manchester Metropolitan University, John Dalton Building, Manchester, U.K.*
[2]*Image Metrics Ltd, City Tower, Piccadilly Plaza, Manchester, U.K.*

Keywords: Facial Landmarking, Android, Deep Learning.

Abstract: Many modern mobile applications incorporate face detection and landmarking into their systems, such as Snapchat, beauty filters and camera auto-focusing systems, where they implement regression based machine learning algorithms for accurate face landmark detection, allowing the manipulation of facial appearance. The mobile applications that incorporate machine learning have to overcome issues such as lighting, occlusion, camera quality and false detections. A solution could be provided through the resurgence of deep learning with neural networks, as they are showing significant improvements in accuracy and reliability in comparison to the state-of-the-art machine learning. Here, we demonstrate the process by using trained networks on mobile devices and review its effectiveness. We also compare the effects of employing max-pooling layers, as an efficient method to reduce the required processing power. We compared network with 3 different amounts of max-pooling layer and ported one to the mobile device, the other two could not be ported due to memory restrictions. We will be releasing all code to build, train and use the model in a mobile application. The results show that despite the limited processing capability of mobile devices, neural networks can be used for difficult challenges while still working in real-time. We show a network running on a mobile device on a live data stream and give a recommendation on the structure of the network.

SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

Face identification and landmarking are often used as the backbone to many camera based applications. Features, such as the auto-focus built into most modern day cameras use a face detector to identify the regions that require focusing. Commercial applications, such as Snapchat (Inc, 2017b), BeautyPlus (Co, 2017) and Makeup Genius (L'Oreal, 2017) are based around identifying the facial region and performing image manipulation to the facial features, these applications are extremely popular and have a high number of users world-wide. Many mobile applications rely on the use of machine learning algorithms to identify patterns in image data. Machine learning uses pre-annotated data to learn the difference between the data values, but it requires a large amount of annotated data to learn which is time consuming even with the available tools (Kendrick et al., 2017). However, Mathias (Mathias et al., 2014) showed that if a strict data annotation regime was implemented, a small dataset can match the accuracy of methods trained on large datasets. We aim to analyse how well neural networks can be integrated into mobile devices to increase the accuracy and reliability of the applications.

Deep learning is restricted and inefficient for low-powered machines, as it requires high-end machines with powerful Graphical Processing Units (GPUs) capable of processing the required quantities of data. However, after the model has been trained, running the network requires significantly less processing power even without optimization, possibly allowing the models to run on low powered mobile devices. In recent years, with the increase in power and affordability of GPUs, deep learning has become more commonly used and in many recent tests, this method outperformed machine learning algorithms significantly ("Grother" and Ngan, 2016). The improved performance of deep learning architectures has potential to improve available application, but the method of implementing deep learning into mobile applications is still in its infancy. Deep learning based software require large amounts of memory and processing capa-

Kendrick, C., Tan, K., Walker, K. and Yap, M.
The Application of Neural Networks for Facial Landmarking on Mobile Devices.
DOI: 10.5220/0006623101890197
In *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2018) - Volume 4: VISAPP*, pages 189-197
ISBN: 978-989-758-290-5
Copyright © 2018 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

bilities, which is not currently efficient for mobile devices. However, the processing power and memory of mobile devices is increasing at a rapid rate, allowing for multi-core processing of large quantities of data, which has the possibility to allow even some of the deeper neural networks to run on the devices.

As deep learning is outperforming machine learning for real-world applications, such as medical healthcare (Goyal et al., 2017), skin assessment (Alarifi et al., 2017), security (Yi et al., 2014) and waste management (Sudha, 2016), we demonstrate the capabilities of implementing deep learning algorithms to mobile devices and analyse its effectiveness. This paper will evaluate the current methods of integrating trained neural networks onto mobile devices.

2 RELATED WORKS

Deep learning and facial landmarking are well researched areas. Deep learning can perform many different tasks, such as object classification and segmentation, to do these the networks structure and layer methodology must be suited to that task. Facial landmarking is best suit for a regression based approach as it has been commonly proven most effective, because the landmarks will shift with different facial expressions. In deep learning, regression is performed by ensuring the networks layer do not perform any 'squashing' of the data. Furthermore, this means the exclusion of methods, such as Softmax (Krizhevsky et al., 2012) and using Root Mean Squared error (RMSE) to calculate the loss, as it provides the distance from the ground truth which is required for regression over the traditional hit or miss methods for classification (Litjens et al., 2017). Furthermore, consideration also has to be based on the type of activation method to prevent restricting the networks outputs, such as Rectified Linear Unit (ReLU) can be used for regression. Other methods to perform regression are provided through APIs, such as Keras (Chollet, 2016) which provides a regression function, but after some initial tests, it shows little difference in performance with an increase in training times.

To make a neural network more efficient, there are many different options, such as use smaller kernels in the convolutions and output less convolution images or use less filters, a different output image is produced for each filter requiring more memory, at each of the layers. However, another method to reduce the strain of the network is to implement max-pooling layers (Srivastava et al., 2014) into the network. The max-pooling layer split the images into a number of regions, from each region the highest value is taken a placed into a new matrix, illustrated Fig 2. This also helps with over-fitting as the data becomes more abstract. The issue with implementing max-pooling layers is that by shrinking the data, data is lost, possibly reducing the accuracy, but reducing processing time. For facial landmarking, the data should be preserved as much as possible, as it is easy to lose key facial features, such as the eye corners and nose tip when employing max-pooling layers. Implementing max-pooling allows networks to run much faster on lower power devices. The max-pooling acts as a cost/reward system which will be analysed in this paper on the trade-off between speed and accuracy.

2.1 Deep Learning for Facial Landmarks

Deep learning for facial landmarking is a known area of research and is used in some commercial applications, such as face++ (Megvii, 2015) which utilises deep learning to give high accuracy results. However, this application requires a connection to their servers and cannot run on devices in real-time. Researchers have shown that deep learning can be used effectively to solve regression problems, such as predicting a point on an image (Zhang et al., 2016; Lai et al., 2015). Recent methods use deep learning for face landmarking are performed by Zhang et al. (Zhang et al., 2016), where they use auxiliary attributes to aid in deciphering the data. During the training stages of the model they provided additional parameters to identify, such as the gender, glasses, expression and pose. These additional parameters can help the network, identify key pieces of information that can affect the results drastically, such as facing left or right can aid the network to identify, when a point is occluded. The results showed increased accuracy in comparison to other systems.

Hochreiter et al. (Hochreiter and Schmidhuber, 1997) proposed a recurrent Long Short Term Memory (LSTM) network with a focus on the inherited regression problem (Lai et al., 2015) in facial landmarking. The network defined by Lai et al. (Lai et al., 2015) uses deconvolution layers which pad out the convolutions, effectively up-scaling the images. The mix of max-pooling, deconvolution and the nature of LSTM, help avoid over fitting while still learning the core features of the facial landmarks position. Other methods, such as Zhou et al. (Zhou et al., 2013) uses a cascading style detector to identify sections of the face and uses the bounding box region to align points to the face. Similarly, Luo et al. (Luo et al., 2012) who also uses a cascade style classifier to retrieve the facial features, but then performs segmentation on the features

instead of using the bounding boxes.

Bulat and Tzimiropouloas (Bulat and Tzimiropoulos, 2017a), focus on using binary convolution neural networks by changing the hierarchy style of the networks to parallel style network. They reduce the amount of data being restricted by processing the previous convolution layers. Their method manages to match the accuracy of existing techniques while reducing the bottleneck and the total required parameters, allowing the network to run on lower powered devices.

2.2 Neural Network Compression

Neural networks can be of varying sizes and scales. The types of layers can drastically increase the amount of processing required to run the network, such as the Inception (Szegedy et al., 2016) and LSTM (Hochreiter and Schmidhuber, 1997) when compared to a convolution layer. This section will describe the work done to compress and analyse neural network performance on mobile devices. The most commonly used methods to compress a neural network is the hashing trick or HashedNets by Chen et al. (Chen et al., 2015). Hashed networks function by grouping random connection weights together into a single 'bucket', the connection weights are all tuned by one parameter reducing the networks total size and memory requirements. However, by joining multiple weight values into one can cause the network to become less diverse, meaning the networks accuracy can be negatively affected. Another method called One-Shot Whole Compression by Kim et al. (Kim et al., 2016) focuses on shrinking the entire convolution based network. Kim et al. (Kim et al., 2016) splits the compression into three main stages:

- Identification. Using Bayesian matrix factorisation the sections of the neural network, which contributes most to the success of the network are identified. By identifying the highest contribution sections, the method can better preserve its accuracy.
- Reduction. Using the ranking from the Bayesian matrix factorisation, the method can apply one of two versions of Tuckers decomposition, the first analyses the core components and then merges them to a single tensor. Whereas, the second performs Single Value Decomposition (SVD).
- Fine Tuning. As the network has been modified, the results produced can differ significantly. Kim et al. (Kim et al., 2016) retrains the network with pre-built models to aid in accuracy recovery.

Overall, Kim et al. (Kim et al., 2016) method allows for large decreases in file size with only small losses in accuracy. Research has also been done to see how different layers, such as recurrent layers can be used to shrink the network size (Robinson, 1994).

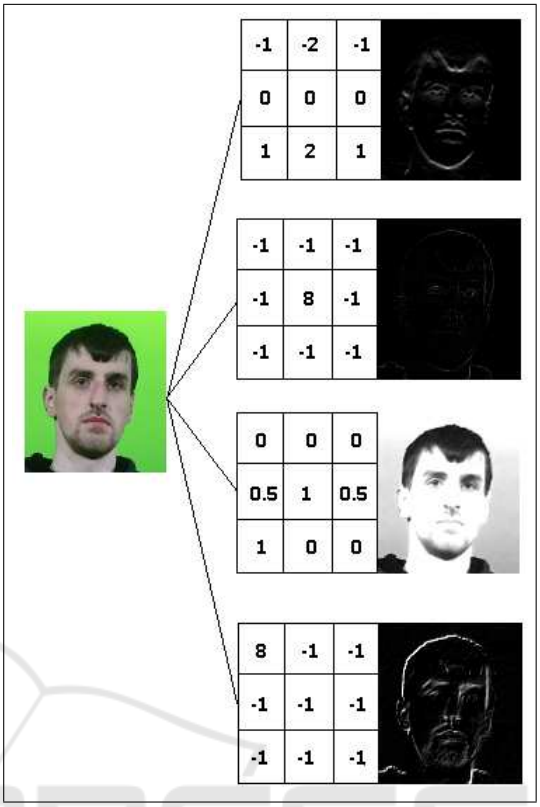

Figure 1: A example of four 3*3 filters.

3 METHODOLOGY

This section will demonstrate and analyse the networks designed for this experiment and give details into the reasoning behind them. After this, a description of the dataset and the elaboration of the preprocessing steps.

To test the processing capability of the devices when implementing neural networks, three basic neural networks have been defined, each with similar layouts designed to maximise the efficiency of the network. Each of the networks follow the same structure as the Base network, as the purpose of this experiment is to discriminate how the max-pooling layer affects the applications capabilities, a simple neural network (henceforth, basic network) is chosen. The basic network consists of three layers: convolutions, max-pooling and activation, followed by two fully connected layers with intermediate activation layers and the final output layer. The initial convolution layer

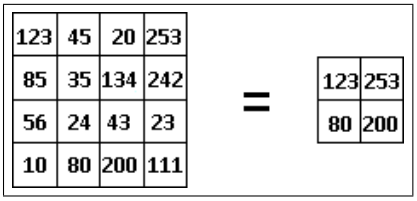

Figure 2: A example of a 2*2 max-pooling layer.

uses a 3×3 kernel, whereas the following convolutions used a 2×2 kernel. The first two convolutions used 32 filters on each image and the last convolution uses 64. Filters are used to widen the neural network as illustrated in Fig 1, they work by creating different convolution kernels for using on the input image. Having multiple filters aid the network, as different filters focus on the type of variation the network could encounter, such as a frontal face filter or side face filter. By having filter account for variation more reliable network can be produced. The max-pooling layers were fixed to 2×2, effectively shrinking the convoluted images by half as shown in Fig 2. The initial input size of the image was 96×96 meaning the final image size for the fully connected layer was 12×12 which equals to the required output of 136 (68 points × 2 for the X and Y). The two activation layers uses 1000 and 500 connected neurons before connecting to the output layer. The goal of these networks is to give a clear view of the cost/reward for implementing max-pooling layer to reduce the workload of the network and how that impacts the networks accuracy. The max-pooling layer were removed starting with the first, as the longer the network maintains the full data the more it can learn. The seed for generating the neurons initial weighting was fixed to 7 to improve reproducibility. Table 1 shows the number of max-pooling layer in each model.

A network with no max-poolings was not considered due to the memory limitations. The memory requirements for training could be improved by reducing the batch size, however, this would have a detrimental effect on the accuracy and increase training times. This is a complicated process, in terms of requirements, as each filter produces an additional image for each layer, so from a single input image, 65536 images of 96*96 are produced (1*32*32*64), this would required 604MB, much higher then what current phones have. For the training stage, even the Titan Pascal (12GB) card with reasonable batch sizes were not possible. The networks were trained over 300 epochs, this is because more data was retained through the removal of max-pooling layer, the longer the networks need to converge and this gave the fairest option for comparison, as this shows when convergence occurs.

For each of the networks, ReLU was implemented as well as RMSE (eq. 1) for loss calculation equation:

$$RMSE = \sqrt{\sum_{i=0}^{n} \frac{(y_i - y'_i)}{n}} \quad (1)$$

where:

- n is the number of samples in the training batches.
- y_i is the ground truth output for the training image.
- y'_i is the predicted output for the training image.

These methods prevent any 'squashing' of the data between 0 - 1, which prevents a fully regressive learning system. The Adam optimizer (Kingma, 2015) was used. The accuracy for the network was calculated using binary accuracy (eq. 2), as with landmarking the chances of early networks getting an exact hit of the landmarks is slim. As a result, the distance for the point is required and this gives a clearer indication of how well the trained model is performing.

$$Accuracy = \frac{\sum(y_i = [y'_i])}{n} \quad (2)$$

where:

- n is the number of samples in the training batches.
- y_i is the ground truth output for the training image.
- $[y'_i]$ is the rounded predicted output of the training image.

The networks were designed and trained using tensorflow (Abadi et al., 2016), thus the tensors were saved in graph form with both weights and structure in the same file. Optimization was then performed to remove the training, testing and none require variables from the graph. The networks were trained using the GTX 1080Ti GPU (11GB) with a batch size of 120. The batch size was determined by modifying the Basic network multiple times and comparing the accuracy which have a high impact. To ensure consistency, all networks were trained with the same batch size and the random weights initializer seed was set to 7 to increase reproducibility. We also split 10% from the training set to be used for validation. The data chosen for validation was performed by the Keras interface.

In this section, we review the methodology behind the training of neural network and the preparation of data. The network was trained on the Morph dataset (Albert and Ricanek Jr, 2008) using 49828 out of the 55134 available within the dataset, images were excluded if they were taken in incorrect lighting, deformed or unrecognisable, had most of the face obscured or failed in both the OpenCV and Dlib (King, 2013) face detectors, as both are commonly used face

Table 1: A comparison of the accuracy and loss for three types of network.

Network	Max-Pooling layers	Accuracy	Loss
Basic	3	0.63	0.17
Two max-poolings	2	0.64	0.12
One max-pooling	1	0.69	0.38

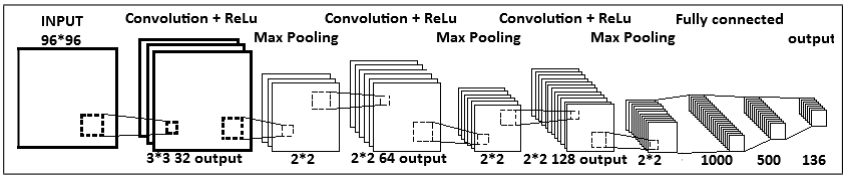

Figure 3: A visualisation of the basic network used for this experiment.

detectors. Images that met the required standard were passed to the next stage of annotation. To speed up the landmark annotation and maintain consistency, which shows to boost system accuracy (Mathias et al., 2014), an automatic approach was taken, similar to (Bulat and Tzimiropoulos, 2017b; Cao et al., 2014). The dataset was loaded into a created program that would first transform all the images into grayscale, the faces were then cropped out using Dlibs face detector and resized to 96×96 images. The images were passed into Dlibs face point detector to retrieve 68 facial landmarks. The points and 96×96 images were saved into a csv file as input data for training. The format for the csv file was face point x coordinate, face point y coordinate for all 68 points on the face and the final cell was the raw 96×96 pixel information with a space between each pixel value. The morph dataset consists of frontal face only and with limited expressions.

The neural networks were trained to take in a 96×96 grayscale face image and process to retrieve the facial landmarks. To retrieve the facial image, the OpenCV library (Corporation et al., 2000) was used as it has high accuracy (Zhu and Ramanan, 2012), the library is compatible to mobile platforms making it suitable for this experiment. The OpenCV library uses a Viola and Jones (Viola and Jones, 2004) style cascade classifier to detect the faces. To prepare the images for the network, the images were first converted to grayscale from RGBA (Red, Green, Blue, Alpha) to reduce the processing time and also resized to 96×96. The resulting image after cropping and rescaling was then processed by the network. The process of preparing data was integrated into the mobile application using the phones backwards facing camera (faces away from the user's face) as the live image feed allowing for the real-time cropping of the face and landmarking.

4 NEURAL NETWORK INTEGRATION

Neural Networks have seen an increase of use in recent years (Learning, 2017; Schmidhuber, 2015), but focusing on how accurate and effective the networks were. Whereas, another major focus should be the implementation of neural networks into consumer and industry based systems. As neural networks are showing increase accuracy and reliability, it shows that many current consumer based applications could be improved by using these techniques. It is possible once the model is trained, to run the model on devices of significantly less processing power as the removal of the back propagation and training layers allows faster computation. However, the memory for processing the image is still required which is still significant amount for a mobile device. Neural networks are stored usually as two separate files. The first file is the model, the networks layout and structure. The second file is the networks weights, which is larger size and controls how the network processes the data. For porting to a mobile device, merging the weights and network structure files is required saving loading times and memory. To run a trained neural network, the device has to be able to load the file into memory, which is an issue on mobile devices as even the most modern mobile devices contain little RAM, such as the iPhone 7 which has 2GB. In addition, tensorflow can have issues with file sizes over 68mb, due to android compression. To load the model onto the device, Tensoflow has a mobile specific libary (Abadi et al., 2016) for android, this allows the device to open a tensor graph containing the model and weights. To build an Android Package (APK), Android Studio (Inc, 2017a) was used with a build targeting the API level 25 on x86 and x64 CPU architecture, this was to ensure compatibility with the Tensorflow library and OpenCV. By using the TensorFlow library, the model

can be loaded onto the device if sufficient memory is available. OpenCV is then used to access the mobile camera to detect faces and pre-process the images for the neural network.

5 RESULTS AND DISCUSSION

Using multiple networks, we showed that to track a large number of points on the face, max-pooling layers can have a detrimental effect on the results of the tracking as facial feature and expression were lost to the degradation of the data. We compare the accuracy and the loss of different max-pooling layers in Table 1.

As the experiments are to determine the effectiveness of how deep learning can be implemented on a mobile device, the file size of the trained net is compared and illustrated in Table 2. The frames per second is also given on the mobile devices as well as the image size before the fully connected layers.

Table 2: Table of the result file sizes.

Network	File size	Image size	FPS
Basic	32.5MB	12×12	15
Two max-poolings	120MB	24×24	N/A
One max-pooling	518MB	48×48	N/A
Face Detector Only	50KB	864×480	17

As shown in Table 2, the number of max-pooling layers has a incremental impact of the resulting weights file. For reference of how this would affect an application, the current Google Play store (Google, 2017) rules for publishing an application are provided. An application on the Play store cannot exceed the files size of 100MB. Compression on the optimized file also cannot be performed as standard methods negatively effect the reading and use of the files. As a result, the file size of the one max-pooling and two max-poolings could not be loaded onto the mobile device. Fig 4 illustrates that as the network begins to retain more data by removing max-pooling layer the network gets significant boosts in accuracy. The increase is much more prominent from One max-pooling to Two max-pooling, this could be as it loses more data through max-pooling compared to Two max-poolings to Basic max-poolings. Fig 5 shows that the more max-pooling layer lowers the loss, but is not as significant as the impact on accuracy. Fig 6 illustrates that with max-pooling layers removed the networks accuracy increases this is much more prominent between Basic max-poolings and Two max-poolings. The reason behind the accuracy drops is that, by using max-pooling on the neural face and large movements can be learned, such as mouth open, because subtle movements and deviation a lost though pooling. However, the method fails to track expressions, such as mouth widen and struggle with accurate placement of the eyes. However, With each improvement we also receive a corresponding drop in performance.

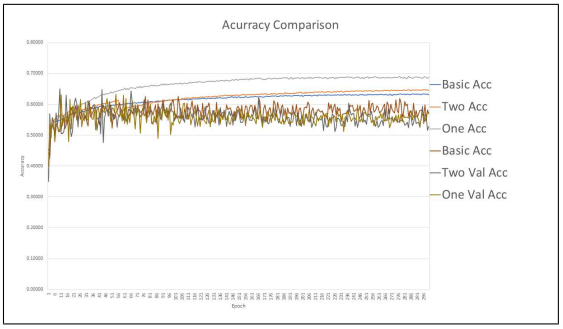

Figure 4: The networks accuracy, both training and validation.

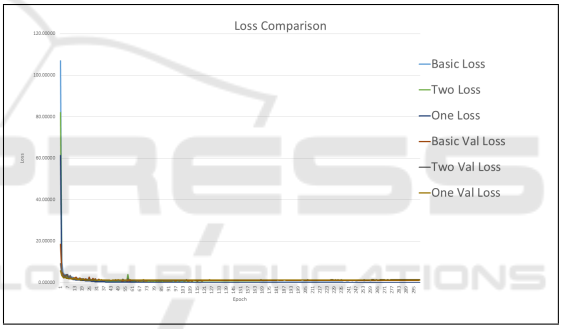

Figure 5: The networks loss, both training and validation.

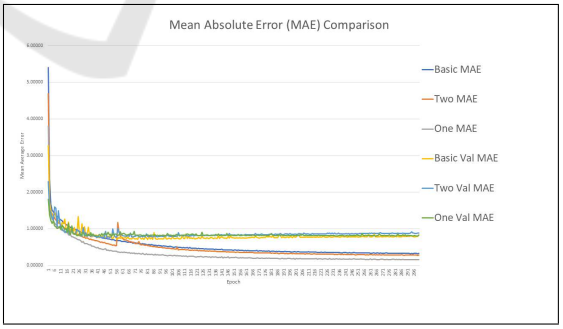

Figure 6: The networks MAE, both training and validation.

We ported each of the models to a HTC 10 mobile phone (4GB Ram, Quad-core (2×2.15GHz and 2×2.16GHZ) CPU, Adreno 530 GPU) to test the effectiveness on live data. The live tests show that even though the basic network runs the fastest it cannot distinguish between many of the facial movement include mouth widen, the system favours the neutral face. As shown in Table 2 all trained models struggle

to reach a real-time landmarking time, but the system also had to perform features, such as the face detection. When comparing the performance of the systems we recommend to implement one max-pooling as it aids performance significantly with little impact on accuracy, giving the best cost/reward trade-off, but further research needs to be done to optimize the size for mobile devices. However, if hardware is restricted and accuracy is not too important, the the two max-poolings will suffice. Image from the live capture is shown in Fig. 7, where we tested in both controlled light conditions and uncontrolled lighting Fig. 8. We also performed a comparison of glasses on/off as in Fig. 9. We provide all codes use to build the android application (Android studio project), codes to train / export (Python 3.5) the model and the models (.pb) used for this experiment publicly available.

Figure 7: Example output of the android application.

6 DISCUSSION

We have demonstrated the effectiveness of basic neural networks on mobile devices and compared the subtle changes in the network design and its effect on the performance, both on the device and accuracy of the network. More processor intensive networks can achieve better accuracy for the system, but its performance on a mobile is unregistered and an open area of research. By testing higher resource required networks on mobile devices and showing their effectiveness in comparison to the one shown in this paper will aid in benchmarking the progression of systems in the future. A major requirement for future work is to perform optimization on the application to increase the frame rate to real time, methods, such as Fagg et al. (Fagg et al., 2017) fast fourier transform face detector, that can run over 60fps freeing up memory and time for the neural network. Other methods could also be tested, such as Ranjan et al. (Ranjan et al., 2017) to allow a neural network to handle everything from face detection to landmarking.

Figure 8: Example output of the android application in a controlled environment(bottom) and wild environment (top), with both male and female faces.

Figure 9: Same Scenario as Fig 8, but with glasses.

7 FUTURE WORK

A future path of our research is to determine how the variation of phone hardware affects the neural networks. Furthermore, this include the phone specific chips that are designed for image processing, such as the chips that allow these “low power” devices to record and play 4K images in real-time, in which some modern day computers have difficulty doing. From observation the training times varied greatly behind the different levels of max-poolings used and if the network implement the regression learning. This would lead to further research to investigate how the different number of max-pooling affect the training time of deep learning networks. A limitation of the system is that it relies on face detector to identify and segment the face for landmarking. Another, limitation is that the network has not been train on none-face images, meaning false detections from the face detector, as the system does not understand none-face images, causes the network to return an average face shape points. A future path of research would be to expand the networks capabilities and training set to include none-face images so this system can handle false positives from the face detector. As even some of the networks shown in this paper could not be ported to the phone for memory issues, future work can be done on

test how the filter sizes effects the results of the network.

8 CONCLUSION

We have demonstrated that the max-pooling layers have high potential to aid in compressing a networks size to operate on mobile devices. Employing max-pooling layers allows networks to become 'lighter' and be trained faster, but with reduced accuracy. With the basic network, the standard learning libraries in networks that don't employ the max-pooling layers have improved results. We have implemented a deep learning network onto a mobile platform, tested the performance on real-world data. We have also show that the outputs on an array of varying neural networks to demonstrate how the networks were effected by the max-pooling layer. By comparing the cost/reward, we recommend the Basic max-pooling network as it has high accuracy, with little impact on the phones memory or processing capabilities for mobile platforms.

REFERENCES

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., Zheng, X., Brain, G., Osdi, I., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. (2016). TensorFlow : A System for Large-Scale Machine Learning. *Osdi*.

Alarifi, J., Goyal, M., Davison, A., Dancey, D., Khan, R., and Yap, M. H. (2017). Facial Skin Classification Using Convolutional Neural Networks. In *Image Analysis and Recognition: 14th International Conference, ICIAR 2017, Montreal, QC, Canada, July 5–7, 2017, Proceedings*, volume 10317, page 479. Springer.

Albert, A. M. and Ricanek Jr, K. (2008). The MORPH database: investigating the effects of adult craniofacial aging on automated face-recognition technology. *Forensic Science Communications*, 10(2).

Bulat, A. and Tzimiropoulos, G. (2017a). Binarized Convolutional Landmark Localizers for Human Pose Estimation and Face Alignment with Limited Resources.

Bulat, A. and Tzimiropoulos, G. (2017b). How far are we from solving the 2D & 3D Face Alignment problem? (and a dataset of 230,000 3D facial landmarks).

Cao, C., Weng, Y., Zhou, S., Tong, Y., and Zhou, K. (2014). FaceWarehouse: A 3D facial expression database for visual computing. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):413–425.

Chen, W., Wilson, J. T., Tyree, S., Weinberger, K. Q., and Chen, Y. (2015). Compressing Neural Networks with the Hashing Trick. *Proceedings of The 32nd International Conference on Machine Learning*, 37:2285–2294.

Chollet, F. (2016). Keras.

Co, X. M. T. (2017). BeautyPlus.

Corporation, I., Garage, W., and Itseez (2000). OpenCV.

Fagg, A., Lucey, S., and Sridharan, S. (2017). Fast , Dense Feature SDM on an iPhone. pages 95–102.

Google (2017). Google Play Store.

Goyal, M., Reeves, N., Rajbhandari, S., Spragg, J., and Yap, M. H. (2017). Fully Convolutional Networks for Diabetic Foot Ulcer Segmentation. *Systems, Man, and Cybernetics (SMC), 2017 IEEE International Conference on*.

"Grother", P. and Ngan, M. (2016). The IJB-A Face Identification Challenge Performance Report.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Inc, G. (2017a). Android Studio.

Inc, S. (2017b). Snapchat.

Kendrick, C., Tan, K., Williams, T., and Yap, M. H. (2017). An Online Tool for the Annotation of 3D Models. pages 362–369.

Kim, Y.-D., Park, E., Yoo, S., Choi, T., Yang, L., and Shin, D. (2016). Compression of Deep Convolutional Neural Networks for Fast and Low Power Mobile Applications. *Iclr*, pages 1–16.

King, D. (2013). Dlib.

Kingma, D. P. (2015). A : a m s o. pages 1–15.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, pages 1–9.

Lai, H., Xiao, S., Pan, Y., Cui, Z., Feng, J., Xu, C., Yin, J., and Yan, S. (2015). Deep Recurrent Regression for Facial Landmark Detection. pages 1–13.

Learning, D. (2017). Deep Learning for Consumer Devices and Services. (APRIL).

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., van der Laak, J. A. W. M., van Ginneken, B., and Sánchez, C. I. (2017). A Survey on Deep Learning in Medical Image Analysis. (1995).

L'Oreal (2017). Makeup genius.

Luo, P., Wang, X., and Tang, X. (2012). Hierarchical Face Parsing via Deep Learning. pages 1–8.

Mathias, M., Benenson, R., Pedersoli, M., and Van Gool, L. (2014). Face detection without bells and whistles. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8692 LNCS(PART 4):720–735.

Megvii, I. (2015). Face Plus Plus.

Ranjan, R., Sankaranarayanan, S., Castillo, C. D., and Chellappa, R. (2017). An All-In-One Convolutional Neural Network for Face Analysis. pages 17–24.

Robinson, A. J. (1994). An Application of Recurrent Nets to Phone Probability Estimation. *IEEE Transactions on Neural Networks*, 5(2):298–305.

Schmidhuber, J. (2015). Deep Learning in neural networks: An overview. *Neural Networks*, 61:85–117.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Sudha, S. (2016). an Automatic Classification Method for Environment Friendly Waste Segregation Using Deep Learning. (Tiar):65–70.

Szegedy, C., Ioffe, S., and Vanhoucke, V. (2016). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *Arxiv*, page 12.

Viola, P. and Jones, M. J. (2004). Robust real-time face detection. *International journal of computer vision*, 57(2):137–154.

Yi, D., Lei, Z., Liao, S., and Li, S. Z. (2014). Learning Face Representation from Scratch. *arXiv*.

Zhang, Z., Luo, P., Loy, C. C., and Tang, X. (2016). Learning Deep Representation for Face Alignment with Auxiliary Attributes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(5):918–930.

Zhou, E., Fan, H., Cao, Z., Jiang, Y., and Yin, Q. (2013). Extensive facial landmark localization with coarse-to-fine convolutional network cascade. *Proceedings of the IEEE International Conference on Computer Vision*, pages 386–391.

Zhu, X. and Ramanan, D. (2012). Face detection, pose estimation, and landmark localization in the wild. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2879–2886.

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS