IMC-5103B Machine Learning 2

Report Modern face recognition with deep learning

Sahar Hosseini

October 2019

This report, follows as Face detection, faces are extracted from images, Pose estimation face alignment has done, Face encoding, encoding the extracted faces, Face recognition, some classifier has defined, Result, compere result of classifier and Conclusion.

Face detection

For this part `saveFaces` function has written, which have 3 arguments, first path of data, second is the output file name and the third is the label file names. Then read images and pass them in to `extract_face` with hog feature detector.

The HOG features are widely used for object detection. HOG (Histogram of Oriented Gradients) decomposes an image into small squared cells, computes a histogram of oriented gradients in each cell, normalizes the result using a block-wise pattern, and return a descriptor for each cell. In order to detect face features we convert images to gray scale the pass to the `face_locations` function. The output is in the form of a list of faces with the (x, y) coordinates of the diagonal corners. Based on output coordination crop the faces from images by 128*128 size. Extract the person names as labels from the image file names. Both labels and images save as npy file. To be accessible for further operation.

Some advantages of HOG

1. Fastest method on CPU
2. Works very well for frontal and slightly non-frontal faces
3. Light-weight model as compared to the CNN.

Some disadvantages of HOG

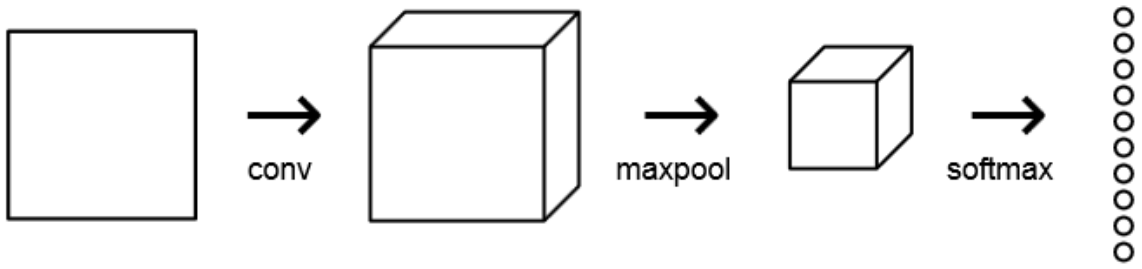
4. The major drawback is that it does not detect small faces as it is trained for minimum face size of 80×80.
5. The bounding box often excludes part of forehead and even part of chin sometimes.
6. Does not work for side face and extreme non-frontal faces, like looking down or up.

In order to train our new data reading form the save files in previous step. First we need to normalize our data, as you know some classifications algorithm like Neural networks using small weight values as input, and inputs with large integer values can disrupt or slow down the learning

process. It is valid for images to have pixel values in the range 0-1 and images can be viewed normally. This can be achieved by dividing all pixels values by the largest pixel value; that is 255. This is performed across all channels, regardless of the actual range of pixel values that are present in the image, then we normalize the output by dividing to 255 for each pixel, we vectored the labels to be suitable for convolutional model, then create the test set 30% of data and train set 70% of data. In order to define our training model `create_model` function has written in this function we define convolutional neural network.

A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that convolve with a multiplication or other dot product. The activation function is a RELU layer, and is subsequently followed by additional convolutions such as pooling layers and fully connected layers. In other word Convolution is the act of taking the original data, and creating feature maps from it. Pooling is down-sampling, most often in the form of "max-pooling," where the region select, and then take the maximum value in that region, and that becomes the new value for the entire region.

In our case, the model by using the Sequential class, which represents a linear stack of layers with 32 filter by 3*3 kernel with Relu activation following by additional convolutions and pooling layers has built. Then feed the last output from the convolutional base into dense layer to perform classification. Dense layers take vectors as input which are 1 dimension, while the current output is a 3 dimension (64, 3, 3). So flatten the 3 dimension output to 1 dimension, and add one more dense layers on top with Relu activation. Then we create the final dense layer with 6 outputs which is number of classes with the Softmax activation (for non-linear regression). Below pictures is represents a simple sequential of layers in our model. To compile our model we used Adam optimizer it is a gradient-based optimizer. Then for the loss function categorical_crossentropy has been used because we used Softmax for the output layer if we used sigmoid we should use binary_crossentropy. Since this is a classification problem, the accuracy metric has set.



Training a model in Keras consists only of calling fit() and specifying some parameters. The training data (images and labels), known as X and Y, respectively. The number of epochs iterations over the entire dataset to train for. The batch size is a hyper-parameter that defines the number of samples to work through before updating the internal model parameters.

The result:

```
Test accuracy: 72.91666865348816 %
```

Pose estimation

The goal of this part is align the extracted faces from previous part then train and see what differences between the results is. By using so read extracted faces from the file and apply landmark and alignment function to align faces which outputs are the face-centered to the image, rotated such that line joining the center of two eyes is parallel to the horizontal line and it resizes the faces to identical scale. Then we normalized, vectored and train the model which create in previous part by new data. As you could see the accuracy is getting better.

```
Test accuracy: 79.16666865348816 %
```

Then we could conclude the face alignment helps to improve face recognition accuracy.

Face encoding

In this part, we encode the face which extracted in the first part to extract a few basic measurements from each face. This is very useful due to avoid creating a big convent model when we have trillion faces. So we create the simple convolutional model with `create_model_simple` function with the Sequential class with two dense layer by Rule activation with input dimension of images and Softmax activation for output layer with number of classes. Then compile the model with Adam optimizer it is a gradient-based optimizer and the loss function categorical_crossentropy. In this part no need to normalized the data by dividing 255, because by encoding the data and shape of data now is different although new data by encoding normalized. So train the model with new endcoing data, as you could see the accuracy is getting better respectively to previous parts.

```
Test accuracy: 91.66666865348816 %
```

Then we could conclude the face encoding helps to improve face recognition accuracy.

Face recognition

In this part, 3 classifier as KNN (K-nearest neighborhood), SVM (Support Vector Machine) and logistic regression has defined. Encoding data has trained by 3 classifiers. In order to create test set and train set for these classifier the labels does not vectored.

KNN classifier, is a supervised learning and predict the target label by finding the nearest neighbor class. The closest class will be identified using the distance measures like Euclidean distance. I choose the I read some articles that mentioned the small value of K means make noise and will have a higher influence on the result i.e., the probability of over-fitting is very high. A large value of K makes it computationally expensive and defeats the basic idea behind KNN (that points that are near might have similar classes). So I chose K=5 and test the result by example images.

SVM classifier, is a supervised learning technique from the field of machine learning applicable to both classification and regression. In the case of classification, construct an optimal separating hyper plane in this space (a maximal margin classifier), in our classifier we define a SVM with kernel type linear and C parameter as Error Penalty in 1.0 to avoid over-fitting of classifier.

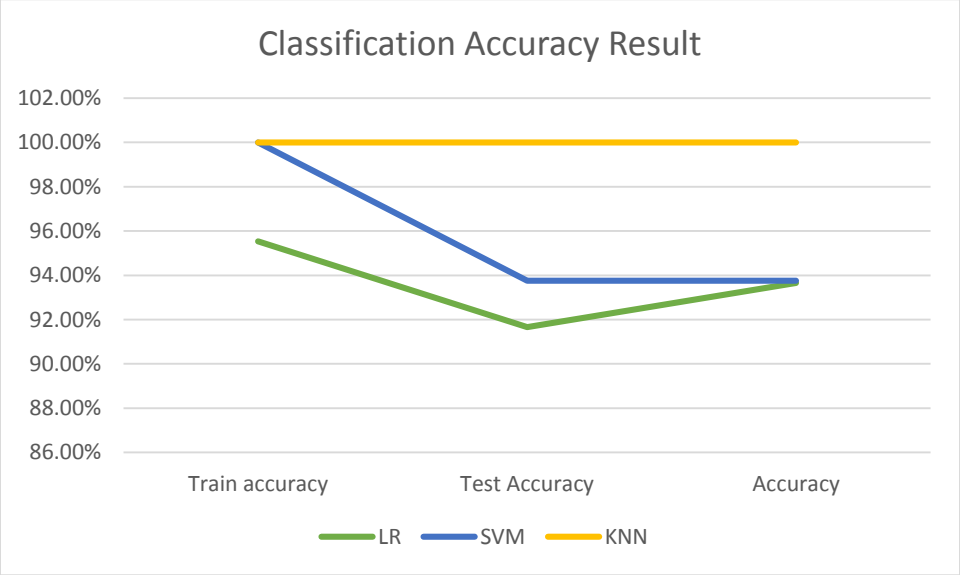
Logistic Regression classifier, is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables. It uses the logistic function to find a model that fits with the data points. It used sigmoid function to classify and cross-entropy cost function measures the performance of a classification model whose output is a probability value between 0 and 1.

Result

We predict test set and train set data with trained model in previous part with three classifier. Below table and chart represent the result.

Logistic regression (LR) classifier has 95.53% accuracy on train set and 91.66% accuracy on test set. SVM has 100% accuracy on train set and 93.75% accuracy on test set. KNN has 100% precision on both train and test set.

	Train accuracy	Test Accuracy	Accuracy
LR	95.53%	91.66%	91.66%
SVM	100%	93.75%	93.75%
KNN	100%	100%	100%



Conclusion

Based on result KNN and SVM classifier have 100% accuracy. But on example images KNN classifier had better detection than SVM. Then KNN classifier chose to do face recognition on provided video.