RECOGNITION OF LANDMARKS ON THE NOSE

Ha Duc Anh, Hcm University of
Technology and Education
19134073@student.hcmute.edu.vn
December 12, 2022

Abstract

Nowadays, facial keypoints detection has become a very popular topic. This paper describes an approach to predicting keypoint positions on greyscale images of faces, as part of the Facial Detection (2016) Kaggle competition. Facial keypoints include centers and corners of the eyes, eyebrows, nose and mouth, among other facial features. The objective of facial keypoints detection is to find the facial keypoints in a given face, which is very challenging due to very different facial features from person to person. The idea of deep learning has been applied to this problem, such as neural network and cascaded neural network.

In our project, we would like to locate the keypoints of the nose in a given image using deep architectures to not only obtain lower loss for the detection task but also accelerate the training and testing process for real-world applications. We have constructed two basic neural network structures, one hidden layer neural network and convolutional neural net-work as our baselines. And we have proposed an approach to better locate the coordinates of facial keypoints with in-troduced features other than raw input. Specifically, we create a CNN model that uses Inception Architecture to extract features and using different deep structures to compute the final output vector.

1. Introduction

1.1 Motivation

With the fast development in computer vision area, more and more research works and industry applications are focused on facial keypoints detection. Detecting keypoints in a given face image would act as a fundamental part for many applications, including facial expression classification, facial alignment, tracking faces in videos and also applications for medical diagnosis. Thus, how to detect facial keypoints both fast and accurately to use it as a preprocessing procedure has become a big challenge.

1. Facial features are very different

Facial features are very different from people to people, which result in the difficulty of training the regression model. And similar as object detection task, under different illumination conditions, positions, sizes, detecting facial keypoints would be very challenging.

2. Detecting keypoints has to be fast

Detecting keypoints in face images is one of the first few steps for many applications as we mentioned before, for example analyzing facial expression or detecting faces in images and videos. Moreover if we would like to fit the detection procedure to a real-time mobile app, we have to complete the detection within seconds. Therefore, the computation complexity of keypoints detection have to be lower than traditional image classification tasks. Unlike other image classification task that we only evaluate the accuracy, we have to focus on the time that we spend on the task. When using traditional deep structures the training and testing process tend to be much slower, which is not we want for facial keypoints detection.

With the development of deep learning, many deep structures designed for this task have been explored recently. Different deep structures have been proposed for facial keypoints detection, like deep convolutional cascade network, which can better deal with two main challenges we mentioned before.

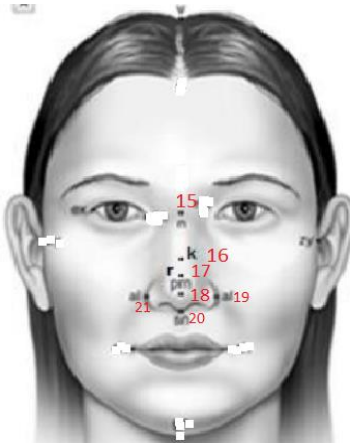
1.2 Problem stament

In this project we detect landmarks of nose image at three different angles. Our objective is to locate facial keypoints of nose image when given a raw facial image. The input is a set of 96 × 96 raw facial images with only the grayscale pixel values, and the output are 14, 10 and

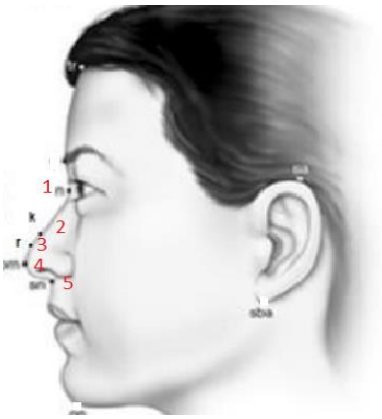
18-dimentional vector, which indicate the (x, y) coordinates each image at three different angles.

In our project, we are going to use deep structures for facial keypoints detection, which can learn well from different faces and overcome the variance between faces of different person or of different conditions to a great extent. Two widely-used model, One Hidden Layer Neural Network and Convolutional Neural Network, are designed as baselines in our project. Most importantly, we have used the Inception architecture to explore some techniques to reduce the computation complexity for detecting facial keypoints. With sparsely connected layers and different number of different size of filters, Inception Model have the ability to better capture local features and reduce computation complexity at the same time.

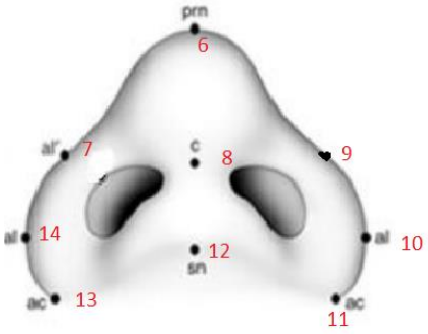
And in this project we wil train three modles for three different angles of nose image. The first case is Face-to-face angle of nose, we will detect 7 facial keypoints. And the output is a 14-dimentional vector, which indicate the (x, y) coordinates of 7 sets of facial keypoints. We can look at the picture below:



The second case is side angle of face, we will detect 5 facial keypoints. And the output is a 10-dimentional vector, which indicate the (x, y) coordinates of 5 sets of facial keypoints. We can look at position of the point in the picture below:



The third case is image of the bottom angle of the nose, we will detect 9 facial keypoints. And the output is a 18-dimentional vector, which indicate the (x, y) coordinates of 9 sets of facial keypoints. We can look at position of the point in the picture below:

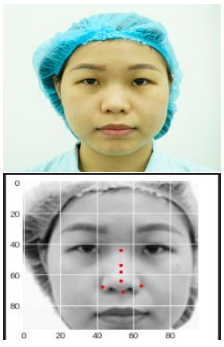


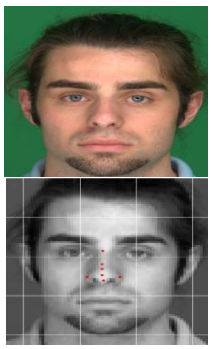
The problem is essentially to predict the near-exact coordinate locations of points on an image of a face. Given three set of facial points of three different angles image, there are total 42 numerical values that we need to predict: a 2D (x, y) coordinate representation for each facial point. The loss is calculated as the total mean squared error (MSE), an effective measure of the deviations in distances between the 21 real and predicted facial point coordinates:

The formula for Mean Squared Error is shown above.

$$MSE = \frac{1}{n}\Sigma\underbrace{\left(y - \hat{y}\right)}^{2}$$

Here n = 14,n = 10, n = 18, and there are 14, 10 and 18 dimensions to the output for three case of image, yi is the i-th dimension of the expected output and ˆyi is the i-th dimension of the predicted output. Our approach involves the use of deep learning and convolutional neural networks to predict the x and y coordinates of a given keypoint class in an image containing a single face. Therefore, as shown below, our input is a greyscale image from the dataset (96x96x1), and our output is floating point values between 0.0 and 96.0, indicating where on the image the corresponding x and y coordinates are for a given keypoint feature. Below are example inputs (first row) and outputs (second row).





1.3 Data set

Our dataset [1], provided by my teacher take from the data of hospital, includes training data of about 1250 images. I crop change them into grayscale 96x96x1 images, with the labelled (x, y) coordinates for facial keypoints. Our task is to develop a model that can predict the specific locations of these facial keypoints on test set images, minimizing Mean Squared Error.

2. Related work

Recent works have focused on deep architectures for this detection task since these structures can better capture the high-level features of a image, in our problem a given face proposed a three-level carefully designed convolutional networks, which at first capture the global high-level features, and then refine the initiliation to locate the positions of keypoints. On the other hand, [3] used pretrained DBN on surrounding feed-forward neural network with linear Gaussian output layer to detect facial keypoints. Different from our work, the deep structures mentioned before have not focused on the time complexity but only on the correctness of the detected keypoints. But using Inception Model we can train a much more complexity model in less time, thus the detection process would be faster than the traditional deep structures considering the number of parameters, which can be adapted to this task.

3. Technical Approach

3.1 Problem Formulation

We have split our dataset into three parts X_train, X_val, X_test and we define the ground truth corresponding to these three sets to be y_train, y_val, y_test. Given a training dataset X_train we have:

$$X_{train} = \{X_{train}^{(1)}, \cdots, X_{train}^{(n_{train})}\}$$

$$y_{train} = \{y_{train}^{(1)}, \cdots, y_{train}^{(n_{train})}\}$$

Where $X^{(i)}_{(train)}$ represents a given image $Y^{(i)}_{(train)}$ represents the keypoints vector for $X^{(i)}_{(train)}$. We will train our model M on (Xtrain, ytrain) and evaluate model on (X_val, y_val) to tune parameters. After the training process, we would evaluate our model on the test set (X_test, y_test).

The performance of our network models is evaluated based on two major metrics. The first one is detection accuracy, which is represented by regression loss. To be more specific, in our case, we quantize the loss using mean squared error (MSE) between the ground truth y and predicted keypoints vectors y. The second one is mean absolute error (MAE)

3.1 Architectures of network

I create an inception module base on inception architecture:

```
# We will create CNN model that uses Inception Architecture
def inception_module(inputs,f1,f2):
    # defining convolution 2d layer
    x1=Conv2D(f1,3,padding="same")(inputs)
    # batchnormalize the x1
    x1=BatchNormalization()(x1)
    x1=ReLU()(x1)
    # it was f1 change it to f2
    x2=Conv2D(f2,5,padding="same")(inputs)
    # batchnormalize the x1
    x2=BatchNormalization()(x2)
    x2=ReLU()(x2)
    # combine x1 and x2
    return Concatenate()([x1,x2])
```

In main network, input will pass the inception 5 times

```
def build_model():
    inputs = Input((96, 96, 1))
    x = inception_module(inputs, 64,  32)
    x = MaxPool2D()(x)
    x = inception_module(x, 64,  32)
    x = MaxPool2D()(x)
    x = inception_module(x, 128, 32)
    x = MaxPool2D()(x)
    x = inception_module(x, 128, 32)
    x = MaxPool2D()(x)
    x = inception_module(x, 256, 64)
    x = MaxPool2D()(x)
    x = Flatten()(x)
    x = Dense(1024, kernel_regularizer=L2(l2=0.05))(x
    x = BatchNormalization()(x)
    x = ReLU()(x)
    x = Dense(512, kernel_regularizer=L2(l2=0.02))(x)
    x = BatchNormalization()(x)
    x = ReLU()(x)
    x = Dense(128, kernel_regularizer=L2(l2=0.01))(x)
    x = BatchNormalization()(x)
    x = ReLU()(x)
    x = Dense(14)(x)
```

We can see total Network Architectures at table below:

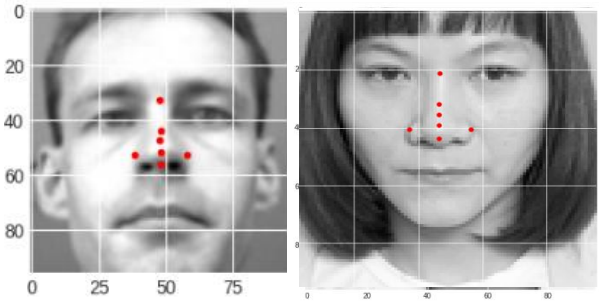
Layer		Name	Size
0		Input	1x96x96
1		Inception_module	96x9696
2		MaxPooling2D	48x48x96
3		Inception_module	48x48x96
4		MaxPooling2D	24x24x96
5		Inception_module	24x24x160
6		MaxPooling2D	12x12x160
7		Inception_module	12x12x160
8		MaxPooling2D	6x6x160
9		Inception_module	6x6x320
10		MaxPooling2D	3x3x320
11		Flatten	2880x1
12		Dense	1024
13		BatchNormalization	1024
14		ReLu	1024
15		Dense	512
16		BatchNormalization	512
17		ReLu	512
18		Dense	128
19		BatchNormalization	128
20		ReLu	128
21		**Dense**	14 or 10 or 18

In the layer 21:

With the first case is Face-to-face angle of nose, we will detect 7 facial keypoints. I will set out is 14.
With the second case is side angle of face, we will detect 5 facial keypoints. I will set out is 10.
With the third case is the bottom angle image of the nose, we will detect 9 facial keypoints. I will set out is 18.

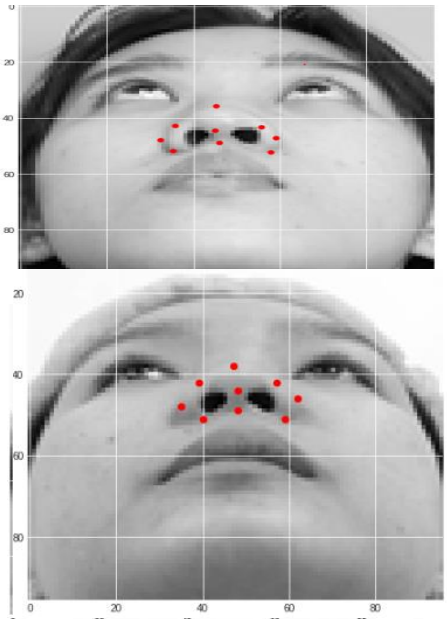
4. Result

With the first case is Face-to-face angle of nose, we will detect 7 facial keypoints. Training 500-700 eporchs with 784 images and get MAE approximately 1.5026 (px) in validation data. Apply model on test data. We can see some picture below:



With the second case is the side angle of nose, we will detect 5 facial keypoints. Training 700 eporchs with 350 images and get MAE approximately 2.7939 (px) in validation data. Apply model on test data. We can see some picture below:

With the third case is image of the bottom angle of the nose, we will detect 9 facial keypoints. Training 600 eporchs with 412 images and get MAE approximately 2.0194 (px) in validation data. Apply model on test data. We can see some picture below:



5. Conclusion

In this paper, we have focused on the task of detecting facial keypoints when given raw facial images. Specifically, for a given 96 x 96 image, we would predict some sets of (x, y) coordinates for facial keypoints. Two traditional deep structures, One Hidden Layer Neural Network and Convolutional Neural Network, are implemented as our baselines. We further explored a sparsely connected Inception Model to reduce computational complexity to fit the requirements for detecting facial keypoints. As for our future work, we can explore from these few aspects:

- Add more data in different cases to increase model accuracy.
- Different resolution can greatly affect the results of the facial keypoints detection, thus what we can do is try to reduce the resolution of our given raw images to see the variance of the performance to further evaluate our model.

6. References

[1] *https://www.kaggle.com/c/facial-keypoints-detection/data.*

[2] *https://www.kaggle.com/c/facial-keypoints-detection/overview/getting-started-with-r.*

[3] *https://arxiv.org/pdf/1710.05279.pdf.*

[4]*_https://www.hindawi.com/journals/cin/2022/6205108/*