Machine Learning Engineer Nanodegree

Capstone Proposal

Pengfei Shi
Feb 27th, 2017

Proposal

This capstone proposal is chosen from Kaggle competition: **Facial Keypoints Detection**. The official homepage of this competition is: https://www.kaggle.com/c/facial-keypoints-detection.

Domain Background

Computer vision is a very important technology, and it has been applied in many areas. The objective of this work is to predict keypoint positions on face images. This can be used as a building block in several applications, such as:

- tracking faces in images and video
- analysing facial expressions
- detecting dysmorphic facial signs for medical diagnosis
- biometrics / face recognition

Figure 1

Detecing facial keypoints is a very challenging problem. Facial features vary greatly from one individual to another, and even for a single individual, there is a large amount of variation due to 3D pose, size, position, viewing angle, and illumination conditions. Computer vision research has come a long way in addressing these difficulties, but there remain many opportunities for improvement.

In my **Image Processing** course, our professor had given us an introduction in this area, which are widely used in computer vision, so I want to choose this topic for my capstone project.

Problem Statement

The aim of this competition is to detect the location of keypoints on face images.

There are 15 keypoints need to be located, each predicted keypoint is specified by an (x, y) real-valued pair in the space of pixel indices, which represent the following elements of the face:

left_eye_center, right_eye_center, left_eye_inner_corner, left_eye_outer_corner, right_eye_inner_corner, right_eye_outer_corner, left_eyebrow_inner_end, left_eyebrow_outer_end, right_eyebrow_inner_end, right_eyebrow_outer_end, nose_tip, mouth_left_corner, mouth_right_corner, mouth_center_top_lip, mouth_center_bottom_lip.

Left and right here refers to the point of view of the subject.

The input image is consists of a list of pixels (ordered by row), as integers in (0,255). The images are (96x96) pixels.

We need to use the given images as the inputs, and the given keypoints as outputs to get a prediction model, and use this model to predict the test dataset. Because the inputs and outputs are numerical data, so in matching learning, it is a regression problem.

Datasets and Inputs

This competition gives us three data files:

- **training.csv**: list of training 7049 images. Each row contains the (x,y) coordinates for 15 keypoints, and image data as row-ordered list of pixels.
- **test.csv**: list of 1783 test images. Each row contains ImageId and image data as row-ordered list of pixels
- **submissionFileFormat.csv**: list of 27124 keypoints to predict. Each row contains a RowId, ImageId, FeatureName, Location. FeatureName are "left_eye_center_x," "right_eyebrow_outer_end_y," etc. Location is what you need to predict.

In some examples, some of the target keypoint positions are misssing (encoded as missing entries in the csv, i.e., with nothing between two commas).

- **For training parts**: inputs are (96*96) images in **training.csv**, and the labels are 15 keypoints for outputs in **training.csv**.

- **For testing parts**: inputs are (96*96) images in **test.csv**, and the prediction results should be stored in **submissionFileFormat.csv** for submitting to Kaggle.

Solution Statement

This task is complex to train the model, because it has large scale high-dimension input data, and the multiple outputs. **Convolutional neural networks (CNN)** is a powerful matching learning method for solving this kind of problems, which data are in high-dimension, and it has been proved very useful.

According to Reference 4, compared with ordinary Neural Networks, CNN architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture, and these make the forward function more efficient to implement and vastly reduce the amount of parameters in the network. Also, CNN can be combined with GPU for parallel computing, which can help accelerate computing for saving the model training time.

The tool I will choose **Tensorflow**, which is an open-source software library for machine intelligence. And it has been widely used in many companies.

Benchmark Model

In order to measure the model trained from the training dataset, this project will use the scores in Kaggle official leaderboard as the benchmark. And according to the offical announcements, this leaderboard is calculated with approximately 50% of the test data (total 1783 images). And some particular scores in the public leaderboards are:

#	Score
1	1.53319
10	2.03259
50	2.56286
100	3.80685
...	...

This capstone will use the model to predict the test dataset and submit the results into Kaggle platform for getting the final score, then compare our score with these scores as benchmark to measure the performance of my model for analysis.

Evaluation Metrics

According to the official competition judgements, it uses **Root Mean Squared Error (RMSE)** for metric.

RMSE is commonly used general-purpose error metric. In this case, the position of keypoints are high-dimension data, so this metric is suitable. Compared to Mean Absolute Error (MAE), it will punish large errors, the description is shown in below:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

where $\hat{y}_i$ is the predicted value and y_i is the original value.

Project Design

- Setup the basic programming environments, and configure the library for use: Numpy, Pandas, Tensorflow.
- Download three data file for test, and load them into python variables using `pd.Dataframe()`.
- Explore the data, an example is shown in **Figure 2**, the gray picture is one person face, and the blue point are the marked keypoints in face.

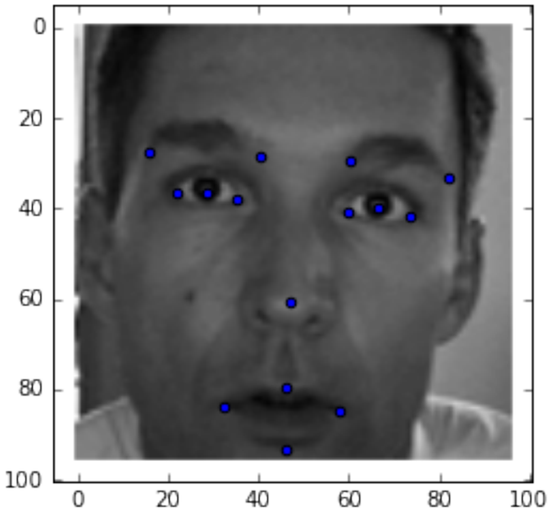

Figure 2

- Do some actions for preprocessing, drop the nan data, and reshape the image data (because the origin image data is one list which is row-ordered).
- Shuffle and split data for creating validation dataset for test. And because of the large num of images, in order to train the model efficiently, we may need to divide the data into some batches, and load them into tensorflow initial variables.
- The most important part, design the CNN model, CNN model consists of many layers. How many convolutional layers, what the size of each convolutional kernels and where to put the max-pooling layer,.etc, these are all need take into consideration. So a good model leverages very about all these components. And these may need a lot of attempts, and choose the least cost model in validation dataset.
- Use the trained model to predict the testing dataset, and upload the results into Kaggle platform for judgements, compare the results with leaderboard scores and do some analysis.

References

1. Kaggle Competition HomePage: https://www.kaggle.com/c/facial-keypoints-detection.
2. CNN tutorial in deepleanring.net: http://deeplearning.net/tutorial/lenet.html

3. Udacity deep learing course: https://www.udacity.com/course/deep-learning--ud730
4. Convolutional Neural Networks for Visual Recognition: http://cs231n.github.io/convolutional-networks