Recognition of Face and object

Xueyi Fan, Yuchen Xiao

Northeastern University

Abstract

We first apply Cascade Classifier and Fisher Face Recognizer in the OpenCV library to perform face detection and recognition in a real-time video. And then, By implementing the convolutional neural network (CNN) and the corresponding modules provided by Caffe library, we achieve face detection and recognition on a set of given pictures. Regarding the object detection and recognition in either a given picture or a real-time video, we finish it by using YOLO algorithm and its system. At last, by combining face/object recognition, manipulation and navigation techniques, we enable a fetch robot to deliver an object to a right person.

Introduction

Human starts to use the eyes to gather and learn the information about the around world as soon as he was born. Human's brain and eyes constitute a very strong, fast and accurate visual system, which enables us to easily detect and recognize object, animal and people's face in an image, video and daily life, as long as these things have been known. The interesting questions are how our visual system addresses the detection and recognition task, how we analyze an image and how our brain process and encode the information, and then return a result. The neurophysiologists have shown us that there exists some specialized nerve cells in our brain that are very sensitive to the specific local features of a scene, such as lines, edges, angles or movements, and will give us the corresponding responses to these features. Every time when we see a new object or a new face, our visual cortex captures its special features to learn and builds a pattern which will be used for recognition in the future. Next time, when we see the similar object or the same face, our visual system will compare the features with the patterns existing in our brain to make the classification.

Along with the research in computer vision and machine learning, fast and accurate algorithms for object and face recognition have been designed and widely applied in the variety of areas including robotics, surveillance, security system, self-driving car and human-computer interaction. These algorithms allow the computer to learn the features of the pixels in an image or a video, and then to evaluate them for detection and recognition. Especially, by implementing Deep-leaning technique, the speed and accuracy of object/face detection and classification have been significantly improved.

Basically, object recognition falls into object detection problem, which is to localize objects in an image or a video, and classification problem, which is to predict the label of the target object in the predefined labels. Face recognition is one typical application of object recognition algorithms, meanwhile, the diversity and complexity of human face's expressions and features make this problem more interesting and challenging.

In this project, we finish the following tasks:

- By using OpenCV to perform face recognition, we first prepare our own pictures as the requirements and add them to the AT&T Face Database. Second, we train a Fisherfaces model based on the given images in the AT&T Face Database. After the model is ready, we first apply the Cascade Classifier to detect faces in a real-time video and then use the trained Fisherfaces Recognizer to predict the face.
- To extend the experiments on face recognition, we build a CNNs LeNet5 model with Caffe framework (Jia et al. 2014). The same face data are used. Firstly, We randomly separate images into three parts, a training set (80%), a validation set (10%) and a testing set (10%). Then we train a LeNet5 model using training set and validation set to get an optimized model. Finally, we apply the model to predict the classification of images in the testing set.
- Regarding the object recognition, we apply YOLO system. We first train the model based on the given database to get the convolutional weights file, and then transfer the YOLO model to Caffe model. In the experiment, we perform the object recognition on a set of images and a real-time video.
- In the last experiment, we successfully enable a fetch robot to pick up an object from a table and then to navigate around the room to search for the target person for delivering the object.

Background and Approach

A. Cascade Classifier and Fisherfaces

The first step of face recognition is face detection. There

Copyright © 2017, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

are several different Haar Feature-based Cascade Classifiers provided by OpenCV library for face/object detection, which have been already trained with hundreds sample images of the particular face/object. The one we applied in our project is called "haarcascade-frontalface" classifier. For searching a face in an image, the classifier will be used as a searching window to move across the image to check all the interest regions that if it is likely a face shown there. Different classifier uses different Haar-like features, such as Edge features, Line features, Center-surround features, which is used to evaluate the interest regions during the searching process.

After a face is detected in an image, we use the Fisherfaces recognizer to predict the label of the face among the labels of the training images. Fisherfaces algorithm is an example of a class specific linear methods for dimensionality reduction proposed by Sir R. A. Fisher. Instead of maximizing the overall scatter, it tries to maximize the ratio of the between-classes to within-classes scatter in order to make it more reliable for classification.

To compute the Fisherfaces model, let X is a set including c classes, and each X_i is a vector with n samples selected from the ith class:

$$X = \{X_1, X_2, ..., X_c\} \tag{1}$$

$$X_i = \{x_1, x_2, ..., x_n\} \tag{2}$$

The within class differences are computed using the within-class scatter matrix which is defined as

$$S_W = \sum_{i=1}^{c} \sum_{x_j \in X_i} (x_k - \mu_i)(x_k - \mu_i)^T \tag{3}$$

and the the between-class scatter matrix is defined as

$$S_B = \sum_{i=1}^{c} N_i(\mu_i - \mu)(\mu_i - \mu)^T \tag{4}$$

where μ_i is the mean of the ith class, μ represents the mean of all classes, and N_i is the number of samples in the ith class.

The next step is to search for the optimal projection W_{opt} which maximizes the ratio of the projected between-class scatter matrix to the projected within-class scatter matrix.

$$W_{opt} = \arg\max_W \frac{|W^T S_B W|}{|W^T S_W W|} \tag{5}$$

The solution of this problem is given by the generalized eigenvalue decomposition

$$S_B V = S_W V \Lambda \tag{6}$$

where V is the matrix composed of eigenvectors, and Λ is diagonal matrix with eigenvalues.

B. Convolutional Neural Networks

Convolutional Neural Networks (CNNs), are hierarchical neural networks. Like the regular neural nets, CNNs consist of layers, and each layer contains a set of neurons. Each

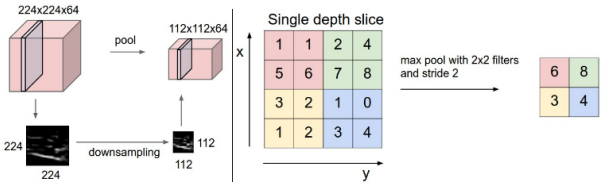

Figure 1: Pooling layer.

layer receives input and transforms data applying a dot product. A series of downstream calculations optionally are selected to execute. CNNs also have a loss function and a full connection layer. However, CNNs are specially designed for images as the input with specific architectures.

The basic CNNs contain three layers: Convolutional layer, Pooling layer, and Normalization layer. People can use these three main layers to construct the CNNs architectures.

- Convolutional Layer:
  The parameters of this layer are a set of features. Every feature has a fixed size which represents a multi-dimension data (i.e. $3 \times 3 \times 3$). During forward pass, each feature will be convolved across the whole image data and compute dot products between the entries of the filter and the input at any position.
- Pooling Layer:
  To reduce the spatial size of network data and control overfitting, a pooling layer will insert between two successive convolutional layers. An operation $\max$ will be called here. People can set a size of filter with a stride, and selected the maximum number among numbers in the range of the filter at every depth slice. Finally, the depth dimension remains unchanged but width and height reduce (see Figure 1).
- Normalization Layer:
  A rectification layer(ReLU), follows convolutional and pooling layers, which normalizes the dot product from convolutional layer. There are multiple types of normalization, and the $max(0, x)$ thresholding at zero is a common method applying elementwise non-linearity and could lead to faster convergence.

C. Yolo Real − Time Object Detection Algorithm

Yolo is an extremely fast object detection system, even 100x faster than Fast R-CNN, which is the significant reason why we would like to learn the algorithm and apply it in our project (Redmon and Farhadi 2016).

The approach used by R-CNN is first to generate the possible bounding boxes with different scales at multi-regions in the image where is likely to have objects, and then separately run the classifier on these bounding boxes. After this, a refinement process, called post-processing, will remove the duplicate detections to derive a better outcome (Redmon et al. 2015). The big disadvantage of R-CNN is that it is not able to evaluate all the bounding boxes on different regions at the same time for prediction, which means that the network evaluation process probably needs to be run thousands

of times for a single image. Since this drawback, R-CNN runs slowly.

Unlike R-CNN, YOLO uses a totally different approach. It frames the detection as a regression problem and implements a single neural network to the whole image, and it is able to capture all the features in the image to do the prediction for all bounding boxes simultaneously. YOLO neural networks architecture includes 24 convolutional layers followed by 2 fully connected layers. The final prediction of this network is a $7 \times 7 \times 30$ tensor.

Experiment and Results

A. Face Recognition in Real − Time Video (OpenCV)

The first experiment is to apply the classifier and recognizer provided by OpenCV library to perform face detection and recognition. The result is shown in Figure 2.

During the experiment, we find that the accuracy of the recognition is not very stable, which depends on the distance from our faces to the camera and the illumination of the environment. We think there are two essential reasons: first, our training data are gray-scale images, which causes that the available features are not good enough to learn; Second, Fisherface algorithm doesn't concern any neural network structure, which results in the learning effective and the accuracy of prediction cannot be competitive with any Deep-learning algorithm.

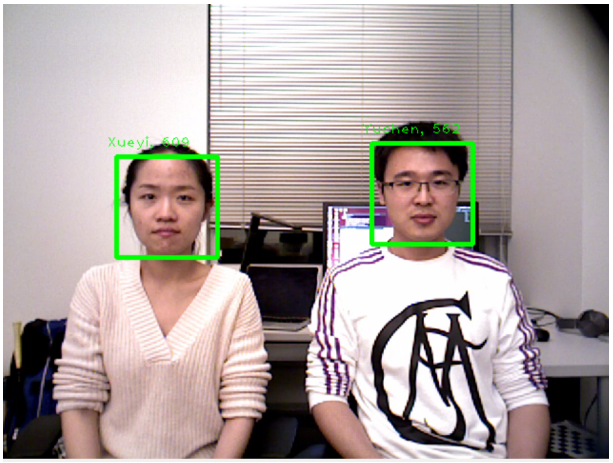

Figure 2: Face Recognition

B. Face Recognition (CNN)

When we implement CNNs with caffe on face recognition, we try LeNet and AlexNet architectures on our database. Yann LeCun developed LeNet on 1990s which is the first successful application of CNNs. There are other several famous architectures, i.e. GoogLeNet, VGGNet, Micriosoft ResNet. They all have great performances on image classification. A branch work for deep face recognition with caffe implementation named caffe-face provides a single model trained by CAISA-WebFace and achieves 99% expected accuracy on LFW. However, the main concern for us to use these architectures is the memory bottleneck, even though we use GPU to accelerate the calculation speed.

We download AT&T face database and adding our own pictures and build a new image library. It contains 42 classes which refer to 42 people and each person has 10 images. We randomly separate these images into three parts: a training set(338 pictures), a validation set (41 pictures), and a testing set(41 pictures) and create their label files. By using the tool provided by caffe, we transform the training set and validation set into LMDB format respectively and calculate the mean of the training images.

Then we write a python script to build layers of LeNet5 model. The architecture of this model is [Conv1 – Pool1 – Conv2 – Pool2 – InnerProd1 – RELU1 – InnerProd2 – Loss] The first layer is input, and we use training and validation data with batch size 41. The second and third layer are similar. Each of them consists of a convolution layer and a pooling layer. We select 5 kernels and use xavier as weight filter. In pooling layer, we set the kernel size and strider as 2 and utilized max operation. The fourth layer combines two inner product neurons and one rectification function. The last layer is a loss function and assigned the number of final classification of LeNet5 model is 42. (see Figure.3)

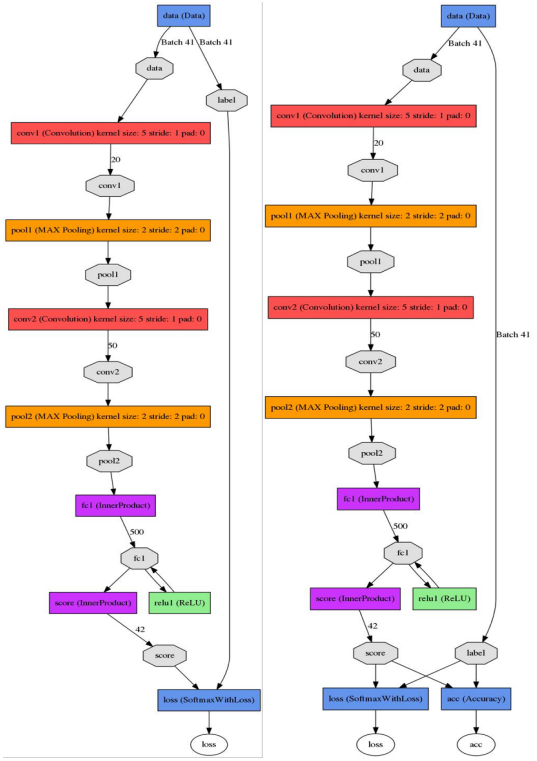

Figure 3: LeNet Architecture

Next, we generate a solver.prototxt file containing other important parameters. In the solver file, we set base learning rate as 0.001, the iteration number is 3000. The value of test interval and test iteration time are based on the size of testing set.

We try three combinations of iteration time and learning rate and compare the performances among them (see Table 1). As we decrease the learning rate from 0.01 to 0.001, the accuracy rate increase from 0.049 to 0.95. We continue to decrease the learning rate, but it doesnt have much effect on accuracy rate. Finally, we choose the model with learning rate 0.001 and 3000 iterations. After training the model with our face data, we get average 0.95 accuracy rate and loss 0.254038.

Table 1: Result of LeNet

Iteration	Learning Rate	Accuracy Rate	Loss
3000	0.01	0.0487805	87.3365
3000	0.001	0.951219	0.254038
1500	0.0005	0.951219	0.180764

We also try Alexnet architecture which is made up of five convolutional layers, max-pooling layers, dropout layers, and three fully connected layers(see Figure 4). However, this architecture cannot show us a good result. Using 10000 iterations and 0.0001 learning rate, we get only 0.24 accuracy rate(see Table 2).

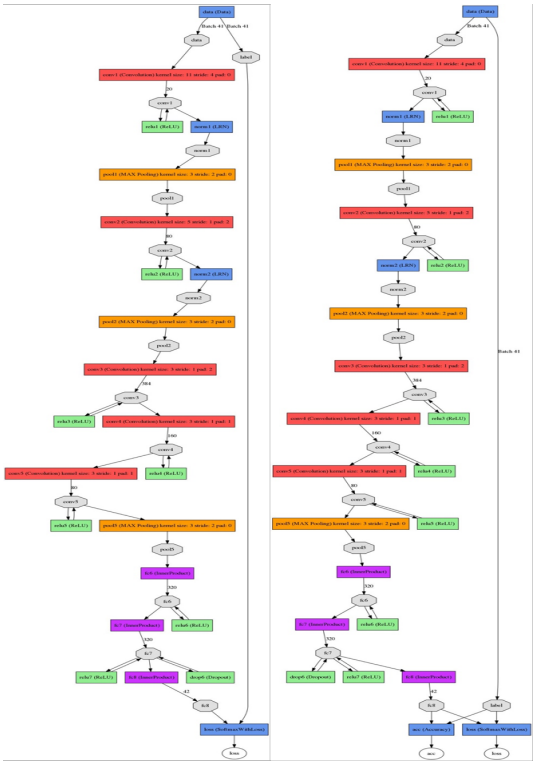

Figure 4: LeNet Architecture

By checking the model efficiency, we create a prediction prototxt file with the same setting as validation in LeNet. However, the prediction result is bad. The accuracy rate for the testing set is 2.43%.

There are several reasons for the results of our experiment on CNNs model. First, we only use 420 images to train the model. The data size is too small to use in a deep learning model like AlexNet and it cannot provide enough information for the computer to learn. Second, we set 10000 iterations for training Alexnet model which may be far away from convergence. Third, the contradiction of high validation accuracy rate and low testing accuracy rate might be caused by model overfitting. A change of gamma value may be a solution for it.

Table 2: Result of Alexnet

Iteration	Learning Rate	Accuracy Rate	Loss
3000	0.01	0.0243902	3.72706
3000	0.001	0.0243902	3.72245
10000	0.0001	0.243902	3.72639

C. Object Recognition

After finishing training the YOLO model, we input a set of images to YOLO system to check the object recognition performance, the result of which is shown in Figure 3.

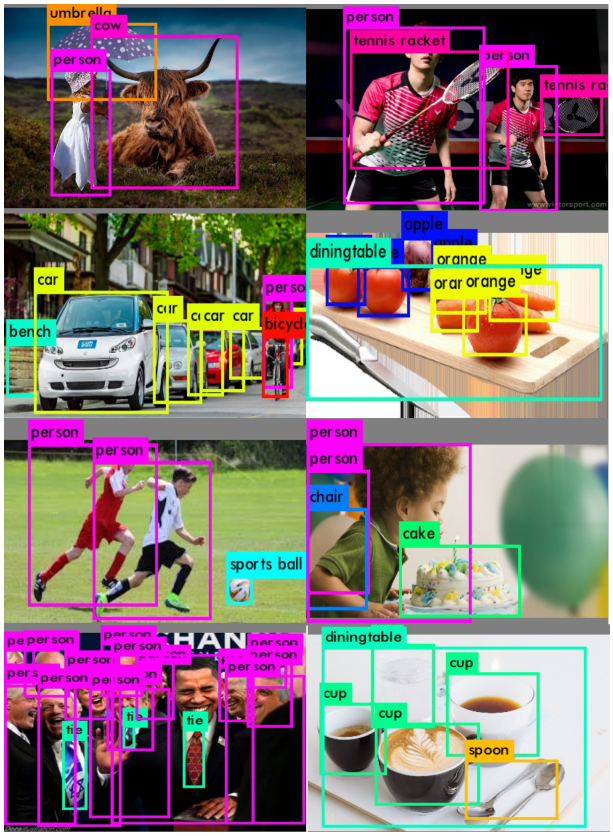

Figure 5: YOLO Objects Recognition.

D. Fetch Robot Delivery

In the last experiment, we would like to make a real robot to be more assistive and collaborative for human. We successfully launch our face/object recognition code on a real

robot and combine the ROS navigation stack to enable the robot to move around in an office room and send an object to a particular person. A video has been uploaded to our git repository, please watch it if you are interested.

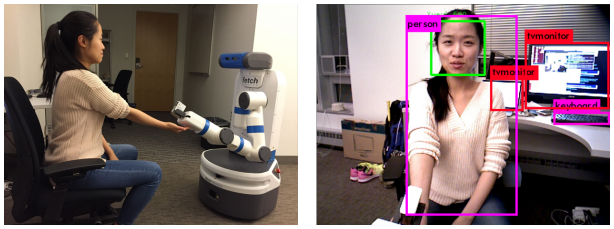

Figure 6: Fetch Robot Delivery

Conclusion

We successfully implement image recognition based on CNNs and related frameworks. We apply YOLO on object recognition and achieve high-speed performance. We make object recognition by the average speed of 155 frames per second. To implement face recognition, we compare CNNs and traditional face recognition algorithms and utilize popular open source framework Caffe and OpenCV. Two of CNNs architectures, LeNet and AlexNet are built by caffe-python and parameters such as number of hidden layers, learning rate, batch size, iteration number are selected to optimize the prediction model. On LeNet, the best model achieve a recognition accuracy rate of 0.95. Visualization of results for face detection and recognition are accomplished using OpenCV. By setting up a connection with PC camera, we capture our face images and label our faces by showing rectangles around face regions. Good results require big data collections and deep learning networks, our small project on the face and object recognition raise a good start of studying CNNs.

References

Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R. B.; Guadarrama, S.; and Darrell, T. 2014. Caffe: Convolutional architecture for fast feature embedding. *CoRR* abs/1408.5093.

Redmon, J., and Farhadi, A. 2016. YOLO9000: better, faster, stronger. *CoRR* abs/1612.08242.

Redmon, J.; Divvala, S. K.; Girshick, R. B.; and Farhadi, A. 2015. You only look once: Unified, real-time object detection. *CoRR* abs/1506.02640.