Facial Keypoint Detection

Implemented in Keras, Tensorflow

Sarthak Agarwal
Northeastern University
sarthak.ag@gmail.com

***Abstract*—This paper describes an approach to detect facial key-points on greyscale images of faces. The dataset is from 2016 Kaggle competition. Facial key-points are centers and corners of the eyes, eyebrows, nose and mouth, among other facial features. The first step discussed here is to preprocess the data for the model then build a baseline model with one layer and different parameter combinations. Next step is to build a Convolutional Neural Network with many layers. The second stage discuss the data augmentation technique to increase the data and build a convolutional neural network on top of it.**

I. Introduction

A. Business value

My primary motivation for this project was my interest in applying deep learning to significant problems with relevant uses. The applications of this research are numerous and significant, including facial expression analysis, biometric or facial recognition, medical diagnosis of facial disfigurement and even tracking of line of sight. I specifically chose the Facial Keypoint Detection dataset because it gave me ample opportunity to experiment with a wide variety of approaches and neural net models.

B. Problem Statement

The task is to predict the coordinate locations of points on an image of a face. For 15 facial points, there are 30 numerical values that needs to predicted: a 2D (x, y) representation for each facial point. Therefore, the input is an array of greyscale images from the dataset and the output is 30 floating point values between 0 and 96 indicating the location of the facial keypoint i.e. the (x, y) coordinate.

In the below images, each greyscale image is the input and the coordinates marked in blue are the facial keypoint.

Fig. 1. Data

II. Dataset

This dataset is provided by the University of Montreal Medical school which contains 7049 greyscale images. The images are of 96x96x1 dimensions in which facial keypoints are labelled as (x, y) coordinates for 15 facial keypoints. The facial keypoints are: 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.

III. Architectures

A. Baseline Model

I first implemented a baseline model against which I would compare the other successive models. This model is a basic neural network with one hidden layer that has 100 neurons and hyper parameters were set as:

Parameter	**Value**
Learning rate	0.01
Momentum	0.9
Nesterov	True

Fig. 2. Hyperparameters for baseline Model

I also ran the model with the default parameter values that Keras provides, which were as:

Parameter	**Value**
Learning rate	0.01
Momentum	0.0
Nesterov	False

Fig. 3. Another Hyperparameter setting for baseline model

B. *Convolutional Neural Network Model*

The second model that I implemented was a neural network which consisting of alternating convolutional layers and max pooling layers, followed by fully connected hidden layers. There are total of three convolutional layer where the first layer had 32 filters of size 3x3, second layer had 64 filters of size 2x2 and third layer had 128 filters of 2x2. The pooling size for each was 2x2. After the convolutional layers, the next layer is flatten layer where the images are flatten to input to the series of two hidden layer with 500 neurons each.

Layer	Name	Size
0	Input	1x96x96
1	Conv2d	-
2	Maxpool	-
3	Con2d	-
4	Maxpool	-
5	Conv2d	-
6	Maxpool	-
7	Flatten	-
8	Dense	500
9	Dense	500
10	Output	30

Fig. 4. Multiple layer CNN

C. *Data Augumentation*

Data augmentation lets us artificially increase the number of training examples by applying transformations, adding noise etc. That's obviously more economic than having to go out and collect more examples by hand. In my case, I flipped the image on a vertical axis and then remapped the images with the facial keypoints. For flipping I define a tuple that holds the information about which columns in the target vector need to swap places when we flip the image horizontally. For example, left_eye_center_x will need to swap places with right_eye_center_x, hence we write down the tuple (0, 2). left_eye_center_y needs to swap places: with right_eye_center_y. we write (1, 3) and so on. I then ran the neural network shown in fig 4 on the augmented data.

IV. RESULTS

For each model, I spent a significant amount of time on tuning and trying out different hyperparameters. I ran the two baseline models on CPU and for convolutional neural network, I leveraged a GPU instance on AWS.

Model	**Model Description**	**Epoch**	**Time**	**Train Loss**	**Val Loss**
Model 1	NN with 1 hidden layer, 100 nodes	100	3mins CPU	0.0024	0.0037
Model 2	Model 1 + momentum=0, nesterov=False	100	3mins CPU	0.0070	0.0080
Model 3	CNN	500	40mins GPU	0.0012	0.0020
Model 4	CNN – Flipped images	Not available	-	-	-

Fig. 5. Results

As it can be observed from the above table, that model 3 has the least training and validation loss. Below is the prediction result on a new image.

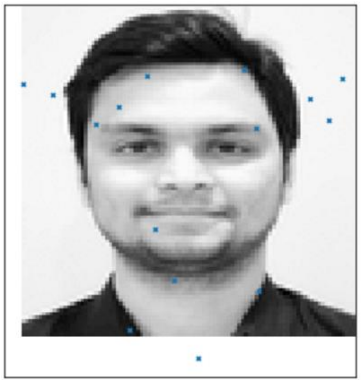

Fig. 6. Baseline NN v1

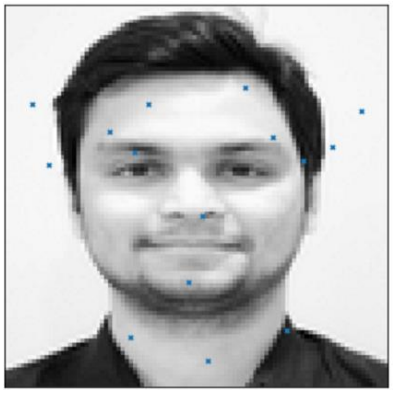

Fig. 7. Baseline NN v2

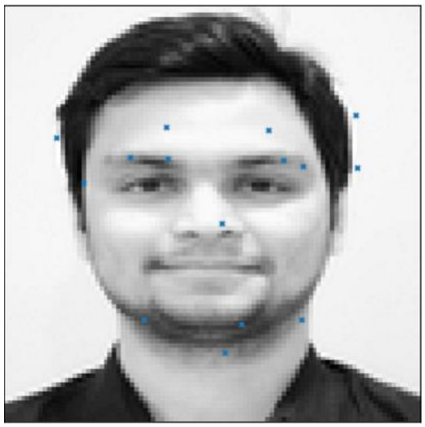

Fig. 8. CNN

We can observe that although not perfect, the facial key-points are getting better.

V. Tools

The tools used were:
Keras 1.2.2
Tensorflow 1.0.1
AWS GPU g2.2xlarge
CUDA 8.0
Python 3.5.2

VI. Future Work

In the future, I would like to run the models for 3 times more epochs. In addition to that a deeper neural network will also result in lesser loss. Thirdly, I would like to run the neural network on the augmented data as well.

References