EB5206 Computational Intelligence:
Classification Assignment

Facial Expression Recognition Classification

Lecturer: Dr. ZHU Fang Ming
Dr. TIAN Jing

Student ID	Name	E-mail
A0178431A	Huang Qingyi	e0267742@u.nus.edu
A0178415Y	Jiang Zhiyuan	e0267726@u.nus.edu
A0178222H	Zhu Huiying	e0267533@u.nus.edu

Table of Contents

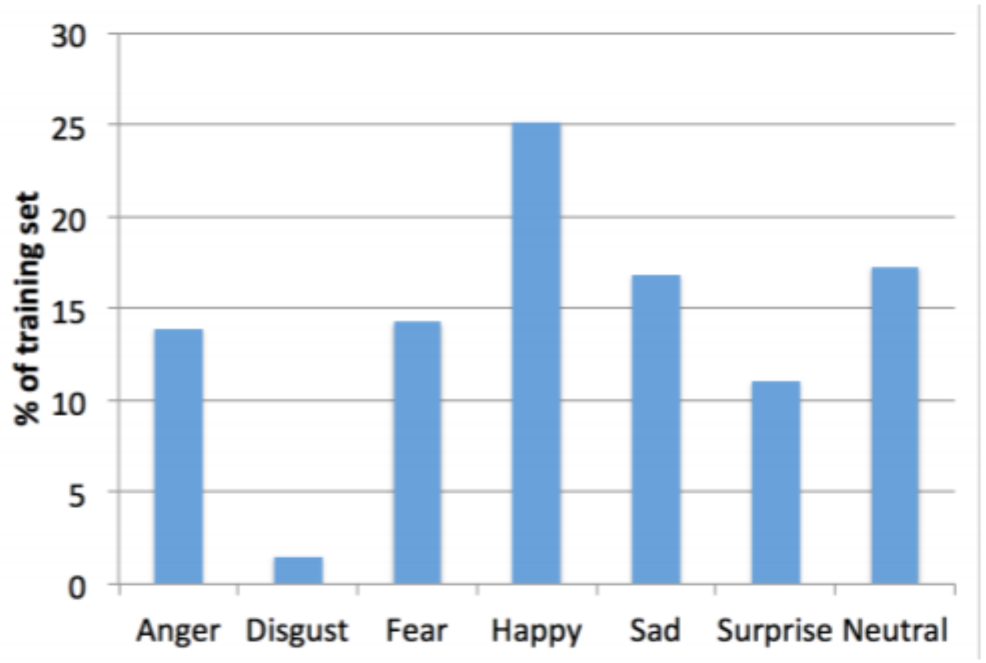
Facial Expression Recognition Classification

Abstract

Facial expressions convey non-verbal information between humans in face-to-face interactions. Automatic facial expression recognition, which plays a vital role in human-machine interfaces, has attracted increasing attention from researchers since the early nineties. Automated facial expression recognition has numerous practical applications such as psychological analysis, medical diagnosis, forensics (lie-detection), studying effectiveness of advertisement and so on. Classical machine learning approaches often require a complex feature extraction process and produce poor results. In our experiment, we apply recent advances in deep learning to propose effective deep Convolutional Neural Networks (CNNs) that can accurately interpret semantic information available in faces in an automated manner without hand-designing of features descriptors. We compare different types of CNN models and training tricks in order to learn CNNs with a strong classification power. To achieve better prediction results, we ensemble different models using confusion matrix. The experimental results show that our proposed networks get close to state-of-the-art methods on the well-known FERC-2013 dataset provided on the Kaggle facial expression recognition competition. In comparison to the winning model of this competition, the number of parameters in our proposed networks intensively decreases, that accelerates the overall performance speed and makes the proposed networks well suitable for real-time systems. To run this large dataset, we use GPU on Google Colab, Python 3 and National Supercomputing Centre Singapore (NSCC) using configuration: 24 CPUs, 80G memory and GPU (nVidia Tesla K40).

1. Data Understanding

The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centered and occupies about the same amount of space in each image. The task is to categorize each face based on the emotion shown in the facial expression in to one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral) using three tools.
Fer2013.csv contains three columns, "emotion", "pixels" and "usage". The "emotion" column contains a numeric code ranging from 0 to 6, inclusive, for the emotion that is present in the image. The "pixels" column contains a string surrounded in quotes for each image. The contents of this string a space-separated pixel values in row major order. The "usage" column contains two kinds of strings indicating two usages of the data, "training" and "test". The training set consists of 28,709 examples. The test set used for the leaderboard consists of 3,589 examples. Additionally, the distribution of each kind of expression in the training dataset is shown in picture 1.1.



Picture 1.1 Distribution of Expressions in Training Dataset

2. Data Preparation

Data preparation has significant effect on results and it determines the performance of models. We did four steps as followed:

2.1 Check for missing data and outliers

The data are intact and each pixel is in range from 0 to 255. As such, there is no missing data and outliers.

2.2 Fit format for model

- For Input
  Input shape of our model is "channel*height*width". The raw training data has one dimension(1*2304), so we change each sample to three dimensions([1*48*48]).
- For target
  Due to choosing categorical_crossentropy as loss function, we should convert targets to categorical format (e.g. targets have 7 classes, the target for each sample should be a 7-dimensional vector that is all-zeros except for a 1 at the index corresponding to the class of the sample)

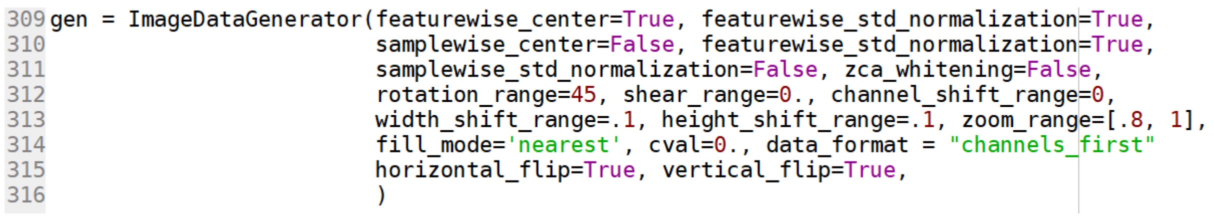
2.3 Feature-wise standardization

Subtracting the mean centers the input to 0, and dividing by the standard deviation makes any scaled feature value the number of standard deviations away from the mean.
CNNs learn by continually adding gradient error vectors (multiplied by a learning rate) computed from backpropagation to various weight matrices throughout the network as training examples are passed through, as shown in formula 2.1. If we didn't scale our input training vectors, the ranges of our distributions of feature values would likely be different for each feature, and thus the learning rate would cause corrections in each dimension that would differ (proportionally speaking) from one another. We might be over compensating a correction in one weight dimension while undercompensating in another. To avoid that, we standardise the each input feature. The standardized images are shown in picture 2.2.

$$\Delta w_{ij}^{k} = -\alpha \frac{\partial \mathrm{E}(X,\theta)}{\partial w_{ij}^{k}} \tag{2.1}$$

2.4 Image Augmentation

Due to the small amount of training data, we apply data augmentation techniques to increase the amount of training samples in order to avoid overfitting and improve recognition accuracy. Keras provides the ImageDataGenerator class that defines the configuration for image data preparation and augmentation, Shown as picture 2.1.



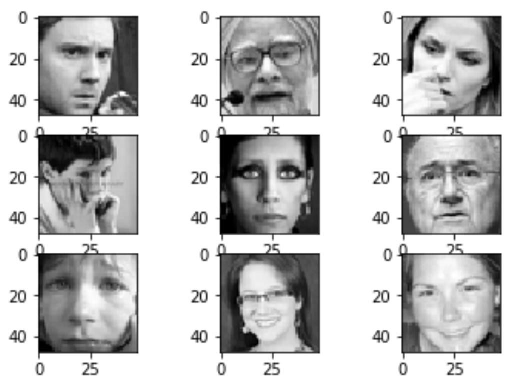
```
gen = ImageDataGenerator(featurewise_center=True, featurewise_std_normalization=True,
                         samplewise_center=False, featurewise_std_normalization=True,
                         samplewise_std_normalization=False, zca_whitening=False,
                         rotation_range=45, shear_range=0., channel_shift_range=0,
                         width_shift_range=.1, height_shift_range=.1, zoom_range=[.8, 1],
                         fill_mode='nearest', cval=0., data_format = "channels_first"
                         horizontal_flip=True, vertical_flip=True,
                         )
```

Picture 2.1 Image Augmentation by ImageDataGenerator in Python

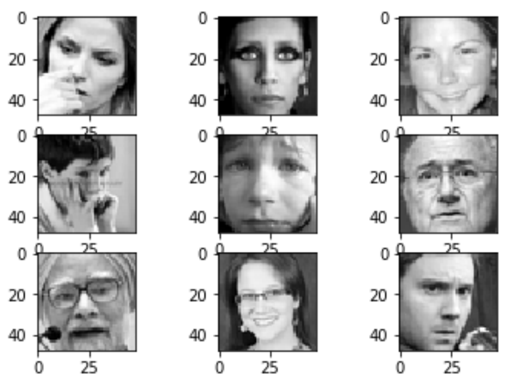
For each image, we perform the following successive transforms::

- Rotate the image with a random angle from -45 to 45 (in degrees).
- Shift the width or height of image with a random 20 percent of total width or height.
- Randomly flip images horizontally or vertically.

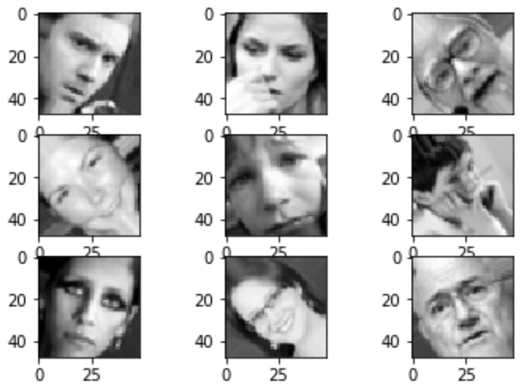
Then we compare the images after processing with originals shown in picture 2.2 with random order.

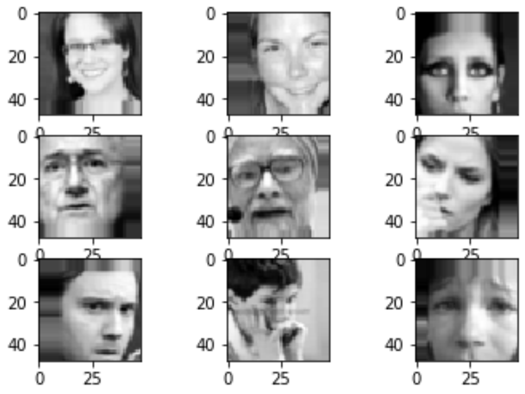


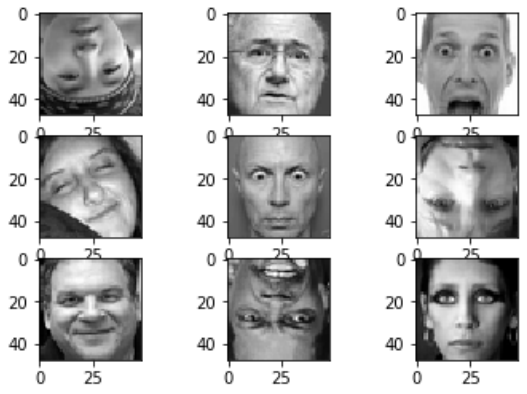
(a) Original image.



(b) Standardized images


(c) Rotation range = 45°


(d) Shift = 0.2


(e) Randomly flip

Picture 2.2 Images comparison before and after processing

From picture 2.2, we find that there is no obviously observed change between original and standardised images. And we exploit augmentation techniques to create more data for training. Moreover, we will compare the accuracy rate between augmentation and not augmentation in sector 3.3.2.

2.5 Comments

In this sector, we do suitable data pre-processing work, including missing data and outliers, fit format for model, normalization and image augmentation, which can promote models performance dramatically. After this, we can build models in sector 3.

3. Build three similar VGG models using Keras

With the fast growth of deep learning, the state-of-the-art in many computer vision tasks has been considerably improved. In image classification, there are some well-known deep CNNs can be mentioned as follows. The first network we want to mention is AlexNet, the winner of ImageNet ILSVRC challenge in 2012. This network has a very similar architecture to LeNet, but is deeper and bigger, and its convolutional layers stack on top of each other. Previously it is common to have only a single convolutional layer followed by a pool layer. The next CNNs are called VGGNet, the runner-up in ILSVRC 2014. One important property of VGGNet is that there are many convolutional layers with small filter size 3×3 that stack on top of each other instead of using a single convolutional layer with larger filter size as in previous CNN generations. The winner of ILSVRC 2015 is ResNet[2] which can be characterized by skip connections and heavy use of batch normalization. Recent improved variants of ResNet called Wide ResNet or ResNeXt also demonstrate their impressive power in image classification tasks.

3.1 Network Structure

VGGNet[3] has simple designing idea of architecture and has better results in the problem of facial expression recognition, and we refer to a thesis "Facial Expression Recognition Using Deep Convolutional Neural Networks"[1], following their models, we build three VGG models shown as table 3.1. Then we compare their performance in Section 3.3.

ConvNet Configuration		
VGG10	VGG12	VGG14
10 layers	12 layers	14 layers
Input (1 * 48 * 48)		
Conv3 - 32	Conv3-32	Conv3-32
	Conv3-32	Conv3-32
maxpool		
Conv3-64	Conv3-64	Conv3-64
	Conv3-64	Conv3-64
maxpool		
Conv3-128	Conv3-128	Conv3-128

Conv3-128	Conv3-128	Conv3-128 Conv3-128
maxpool		
Conv3-256 Conv3-256 Conv3-256	Conv3-256 Conv3-256 Conv3-256	Conv3-256 Conv3-256 Conv3-256 Conv3-256
FC-256		
FC-256		
FC-7		

Table 3.1 Three VGG model architectures

All of these architectures follow the general designing principles of VGG. They differ from each other only in depth: from 10 layers in VGG10 (7 convolutional and 3 fully connected layers) up to 14 layers in VGG14 (11 convolutional and 3 fully connected layers).

For example, VGG12 is described in more details as follows. Overall, VGG12 consists of four blocks. The first block has two 3 × 3 convolutional layers, the number of filters is 32, the stride is 1. The second block has two 3 × 3 convolutional layers, the number of filters is 64, the stride is 1. The third block has two 3×3 convolutional layers, the number of filters is 64, the stride is 1. The fourth block has three 3×3 convolutional layers, the number of filters is 64, the stride is 1. After each block except the last one, there is a max pooling layer with filter size 2 × 2 and the stride is 2. After the four blocks, there are two fully connected layers with 256 neurons per each layer. Finally, the output layer is a fully connected layer with 7 neurons associated with 7 emotion classes. The softmax activation function is applied to output layer and cross-entropy loss function is used in training process.

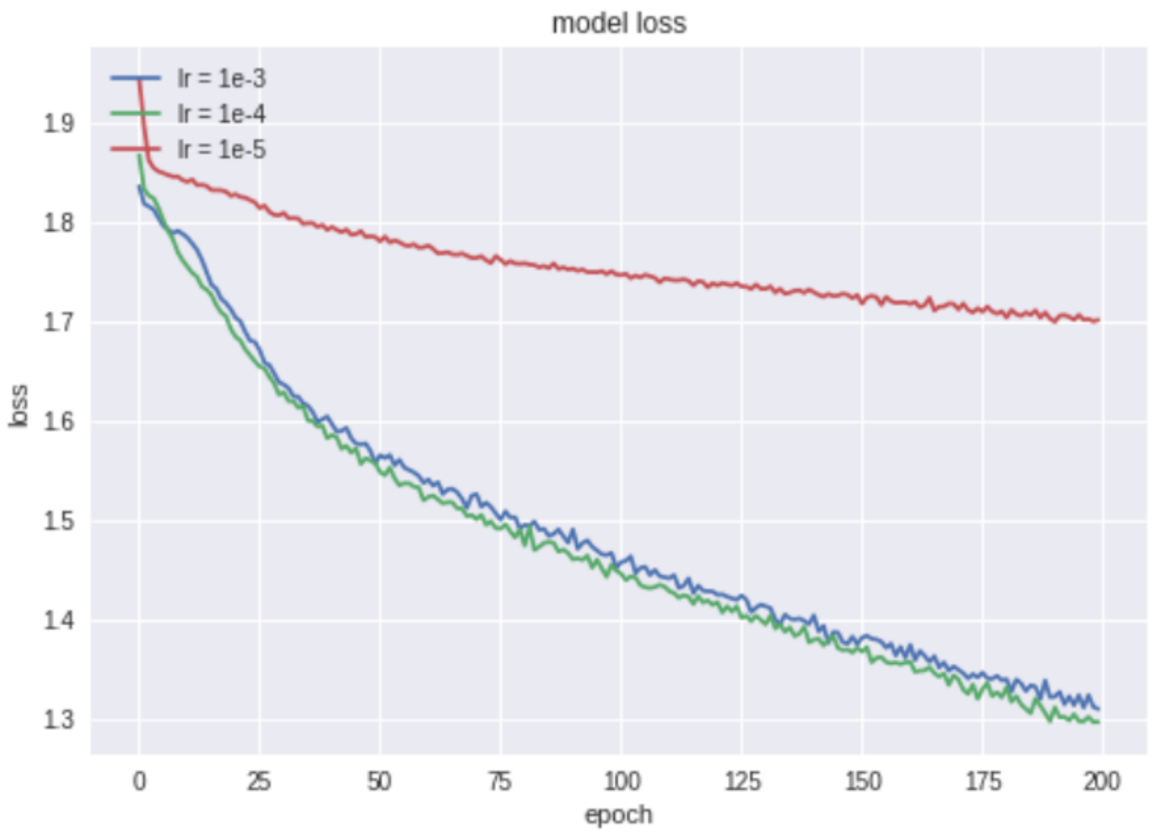
3.2 Parameter Choosing

In section 3.2 we use VGG10 model to compare different parameters performance, and then explore the optimum parameters in VGG12 and VGG 14.

3.2.1 Optimizer

Compared three popular optimizers Adam, Adadelta and RMSprop, we find Adam works well in practice and compares favourably to other adaptive learning-method algorithms as it converges very fast and the learning speed of the Model is quiet Fast and efficient. Moreover, it rectifies every problem that is faced in other optimization techniques such as vanishing Learning rate, slow convergence or High variance in the parameter updates which leads to fluctuating Loss function. As such, we choose Adam as optimizer.

Then we test different learning rate effect, shown in picture 3.1. After 200 epochs, learning rate equal to 0.0001 has the lowest loss, so we choose 0.0001 as learning rate. By setting that learning rate, the model can cost less study time and acquire the best convergence result.



Picture 3.1 Model loss based on various learning rate

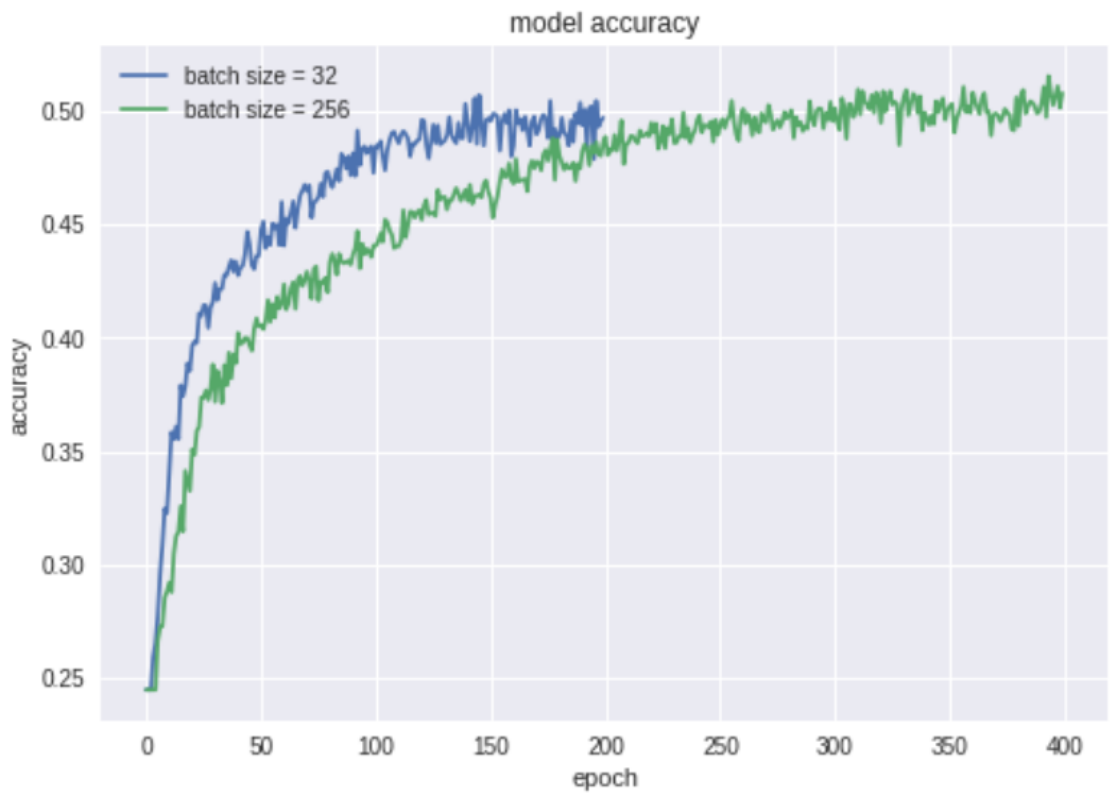
3.2.2 Batch Size

To train all the data in each epoch, the product of batch size and steps_per_epoch must be greater than length of training dataset, so we set **steps_per_epoch=len(x_train)//batch_size** and get different time consuming per epoch among different training batch size, shown as table 3.2.

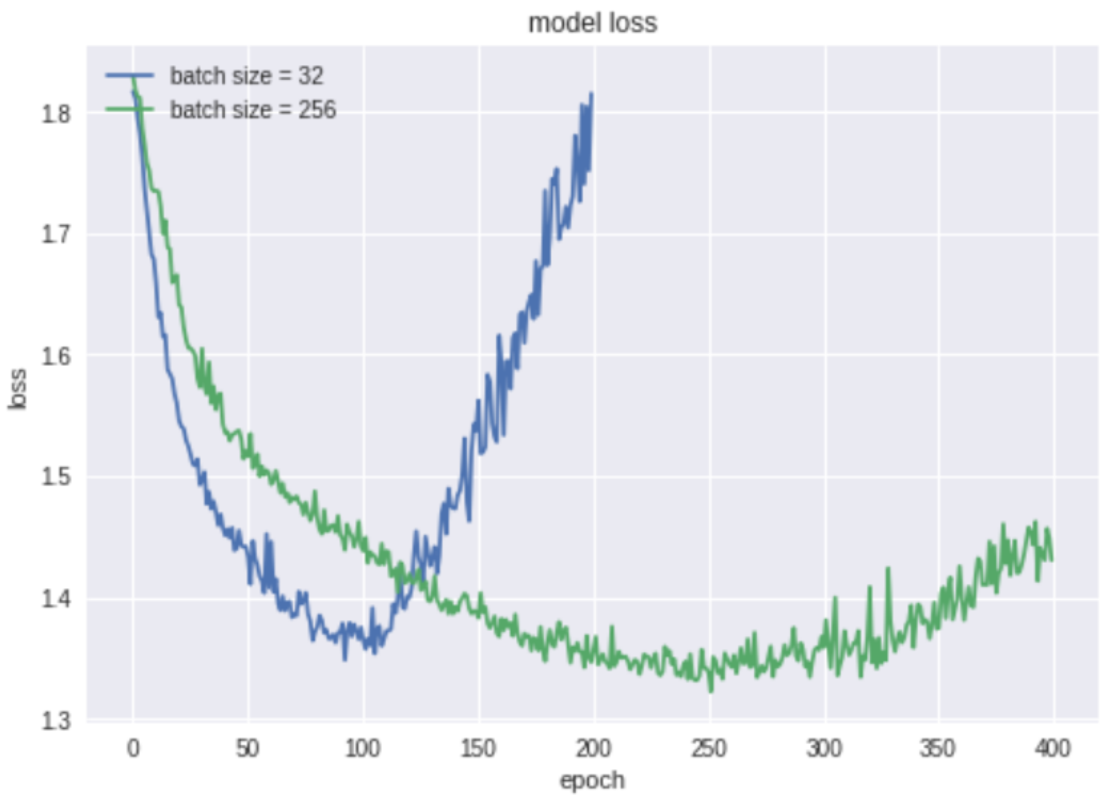
Batch Size	10	32	256	480
Seconds/epoch	74	26	12	11

Table 3.2 Time consuming among different batch size

Delete 10 batch size for quick training. And with batch size increasing, the number of epochs is also required higher to convergence. Based on the two points, we finally compare model validation accuracy among 32 and 256 training batch size and the results are shown as picture 3.2 and picture 3.3.



Picture 3.2 Model loss based on various training batch sizes



Picture 3.3 Model loss based on various training batch sizes

When training in 32 batch size, the model converges after 200 epochs, and when training in 256 batch size the model converges after 250 epochs. So we can get conclusion that the larger batch size, the more epochs needed in the picture 3.3. But when training in 256 batch size, model has the lowest validation loss and higher accuracy, so we choose 256 as training batch size.

3.3 Avoid Overfitting

3.3.1 Dropout

To avoid overfitting[4], we drop different 20% and 50% in fully connected layers, and compare the accuracy rate with none dropout, shown in table 3.3.

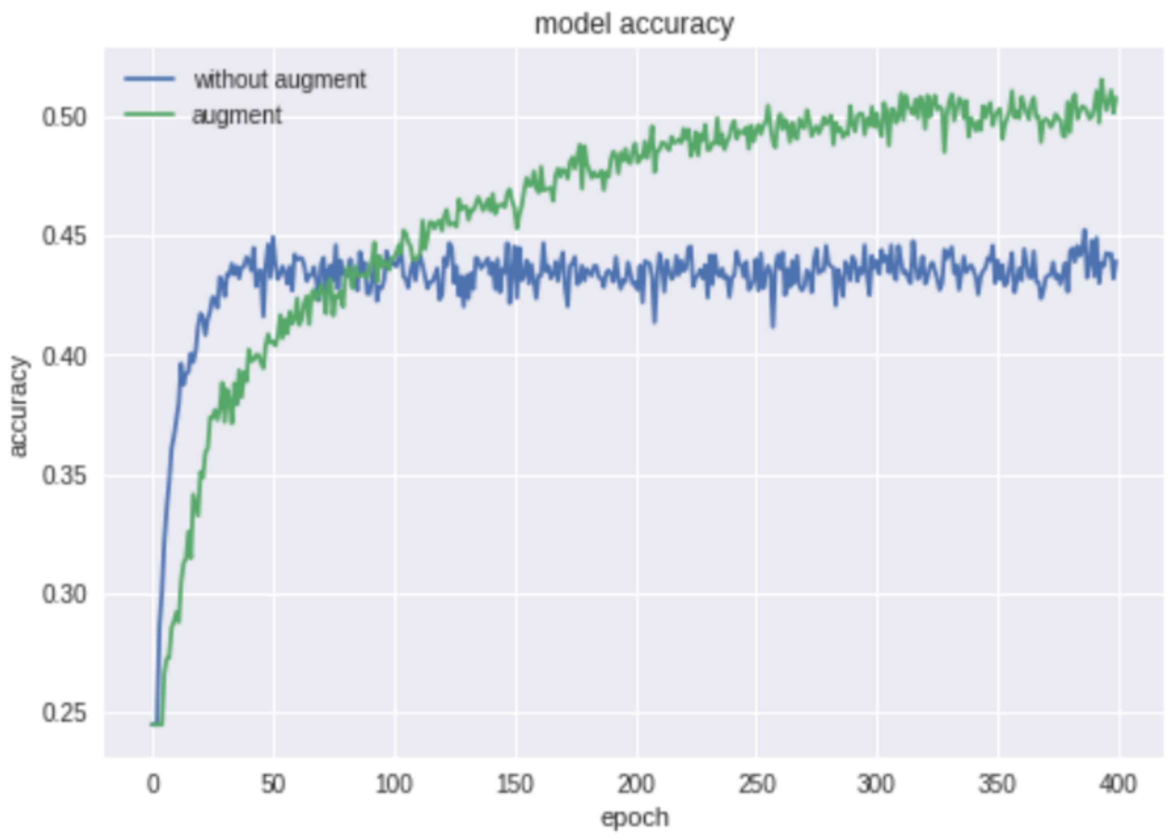
Dropout	None	20%	50%
Accuracy(percent)	49.97	50.73	52.36

Table 3.3 VGG10 test accuracy by different dropout

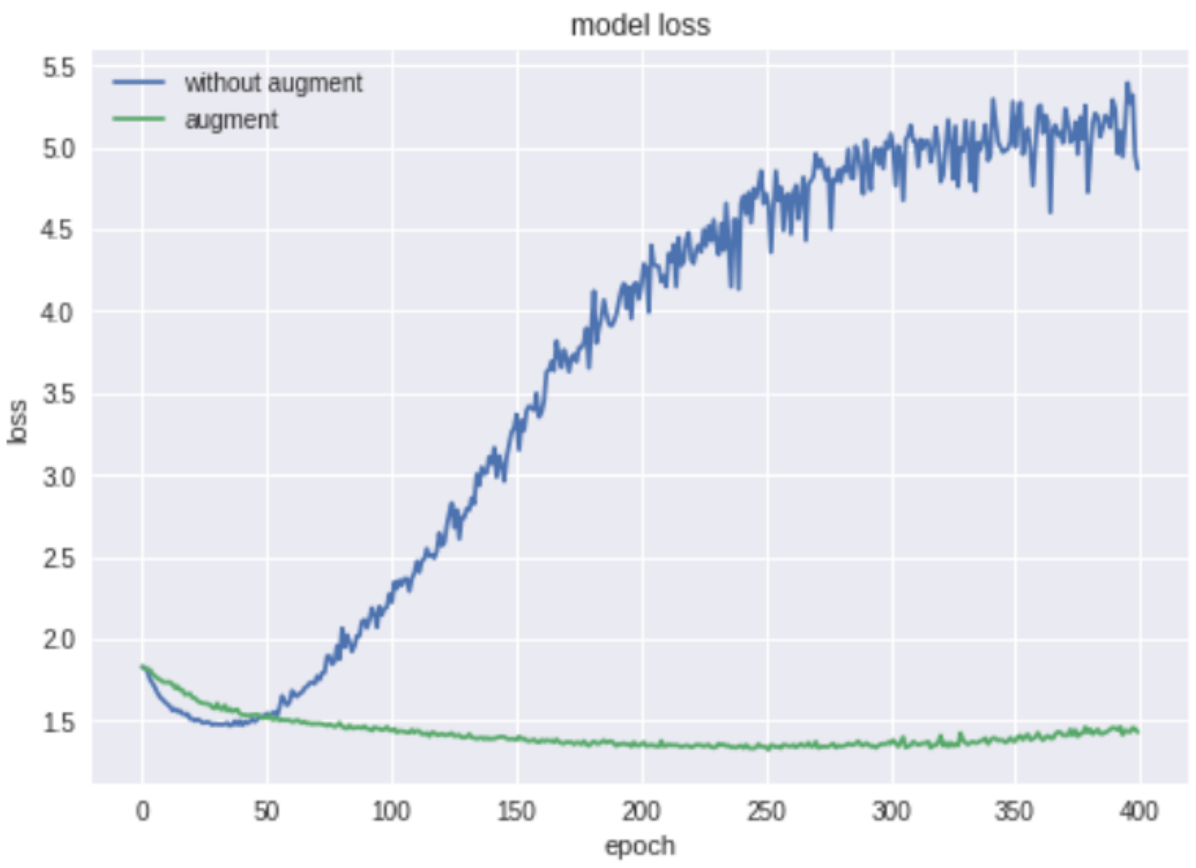
From table 3.3, we find 50% dropout has highest accuracy, so we adopt 50% dropout in later models.

3.3.2 Image Augment

As mentioned in sector 2.4, we exploit Image Augment[5] to create more data for training, then we draw accuracy- epoch curves in training dataset and test dataset, shown in picture 3.4 and picture 3.5. From this picture, we get Image Augment can promote the model performance.



Picture 3.4 Model accuracy with augmentation vs without augmentation



Picture 3.5 Model loss augmentation vs without augmentation

Another point we find is that training without augment only need 5 seconds per epoch vs 12 seconds per epoch with augment. Although the time is less, but result is not so good. Therefore, we adopt image augment to later models.

3.4 Experiment Results

Based on optimised parameters, we build VGG10, VGG12 and VGG14 and get their loss and accuracy curve, shown in table 3.4. VGG10 to VGG14 the model becomes more and more complex thanks to increasing depth and can fit better the dataset.

Model	VGG10	VGG12	VGG14
Accuracy	52.36	53.59	56.36

Table 3.4 Model Accuracy

In table 3.5, we compare the model complexity of our model with the champion team of the competition. Although the result of the campion team is 69.4% on the test dataset, our models have much smaller number of parameters, so our results can be acceptable.

Architecture	Number of parameters
Champion team’s model	7.17M
VGG10	4.14M
VGG12	4.19M
VGG14	4.92M

Table 3.5 Number of parameters

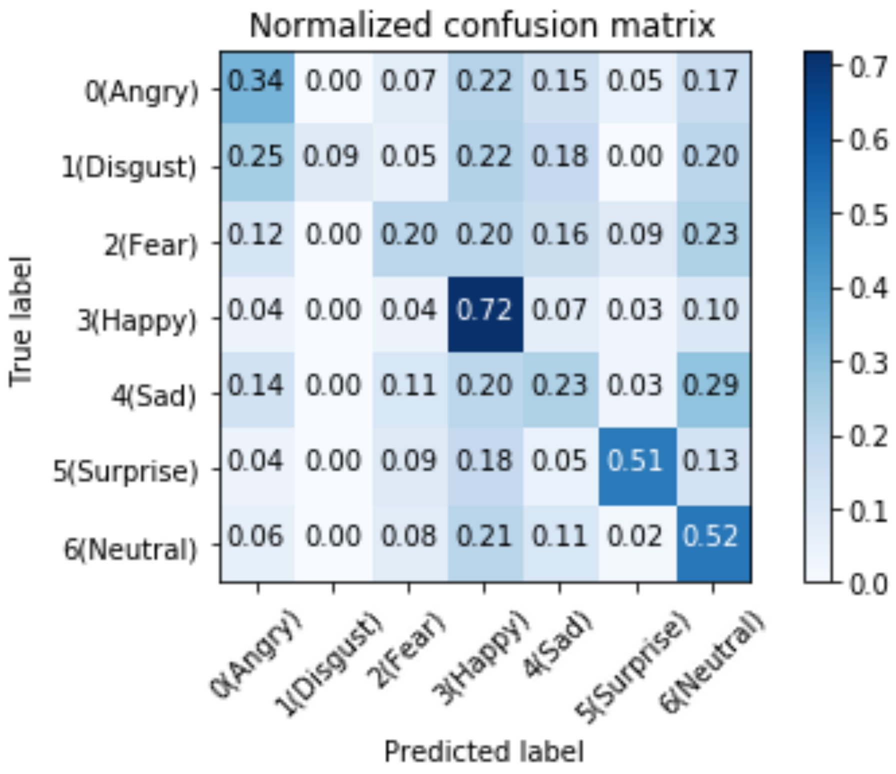
3.5 Comments

In this sector, we optimize parameters based on convergence speed and accuracy, then use dropout and augmentation to avoid overfitting effectively. Finally get three VGG models and comparing with the champion model, our models are good.

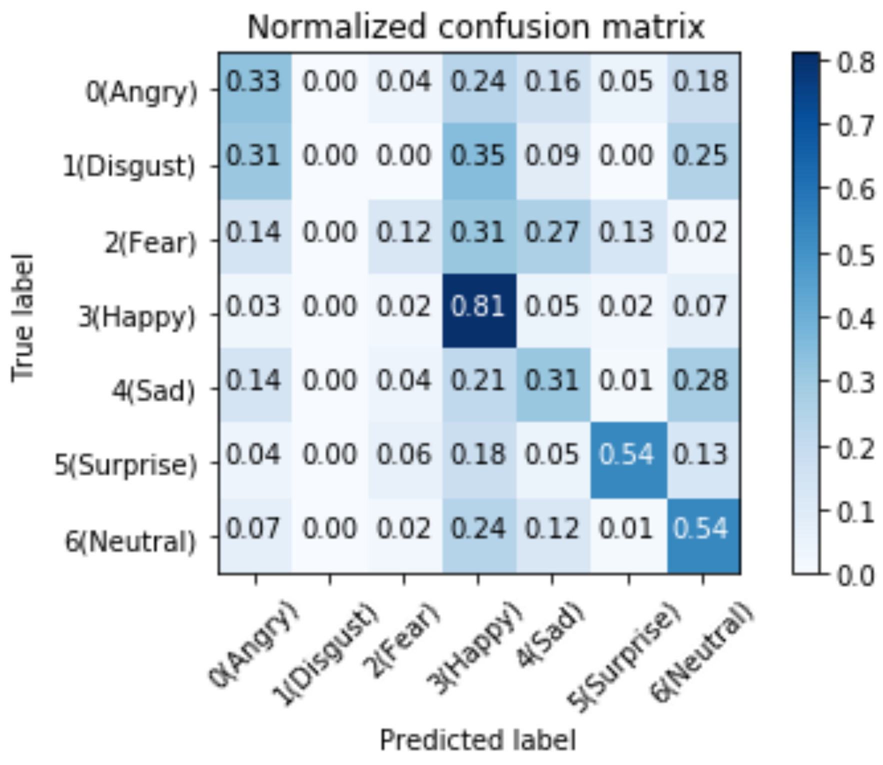
4. Comparison and Ensemble

4.1 Confusion Matrix

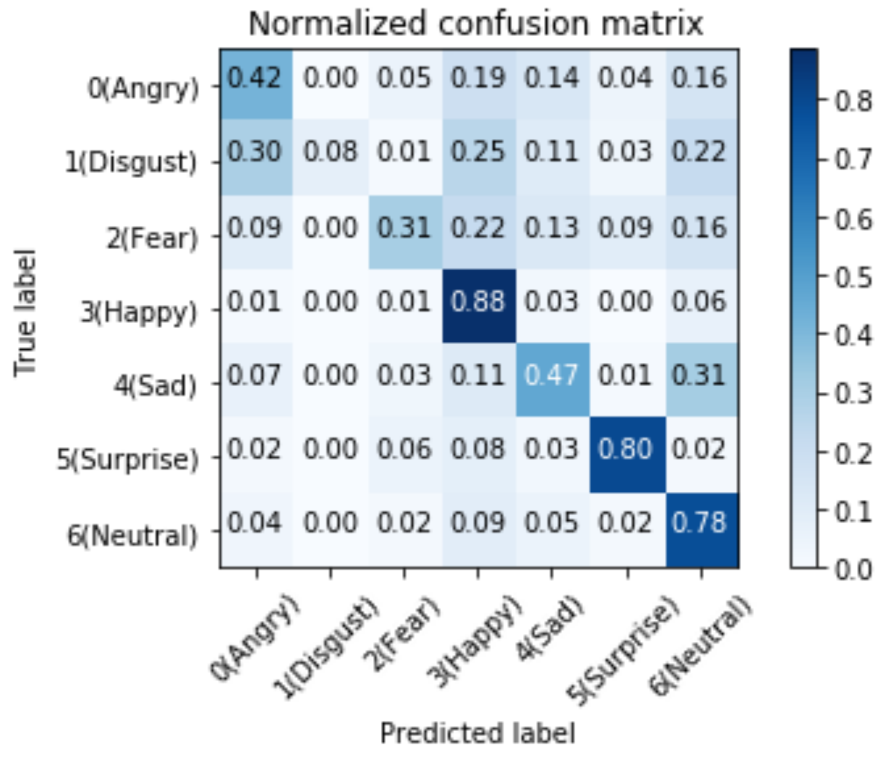
Picture 4.1 presents the confusion matrix of recognition result achieved by VGG10,VGG12 and VGG14 architecture. As convolutional layers are deeper, each kind of expression accuracy is increasing. High accuracies are obtained with happy (88%), surprised (80%), and neutral (78%) classes. In fact, they are the most distinguishable emotions for human. The angry, fearful, sad classes are more often confused together, since they share many similar expressions. Finally, the disgusted class get a relatively good accuracy 9%, despite the low amount of disgusted samples in the training set.



(a) VGG10



(b) VGG12

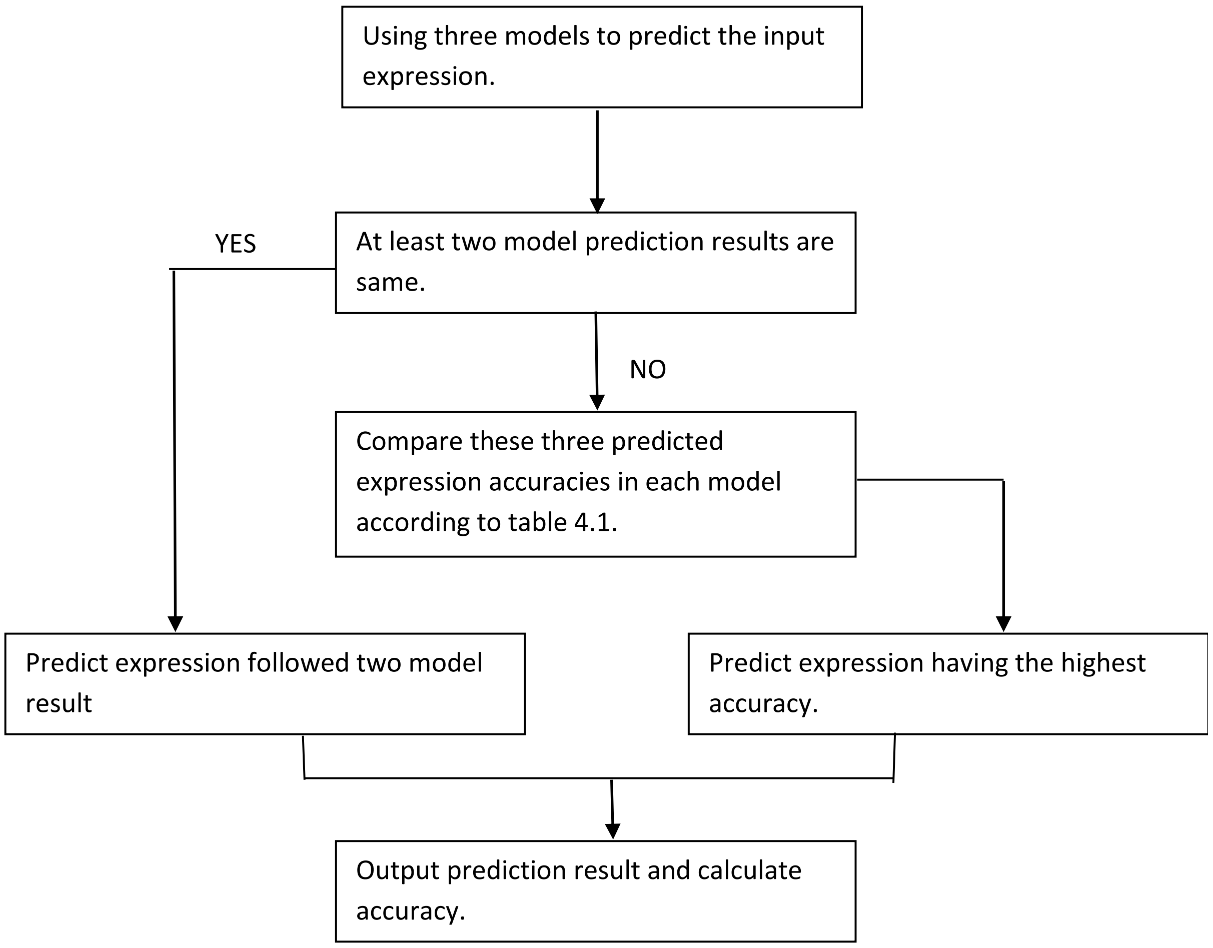


(c) VGG14

Picture 4.1 Normalized Confusion Matrix

4.2 Ensemble three models

To get higher accuracy, we follow the rule shown in picture 4.2 to make the three VGG models ensemble.


Using three models to predict the input expression.
At least two model prediction results are same.
YES
NO
Compare these three predicted expression accuracies in each model according to table 4.1.
Predict expression followed two model result
Predict expression having the highest accuracy.
Output prediction result and calculate accuracy.

The final ensemble accuracy is shown in table 4.1.

Picture 4.2 Model ensemble block diagram

Type	Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral	Average
Accuracy(%)	45.23	11.73	39.88	89.56	50.77	79.77	75.36	59.34

The final ensemble accuracy is shown in table 4.1.

4.3 Comments

In this section, we build VGG10, VGG12 and VGG14. The accuracy is higher as convolutional layer are deeper. Moreover, we adopt ensemble to achieve better result. We find not only average accuracy is higher, each type of expression accuracy is also higher, so ensemble model has an significant influence on the performance of the whole system.

5. Conclusion

In this project, inspired by the designing principles of VGG, we adopt effective architectures of CNNs to tackle the problem of facial expression recognition. The proposed networks are composed of a stack of convolutional blocks. Each block has a few 3×3 convolutional layers followed a max pooling layers. Despite having much smaller number of parameters, the model becomes more and more complex thanks to increasing depth and can fit better the dataset. The results prove the power of small filter and very deep network in classification tasks. We also show data augmentation is be an important trick in training deep neural network. Additionally, ensemble models can promote the final accuracy.

Files and codes related to the experiments, we have pushed to the github remote repository url is https://github.com/alex44jzy/NUS-CI1-CA1. And the facial expression dataset is sourced from Kaggle competition (Challenges in Representation Learning: Facial Expression Recognition Challenge), link is https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge.

6. Future Work

Here, we propose two approaches to further improve the performance of our VGG neural networks as our future works.

As the percentage of disgust expression images among the whole dataset is only 2%, very less, and they are quite similar to angry expression images, as a result the accuracy of disgust is relatively low(11.73 %). Such a low accuracy has influenced the overall testing accuracy. So we plan to investigate how to improve the testing accuracy by concentrating on the disgusting expression data.

Firstly, since the training images with label of disgusting expression accounts for a quite small percentages, we hypothesize that if we remove the small group of images from the training and testing dataset, we would acquire a higher testing accuracy. We believe that the disgusting group images are noises for the dataset, because they are very similar to and always being wrongly classified as “happy”, “angry” and “neutral” (Picture 4.1).

Secondly, we suggest to manually increase the percentage of the small disgusting expression images group by data augmentation technology. By the reason that the training samples are too less, the neural network might not converge for the disgusting expression images. So if we increase the number of samples from this label group, it is expected to give out a better testing accuracy.

Reference

[1] Sang, D.V. and Van Dat, N., 2017, October. Facial expression recognition using deep convolutional neural networks. In *Knowledge and Systems Engineering (KSE), 2017 9th International Conference on* (pp. 130-135). IEEE.

[2] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[3] Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[4] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, *15*(1), pp.1929-1958.

[5] Wong, S.C., Gatt, A., Stamatescu, V. and McDonnell, M.D., 2016, November. Understanding data augmentation for classification: when to warp?. In *Digital Image Computing: Techniques and Applications (DICTA), 2016 International Conference on* (pp. 1-6). IEEE.