Facial Keypoints Detection

Sara Kartalovic
sara.kartalovic@studenti.unipd.it

Sercan Albut
sercan.albut@studenti.unipd.it

Abstract

Facial keypoints detection is a machine learning and computer vision problem whose solutions have many different applications. It can be used to track and capture human faces in images and videos, and to recognize faces as well as facial expressions. It is a crucial component for face recognition technologies. The problem we face includes the prediction of 15 facial keypoint coordinates on a given input image of different human faces. Facial keypoints recognition is challenging in the sense of machine learning because people have different facial features. With the rise of Deep Learning features the problem became more interesting because Convolutional Neural Networks are proven to be very effective when it comes to solving facial recognition problems. With this being said, in this project we are introducing and sampling state-of-the-art deep convolutional neural network approaches starting from AlexNet on the task of facial keypoint detection problem. Models are trained and tested on 96x96 grayscale images and to detect the facial keypoints, deep learning techniques have been used.

1. Introduction

The scope of this project is facial keypoints detection on images of human faces. With the new developments in the technology, facial keypoints recognition task has become crucial in today's world as it is used widely on CCTV, ImageNet searches, basically involving everything which takes into account the extraction of keypoints on visuals. The reason why this detection problem is challenging is because people have different face features; moreover, the image itself can make it challenging depending how, when and where it was taken. Some of the challenges that are present not only in this problem, but in many computer vision problems, are: illumination, scale, viewpoint, and position. To deal with the problem, deep convolutional neural networks models will be used to solve the problem and find the mappings between the image pixels and facial keypoints. Different face recognition and visual key point extraction methods and applications will be briefly explained before experimentation. The implementation will start with a basic model which is considered as a cornerstone of the process: AlexNet. Later, it will be continued by introducing more complex and deeper models by adopting some VGGnet properties. The experiments, implementations and improvements will be completed on a publicly available data set. The process of obtaining the deep convolutional neural networks will be explained as it has been used as layerwise and basic structures of itself. At the same time there will be a process of the evaluation of the data and model. Since the task is widely used and still in process of improvement, the experiments that have been conducted in this paper will have different approaches with the aim of increasing the accuracy and decreasing the loss of the model and detection.

Components of network	Number and specifications
Convolution layer	Number of layers: 15 Filters: 32, 32, 64, 64, 96, 96, 128, 128, 256, 256, 512, 512, 1024, 1024, 1024 Kernel size: 3x3 Activation: Leaky ReLU
Dense layer	Number of layers: 1
Batch Normalization	Applied after each convolutional layer
Max-pooling layer	Number of layers: 5 Size: 2x2

Table 1. Description of the architecture of the network

Table 1 shows the structure and architecture of the

model that we have obtained after conducting numerous tests. This model showed a good performance when it comes to predicting the coordinates of the facial keypoints during our experiments and was the model that provided the highest accuracy as well as the lowest loss. The path of obtaining this model is explained in this paper.

2. Related work

Facial keypoints recognition has been a topic of research for a while. There are numerous models and applications that are related to this task. Previous work includes approaches using deep learning and CNN which is the structure that we are going to exploit as well. In this section, we describe two applications and different artificial neural networks that have been used to recognize faces and facial features.

2.1 FaceReader application

An application FaceReader, for example, is used for emotion analysis. Without any bias, this app records the primary reactions of the test subject and provides a validated data analysis of facial expressions. The unique feature of this application distinguishes the intensity of the active muscles by using the groups of facial muscles called 'action units.' This is performed for the left and the right side of the face separately.

The application follows the 4 main steps. At the first step it uses Viola-Jones algorithm to find a face in the picture. Next, it uses 500 facial keypoints to make precise 3D modeling of the face. At the third step, it uses Artificial Intelligence and Deep Learning to analyze the face in detail. Furthermore, at the last step, an artificial neural network is used for classifying the emotional facial expression.[1]

It is reported that FaceReader 6, according to their validation study, achieves the accuracy of 88%, whereas a newer version of the application named FaceReader 8.1 achieves much higher accuracy of 96%. According to Google Scholar, so far FaceReader is the most cited facial expression recognition software in nearly 1,300 publications since 2005. Moreover, more than 1,000 universities (including 6 out of 8 Ivy League universities), research institutes, and companies in many markets such as psychology, consumer research, user experience, human factors, and neuromarketing use this application.

2.2 Luxand FaceSDK application

FaceSDK is another application that uses facial keypoint detection for different purposes. "FaceSDK is a high-performance, multi-platform face recognition, identification and facial feature detection solution." [**2**] It is used for face recognition, live video recognition, face detection as well as mask-on face detection, facial keypoints recognition, facial expression recognition, gender and age recognition, IP camera support and thermal face detection.

When it comes to 70 facial points recognitions, the SDK processes the image to find a human face on it. Then, it returns the coordinates of the 70 unique facial keypoints. These points include the eyes, eye contours, eyebrows, lip contours, nose tip, and others. The detection of keypoints works in real-time on desktop or mobile, which allows this software to be used on video for tracking and transformations of facial features.

2.3 Convolutional neural networks (CNN)

One sort of artificial deep neural network that is used particularly for image recognition, classification and object detection is Convolutional neural networks (CNN). Because our dataset consists of images and CNN's have proven that they can be successfully used for solving the kind of problem that we are facing, we decided to exploit this structure and use CNN to correctly recognize facial keypoints in each image. One of the reasons why CNN's are so successful is because they extract features directly from the image and these features are learned while the training is performed. For instance, the Alex Net, LeNet, VGG, GoogleNet, ZFNet and ResNet have been used to solve the facial keypoints detection problem. [3] Using AlexNet and VGGNet as examples, we try to create our own CNN architecture to detect keypoints in the facial images.

"Network framed by Alex Krizhevsky, et al. AlexNet has been trained and tested for classification of high-resolution images, with results better than conventional feature extraction methods like SIFT. "[3] The part of AlexNet has been used, for example, to improve vehicle image detection and classification [4]. Also, more recently, AlexNet features are applied in order to get better classification of scenes. [5] This network is composed of 5 convolutional layers, 5 ReLU layers, and 5 max pooling layers which are followed by 2 fully connected dense layers. [3] "Layer1 gives edge and blob of the input image, layer2 performs the conjunctions of these edges or responds to corners and other edge or color conjunctions, layer3 output is texture of a image, Layer5 identifies object parts", where the last two fully connected layers give the image features and are used as feature descriptors where region around the keypoint of image is detected using Scale-invariant feature transformer or shortly SIFT.[3] The scale-invariant feature transform is a feature detection algorithm to detect and describe local features in images.

Karen Simonyan and Andrew Zisserman proposed VGGNet in their paper *Very deep convolutional networks for large-scale image recognition* [6]. Since it has been introduced to the public, it is still accepted as a state-of-art model for specifically image classification. VGG, which is short for Visual Geometry Group, or another known name as

oxford model, has a key value over the definition and repeated usage of VGG-blocks to clarify the image filtering. The concept is basically starting with as small as 3x3 pixel filtering followed by max pooling layer. In their research paper of Very Deep Convolutional Networks for Large-Scale Image Recognition, 2014, the process has been basically defined as: "The image is passed through a stack of convolutional (conv.) layers, where we use filters with a very small receptive field: 3 x 3 (which is the smallest size to capture the notion of left/right, up/down, center). […] Max-pooling is performed over a 2x2 pixel window, with stride 2.". The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. The model has made improvements over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. The VGG Net-16 consists of 16 convolutional layers, and it is considered appealing because of its very uniform architecture. In truth VGGNet consists of 138 million parameters, which at first could be hard to handle.

3. Dataset

The dataset used for the project is available on Kaggle and comes from the Facial Keypoints Detection challenge where the goal is to detect the location of keypoints on face images. [7] The dataset is split into train and test sets which include lists of 7049 and 1783 images respectively. The images used only one channel and were grayscale where each image was 96*96 pixels as shown in Figure 1. They were composed of a list of pixels which are ordered by row, as integers in range (0,255). For the training dataset, the coordinates for 15 keypoints are also provided along with images. As shown on Table 2, those 15 keypoints represent different elements of the human face such as: center and corners of both eyes, inner and outer ends of the eyebrows, tip of the nose, corners of the mouth and center of the top and bottom lips. On the Table 2 and in the dataset, the left and right are the sides from the view of the subject. Because these 15 coordinate features are in the form of (x, y), there are 30 points features for each image of the face.

The facial keypoints	
Left eye center	Right eye center
Left eye inner corner	Right eye inner corner
Left eye outer corner	Right eye outer corner
Left eyebrow inner end	Right eyebrow inner end
Left eyebrow outerend	Right eyebrow outer end
Left mouth corner	Right mouth corner
Top lip mouth center	Bottom lip mouth center
Nose tip	

Table 2. The 15 facial features

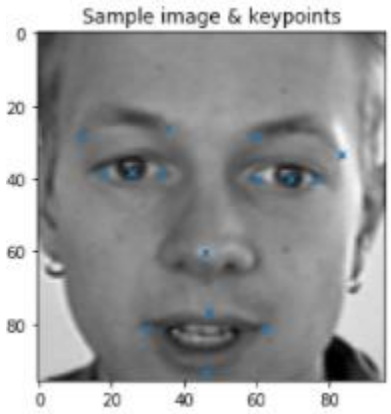

Figure 1. Sample of training image of size 96x96 and corresponding 15 facial keypoints

Immediately after loading the datasets, the images were scaled. The training dataset had numerous missing facial keypoints where 4909 rows have at least one missing keypoint. After removing the data with missing keypoints coordinates we were left with less than half of the training dataset which now consisted of 2140 images that include all 15 keypoints.

In computer vision and machine learning tasks it is almost always better to have more data to experiment with. We modified our original dataset to make it more flexible to variations and obtain more training samples. Firstly, we flipped pictures horizontally from left to right which doubled our training dataset. This way the x coordinates were flipped, while y coordinates remain the same. Then, we rotated pictures along with the keypoints for 12 degrees to the left and continued with changing the brightness of the pictures. The brightness was increased by factor 1.5 and decreased by factor 0.8. The images and corresponding key points were shifted to bottom, up, right and left and this way 6350 new images were obtained. Lastly, we added the random normal noise to the original training image. After all alterations we ended up with a much larger data set for training which included 26239 images and corresponding keypoint features.

4. Methods

After thoroughly analyzing the dataset and creating more data to train our model on, we proceeded by exploring the work that was done to solve this problem. Because it has been proven that deep convolutional neural networks work well when applied to computer vision problems, we decided to take this approach. Also, because we had 26239 samples in our training dataset, as a starting point AlexNet was being explored and ultimately few adoptions were taken from VGGnet. Also, it has been considered to use deeper VGGnet as our starting point; however, it was impossible to know if we would obtain better results and prediction by using

exactly 16 convolutional layers or more. We wanted to see how the model would behave while making deeper networks, so using model that had similar architecture as AlexNet was a good start. Also, it is easier to notice properties and behaviors of the model on relatively smaller networks. Throughout the experiment, this proved to be a good decision because the model that we explored more and that gave us better results than VGGnet-16 is a CNN consisting of 15 convolutional layers. By trying to find the middle between complexity and accuracy while keeping prediction loss low, we started with a less complex model. Eventually, to obtain better results, we explored and created a deeper and more complex neural network.

On the first step before creating and testing any models, 15% of our training data was used for creating test set. This set is going to be used to test models on unseen data and measure its accuracy and loss. After the split, training dataset contained 22303 pictures and corresponding coordinates of the facial keypoints, while test set contained 3963 samples.

At the beginning, for the sake of comparison, all models were trained using 100 epochs. Moreover, 90% of the data was used for training, while the remaining 10% were used for the validation purposes. When improved models were found, the number of epochs was increased to 150 so the models could benefit from it.

Next, when started with testing, the property that was adopted from VGGnet at the very beginning is the kernel size of 3x3 on all convolutional layers. The reason why this was done is because the kernel size of AlexNet was 11x11 and 5x5 on the first two layers and this was too big if we consider that our pictures were 96x96 grayscale images and AlexNet takes as input 224x224x3 images.

Throughout experimentation, we monitored accuracy and used ‘mean squared error loss’ to compare our models and see if there is an improvement. The loss function was used because we are comparing the predicted coordinates of the keypoints with the actual coordinates. Accuracy and loss curves were used to monitor if there is overfitting present and if the learning of the model is stable and consistent.

5. Experiments

The main goal was minimization of the loss and obtaining high accuracy which would mean that our model has good performance and recognizes facial keypoints well. In our experiments we created models by using Keras library and started with a neural network model that consists of 5 convolutional layers followed by 3 dense layers which is modeled after AlexNet and continued experiments by creating deeper CNNs and adopting features from VGGNet.

The optimizer used for all models is ‘adam’ which is “an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments.” [8] Testing different batch sizes did not influence the model very much, so we decided to use a smaller batch of size 64. To stabilize and improve the learning process, batch normalization is added to standardize the input to each layer for each mini batch.

The first activation function we tried to incorporate in our model was ‘ReLU’. However, to obtain smaller loss and make better model, we discovered that ‘Leaky ReLU’ works better for solving the given task. It was found that this activation speeds up the training process a lot and makes models more accurate. It has previously been proven that this activation is good for models that suffer from sparse gradients or exploding gradients. This specific activation function is used as an alternative and to face the problem called ’dying ReLU’ where the neuron is considered to be ‘dead’. This neuron is ‘dead’ if it always gives the output 0 and is stuck on the negative side. Additionally, activation is not applied on the final output dense layer because we need directly the coordinates of facial features keypoints.

In the experiments we also used a technique called callbacks.ReduceLROnPlateau; it simply inherits from callbacks and reduces the learning rate when a metric has stopped learning. Models often benefit from reducing the learning rate by a factor of 2-10 once learning stagnates. This callback monitors a quantity and if no improvement is seen for a 'patience' number of epochs, the learning rate is reduced.

Also, another technique called early stopping is used. Early stopping is a method that allows you to train a large number of training epochs and stop training once the model performance stops improving on a holdout validation dataset. It is used to determine the choice of the number of training epochs to be used.

When it comes to creating a deeper network and discovering a model that would have a good performance, we started adding more convolutional layers and monitoring the results until we reach a similar architecture as VGGnet that is explained in section 2.3. Table 3 shows the results that were obtained by testing models that have different number of convolutional and max-pooling layers. By looking at the Table 3, it can be seen that there is an increase in accuracy after adding convolutional layers. The accuracy was at its peak point while using 15 convolutional layers and dropped significantly after switching to 16 convolutional layers and this was one of the reasons why we chose to explore model that uses 15 convolutional layers. Also, another reason was a big jump when it comes to loss. This model had better performance and provided smaller loss of 0.7246 and accuracy of 91.99% compared to the first model that was built after AlexNET model. Moreover, when it comes to accuracy, the model that we chose is very close to the model using 14 convolutional layers, but if we look at the loss, our model has smaller loss. This is another reason why we chose model using 15 convolutional layers.

After making a decision which model to exploit, we increased number of epochs to 150. To get the best results, we experiment with the number of dropout layers as well as dropout rate, dense layers, and max pooling layers.

Model	val_loss	val_acc	test_loss	test_acc
5cl, 3mpl	1.3782	0.8898	1.5209	0.8915
6cl, 3mpl	1.461	0.8978	1.334	0.9017
7cl, 3mpl	0.9367	0.9131	1.2849	0.9009
8cl, 3mpl	1.5881	0.8961	1.3861	0.9017
9cl, 3mpl	1.1488	0.8987	1.1798	0.9103
10cl, 3mpl	1.788	0.9185	1.1013	0.9096
11cl, 3mpl	1.3769	0.914	1.3842	0.9162
12cl, 3mpl	0.6517	0.922	0.9463	0.9192
13cl, 3mpl	1.4289	0.9149	1.0213	0.8991
14cl, 3mpl	0.9924	0.9068	1.2149	0.8996
14cl, 4mpl	0.6926	0.9229	0.9192	0.9194
14cl, 5mpl	0.7169	0.9122	0.7051	0.9184
15cl, 5mpl	0.7337	0.9337	0.7246	0.9199
16cl, 5mpl	2.3566	0.845	2.7675	0.827
16cl, 6mpl	7.1156	0.7536	5.9656	0.7581
16cl, 7mpl	3.5023	0.7867	3.5831	0.7919

Table 3. Validation and testing results obtained by testing models with of different size

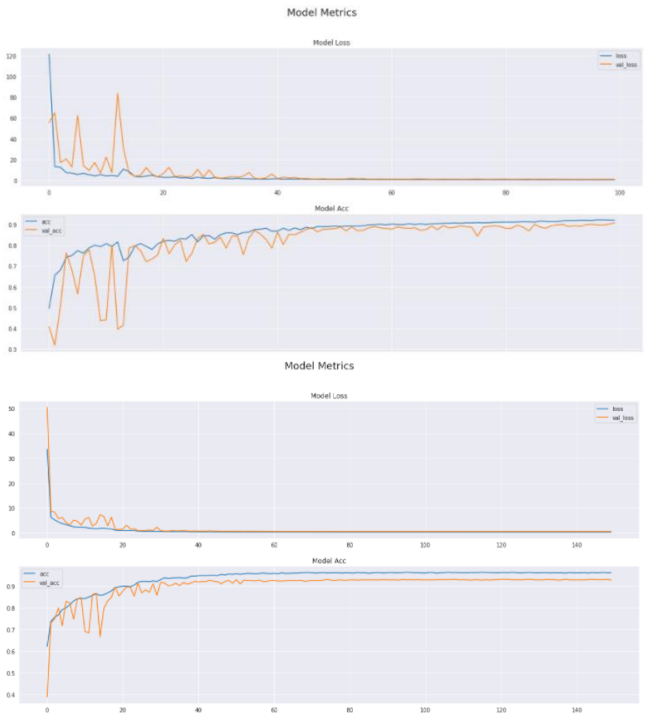

Figure 2. Plots showing the difference between the models using 3 max pooling layers (on the top) and 5 max-pooling layers (on the bottom)

Table 3 also shows that by adding 2 more max-pooling layers, the model started performing better. These 2 additional layers did not only improve performance but also made learning more stable. Figure 2 demonstrates how unstable the learning of our model is before we introduce 2 additional max-pooling layers. The top two plots show that it takes more time for the model to become stable and the curve is drastically increasing and decreasing on the beginning. The two plots on the bottom show that model needs more time to reach consistency when learning and has better performance overall.

Next step in the experiment was adding dropout layer and testing different dropout rates. This was done to reduce overfitting and improve the performance of the model even more. Without using dropout layer models show overfitting, smaller accuracy, and bigger loss. Table 4 shows how the previously mentioned model behaves when added dropout layer with different dropout rates. It is easy to see that model performs drastically better with dropout layer in terms of loss and accuracy was better also. Additionally, Table 4 shows that model performs the best when using dropout rate 0.1 so we adopted the dropout layer with dropout rate of 0.1 for our model.

Dropout rate	val_loss	val_acc	test_loss	test_acc
No dropout	1.2837	0.914	1.0102	0.9205
0.1	1.2842	0.9274	0.6964	0.9238
0.2	0.7538	0.9247	0.8698	0.9182
0.3	0.7842	0.9158	0.9723	0.9062

Table 4. Results obtained by model using different dropout rates

When it comes to dense layers, our first model was based on AlexNET which includes 3 dense layers. We wanted to see how our model would behave if we removed some of these layers and the results are reported in the Table 5. Table 5 shows that the model that has highest test accuracy of 93.09% is model using only one dense layer. This model also has the smallest mean squared average loss of 0.6964.

Number of layers	val_loss	val_acc	test_loss	test_acc
3	1.2691	0.9059	0.9305	0.9238
2	0.9777	0.8862	1.2785	0.8941
1	1.2842	0.9274	0.6964	0.9309

Table 5. Results obtained by model using different number of dense layers

6. Conclusion

To conclude, the focus of this project was prediction of the coordinates of 15 facial keypoints on raw facial images. Particularly, we would map the coordinates of keypoints to the raw pixels on 96x96 grayscale pictures. So, the purpose was to find out how well this deep convolutional neural network will learn facial keypoints and detect them on unseen data. The AlexNet architecture was used as a baseline, but since there was a need for a deeper model, we experimented with different deeper CNN structures and try

to exploit them. Furthermore, different data augmentation techniques were performed such as scaling, rotating, flipping, shifting, changing the brightness of the picture, and adding noise.

Starting from a neural network that has 5 convolutional layers and 3 dense layers, we provided a deeper artificial neural network which has 15 convolutional layers, 5 maximum pooling layers of size 2x2, 1 dense layer and 1 dropout layer with dropout rate 0.1.

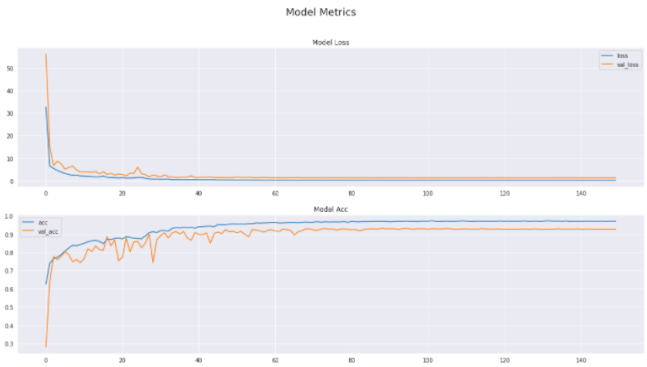

Figure 3. Validation and accuracy plots of the final model

Figure 3 shows the curves of our final model. It can be seen from the plots that as loss decreases the accuracy increases. The distance between curves states that adjusted learning rate is fit for the model. Furthermore, the very small difference between loss curves indicates that there is no overfitting present. The training is relatively stable due to the batch normalization after each convolution layer and max-pooling layers.

Figure 4. Visualizing test predictions

Figure 4 represents two pictures from our testing data set that was provided on the beginning. Blue dots represent the 15 facial keypoints that our model predicted. As we can see, the model performed well even though there is a difference in brightness, facial expression, and angle from which the pictures were taken.

After experimentation and through many processes, the results proved that deeper convolutional neural networks dealt better with detection and prediction of facial keypoints which was our main objective. In future, deeper and different architectures of convolutional neural networks should be explored and tested in order to improve the overall performance and solve the challenge more accurately and efficiently.

References

[1] *Facial expression recognition software: FaceReader*. Facial expression recognition software FaceReader.(n.d.).https://www.noldus.com/facereader.

[2] *Detect and Recognize Faces and Facial Features with Luxand FaceSDK*. Luxand. (n.d.). https://www.luxand.com/facesdk/.

[3] K.Kavitha, B. Thirumala Rao, & B. Sandhya. (2016). *Evaluation of Distance Measures for Feature based Image Registration using AlexNet*., Vol. 9, No. 10

[4] Yiren Zhou, Hossein Nejati, Thanh-ToanDo, Nagai-Man Cheung, Lynette Cheah, *Image-based Vehicle analysis using Deep Neural network: A systematic Study*, IEEE International conf. on digital signal processing, 2016.

[5] Lisha Xiao, Qin Yan, Shuyu Deng, *Scene Classification with improved AlexNet Model*, 12th international conf. on intelligent systems and knowledge engineering, 2017.

[6] Simonyan, K., & Zisserman, A. (2014). *Very Deep Convolutional Networks for Large-Scale Image Recognition*, 1556.

[7] A. Krizhevsky, I. Sutskever, G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, in: F. Pereira, C. J. C. Burges, L. Bottou, K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 25*, Curran Associates, Inc., 2012, pp. 1097–1105.

[8] Gao, R., & Liu, Q. (2018). Facial Keypoints Detection with Deep Learning. Journal of Computers, 13, 1403–1410. https://doi.org/http://www.jcomputers.us/vol13/jcp1312-07.pdf