ADVANCED SIGNAL PROCESSING LABORATORY COURSE
IMAGE RECOGNITION

Le Hai Nam, Tran Duy Hung

281822, 268824
nam.le@tuni.fi , duy.tran@tuni.fi

ABSTRACT

The purpose of this work is to build a model which can classify human face images into smile or no-smile classes by using Convolutional Neural Network (CNN).

Dataset used for training is GENKI-4k consisting of 4000 face images (2162 smiles and 1838 non-smiles).

1. INTRODUCTION

Image recognition has become an exploding area in computer vision since CNN appeared. With the help of CNN, computer vision has reached much higher accuracy in many image recognition and object detection tasks. In this project, the objective is to classify human face images to either smile or non-smile class. The model was built to solve the task was a CNN model which consists of 6 convolutional layers and 2 dense layers.

2. DATA

Dataset GENKI-4k has 2162 smile images and 1838 non-smile images. Labels include many other attributes of images like expression and pose, but the scope of this task is to classify images into smile or non-smile class, so we only considered this part of the labels.

Before feeding the images to our model, we did some simple pre-processing to obtain better outcome. Firstly, we resized all images to 64x64. Secondly, we converted them to gray scaled images. Finally, we normalized them to range 0 to 1.

After pre-processing, we divided the dataset to training and test set with ratio 80/20.

Layer (type)	Output Shape	Param #
conv2d_7 (Conv2D)	(None, 64, 64, 16)	160
conv2d_8 (Conv2D)	(None, 64, 64, 16)	2320
max_pooling2d_5 (MaxPooling2	(None, 32, 32, 16)	0
dropout_6 (Dropout)	(None, 32, 32, 16)	0
conv2d_9 (Conv2D)	(None, 32, 32, 32)	4640
conv2d_10 (Conv2D)	(None, 32, 32, 32)	9248
max_pooling2d_6 (MaxPooling2	(None, 16, 16, 32)	0
dropout_7 (Dropout)	(None, 16, 16, 32)	0
conv2d_11 (Conv2D)	(None, 16, 16, 64)	18496
max_pooling2d_7 (MaxPooling2	(None, 8, 8, 64)	0
dropout_8 (Dropout)	(None, 8, 8, 64)	0
conv2d_12 (Conv2D)	(None, 8, 8, 128)	73856
max_pooling2d_8 (MaxPooling2	(None, 4, 4, 128)	0
dropout_9 (Dropout)	(None, 4, 4, 128)	0
flatten_2 (Flatten)	(None, 2048)	0
dense_3 (Dense)	(None, 64)	131136
dropout_10 (Dropout)	(None, 64)	0
dense_4 (Dense)	(None, 2)	130

Total params: 239,986
Trainable params: 239,986
Non-trainable params: 0

Figure 1: Model architecture.

3. MODEL

As mentioned above, our CNN model consists of 6 convolutional layers and 2 dense layers. Additionally, there are also some dropout and max pooling layers between those layers. Max pooling layers help the model to extract local features of the input image while dropout layers help the model generalize better by shutting down some nodes of the model during training.

The detailed architecture of the model is shown in Figure 1. The total number of parameters of the model is 239 986 and all of them are trainable parameters.

All layers except the last one uses ReLU for activation function to prevent exploding or vanishing gradient problem. The last dense layer uses softmax for activation function.

We also initialized weights of all layers with He normal initialization. This weight initialization helps the model converge faster.

4. TRAINING

After pre-processing data and building model, we fed the data to the model to make learn to classify images to smile and non-smile classes.

We used Adam algorithm for training to make the training session converge faster. Loss function to evaluate the model is categorical cross-entropy. We trained the model with 100 epochs. Additionally, because our dataset is not balanced, we have more smile then non-smile images. Therefore, we calculated the percentages of each class in the whole dataset and gave this information to the model as prior knowledge.

5. RESULT

After training the model with 100 epochs, we got the validation accuracy of 87%.

From Figure 2, we can see that loss is reduced at the beginning of both training and validation. However, after epoch 40 the loss starts to increase in validation while training loss still decreases. This happened because from epoch 40, the model starts to overfit by just memorizing all the input data.

Figure 3 shows accuracy of our model during training and validation. Accuracy of both training and validation increases over number of epochs. Training accuracy reached almost 100% while validation accuracy stopped at around 87%.

Figure 2: Loss over epoch.

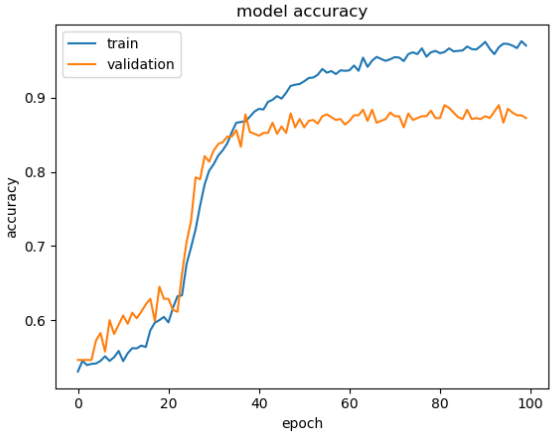

Figure 3: Accuracy over epoch.

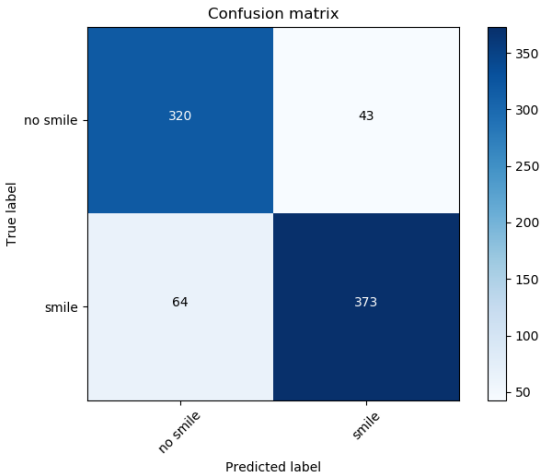

Figure 4: Confusion matrix.

Figure 4 is the confusion matrix of validation. There are 473 smile images in validation test and the model classified 373 samples of them correctly. And the model classified correctly 320 of 363 non-smile images.

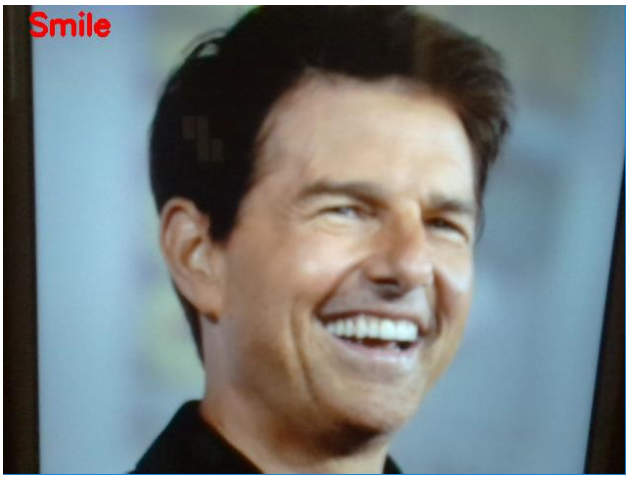

Figure 5: Smile test

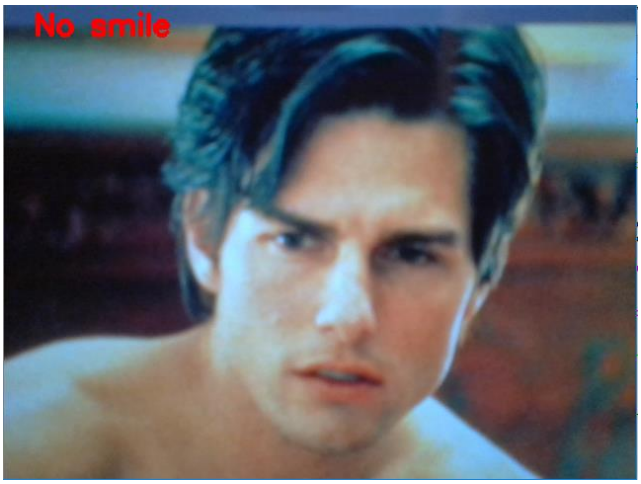

Figure 6: Non-smile test

Figure 5 and 6 are two images which are not included in GENKI-4k to test the accuracy of our model.

6. CONCLUSION

In this project, we have successfully implemented a simple CNN model to do an image recognition task. Even though the accuracy is not perfect but there are still many places for improvement in our model.

There are many ways to improve the accuracy of the model. Firstly, we could collect more extra data to train our model. The more data we have, the better the model could become. Additional to collect more data, we could also expand the architecture of our model by adding more layers to give it more capacity to learn better. Lastly, we could also use more complicated techniques in pre-processing phase like edge detection, face crop, etc. to improve the final outcome.

APPENDIX

In our submission, there are two Python scripts smile_detection.py, camcpture.py and our model's trained parameters model.h5. The script smile_detection.py does all the jobs including reading data, pre-processing, building model, training, plotting graphs and saving model parameters. The script camcapture.py reads the model paramteres and takes image from computer webcam and classifies the image.

In order to test the model, only script camcapture.py is needed. When the script is run, a webcam window will pop up then press key C to take image. The image taken is fed directly to our model to be classified. Another window will pop up after the image is classified and shows the result of the classification like in Figure 5 and 6. Pressing key Q will stop the script.