Machine Learning Final Project Report

Peiwen Zhao
Virginia Tech
Blacksburg Virginia
peiwenz@vt.edu

Abstract

Nowadays, face recognition is one of the key technologies behind many popular applications. It has been used in filtering, photographing, video, movie and many other areas. Behind its magic sound like name, machine learning plays the major role for supporting the face recognition performance. By using Convolutional Neural Networks, the input images can be recognized and the human face can be distinguished though algorithms with reasonable error rate. By using the output message from the neural networks, people can do many things with the faces, including remembering each faces' different features and classify who the person is. This paper will discuss a specific application of using neural networks to make face recognition working and apply different filters on the input data. This paper will discuss the steps and outcomes of the application.

1. Introduction

Face recognition is increasingly being used for solving various social-problems such as personal protection and authentication. As with other widely used biometric applications, facial recognition is a biometric instrument such as iris recognition, vein pattern recognition, and fingerprint recognition.[1]

Image filters are one of the most popular things on the internet that has machine learning and face recognition technology in it.

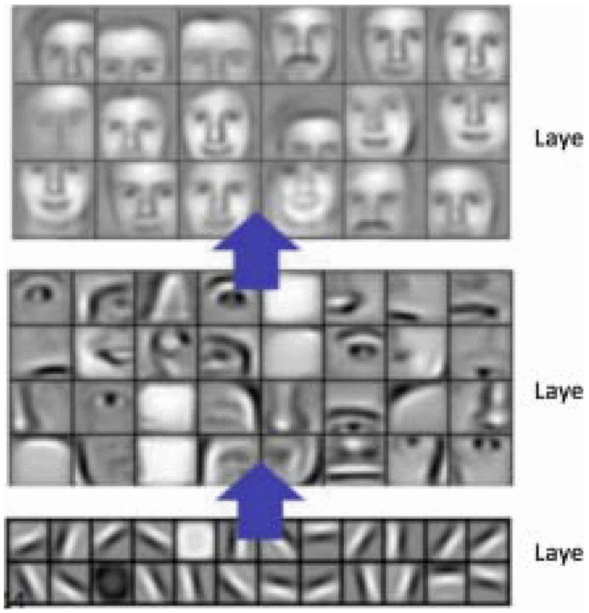
Image Filter Examples[5]

It is extremely fun to play around and also quite fascinating since people can use those filters to do verity of things. The extensive use of social media platforms, especially during disasters, creates unique opportunities for humanitarian organizations to gain situational awareness and launch relief operations accordingly.[2]

1.1. Current Implementation

The current implementation of face filtering is by using a developed system which uses deep convolutional neural networks (DCNN).[4]Usially, a input image will be transfromed to be a matrix of numerical digits, to represent different color and pixels. Then, the face detection algorithm will go through this code and look for a colour patterns that would eventually represent a face.[4] After the patterns have been extracted, a statistical model of a face can be created by manually pointing out different borders of the facial features with appointing numbers to the different regions of the face.[4]

Then, by mapping different key parts of the face to their designed corresponded part of the filter object, a face filter can be done.



Structure of Convolutional Neural Networks[7]

1.2. Limitation

There are some drawbacks of using Convolutional neural networks to train models for facial recognation. First is the CNN requires a large size of data set to train. Second is CNN is very slow if the functions used in the training is not optimal. Third is CNN requires a lot of resources if the layers number increases.

Also, the algorithm relies on a complete and clear input for creating the face model, if the input face picture is not clear or has face covering, the algorithm error rate will be effect tremendously.

1.3. Target Audiences

The main target audiences of this face filtering application is people who loves using filtering on social media. The application is able to use the trained the model to predict facial key points of different photos, and output the location of these facial key points. These facial key points can be used to produce different face filters.

1.4. Objective

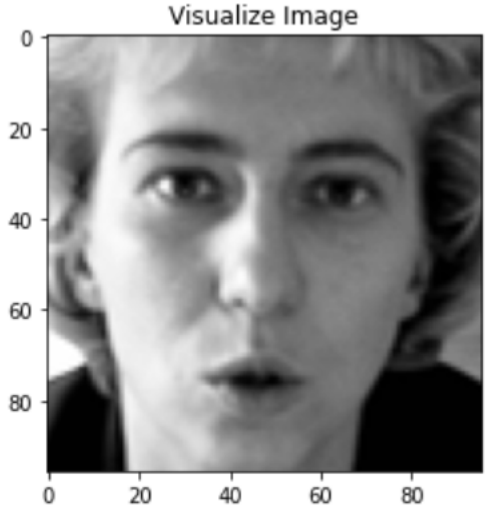
The objective of the application is to have a working face recognition model such that the users may use the application and apply filters on their images.

2. Approach

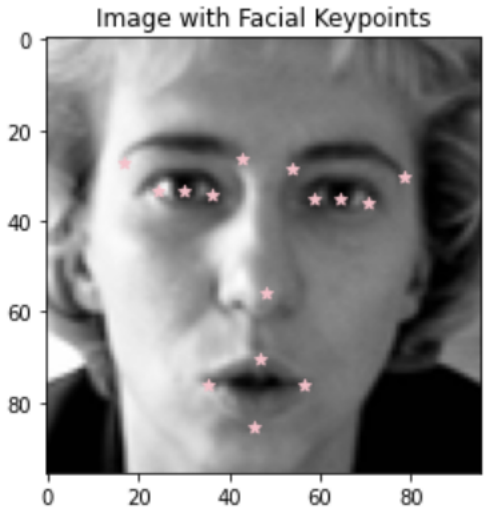
In order to train a working model for adding filters to any input, a training data set is needed for the algorithm. The input data set needs to be filtered since some of the images in the data set is missing some of the parameters. My training algorithm is fully supervised, thus the first step I will do is to correct any input that has missing parameters.

I first read in the training.csv and test.csv friles as arrays. Then I have reference to KHOLOUD SAYED's tutorial[6] for accomplishing completing the incomplete data from the data set. By using .fillna() in python library, I am able to get a data set that all the parameters are not missing.

The second step is to build the basic convolutional neural network with different layers. In the convolutional neural network, I choose to use the 'relu' activation function, combined with no paddings. Since the input image will always be 96 pixels X 96 pixels, shown in picture1,



Picture1: example of a training image



training image with corresponded facial key points

there is no need for padding. 'relu' activation function has the highest training accuracy and performs the best over all the other activation function such as sigmoid function or

normal distribution function.

In order to build the CNN, I will create a X-train list and Y-train list for getting all the data of each image. By using numpy.vstack(), I am able to normalized each pixel to [0,1], then I can use reshape() to convert the images to 96X96X1 sizes. Then, I am able to use X-train as pixels and Y-train as corresponded facial key points to start building the convolutional neural network.

In each big layer of the convolutional neural network, I added a 'relu' activation layer, a Maxpool2D layer, and a dropout layer. In the end, I added a flatten layer, and 2 dense layers. For the optimizer, I choose 'Adam' since it has the best accuracy.

Layer (type)	Output Shape	Param #
conv2d_28 (Conv2D)	(None, 96, 96, 32)	320
activation (Activation)	(None, 96, 96, 32)	0
max_pooling2d_28 (MaxPoolin g2D)	(None, 48, 48, 32)	0
dropout_13 (Dropout)	(None, 48, 48, 32)	0
conv2d_29 (Conv2D)	(None, 48, 48, 32)	9248
activation_1 (Activation)	(None, 48, 48, 32)	0
max_pooling2d_29 (MaxPoolin g2D)	(None, 24, 24, 32)	0
dropout_14 (Dropout)	(None, 24, 24, 32)	0
conv2d_30 (Conv2D)	(None, 24, 24, 64)	18496
activation_2 (Activation)	(None, 24, 24, 64)	0
max_pooling2d_30 (MaxPoolin g2D)	(None, 12, 12, 64)	0
dropout_15 (Dropout)	(None, 12, 12, 64)	0
conv2d_31 (Conv2D)	(None, 12, 12, 128)	73856
activation_3 (Activation)	(None, 12, 12, 128)	0
max_pooling2d_31 (MaxPoolin g2D)	(None, 6, 6, 128)	0
flatten_7 (Flatten)	(None, 4608)	0
dense_14 (Dense)	(None, 256)	1179904
dropout_16 (Dropout)	(None, 256)	0
dense_15 (Dense)	(None, 30)	7710

Convolutional Neural Network Structure

2.1. Data sets

The data sets are from Kaggle, it has thousands of training input pictures. The input image is given in the last field of the data files, and consists of a list of pixels (ordered by row), as integers in (0,255). The images are 96x96 pixels.

Each predicted key-point is specified by an (x,y) real-valued pair in the space of pixel indices. There are 15 key-points, which represent different position of the human face.

There are 7049 images for training, and 1783 images for testing.[3]

Kaggle provides a testing data set, however there is no specific way of validating the accuracy from the model's estimation.

2.2. Anticipated Problems

My original anticipation of the major problem of my application is to read in the data set and setup the CNN. It is quite challenging since I does not have experience with setting up any kind of CNN before, the only experience I have is homework 4 in the class.

I anticipated my training will take a long time since CNN generally requires a lot of resources and time to train.

I do not have a quantitative way to evaluate my model's accuracy, since the data set does not provide a way to perform validation for the testing data.

2.3. Encountered Problems

In order to start building a CNN with no experience, I get helped by looking up others' tutorials and examples of how to set up CNN by using google colab and kersa, I was able to build my CNN and train a model with the training data provided by Kaggle.

The model takes a very long time to train, and the accuracy increasing rate stays almost the same. This is not what I expected, since in homework 4 the model was able to increase accuracy when the number of epochs increases. I tried to change different activation functions and changing the size of each layers, I also tried out different drop out layer percentages. The final result is the best combination I have gotten.

I decided to evaluate the model's prediction with qualitative instead of quantitative. It is very obvious for humans to see weather the output prediction is accurate or not accurate.

3. Experiments and Results

For the evaluation of success, I will use the trained model to predict the testing data set. After I have received all the predictions, I will plot each prediction to its corresponding testing image, and show the image out. By reviewing each image with my eye, I am able to see how accurate the model's predictions are.

The result is qualitative rather than quantitative. And my conclusion of my application is that I am partially successful. In the testing data set, my model is able to predict the faces that are directly facing the camera with high accuracy. The facial key points are very close to the actual facial key points. However, when the face is not directly facing to the camera, or it has a rotation, my model is not able to handle it very well. It will try to fit the direct-camera-face's facial key points' locations to the non-direct-camera faces.

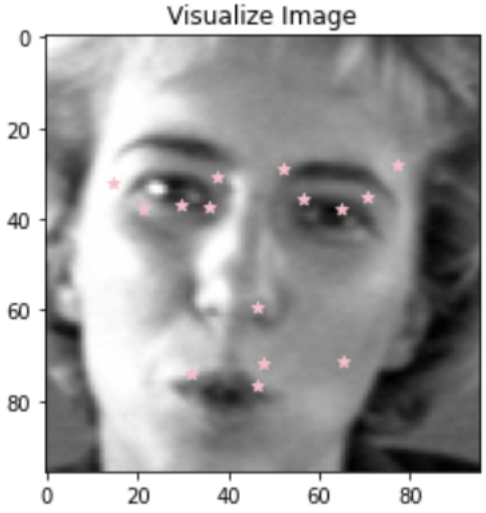
Which causes a not accurate prediction. This is the major issue of my trained model. Even though I keep trying different combinations of activation function, drop out layers, pooling layers, and dense layers, the accuracy did not increase significantly when the number of epochs increases.

For the metrics for evaluation, I did not accomplished NO.1 since my training correctness is not over 70 percent. I keep trying with different CNN structures but I can not reach 70 percent of accuracy.

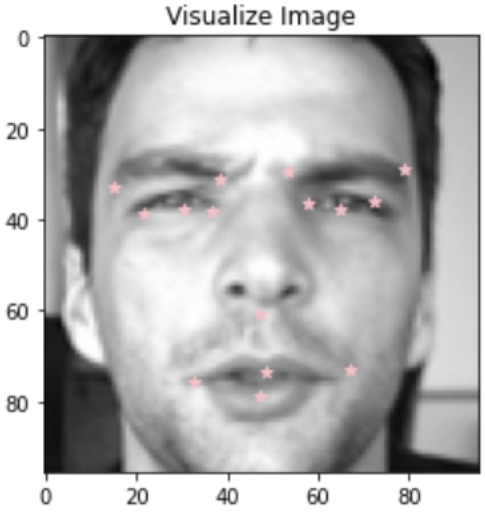
I accomplished NO.2 since my model is able to predict testing images' facial key points. Even though the accuracy of the model does not perform perfect, it can still do a fairly good job for the pictures that have faces directly facing the camera. I will count this as a success.

I did not accomplish the NO.3 of metrics for evaluation since I am not able to design the program to have a functional UI that can change filter based on input photos. I am only able to get the testing data estimated by the trained model, in order to design a functional UI is requiring a lot of extra knowledge and work since it requires some external resources for the design of the filter, and also applying the facial key points of each testing picture to the corresponding facial key points of the filter object.

3.1. Testing result Images



testing image with not accurate facial key points



testing image with accurate facial key points

3.2. Model training process

```
loss: 26.9786 - accuracy: 0.5967 - val_loss: 5.2385 - val_accuracy: 0.6872
loss: 27.3307 - accuracy: 0.6044 - val_loss: 6.3549 - val_accuracy: 0.6872
loss: 27.3571 - accuracy: 0.6049 - val_loss: 6.4130 - val_accuracy: 0.6872
loss: 27.7298 - accuracy: 0.6054 - val_loss: 8.0161 - val_accuracy: 0.6872
loss: 27.0958 - accuracy: 0.6003 - val_loss: 5.9493 - val_accuracy: 0.6872
loss: 26.2495 - accuracy: 0.6054 - val_loss: 8.4662 - val_accuracy: 0.6872
loss: 27.5707 - accuracy: 0.6063 - val_loss: 5.8937 - val_accuracy: 0.6872
loss: 26.8574 - accuracy: 0.6044 - val_loss: 5.2368 - val_accuracy: 0.6872
loss: 27.3233 - accuracy: 0.6038 - val_loss: 5.9662 - val_accuracy: 0.6872
loss: 27.0523 - accuracy: 0.6013 - val_loss: 6.6396 - val_accuracy: 0.6872
loss: 27.5298 - accuracy: 0.6040 - val_loss: 6.8998 - val_accuracy: 0.6872
loss: 26.7672 - accuracy: 0.6063 - val_loss: 19.3804 - val_accuracy: 0.6872
loss: 26.9128 - accuracy: 0.6026 - val_loss: 5.5972 - val_accuracy: 0.6872
loss: 26.7374 - accuracy: 0.6022 - val_loss: 6.5886 - val_accuracy: 0.6872
loss: 26.6116 - accuracy: 0.6045 - val_loss: 7.0356 - val_accuracy: 0.6872
loss: 26.2065 - accuracy: 0.6049 - val_loss: 18.8477 - val_accuracy: 0.6872
loss: 27.7639 - accuracy: 0.5982 - val_loss: 5.5542 - val_accuracy: 0.6872
loss: 27.8073 - accuracy: 0.6037 - val_loss: 5.4205 - val_accuracy: 0.6872
loss: 27.1673 - accuracy: 0.5962 - val_loss: 6.3764 - val_accuracy: 0.6872
loss: 27.0635 - accuracy: 0.6024 - val_loss: 7.5895 - val_accuracy: 0.6872
loss: 26.4852 - accuracy: 0.6051 - val_loss: 6.2408 - val_accuracy: 0.6872
loss: 26.6072 - accuracy: 0.5983 - val_loss: 8.8115 - val_accuracy: 0.6872
loss: 26.8316 - accuracy: 0.6047 - val_loss: 7.0099 - val_accuracy: 0.6872
loss: 27.1132 - accuracy: 0.6017 - val_loss: 6.0679 - val_accuracy: 0.6872
loss: 26.3831 - accuracy: 0.6028 - val_loss: 7.6495 - val_accuracy: 0.6872
```

chart of CNN model training epochs

The picture of one testing image which has an inaccurate facial key points is a classical example of how my model fails due to the face in the image is rotated left in a slight angle. The model is trying to fit the facial key points to the image and assume the face is directly facing the camera.

The picture of the one testing image that has a accurate facial key points is an example of how my model can well predict a face with directly facing the camera. All 15 facial key points are very accurate.

In the 'chart of CNN model training epochs', the main issue of the training is the accuracy does not increase as the epochs increases. The CNN did not sufficiently learn from the previous epoch's mistake and apply the past experience to the new epoch.

3.3. Metrics for Evaluation

1. The final model should reach 70 percent of correctness.

2. The final model shall have the ability to change input's face and add filters on the input.

3. The final model shall be able to personalized and seen in UI.

3.4. Mile Stones

1. The application shall load all the training data and classify them, then build a basic Convolutional neural network based on the features of the input data.

2. The CNN shall detect different key points of each input images, and keep running Convolutional layers and pooling layers to train the model.

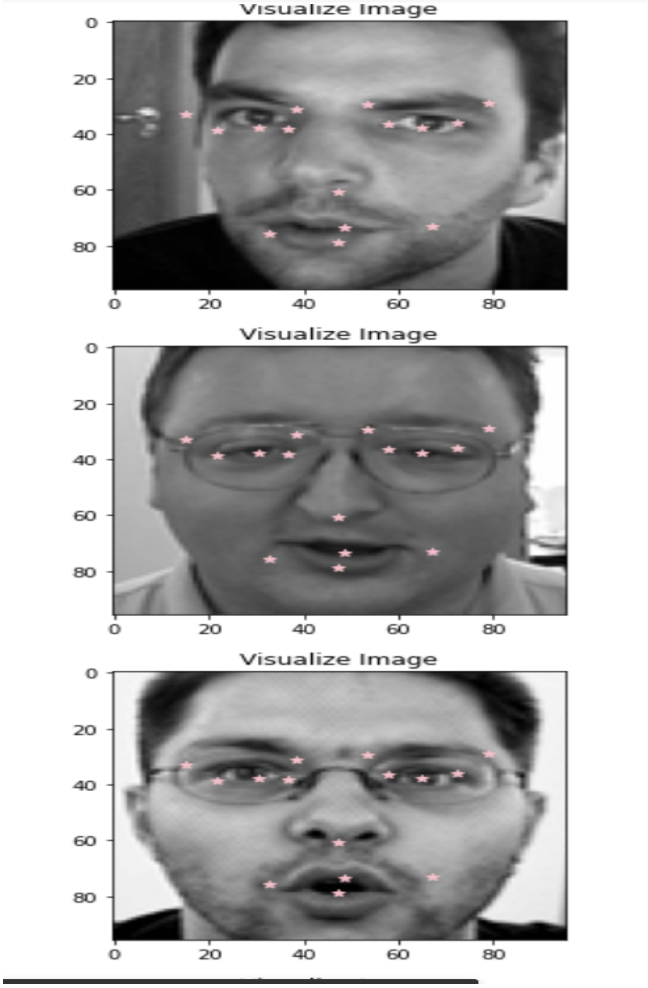
3. After the model has been trained, the model will be tested by using the testing data.

4. The model can change different key points and in this way the filers will be applied.

3.5. Mile Stones report

Over the four mile stones I set for this project, I accomplished mile stone 1, mile stone 2, mile stone 3. I can not accomplish mile stone 4 since it requires a lot more works and know-ledges.

Since the main functionality, building the training model for facial recognition, is accomplished(showing more examples of testing image predictions), mile stone 4 can be accomplished in a timely manner.



Examples of testing predictions

4. Availability

4.1. Code availability

The code, the training data, and the testing data are available on Github.

4.2. How to use the code

Users may download the training data set and testing data set on Github, then the user needs to run the .ipynb notebook code for actually training the model. The training will take around 1 hour to accomplish, however the time will varies due to the computation power of the user's computer.

5. Reproducibility

5.1. How others can use the training model

The other users may run the .ipynb notebook and get the model. They may load the model to other applications and

use the model to predict any other face image's facial key points' locations. The output of the model will be an array which contains the specific location of each facial key points.

5.2. Are model parameters fully reproducible

The CNN which trains the model has a fixed parameters in my version. However, it is very easy to change the parameters of the CNN, the users may add or change layers in the current CNN and experiment different combinations of neural layers.

6. Conclusion

Overall, this project have taught me a lot about how deep learning and neural networks work. Also this project taught me how to slowly accomplish the goal step by step.

I have learned how to start design a project and how to research and gather resources I need for accomplishing the project. The skills and knowledge I learned from the project is important.

7. Reference

[1] Rondik J.Hassan Adnan Mohsin Abdulazeez, 2021. "Deep Learning Convolutional Neural Network for Face Recognition: A Review," International Journal of Science and Business, IJSAB International, vol. 5(2), pages 114-127.

[2] Dat Tien Nguyen. Automatic image filtering on social networks using deep learning and perceptual hashing during crises. IEEE TPAMI, 1(1):234–778, 2017

[3] Facial keypoints detection. Kaggle. (n.d.). Retrieved May 4, 2022, from https://www.kaggle.com/competitions/facial-keypoints-detection/data

[4] wapnil Bhardwaj, Madhurima Hooda, and Saru Dhir. An era of face filters. International Journal of Innovative Technology and Exploring Engineering, 9(1), 2019.

[5] Banuba. (2020, July 7). 10 best face filter apps like Snapchat to spark your creativity [2019]. Medium. Retrieved May 4, 2022, from https://banuba.medium.com/best-face-filter-apps-like-snapchat-to-spark-your-creativity-2019-8c0042300e37

[6] Kholoudsayed. (2020, December 31). Facial-keypoints-detection. Kaggle. Retrieved May 4, 2022, from https://www.kaggle.com/code/kholoudsayed/facial-keypoints-detection/notebook

[7] S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186