Date of submission: 15 Apr 2017

**FYP17019**

2017-2018

Lo Wang Kin (3035186401)

Mak Tin Shing (3035187857)

Supervisor: Dr K.P. Chan

Facial Expression Recognition using Region-based Convolutional Neural Network

Final Report

# Abstract

Facial Expression has always been an important role in face-to-face communication. Messages sent and received using facial expression are easily understood by human without much effort, but it is not an easy task for computers. Understanding human emotions through facial expression is still a challenge for computer system [1].

This project aims to study and experiment different techniques of Facial Expression Recognition techniques. Using new techniques, the precision of identifying the facial expression of human by the computer may be improved.

# Acknowledgement

I would like to express my sincere gratitude to Dr. K.P. Chan in Department of Computer Science in HKU, supervisor of this project, for his guidance and help on this project. Special thanks to my teammate Mak Tin Shing, who helped me in understanding the underlying principle of neural networks and giving suggestions in the implementation details.

Contents

[Abstract 2](#_Toc511590207)

[Acknowledgement 3](#_Toc511590208)

[List of Figures 6](#_Toc511590209)

[Abbreviations 7](#_Toc511590210)

[Distribution of work 8](#_Toc511590211)

[1. Introduction 9](#_Toc511590212)

[2. Objective 10](#_Toc511590213)

[3. Methodology 11](#_Toc511590214)

[3.1 Development tools & technologies 11](#_Toc511590215)

[3.2 Project Details 12](#_Toc511590216)

[3.2.1 Image dataset 12](#_Toc511590217)

[3.2.2 Baseline CNN system 13](#_Toc511590218)

[3.2.3 Faster R-CNN + CNN system 14](#_Toc511590219)

[4. Implementation Details 15](#_Toc511590220)

[4.1 Preparing data for training 15](#_Toc511590221)

[4.1.1 Identifying the emotions of the images in dataset 15](#_Toc511590222)

[4.1.2 Dividing the dataset into training, validation and test dataset 17](#_Toc511590223)

[4.1.3 Preparing the images for baseline CNN system 17](#_Toc511590224)

[4.2 Baseline CNN system 18](#_Toc511590225)

[4.2.1 Training the CNN system 19](#_Toc511590226)

[4.2.2 Testing the CNN system 19](#_Toc511590227)

[4.3 The Faster R-CNN 20](#_Toc511590228)

[4.4 Merging the facial components into one image 21](#_Toc511590229)

[4.4.1 Merging strategy 1: Blackening irrelevant areas 21](#_Toc511590230)

[4.4.2 Merging strategy 2: Placing the components linearly 22](#_Toc511590231)

[4.4.3 Merging strategy 3: Placing the components in fix position 23](#_Toc511590232)

[4.4.4 Merging strategy 4: Pack the components tightly 24](#_Toc511590233)

[4.4.5 Merging strategy 5: Use different colour channel 24](#_Toc511590234)

[4.5 Training the CNN for the merged images 25](#_Toc511590235)

[5. Results and findings 26](#_Toc511590236)

[5.1 Best fitting models 26](#_Toc511590237)

[5.2 Precision of test results 28](#_Toc511590238)

[6. Conclusion 30](#_Toc511590239)

[7. References 31](#_Toc511590240)

# List of Figures

[Figure 1 The Baseline CNN model 13](#_Toc511590241)

[Figure 2 Faster R-CNN + CNN system 14](#_Toc511590242)

[Figure 3 Examples of images which cannot be labelled with any emotions 15](#_Toc511590243)

[Figure 4 The structure of the baseline CNN model. 18](#_Toc511590244)

[Figure 5 An example image of the result from trained R-CNN model 20](#_Toc511590245)

[Figure 6 Merging strategy 1: Blackening irrelevant areas 21](#_Toc511590246)

[Figure 7 Merging strategy 2: Placing the components linearly 22](#_Toc511590247)

[Figure 8 Merging strategy 3: Placing the components in fix position 23](#_Toc511590248)

[Figure 9 Merging strategy 4: Pack the components tightly 24](#_Toc511590249)

[Figure 10 mAP against iteration of the two variations of baseline CNN model 27](#_Toc511590250)

[Table 1 The number of images in each emotion category 16](#_Toc511590251)

[Table 2 Iterations and mAPs of the best fitting models for each network. 26](#_Toc511590252)

[Table 3 The mean average precision and recall of each model. 28](#_Toc511590253)

[Table 4 Average precisions among pretrained models and non-pretrained models respectively. 29](#_Toc511590254)

# Abbreviations

AUs – Action units

BLOBs – binary large objects

CNN – Convolutional Neural Network

FACS – Facial Action Coding System

FER – Facial Expression Recognition

mAP – mean average precision

R-CNN – Region-based Convolutional Neural Network

# Distribution of work

|  |  |
| --- | --- |
| Preparing dataset | Lo Wang Kin |
| Implementing CNN | Lo Wang Kin |
| Automated scripts for training CNN | Lo Wang Kin |
| Generating test results of CNN | Lo Wang Kin |
| Preparing images for R-CNN | Mak Tin Shing |
| Training R-CNN | Mak Tin Shing |
| Testing R-CNN | Mak Tin Shing |
| Implementing merging strategies (except Pack) | Mak Tin Shing |
| Implementing merging strategies (Pack) | Lo Wang Kin |

# 1. Introduction

To facilitate a more natural communication between humans and computer, considerable progress has been made in the field of FER, especially in feature extraction algorithms and classification techniques in the past few years. Much development involves using a technique called Facial Action Coding System (FACS) [2] [3]. FACS is a method that classifies different facial components, such as eyes and lips, into some Action Units (AUs). Combining different AUs, the computer can then recognize the expression. Some other methods involving classifiers like Naive Bayes classifiers and hidden Markov models [4].

The recent success in Convolutional Neural Network (CNN) has drawn much attention in the field of Facial Expression Recognition (FER). Different contributors try to apply modifications on CNN such as applying transformation on images at train time, using a decision tree for combining results of different neural networks, and use different pooling layers [5] [6] [7]. However, most CNN models usually cannot achieve both high recognition rate and high accuracy. Much experiments and research are needed to examine the success and drawbacks of different techniques.

To extend the use of CNN to object detection, the Region-based Convolutional Neural Networks (R-CNN) was developed in 2014 [8]. The system proposes regions using selective search before feeding them to a modified version of AlexNet for classification. The algorithm was further optimized to Fast-R-CNN, Faster-R-CNN and Mask-R-CNN. R-CNN successfully improves the precision on PASCAL VOC 2012 by 30%, compared to CNN [8]. The success of R-CNN could be a key to further improve the performance of FER.

This report will first discuss the objective of the project, and then carry on with the design and implementation details. After that, the result and findings will be discussed.

# 2. Objective

This goal of this project is to experiment the effectiveness of R-CNN on facial expression recognition. This project will start with implementing a functional FER system that uses traditional CNN as a baseline for comparison. Then a system that can extract facial component using R-CNN, and then classify facial expression using CNN, will be built. Different merging strategies of the facial components will be experimented. The performance of different techniques will be compared and analysed.

# 3. Methodology

To achieve FER on computer, two systems based on CNN will be implemented and trained using existing tools and technologies.

## 3.1 Development tools & technologies

3.1.1. Ubuntu

Ubuntu was used as the operating system for most of the training and testing operations, due to its well-developed algorithm for fast computation of image processing.

3.1.2. Python 3

Python 3 was used as the main programming language due to its versatility and its extensibility. Libraries required by this project, such as the CNN and Faster R-CNN, could also be imported to Python 3 codes.

3.1.3. Caffe

Caffe is a popular framework of deep neural network. It has a variety of predefined types of layers, easing the implementation of the models of this project. The CNN and Faster R-CNN libraries are also designed in accordance to Caffe framework, hence less effort is required to build the models.

## 3.2 Project Details

This section will provide an overview on how this project will be done.

### 3.2.1 Image dataset

The neural network systems used by this project will be trained based on a dataset called the Extended Cohn-Kanade Dataset (CK+) [9]. This dataset was made available on July 2010. It contains a total of 593 sequences of images, taken from 123 different people. Each sequence begins with a relatively neutral facial expression, and then proceed to a peek emotion. A total of 11424 images are provided by this dataset. Eight kinds of facial expression categories are provided, which are neutral, angry, contempt, disgust, fear, happy, sadness, and surprise. These images have relatively high resolution (640×480), and therefore is suitable for extracting facial components required by this project.

The CK+ dataset was further divided into the training dataset, validation dataset and the test dataset for this project.

### 3.2.2 Baseline CNN system

The first neural network for FER will be a traditional CNN model (see Figure 1). This system will serve as a baseline for comparison, as it is relatively simpler than the other models.



CNN

Happy: 0.85

Surprise: 0.07

Contempt: 0.05

…

…

Figure The Baseline CNN model

There are two variations of this system: one with transfer learning and one without. For the one with transfer learning, CaffeNet [10], an open source neural network model on the internet, will be used. It is derived from the well-known AlexNet. A pretrained model was provided on the internet, which was trained with the ImageNet database for 310,000 iterations. The same network was used in both variations.

### 3.2.3 Faster R-CNN + CNN system

The second neural network will be a Faster R-CNN, in addition to a tradition CNN as a classifier (see Figure 2). R-CNN is used to extract facial components from an image, such as the eyes, the eyebrows and the mouth. These extracted components will then be fed into the baseline CNN. The CNN model will then classify the input into different emotions.



Left eye:

Left eyebrow:

Mouth

…

…

Happy: 0.85

Surprise: 0.07

Contempt: 0.05

…

…



CNN model

R-CNN model

Figure Faster R-CNN + CNN system

The R-CNN model was trained by providing the detail positions of the facial components of the image. These training data were generated by using a Python library named Dlib [11], which has a classifier for detecting facial components. After extracting the facial components, different strategies of merging them back into one image are used, so that these combined images can be used to train the CNN model. The CNN model uses the same network as the baseline CNN system. Depends on the merging strategy, pretrained model may or may not be used.

# 4. Implementation Details

This section will discuss the implementation details of how the networks are implemented, trained and tested.

## 4.1 Preparing data for training

To start training the neural network required for this project, the dataset will first need to be pre-processed before it can be used.

### 4.1.1 Identifying the emotions of each sequence in dataset

The emotion of each image must be known beforehand in order to train the networks. For the CK+ dataset, 327 sequences out of 593 (55%) was labelled with an emotion category. Since only images with known emotions are usable in training the neural networks, over 4000 images are left unused.

However, the Action Units of the sequences identified was provided for each sequence. By comparing the images with similar Action Units, either by program or manually, 198 more sequences are identified and categorized into an emotion. This increases the amount of training data available for training. Some images can hardly be labelled with any emotions (Figure 3), and therefore were discarded.



Figure Examples of images which cannot be labelled with any emotions

As mentioned before, each sequence started with a neutral emotion and proceed to a peek emotion. For each sequence, the first 20% of the images will be labelled as a neutral emotion, and the last 40% of the images will be labelled as the peek emotion (angry, sad etc.). The middle 40% are usually too vague to be labelled with either neutral or peek emotion, and therefore were discarded.

Table 1 shows the number of images in each emotion category for each dataset.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Training dataset | Validation dataset | Test dataset |
| Neutral | 1017 | 356 | 346 |
| Anger | 437 | 109 | 107 |
| Contempt | 55 | 15 | 32 |
| Disgust | 258 | 127 | 75 |
| Fear | 341 | 95 | 140 |
| Happy | 557 | 188 | 196 |
| Sadness | 391 | 148 | 142 |
| Surprise | 387 | 148 | 132 |
| Total | 3443 | 1186 | 1170 |

Table The number of images in each emotion category

### 4.1.2 Dividing the dataset into training, validation and test dataset

The CK+ dataset was divided into three group: training dataset, validation and test dataset. Two-thirds of the dataset is in the training dataset, one-sixth of it is in the validation dataset, and the remaining one-sixth is in the test dataset.

At first, the division was done by randomly shuffling all images to the dataset. However, after testing the baseline CNN system, the precision was unexpectedly high. After looking further into the problem, this part was reworked. Instead of shuffling all images, images of the same person were grouped up. Each group was then randomly assign each group to a dataset, so that no two images of the same person will be in different datasets.

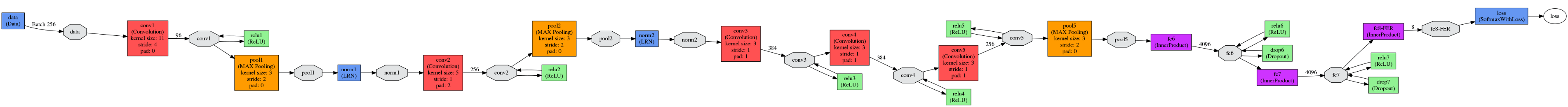
The reason for the unexpected high precision is that the images of the same sequence are very similar. If an image was used in training, and another image of the same sequence is then used in testing, the network may easily identify the image in the testing phase, since the network has already seen a very similar image during training phase. To a lesser extent, images of the same person were also quite similar, and therefore it is best to avoid such situation to prevent a misleading validation results or test results.

### 4.1.3 Preparing the images for baseline CNN system

To put the images to use, they must first be properly pre-processed. Histogram equalization will first be applied to each image to increase the contrast. Then each image will be resized to the required size (227×227) of the network. After that, the images were stored inside a Lightning Memory-Mapped Database (LMDB). The mean value of the images is then computed and stored.

## 4.2 Baseline CNN system

The baseline CNN system uses a similar network as the CaffeNet. Figure 4 shows the structure of the network. Square blocks are the layers, and octagon blocks indicate the flow of the image BLOBs (binary large objects). The numbers above the arrows indicate the number of output images (batch size).



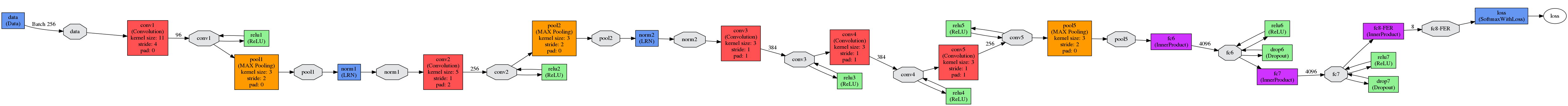
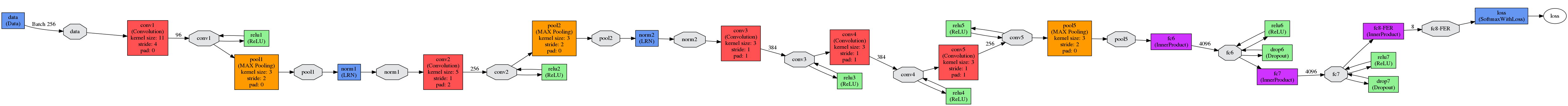


Figure The structure of the baseline CNN model.

### 4.2.1 Training the CNN system

To train the CNN system, a base learning rate of 0.001 and a weight decay of 0.0005 was used. Images from the training dataset are then provided to the network to start the training. For every 50 iterations, a snapshot of the status of the network was recorded and was tested against the validation dataset using the build-in testing network of Caffe, in order to monitor the progress during the training. Based on some predictions on the learning curve of the network, an iteration of 3000 for the network without pretrain model and an iteration of 1000 for the one with pretrain model was used.

After that, the precision of the validation dataset was analysed. The iteration with highest precision is selected as the best fitting model.

### 4.2.2 Testing the CNN system

Normally, tests only need to be conducted on the selected best fitting model. However, due to snapshots of the status of the network take up a lot of disk space, they must be immediately deleted after tests were conduct. Hence, tests were done against all images in the test dataset for every 50 iterations, before the snapshot was deleted. Images were tested one by one in Python codes instead of letting the Caffe framework to do it on its own, since the build-in Caffe framework can only provide little details about the tests. After passing the images through the trained network, the output of the final layer of each image was collected. After the best fitting model was selected, the results of other models were discarded. Mean average precision and recall of each emotion category were then calculated.

## 4.3 The Faster R-CNN

The faster R-CNN was based on a network designed by Zeiler & Fergus [12], which was called the ZF net. In this project this network will be used to identify the facial component on an image.

To train the R-CNN, information on the position of facial components are required. Since the dataset lack such information, an external classifier named Dlib [11] was used to generate such data. After the facial components were located, the R-CNN were trained for 1300 iterations.

After training, all images from the three datasets passed through the network to generate the positions of the facial components. The facial components identified are the left and right eyebrows, left and right eyes, nose, mouth and jaw (Figure 5).



Figure An example image of the result from trained R-CNN model

## 4.4 Merging the facial components into one image

After the facial components were extracted, they need to be combined into one image, since CNN may not able to combine multiple inputs to one output. To combine the facial components, different methods were used in this project.

### 4.4.1 Merging strategy 1: Blackening irrelevant areas

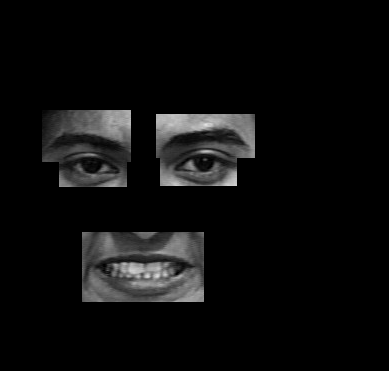
The easiest way to combine the images are to simply blacken all other areas except the facial components (Figure 6). However, this may create an image that has a large black area. To solve this problem, a rough approximation of the size and position of the useful areas are calculated, and the edges were cut off from the image.  

Figure Merging strategy 1: Blackening irrelevant areas

### 4.4.2 Merging strategy 2: Placing the components linearly

Another way is to place each facial component next to each other, so that they are put into one image next to each other.

Figure Merging strategy 2: Placing the components linearly

### 4.4.3 Merging strategy 3: Placing the components in fix position

A variation of the above method is to place the components in some fixed position, instead of placing them next to each other. This allows the same components to be always at the same place for every image.

Figure Merging strategy 3: Placing the components in fix position

### 4.4.4 Merging strategy 4: Pack the components tightly

Since the above methods may causes the output image to be very large, an algorithm is used to pack the components into one image, while minimizing the empty black area on the image. To achieve this, the components was put into the image one by one from the largest in size to the smallest, such that each component may be placed on the edge of another component, while minimizing the total image size.

Figure Merging strategy 4: Pack the components tightly

### 4.4.5 Merging strategy 5: Use different colour channel

Colour channel usually represents the red, green and blue components on an image. In CNN, the colour channels are treated independent of each other. Hence, each component can be put in a separate channel as a new image.

## 4.5 Training the CNN for the merged images

The CNN uses the same network as the baseline CNN. For each merging strategy, the network again trained with two variations: with and without transfer learning. While using colour channels, since more than 3 channels was used, there is no pretrained model for such situation. Therefore, it was only trained without transfer learning. Again, the networks with and without pretrain model were trained for an iteration of 1000 and 3000 respectively. The best fitting model is selected for each model based on the precision of validation dataset. Then the tests were done on each of these models and the precision and recall were calculated.

# 5. Results and findings

This section will cover the results after the tests have been conducted.

## 5.1 Best fitting models

After training a network, the best fitting model was selected based on the precision of the validation dataset. The model with the highest mean average precision will be selected as the best fitting one. Table 2 shows the corresponding iterations and mAPs of the best fitting models for each network.

|  |  |  |
| --- | --- | --- |
| Model | Best fitting model | mAP of validation set |
| Baseline CNN | 2550 | 0.59082 |
| M1: Blacken | 2550 | 0.673828 |
| M2: Linear | 2100 | 0.698242 |
| M3: Fix Linear | 2550 | 0.706055 |
| M4: Pack | 2100 | 0.706055 |
| M5: Channel | 1800 | 0.732422 |
| Baseline CNN w/ pretrain | 800 | 0.724609 |
| M1: Blacken w/ pretrain | 700 | 0.763672 |
| M2: Linear w/ pretrain | 900 | 0.810547 |
| M3: Fix Linear w/ pretrain | 900 | 0.751953 |
| M4: Pack w/ pretrain | 700 | 0.826172 |

Table Iterations and mAPs of the best fitting models for each network.

As expected, networks that use pretrained model reach the peak much earlier than the one without, since they have already learnt to identify general objects instead of learning from scratch. Figure 10 uses the baseline CNN as an example to demonstrate the differences in the changes on the mAP per iteration.

Figure mAP against iteration of the two variations of baseline CNN model

## 5.2 Precision of test results

For each method used, the mean average precision and recall of the best fitting model were calculated. Table 3 shows the result of each method.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Iteration | Mean precision | Mean recall |
| Baseline CNN | 2550 | 0.5139 | 0.5069 |
| M1: Blacken | 2550 | 0.629 | 0.6251 |
| M2: Linear | 2100 | 0.6335 | 0.5635 |
| M3: Fix Linear | 2550 | 0.6907 | 0.653 |
| M4: Pack | 2100 | 0.593 | 0.5507 |
| M5: Channel | 1800 | 0.6243 | 0.6212 |
| Baseline CNN w/ pretrain | 800 | 0.8116 | 0.7372 |
| M1: Blacken w/ pretrain | 700 | 0.8322 | 0.8026 |
| M2: Linear w/ pretrain | 900 | 0.8027 | 0.7469 |
| M3: Fix Linear w/ pretrain | 900 | 0.7131 | 0.7236 |
| M4: Pack w/ pretrain | 700 | 0.8003 | 0.7625 |

Table The mean average precision and recall of each model.

Some interesting findings are:

1. The mean precision and recall of the baseline CNN system are the lowest of all as expected. The models without transfer learning have lower precision and recall than the one with transfer learning in general.
2. Different merging strategies do not make much difference in the test results. This means that the positioning of the facial components does not affect the overall performance of the CNN model.
3. All the pretrained models have very similar precision and recall. This shows that the use of R-CNN to extract facial components did not improve nor reduce the performance of FER. However, among the non-pretrained models, the use of R-CNN did improve the performance. This means that even though R-CNN did not improve the performance, it helps the network to learn, so that the network can reach a point closer to the global maximum.

In addition, the precision and recall of images belongs to each emotion category were also calculated. Table 4 shows the average precisions of each emotion category among pretrained models and non-pretrained models respectively.

|  |  |  |
| --- | --- | --- |
| Emotion | Precision without pretrain | Precision with pretrain |
| Neutral | 0.6795 | 0.7515 |
| Anger | 0.4987 | 0.6090 |
| Contempt | 0.2070 | 0.6832 |
| Disgust | 0.5005 | 0.7752 |
| Fear | 0.6295 | 0.9337 |
| Happy | 0.7386 | 0.8087 |
| Sadness | 0.7747 | 0.8663 |
| Surprise | 0.8549 | 0.9081 |

Table Average precisions among pretrained models and non-pretrained models respectively.

Some findings are:

1. The precision of the emotion Contempt was very low in non-pretrained models. In fact, the baseline CNN model cannot identify any images with Contempt emotion at all. This is probably due to the small dataset size of this emotion category. Also, images with this emotion is usually quite mild, and so it was frequently identified as Neutral.
2. The precision of Anger and Disgust were lower than most other in non-pretrained models. It was found that they were frequently mixed up by the network, i.e. many images of Anger were identified as Disgust, and vice versa. This is most likely due to the fact that both emotions were very similar, usually with tightened eyebrows and lips.
3. The precision of Surprise emotion was unexpectedly high. It is possibly result from the fact that images of Surprise emotion usually have raised eyebrows, opened eyelids and wide-opened lips, and therefore is quite unique among the emotions.

# 6. Conclusion

This report presented the approaches of implementing and experimenting the techniques of R-CNN to achieve FER by the computer.

The improvement in mean average precision of non-pretrained networks shows that by extracting facial components before classifying the image, the network will learn better than using the entire image. The precision was however not improved nor reduced in pretrained network.

Further improvement could be done by using a larger dataset. Since some emotions have few images associated with them, the network may not correctly identify these emotions. The dataset also lacks some exceptional cases, such as side view of a face or wearing glasses. Hence the network may not identify these images. The use of other facial components like nose and jaw can be experimented as well.

# 7. References

|  |  |
| --- | --- |
| [1] | N. Sebe, M. Lew, Y. Sun, I. Cohen, T. Gevers and T. Huang, “Authentic facial expression analysis,” *Image and Vision Computing,* vol. 25, no. 12, pp. 1856-1863, 2007. |
| [2] | Y. I. Tian, T. Kanade and J. F. Cohn, “Recognizing action units for facial expression analysis,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 23, no. 2, pp. 97-115, 2001. |
| [3] | M. S. Bartlett, G. Littlewort, M. Frank, C. Lainscsek, I. R. Fasel and J. Movellan, “Automatic Recognition of Facial Actions in Spontaneous Expressions,” *Journal of Multimedia,* vol. 1, no. 6, 2006. |
| [4] | I. Cohen, N. Sebe, Y. Sun, M. Lew and T. Huang, “Evaluation of Expression Recognition Techniques,” in *Image and Video Retrieval*, Berlin, Heidelberg, Springer, 2003, pp. 184-195. |
| [5] | B. K. Kim, J. Roh, S. Y. Dong and S. Y. Lee, “Hierarchical committee of deep convolutional neural networks for robust facial expression recognition,” *Journal on Multimodal User Interfaces,* vol. 10, no. 2, p. 173–189, 2016. |
| [6] | Z. Yu and C. Zhang, “Image based Static Facial Expression Recognition with Multiple Deep Network Learning,” in *2015 ACM on International Conference on Multimodal Interaction*, Seattle, Washington, USA, 2015. |
| [7] | A. Mollahosseini, D. Chan and M. Mahoor, “Going deeper in facial expression recognition using deep neural networks,” in *2016 IEEE Winter Conference on Applications of Computer Vision*, Lake Placid, NY, 2016. |
| [8] | R. Girshick, J. Donahue, T. Darrell and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, USA, 2014. |
| [9] | P. Lucey, J. Cohn, T. Kanade, J. Saragih, Z. Ambadar and I. Matthews, “The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression,” in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, USA, 2010. |
| [10] | J. Donahue, “BAIR/BVLC CaffeNet Model,” 2014. [Online]. Available: https://github.com/BVLC/caffe/tree/master/models/bvlc\_reference\_caffenet. |
| [11] | A. Rosebrock, “Real-time facial landmark detection with OpenCV, Python, and dlib,” 2015. [Online]. Available: https://www.pyimagesearch.com/2017/04/17/real-time-facial-landmark-detection-opencv-python-dlib/. |
| [12] | M. D. Zeiler and R. Fergus, “Visualizing and Understanding Convolutional Networks,” in *13th European Conference on Computer Vision, ECCV 2014*, Zurich, Switzerland, 2014. |