QUY VU

INM460 – COMPUTER VISION

COURSEWORK REPORT

FACE RECOGNITION AND OPTICAL CHARACTER RECOGNITION

City, University of London

22 April 2018

1. Overview

The aim of this project is to develop a facial and optical character detection and recognition in images and video. To achieve this objective, regions of interest containing faces and optical characters were identified using cascade classifiers. Then, various features were extracted and inputted to a set of classifiers. In face recognition task, the results suggested that Convolutional Neural Network performed the best with 97% accuracy on the test set. For Optical character recognition task, an accuracy of 85% were obtained on images and 97% on videos.

2. Data preparation

A face detection model was trained using the Matlab's built-in function trainCascadeObjectDetector. To create a database of positive examples to train this model, 400 faces were manually labeled using the Matlab's built-in application ImageLabeler. In addition, the database for negative images constitutes of various background images obtained from the internet, and photos of individuals provided in the original dataset with faces covered by a colored rectangle. However, this model was prone to false positives.

As a result, the Matlab pre-trained model CascadeObjectDetector with a merge threshold of 10 was utilized to detect faces This model was applied to images and videos to detect faces, which resulted in bounding boxes marking the detected faces.

The images were cropped according to the resulted bounding boxes and then stored in separate folders. For videos, the above operations were applied to each of the frames. Faces recognized in the group photos were then moved to respective folders according to their labels. Unlabeled faces were then move to the same folder named "555".

To prepare the dataset to train a convolutional neural network, images were stored under the Matlab's imageDatastore format, then partitioned by an 80:20 ratio to create training and test data. For other models, the images were stored as imageSet, applied a 80:20 partitioning and extract relevant features.

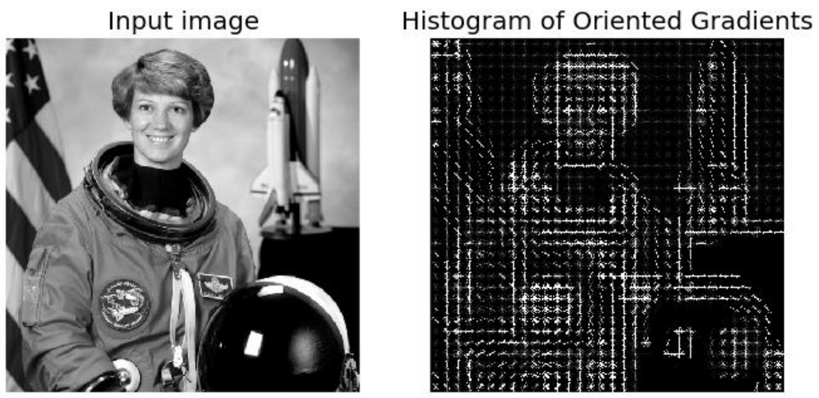
In addition, to test the Random Forest ability to learn with one single example, an average profile of each individual was created by averaging all images corresponding to that individual.

Throughout this process, image data were converted into GPU Array so that the operations can be carried out on a GPU, which hastened the process. However, this is not applicable for all operations.

3. Feature extraction

3.1. Histogram of Oriented Gradients

Figure 1 – Input image and its respective HOG



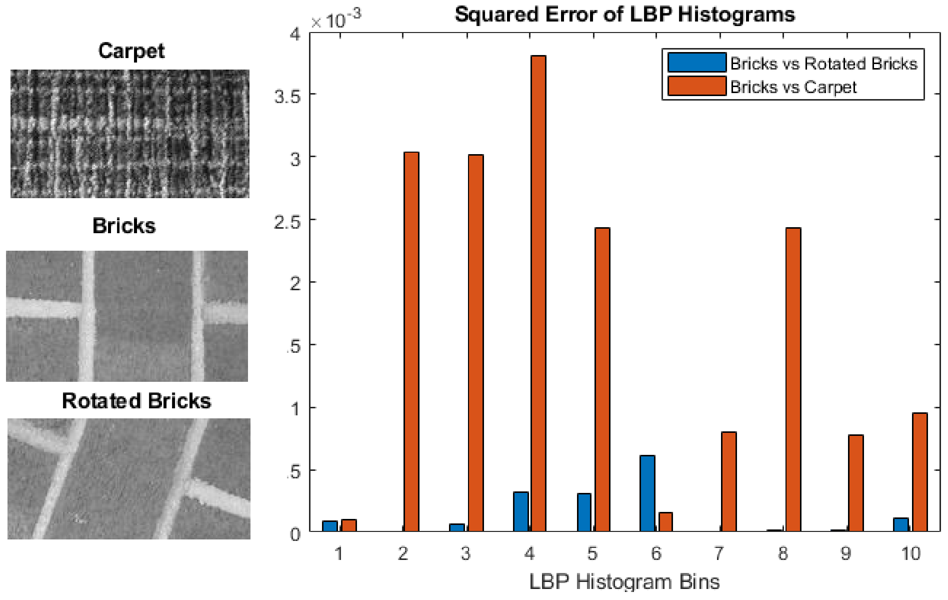
(Source: sci-kit image documentation)

Histogram of Oriented Gradients (HOG) is one of the most popular feature descriptors used in image processing. This technique counts and returns the most dominated gradient orientation located in a specific region of the image. As such, this technique is particularly efficient in detecting edges and corners the magnitude of gradient is large around them.

In this project, images are first converted to grayscale. HOG features were then extracted from the images using a cell of size 32 by 32.

3.2. Local Binary Patterns

Figure 2 – Squared error of LBP Histograms between image of carpet and bricks



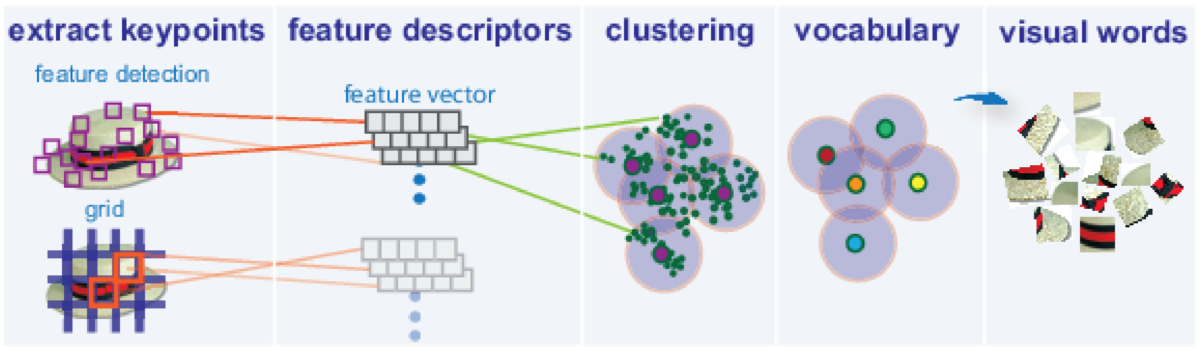
(Source: Matlab documentation)

Local Binary Pattern is another type of feature descriptor introduced by Olaja et. al in 2002. It computes a local representation of the texture, which is constructed by comparing each pixel with its surrounding. This is useful in differentiating corner, surfaces and edges.

In this project, LBP features were extracted from greyscale image using a cell of size 16 by 16. Rotation information is also encoded.

3.3. Bag of visual words

Figure 3 – Illustration of the Bag of visual words model



(Source: Matlab documentation)

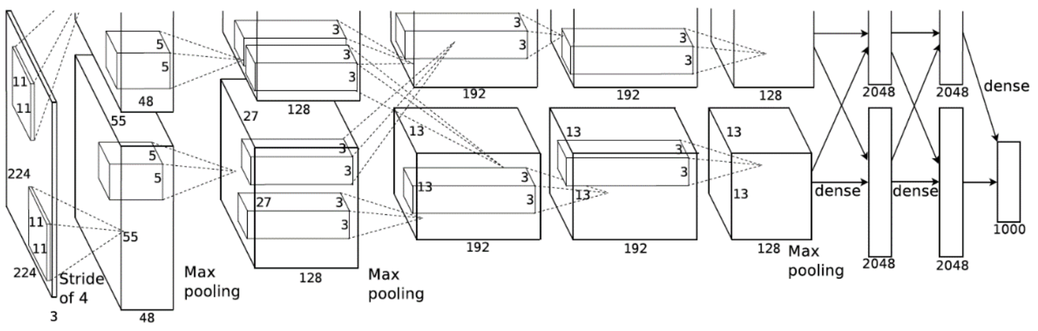
Bag of Visual Words represents a model which extracts features from images then represents them as feature vectors. Afterwards, a clustering technique is then applied to group similar features together, which creates a vocabulary. This output can then be introduced to classifiers.

In this project, the final vocabulary is composed of 500 clusters. This was done by the K-means clustering algorithm after 100 iterations.

4. Classifiers training

4.1. Convolutional Neural Network

Figure 4 – AlexNet's architecture



Source: A. Krizhevsky et. al, 2012

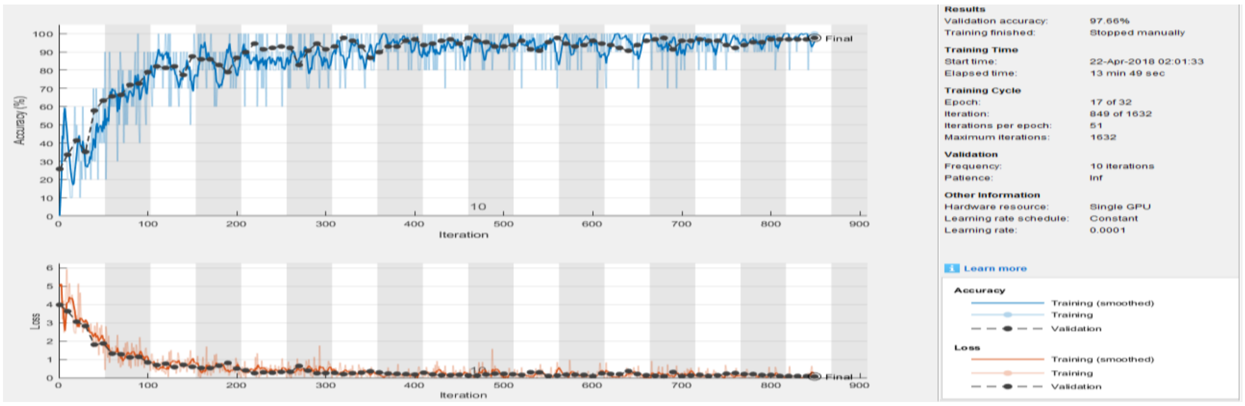
A convolutional neural network in a special class of artificial neural network using local connections and weight-sharing. These networks require minimal pre-processing [1]. They are based on biological processes and have been very successfully used for analysing two-dimensional shapes.

They comprise of an input layer and a series of feature maps which alternate between convolution and subsampling layers. The convolutional layers apply a convolutional operation to the input, passing the result on to the next layer, so that there are a fewer number of free parameters, allowing a deep network without the large number of parameters than would be needed otherwise in a fully connected feedforward network. The weights are the same in convolutional layers, which means the same filter can be used in each receptive field. The subsampling layers either take the largest value or average the values of one section of an image and creates a new reduced number of features maps from the previous layer. So essentially, the convolution stage creates a series of smaller feature maps while the subsampling stage reduces the number and size of these maps.

There are several benefits of these networks, mainly that they have an increased generalization ability. As previously stated, they currently seem to provide one of the best image recognition solutions. However, in real life applications they may not perform as well as expected [2].

In this project, transfer learning was performed to minimize the training effort. The original Alexnet was loaded, had the last 3 layers removed. On top of that, a fully-connected layer of 54 nodes corresponding the 54 classes were added on, followed by a softmax and classification layer. The network was then trained on the training data, optimized with mini-batch gradient descent with momentum. To accelerate the learning, the learning rate of the added layers were set to be high, then quickly decreased. In addition, to avoid overfitting, data augmentation was also implemented, which flipped the image horizontally and rotated by -20 and 20 degrees vertically and horizontally.

Figure 5 – Convolutional neural network training progress



Results	
Validation accuracy:	97.66%
Training finished:	Stopped manually
Training Time	
Start time:	22-Apr-2018 02:01:33
Elapsed time:	13 min 49 sec
Training Cycle	
Epoch:	17 of 32
Iteration:	849 of 1632
Iterations per epoch:	51
Maximum iterations:	1632
Validation	
Frequency:	10 iterations
Patience:	Inf
Other Information	
Hardware resource:	Single GPU
Learning rate schedule:	Constant
Learning rate:	0.0001

Accuracy
— Training (smoothed)
● Training
-- ● -- Validation

Loss
— Training (smoothed)
● Training
-- ● -- Validation

4.2. Support Vector Machine

In summary, Support Vector Machine (SVM) is a supervised learning algorithm that seeks to classify groups so as a maximum margin between each class is obtained. In addition, non-linearity can be addressed with a kernel trick – which is a technique that maps the data into a higher dimensional space where it can be separated by a hyperplane.

SVM has been one of the most preferred classification algorithm for being simple yet efficient. Despite originally being a binary classifier, SVM can works on multi-class classification task by using the one vs all approach. Due to its nature of maximise a margin, it works extremely well on imbalanced dataset without the interference of specific techniques such as undersampling, oversampling and assigning class weight.

In this project, a SVM algorithm with a gaussian kernel, one-vs-one design was trained on the training dataset.

4.3. Decision tree

Decision tree is a popular non-parametric machine learning algorithm for regression and classification tasks. In this algorithm, the following tasks are performed recursively:

- Create root node for the tree
- If all examples are positive, return leaf node ‘positive’
- Else if all examples are negative, return leaf node ‘negative’
- Calculate the entropy of current state H(S)
- For each attribute, calculate the entropy with respect to the attribute ‘x’ denoted by H(S, x)
- Select the attribute which has maximum value of Information Gain (S, x) (IG)
- Remove the attribute that offers highest IG from the set of attributes
- Repeat until we run out of all attributes, or the decision tree has all leaf nodes.

Decision tree is preferred for its interpretability while still outputting good results for some complex problems. However, it is prone to overfitting and may not generalize well in numerous cases.

In this project, a decision tree without maximum depth and a minimum size at each parent being 10 was fitted to the training data.

4.4. Random Forest

Random Forest is an ensemble of decision trees, in which each tree are trained on a subset of the data. Random Forest then output its prediction by either averaging the output of all trees in a regression task, or by major voting in a classification task.

Random Forest is one of the most popular algorithm in machine learning, especially Computer vision. One of the main advantages of Random Forest is performance is almost always guaranteed as the number of trees in the model increases. However, this comes with a trade-off between performance vs. interpretability and computational cost.

In this project, Random Forest models of 50, 100, 150, and 200 trees with minimum leaf size of 1 were created. The best model was selected based on the performance on the test data.

5. Experimental results

5.1. Face recognition

Face detection

As discussed earlier, Matlab's built-in model CascadeObjectDetector was used to detect faces. However, this model produced false negatives in group photos, which means that there are undetected faces. This is due to lighting conditions, partially coverage of faces.

Face classification

The table below summarizes the classification results by classifiers

Classifier	Feature	Training acc.	Validation acc.	Training time
CNN (Alex Net)	N/A	99.25%	97.66%	13 min
SVM	SURF	99.58%	89.51%	< 1 min
SVM	HOG	100%	89.47%	< 1 min
SVM	LBP	98.25%	27.22%	< 1 min
Dec. Tree	LBP	87.8%	32.51%	< 1 min
RF	LBP	24.25%	24.15%	2 min
RF	HOG	88.60%	86.84%	3 min
RF	Averaged LBP	24.25%	12.25%	2 min

The results suggest that AlexNet is the best performing classifier. However, all models experienced difficulties in classifying unlabeled faces. This is due to the fact that this class in composed by different individuals who have differences in facial features. As a result, it is problematic for classifiers. In fact, most models suffer from classifying faces of labeled individuals into this class.

The results also show that all classifiers give modest performance when trained on LBP feature. In addition, Decision Tree and SVM both suffer from overfitting problem. However, with the accuracy in training and validation set are roughly the same, Random Forest proved to be robust against overfitting.

5.2. Optical Character Recognition

Character detection

The character detection model was trained using the Matlab's built-in function trainCascadeObjectDetector. 256 positive examples were manually labeled using the Matlab's built-in application ImageLabeler. In addition, the database for negative images constitutes of various background images obtained from the internet, and photos of individuals provided in the original dataset with the number covered by a colored rectangle.

This method is not perfect, as a few false positives was returned. However, the next section discusses the method to address this problem

Character classification

After obtaining the detections output in the previous step, the Matlab's OCR classifier was used to recognize any character appeared in the marked region. This classifier iterated through all the regions and stored the character detected in a list. Afterwards, all elements of the list not containing the numbers from 0 to 9 were discarded, leaving the result which is the character displayed on the paper sheet as in the images.

For videos, frames were extracted and the above process was then applied. The final prediction is then produced by taking the mode of the outputs. Accuracy in videos is better as in frames because the most frequent prediction of all frames was taken.

6. Reflection and future work

Overall, positive results were obtained except for the case of LBP feature. With more time and a more diverse dataset, the results obtained could be improved. In specific, there are some classes that has only 4 to 5 training examples, which creates obstacles for classifiers from learning the underlying patterns.

The amount of training data available is also limited by the false negative of the Cascade Classifier when detecting individuals from the group photo. As a result, the data for some individuals may not be diverse i.e only limited to individual photo. This also raised a risk of overfitting.

Since unlabeled individuals were grouped in a same class, the classifiers ended up learning a more general facial trait instead of what is specific to an individual. This also justifies the misclassification of some faces into this group. Nevertheless, improvement could be obtained by marking them as "unknown 1", "unknown 2", etc. However, this approach would create an extreme amount of works and does not applicable in settings with a massive number of individuals.

To successfully classify the detected numbers, more sophisticated method can be implemented, which are CNN and MLP. In this setting, to guarantee good performance, the models should output a single digit number (0 to 9) then seek to combine predictions that are in vicinity to produce numbers with more than 1 digit.

In addition, to improve the detection accuracy, one may implement other state-of-the-art algorithm such as YOLO, R-CNN, Faster R-CNN, … Given enough time for data preparation.

Also, in order to reduce the ratio of false positive and false negatives, more data could benefit the face and optical character detectors, in specific: images of obscured faces, images of faces under different lighting conditions, and additional negative images. After carrying out this progress, the merge threshold can be raised for better accuracy.

7. Using the developed function

The functions can be obtained from

https://drive.google.com/open?id=1ZHAHdgfJ9k3QFKu2vuJgZD9qkFQ2pWvX

7.1. RecogniseFace

The RecogniseFace function accepts 3 arguments as follow:

I: Image name – format: char
Classifier: Classifier name – format: char
Feature: Feature use – format: char. For CNN: input a pair of single quote (‘ ‘)

7.2. DetectNum

This function accepts 1 argument, which is the name of the image / video in a character format.

8. References

G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, “EMNIST: Extending MNIST to handwritten letters,” *Proc. Int. Jt. Conf. Neural Networks*, vol. 2017–May, pp. 2921–2926, 2017.

[D. Cireşan, U. Meier, and J. Schmidhuber, “Multi-column Deep Neural Networks for Image Classification,” Feb. 2012.

Y. LeCun and Y. Bengio, “Convolutional networks for images, speech, and time series,” *Handb. brain theory neural networks*, vol. 3361, no. April 2016, pp. 255–258, 1995.

P. Ahmadvand, R. Ebrahimpour, and P. Ahmadvand, “How popular CNNs perform in real applications of face recognition,” *24th Telecommun. Forum, TELFOR 2016*, no. October, 2017.

Taggart, Allison J; DeSimone, Alec M; Shih, Janice S; Filloux, Madeleine E; Fairbrother, William G (2012-06-17)

Leo Breiman, 'Random Forests', Machine Learning 45, no. 1 (2001) 5-32.

Paul Smith, Siva Ganesh, Ping Liu. A Comparison of Random Forest Regression and Multiple Linear Regression for Prediction in Neuroscience. Journal of neuroscience methods (2013). 220:10.1016