CS 543 Final Project: Facial Keypoints Detection using Deep Learning

Mahshid Mansouri (mm64), Alisina Bayati (abayati2), Shivani Ghanta (sghanta2)

1. Introduction

Facial recognition in images is at the forefront of computer vision problems given its social and cultural implications. Identifying key points in an image is a critical part of many applications, such as facial expression classification, facial alignment, tracking faces in videos as well as applications for medical diagnosis [1]. Facial key points refer to the main features on a human face, such as the eyes, nose, lips, or eyebrows. Once accurately detected, these key points can be utilized to train deep learning algorithms to perform a range of classification tasks. However, doing this accurately and efficiently is difficult as facial features may vary greatly from one individual to another, and even for a single individual, there is a large amount of variation due to the 3D pose, size, position, viewing angle, and illumination conditions.

Facial recognition is an important source of information for the extensive work being done in the areas of activity understanding [2] and person tracking [3]. Many studies have also begun to focus on facial expressive analysis to gather affective state [4] or for driving character animations, specifically in MPEG-4 compression [5]. The recognition of visual speech is another rapidly growing field of interest in image processing [6], as seen in the Face Recognition Technology (FERET) tests from Jonathan Phillips [7], which provide an early benchmark of facial recognition. Phillips' report in Face Recognition Vendor Tests [8] evaluates face systems for US government agencies and reveals how performance of current technologies is excellent under ideal conditions but under conditions of changing illumination, expression, resolution, distance or aging, performance begins to degrade dramatically [9]. Deviations from the ideal face image acquisitions reveal the weaknesses of many of these current technologies, highlighting the need for robust facial recognition systems.

Many techniques to detect facial keypoints have emerged in the last decade, including utilizing local image features [10], random forest classifiers [11], and even SVMs [12]. To solve the issue of local minima caused by ambiguity and data corruption in problematic image samples, cascading convolutional neural network architectures [13] and gabor filters [14] were introduced. While these techniques performed fairly well, more refined component detection was required, which gave rise to more regression based approaches. Valstar et al. [15] used Markov Random Fields with support vector regressors, which enabled efficient and robust detection even with changes in expression and pose, with conditional regression forests outperforming regression forests [16]. Among the different architectures, The 2014 ImageNet challenge revealed VGGNet [17] to be the best suited for the task of facial recognition, which we will be analyzing later in this paper.

In this project, we will use one of the Facial Keypoint Detection datasets from Kaggle [18] to identify important facial key points on a set of face images using deep learning algorithms. We plan to use a set of training images dataset with premarked facial key points to train a deep learning model and test the model on a set of testing datasets to evaluate the model prediction accuracy. Additionally, we aim to augment the image dataset and explore and improve the performance of an existing pre-trained deep-learning model [19] on the augmented dataset. We will augment the existing testing dataset by adding noise and rotation to the images, evaluating the performance of the trained model on the augmented dataset, and improving the model if needed to make it as robust as possible.

Taking into consideration the feedback from the initial project proposal, we will also study different facial keypoint detection architectures and losses to improve the prediction results. We plan to implement a baseline Convolutional Neural Network (CNN) as proposed in [20], a baseline Neural Network (NN) architecture, a 5-layer LeNet with dropout, and a Simple VGGNet model as proposed in [20] to compare against the baseline CNN model that we adopted. We intend to analyze factors such as model complexity, computational cost, and the training and validation losses to decide which architecture is most optimal in solving this facial key points detection problem.

2. Details of the Approach

2.1 Data

2.1.1 Original Dataset

We used the available dataset on Kaggle website [18] which contained a list of 7049 training images and 1783 test images in the format of csv files. Each training image included 15 pre-marked facial keypoint locations shown in red circles (e.g., right eye corner, left eye corner, etc.). For the training images, each row of data contained the (x,y) coordinates of the 15 keypoints, and the image data as a row-ordered list of pixels. For the test images, each row contained the image ID and the image data as row-ordered list of pixels.

2.1.2 Augmented Dataset

To augment the original dataset, random zero-mean gaussian noise with the standard deviation of σ together with a rotation with the angle of α were added to all the original images dataset, resulting in two types of data augmentation: 1) Type I augmentation in which α and σ are sampled from uniform distributions from -10 and 10 degrees, and from 0 to 5, respectively and 2) Type II augmentation where the rotation angle is chosen uniformly from -15 to 15 degrees for every image in the dataset and σ can vary from 0 to 10. The augmentation function would take a training image and facial keypoints and return a tuple with the transformed image and training points as described mathematically below.

$$For\ Type\ I\ augmentation: \quad \alpha \sim U([-10, 10]), \quad \sigma \sim U([0, 5])$$

$$For\ Type\ II\ augmentation: \quad \alpha \sim U([-15, 15]), \quad \sigma \sim U([0, 10])$$

An example of the data augmentation is as below:

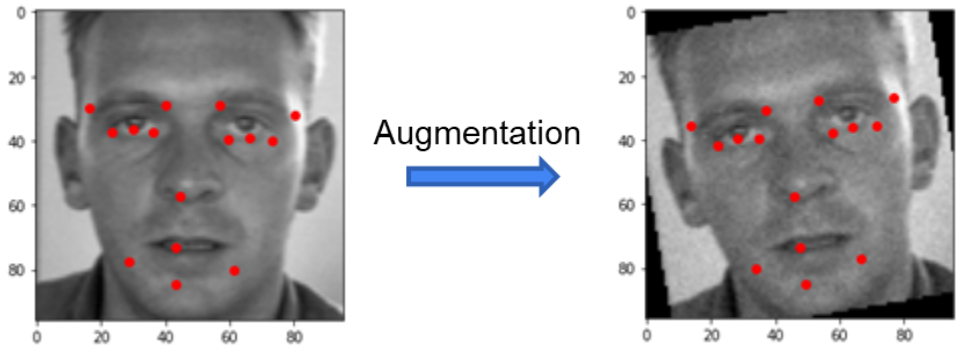

Figure 1. Left: An example of a training image with the pre-marked facial keypoints, Right: The augmented training image with added noise and rotation.

2.2 Deep Learning Architectures

Four different deep learning architectures were tested on the original and the two types of the augmented datasets. Each dataset was split into two parts: 1) training set which included 80% of the data, and 2) validation set which included the remaining 20% of the data. For each architecture, we plotted the training and validation losses after 20 epochs and displayed the facial keypoints prediction results on some sample test images. The details of each architecture will be explained in the following.

2.2.1 Baseline Convolutional Neural Network (CNN)

To start off, we first adopted the CNN architecture proposed in the original code [19] (Table 1). This model consists of a series of alternating convolutional, max pooling, and dropout layers followed by a dense layer at the end.

Table 1. Baseline CNN architecture

Layer	Name	Size
0	input	1×96×96
1	conv2d1	64×94×94
2	maxpool2d2	64×47×47
3	conv2d3	128×45×45
4	dropout4	128×45×45
5	maxpool2d5	128×22×22
6	conv2d6	256×20×20
7	drouput7	256×20×20
8	maxpool2d8	256×10×10
9	conv2d9	512×8×8
10	droupout10	512×8×8
11	maxpool2d11	512×4×4
12	conv2d12	1024×2×2
13	droupout13	1024×2×2
14	maxpool2d14	1024×1×1
15	dense15	512
16	Output	30

2.2.2 Baseline Neural Network (NN)

The next architecture that we used was based on [20], with an intent to identify a “lower bound” or a simple baseline, to compare other architectures to. This basic network consisted of a single fully-connected hidden layer, with 500 neurons detailed in Table 2.

Table 2. Baseline NN architecture

Layer	Name	Size
0	input	1×96×96
1	hidden	500
2	output	30

2.2.3 Simple VGGNet

We also adopted and implemented a simple VGGNet convolutional neural network proposed in [20], consisting of several convolutional layers followed by single max pool layers, with fully connected layers and dropout (Table 3).

Table 3. Simple VGGNet architecture

Layer	Name	Size
0	input	1×96×96
1	conv2d1	64×94×94
2	conv2d2	64×92×92
3	maxpool2d3	64×46×46
4	conv2d4	128×44×44
5	conv2d5	128×42×42
6	maxpool2d6	128×21×21
7	conv2d7	256×19×19
8	conv2d8	256×17×17
9	maxpool2d9	256×8×8
10	dense10	512
11	dropout11	512
12	dense12	512
13	droupout13	512
14	output	30

2.2.4 5-Layer LeNet with Dropout

Lastly, we adopted the 5-LayerLeNet with Dropout architecture proposed in [20] consisting of alternating convolutional layers and max pooling layers, followed by fully connected hidden layers, with filter size 3×3, pooling size of 2×2 with stride 1. All hidden layers contained 1000 neurons (Table 4).

Table 4. 5-Layer LeNet with Dropout architecture

Layer	Name	Size
0	input	1×96×96
1	conv2d1	16×94×94
2	maxpool2d2	16×47×47
3	dropout3	16×47×47

4	conv2d4	32×45×45
5	maxpool2d5	32×22×22
6	dropout6	32×22×22
7	conv2d7	64×20×20
8	maxpool2d8	64×10×10
9	dropout9	64×10×10
10	conv2d10	128×8×8
11	maxpool2d11	128×4×4
12	dropout12	128×4×4
13	conv2d13	256×2×2
14	maxpool2d14	256×1×1
15	dropout15	256×1×1
16	dense16	1000
17	dense17	1000
18	dense18	1000
19	dense19	1000
20	Output	30

3. Results

For each of the three data types (i.e., original, Type I augmented , and Type II augmented), the training and validation losses for each architecture were plotted and facial keypoints prediction results for some sample test images were displayed (Fig. 2-7). For all architectures and for all datasets, the final validation loss (i.e., after 20 epochs) was higher than the training loss which could be a sign of overfitting (Fig. 2-4 and Table 5).

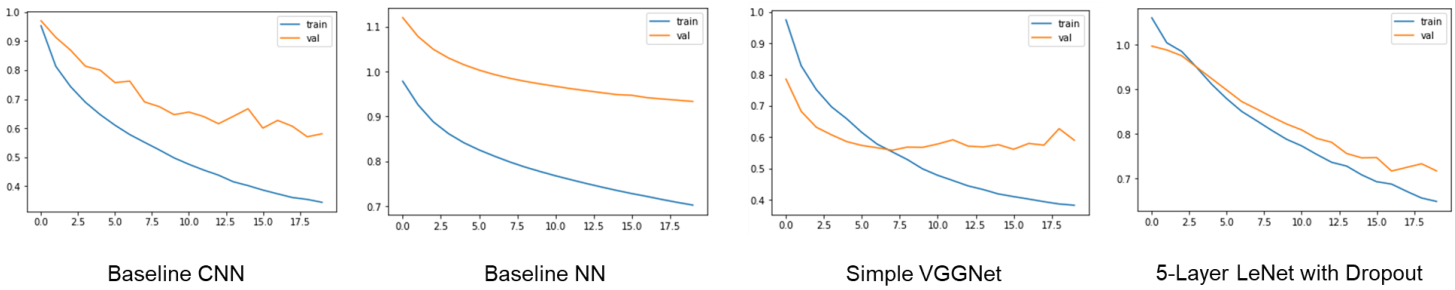

Figure 2. Training and validation losses for the original dataset for different architectures

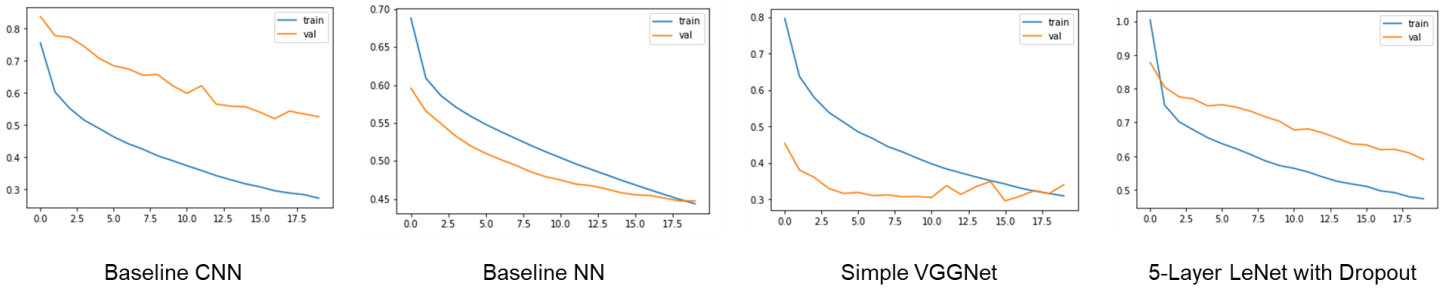

Figure 3. Training and validation losses for the Type I augmented dataset for different architectures

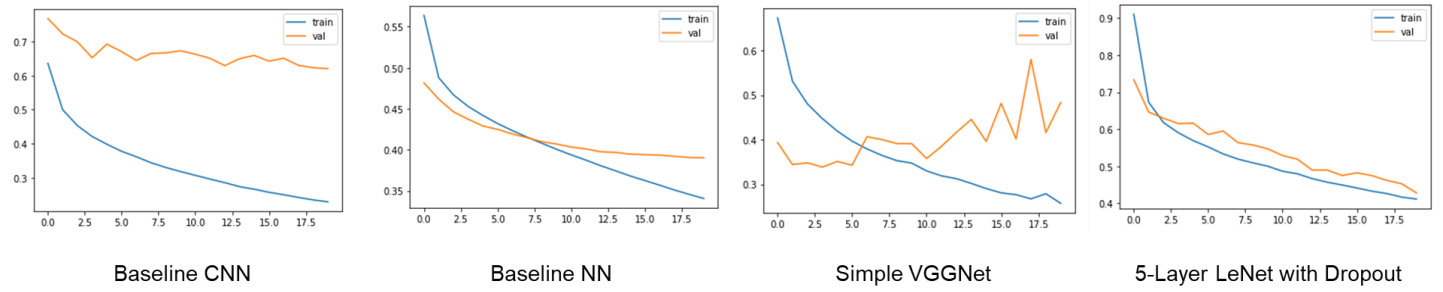

Figure 4. Training and validation losses for the Type II augmented dataset for different architectures

Figure 5. Sample facial keypoints prediction results for some of the test images for the original dataset for different architectures.

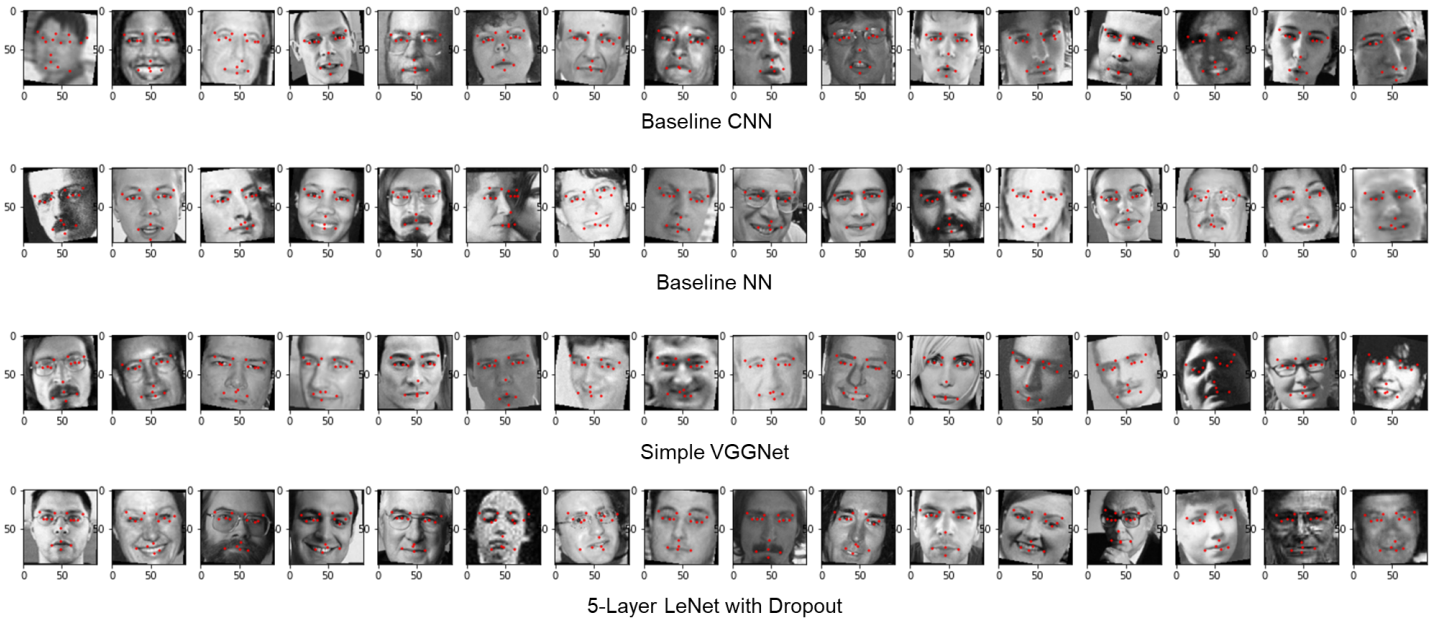

Figure 6. Sample facial keypoints prediction results for some of the test images for the Type I augmented dataset for different architectures.

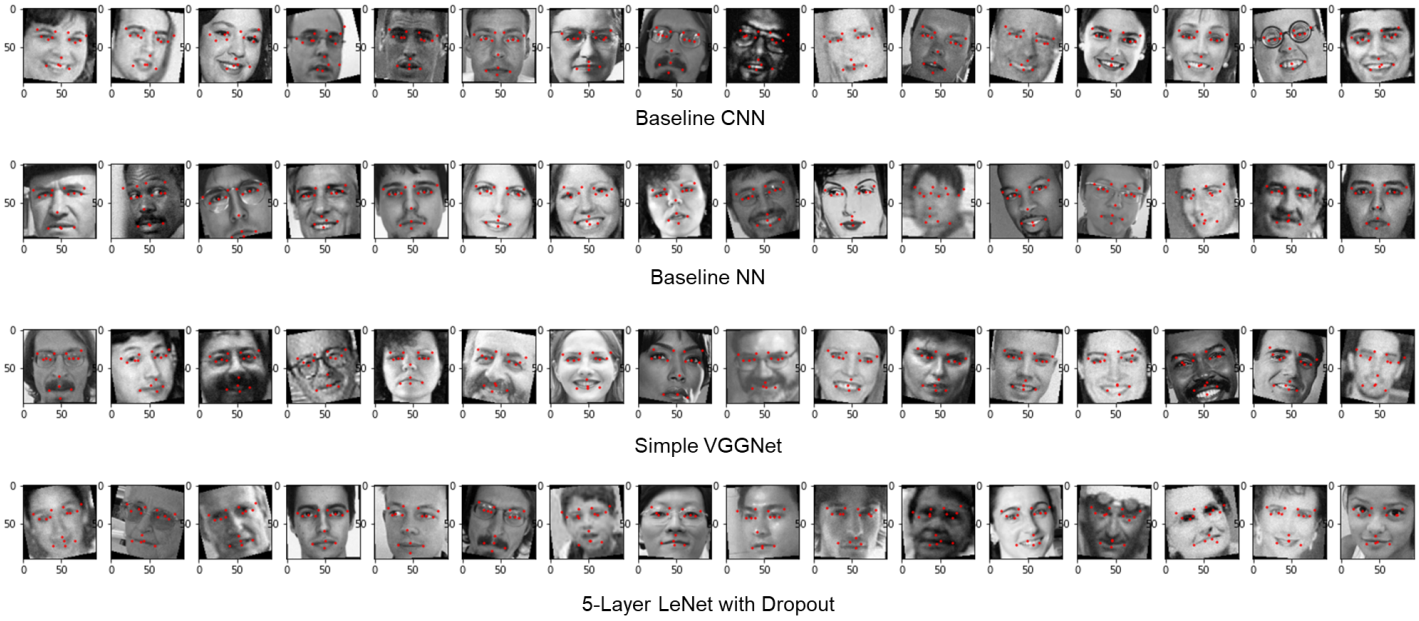

Figure 7. Sample facial keypoints prediction results for some of the test images for the Type II augmented dataset for different architectures.

Table 5. Training and validation losses after 20 epochs for different tested architectures for the original dataset

Architecture	Training Loss			Validation Loss		
	Original	Type I	Type II	Original	Type I	Type II
Baseline CNN	0.34	0.27	0.23	0.58	0.52	0.62
Baseline NN	0.70	0.44	0.34	0.93	0.45	0.39
Simple VGGNet	0.38	0.31	0.26	0.59	0.34	0.48
5-Layer LeNet with Dropout	0.65	0.47	0.41	0.72	0.59	0.43

4. Discussion and Conclusions

For all the network architectures used in this project, the original and the type II augmented datasets had the highest and the lowest training loss values, respectively. For validation, however, the type I augmented dataset had the lowest loss value for the simple VGGNet and baseline CNN architectures, and the type II dataset performed better in other architectures (Table 5).

Interestingly enough, for both types of augmented datasets, the training and validation losses were lower compared to the original dataset among all architectures (Table 5). As suggested in the literature, data augmentation through adding noise or applying transformations such as rotation to the neural network inputs might improve the performance on the neural network [8,21]. One possible reason is that when noise is added to the network, the network is less able to memorize the training samples because they are changing continuously, which results in a more robust network that has a lower generalization error [An 1995]. Some previous work have shown that injecting noise to the neural network is equivalent to some form of regularization which is added to the error function [21].

Additionally, among all architectures, simple VGGNet and baseline CNN in general had the lowest training and largest validation losses for all three types of datasets. Furthermore, for all architectures and for all datasets, the validation loss was higher than the training loss which could be an indication of overfitting. Possible solutions to overcome overfitting include using different regularization techniques, removing layers from more complex models, early stopping of training, etc. In general, as the model got more complicated, the gap between training and validation losses increased. 5-Layer LeNet, although having a relatively complicated architecture, did not perform very differently on training and validation sets and had the second lowest training-validation gap after the baseline fully connected network. This might have happened due to adding the dropout layer, which usually reduces overfitting and makes the model more generalizable. It should be noted that these differences between different architectures and datasets are not easily visible from Fig. 5-7 because the differences are quite small and not large enough to reflect qualitatively on the resultant images.

In terms of training time, simple VGGNet with 14 layers was the most time-consuming architecture to train. 5-layer LeNet and baseline CNN architectures had relatively the same training time, and the baseline fully connected neural network with one hidden layer was the easiest to train.

In general, we hypothesized that a more complicated model would most likely result in a lower training loss. However, as can be seen in Table 5, 5-Layer LeNet for instance, which is the most complicated model, had the highest training loss on all datasets among other architectures. Note that these results might have changed if we had increased the number of epochs while training. Due to the limitation of time and computational resources, we trained all the architectures with the same number of epochs (i.e., # epochs = 20). However, the simple VGG net, due to its significantly larger number of learnable parameters, converges in a greater number of epochs than the other architectures. Therefore, we cannot conclude that the baseline CNN necessarily outperforms the simple VGGNet for the task of learning facial features.

In conclusion, there's a tradeoff between model complexity, computational cost, and the training and validation losses in choosing the best architecture for this problem. A simple neural network such as the baseline NN used in this project might be sufficient for this simple application, but one might need more complex models if they were to predict more keypoints on a single image or in noisier images. One possible future work could be segmenting the best model into more specialized models

that would focus only on a subset of the facial keypoints. For example, one model might specialize specifically on eye features, another specifically on nose features, and so on.

5. Statement of the individual contributions

Mahshid Mansouri was responsible for testing the original code on the already available dataset and evaluating the performance of the pre-trained model. Additionally, she assisted Alisina Bayati in running the original architecture on the three datasets to get preliminary results. She was also responsible for writing the methods and results section of the report.

Alisina Bayati was in charge of augmenting the data through adding noise and rotation to some of the images and testing the pre-trained model on the augmented dataset. He also ran different proposed architectures on the original and augmented datasets to obtain the results. He also contributed to writing the discussion and conclusions section of the report.

Shivani Ghanta was primarily responsible for researching different facial keypoint detection architectures to compare with the baseline. She was also responsible for writing the introduction, statement of the individual contribution, and the reference sections.

The group had synchronous meetings over zoom as well as in-person meetings to discuss each part of the project and maintained constant communication through a group chat. All sections of the report have been proof-read by all group members. The code was maintained and shared between all group members on Google Colab. The original and augmented data well as the prediction results were all maintained on a shared Box folder.

6. References

[1] Zhang, S. and Chenyue, M., 2016. Facial keypoint detection. *Facial Detection Kaggle competition*.

[2] Second International Workshop on Performance and Evaluation of Tracking and Surveillance. IEEE, December 2001

[3] T. Tan: Second IEEE International Workshop on Visual Surveillance. IEEE, 1999

[4] Rosalind W. Picard: Affective Computing. MIT Press, 2009

[5] E. Petajan: The Communication of Virtual Human Faces using mpeg-4 Tools. In International Symposium on Circuits and Systems, Volume 1, pages 307-310, 2010

[6] Chin-Seng Chua & FengHan & Yeong-KhingHo: 3DHumanFace Recognition using Point Signature. In International Conference on Face and Gesture Recognition, Volume 1, pages 307-310, 2010

[7] Duane M.Blackburn & Mike Bone & P.Jonathon Phillips: Facial Recognition Vendor Test 2000 Evaluation Report. Technical Report, Department of Defence Counterdrug Technology Development Program Office, February 2011

[8] An, G., 1996. The effects of adding noise during backpropagation training on a generalization performance. *Neural computation*, *8*(3), pp.643-674.

[9] Shenghao Shi. 2017. Facial keypoints detection. arXiv preprint arXiv:1710.05279.

[10] Liang, Lin, et al. “Face alignment via component-based discriminative search.” Computer Vision–ECCV 2008. Springer Berlin Hei- delberg, 2008. 72-85.

[11] Amberg, Brian, and Thomas Vetter. “Optimal landmark detection using shape models and branch and bound.” Computer Vision (ICCV), 2011 IEEE International Conference on. IEEE, 2011.

[12] Belhumeur, Peter N., et al. “Localizing parts of faces using a consensus of exemplars.” Pattern Analysis and Machine Intelligence, IEEE Trans- actions on 35.12 (2013): 2930-2940.

[13] Sun, Yi, Xiaogang Wang, and Xiaoou Tang. “Deep convolutional network cascade for facial point detection.” Proceedings of the IEEE Con- ference on Computer Vision and Pattern Recog- nition. 2013.

[14] Serre, Thomas, Lior Wolf, and Tomaso Pog- gio. “Object recognition with features inspired by visual cortex.” Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Vol. 2. IEEE, 2005.

[15] M. Valstar, B. Martinez, X. Binefa, and M. Pan- tic. Facial point detection using boosted regression and graph models. In Proc. CVPR, 2010.

[16] M. Dantone, J. Gall, G. Fanelli, and L. J. V. Gool. Real-time facial feature detection using conditional regression forests. In Proc. CVPR, 2012.

[17] Simonyan, Karen, and Andrew Zisserman. “Very deep convolutional networks for large- scale image recognition.” arXiv preprint arXiv:1409.1556 (2014).

[18] Original dataset on Kaggle: https://www.kaggle.com/competitions/facial-keypoints-detection/data

[19] Original code on Kaggle: https://www.kaggle.com/code/rajeevctrl/facial-keypoints-detection-getting-started/notebook

[20] Longpre, S. and Sohmshetty, A., 2016. Facial keypoint detection. *Facial Detection Kaggle competition*.

[21] Bishop, C.M., 1995. *Neural networks for pattern recognition*. Oxford university press.