APPLIED MACHINE LEARNING SYSTEM ELEC0132 19/20 REPORT

SN: 19132770

ABSTRACT

This report mainly focuses on reporting models selection, models implementation and results discussion of four given tasks. These four tasks include gender recognition and emotion detection on a real person, face shape detection and eye colour detection on cartoon faces.

A convolutional neural network (CNN) learning algorithm is used to train the models for all the four tasks. The accuracy score for these four tasks trained model are 90%, 87%, 100% and 84%, respectively. These high accuracy scores suggest that the CNN algorithm works for these four tasks and produces four good trained model to fit the requirements of tasks.

1. Introduction

The problem of this assignment is to use a machine learning method to perform image classification on given datasets. Two datasets are given for four tasks, including faces of a real person dataset and a cartoon face dataset. These four tasks can be split into two binary classification tasks and multi-class classification tasks. These binary classification tasks include gender recognition (male or female) and emotion detection on a real person (smiling or not). The multi-class classification tasks include face shape recognition (5 types) and eye color recognition on cartoon faces (5 types).

Since the dataset contains labels which are provided, supervised learning algorithm is more preferably to be used to accomplish this assignment. Among all possible approaches of learning algorithms, the Convolutional Neural Network (CNN) is selected for this assignment.

The structure of this report is kept the same as the requirement of this report. This introduction is the 1st part of this report. Literature survey, model descriptions and model implementation are discussed separately in the 2nd, 3rd and 4th parts of this report. Furthermore, the experiment results are presented and discussed in the 5th part. Lastly, conclusions, references and supplement materials are presented in the 6th, 7th and 8th parts of this report, respectively.

2. Literature survey

This image classification task can be solved using various supervised leaning methods such as decision trees, support vector machines (SVM), logistic regression and neural networks.

Starting from the decision trees technique, this technique is a supervised learning algorithm and has been extensively used in video games and other areas. A good-constructed decision tree always splits the features on next best attribute to achieve high information gains. However, feature selection must be done before constructing a decision tree to achieve a good trained model. Otherwise, the model may experience overfitting and hence will deal with complex computations.

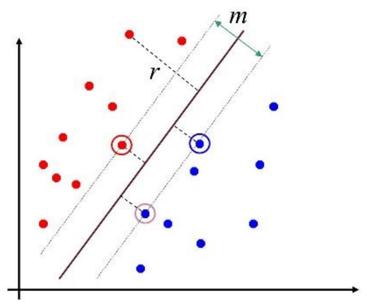

Fig. 1. SVM classification margin [1]

SVM is used to find a hyperplane for n-dimensional features to perform classification. Figure 1 shows a classification margin graph and the distance between the separator and point x can be calculated using equation (1), where b is the bias and w is the weight.

$$r = \frac{w^T x + b}{\|w\|} \tag{1}$$

Only the points closest to the hyperplane are taken into account for determining hyperplane because the principle of SVM is to make decision boundary as far away from data points as possible [1]. SVM experiences the same problem as decision trees algorithm does, that is a fine feature selection needs to be done before training a SVM model.

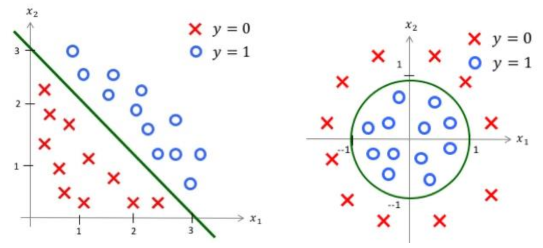

Fig. 2. Linear / non-linear classifier of logistic regression [2]

The left plot in Figure 2 shows a linear classifier of logistic regression whereas the right one illustrates a non-linear classifier of logistic regression.

$$S(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

Logistic regression builds binary decision regions to classify data points using a sigmoid function. The sigmoid function is given by equation (2), which outputs a binary value (0 or 1). However, logistic regression also needs feature selections to be done before training.

The state of the art machine learning algorithm is CNN. Since CNN can select spatial features using convolution layers and max pooling layers, no feature selection is needed before training a CNN model. Furthermore, the CNN training algorithm can better detect edges and shapes because the spatial interactions between pixels are conserved.

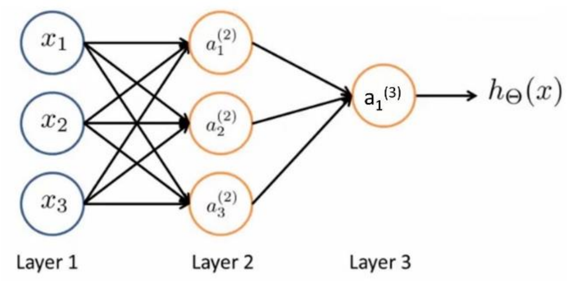

Fig. 3. Convolutional neural networks [3]

As can be seen from Figure 3, fully connected layers then dense to a node which outputs a binary classification result. If a multi-class classification model needs to be trained, then the final dense layer should have multiple nodes.

3. Description of models

There are four image classification tasks in this project, including two binary classification tasks and two multi-class classification tasks. The CNN learning algorithm can perform both classifications tasks with satisfied results.

The reason for choosing CNN algorithm is because it can perform feature selection tasks "automatically" and capture spatial features. Unlike other learning algorithms (e.g., logistic regression and SVM), the CNN learning algorithm does not need feeding with feature vectors. For example, to use SVM to perform an image classification task, first it has to develop a feature selection algorithm for the given task. In comparison, the CNN algorithm can take a pixel level vector of images as input and extract features by going through the convolution layer. This convolution layer is able to preserve spatial interaction between pixels, hence a CNN trained model is more efficient at detecting features such as edges and shapes. Therefore, the CNN trained model can make more accurate predictions on test dataset compared to other classification algorithm trained models.

To choose the best fit model for a given task, a number of test models are trained before choosing one. These test models are built using different hidden layer parameters to obtain various performance. Then, choose the best performed model among all the test models. If the chosen model can make a reasonable predictions for the test set, then this model is the best fit model for this task. Otherwise, train some more test models by tuning parameters based on the previous training experience.

For these 4 tasks, each task has a test model pool which contains 27 models. These 27 models are combinations of 1-3 convolution layers, 32, 64, 128 nodes of layers and 1-3 dense layers. For example, test models can be 1-conv-32-node-1 dense or 2 conv-32-node-2-dense and others.

The method of choosing best fit model from all test models is comparing the lowest validation loss. The reason to choose validation dataset instead of training datasets is that the test model may overfit the training dataset and then cannot make accurate predictions other than images in training dataset. Furthermore, the loss parameter reflects the distance between the predicted values and the true values. The lower the loss, the better fit the model is. Therefore, the method used here is to find the smallest value among all the lowest validation losses of the test models.

3.1. Task A1: Gender detection: male or female

Task A1 is a binary classification task in which a CNN model is trained to recognize the gender of a person in a given picture.

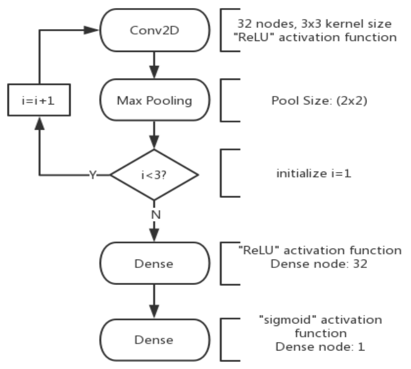
Conv2D
32 nodes, 3x3 kernel size
"ReLU" activation function
i=i+1
Max Pooling
Pool Size: (2x2)
Y
i<3?
initialize i=1
N
Dense
"ReLU" activation function
Dense node: 32
Dense
"sigmoid" activation function
Dense node: 1

Fig. 4. Flowchart of best fit model of task A1

The best fit CNN model of task A1 can be viewed using the flowchart shown as Figure 4. The input pixel vectors passes through 3 convolution layers and 3 max pooling layers in sequence, then these are densified into 32 nodes and 1 node which outputs the predicted result.

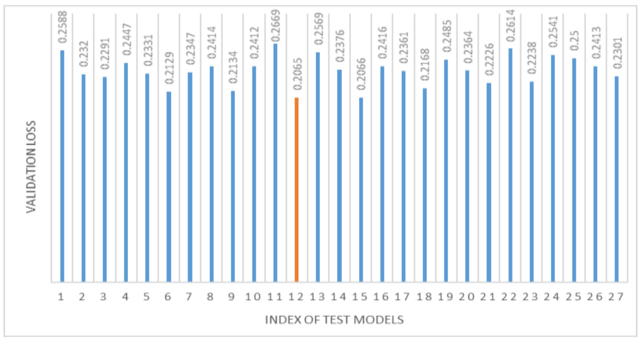

Fig. 5. Histogram of lowest validation loss for 27 different test models of task A1

From Figure 5, it is noticeable that the test model 12 has the lowest validation loss of 0.2065 among all the 27 test models. Therefore, the test model 12 is chosen to be the best fit model for task A1.

3.2. Task A2: Emotion detection: smiling/not smiling

Task A2 is also a binary classification task, but this time a CNN model is trained to recognize whether the person on a picture is smiling or not.

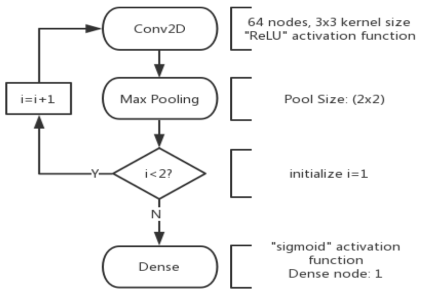
Conv2D
64 nodes, 3x3 kernel size "ReLU" activation function
Max Pooling
Pool Size: (2x2)
i=i+1
i<2?
Y
N
initialize i=1
Dense
"sigmoid" activation function Dense node: 1

Fig. 6. Flowchart of the best fit model of task A2

As can be seen from the flowchart in Figure 6, the best fit model of task A2 has 2 convolution layers, 2 max pooling layers and 1 dense layer. The dense layer has 1 node which outputs the prediction of the class.

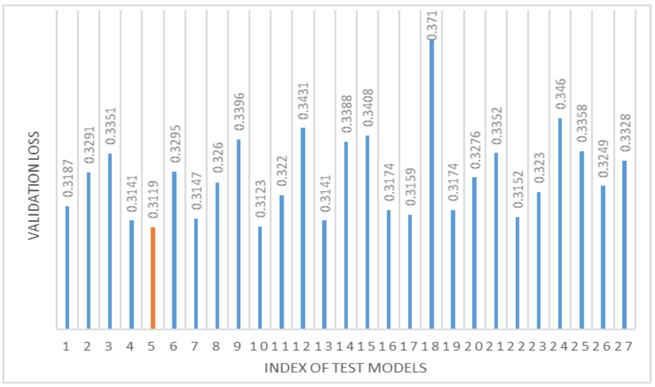

Fig. 7. Histogram of lowest validation loss for 27 different test models of task A2

Test model 5 is chosen to be the best fit model of task A2 because it has the lowest validation loss (0.3119) among all 27 test models, as shown in Figure 7.

3.3. Task B1: Face shape recognition (5 types)

Task B1 is a multi-class classification task where a CNN model is trained to recognize 5 types of cartoon face shapes.

The flowchart presented in Figure 8 shows all hidden layers in best fit model of task B1. This model contains 1 convolution layer, 1 max pooling layer and 3 dense layers. The 5-node dense layer outputs the probability of test image belongs to each class.

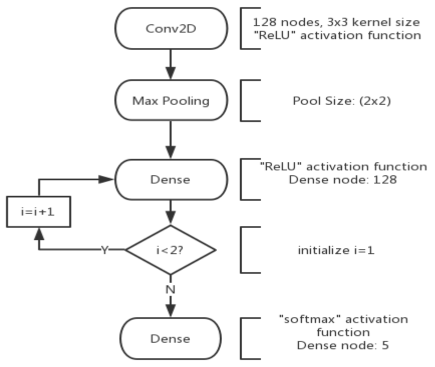
Conv2D
128 nodes, 3x3 kernel size "ReLU" activation function
Max Pooling
Pool Size: (2x2)
Dense
"ReLU" activation function Dense node: 128
i=i+1
i<2?
Y
N
initialize i=1
Dense
"softmax" activation function Dense node: 5

Fig. 8. Flowchart of the best fit model of task B1

Test model 25 is chosen to be the best fit model of task A2 as it has the lowest validation loss of 6.03e-4 among all the 27 test models, as can be seen from Figure 9. This time, there are three outliers in all test models. Models 18, 24 and 27 converge at the first epoch with a validation loss of 1.61, and validation losses of these three models stay the same after that.

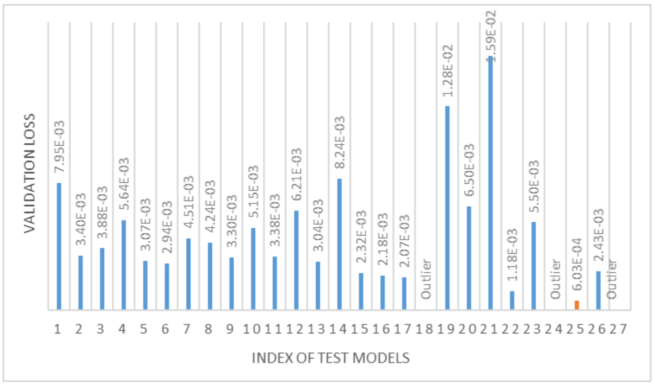

Fig. 9. Histogram of lowest validation loss for 27 different test models of task B1

3.4. Task B2: Eye color recognition (5 types)

Task B2 is a multi-class classification task, a CNN model is trained to recognize 5 eye colours of the cartoon face.

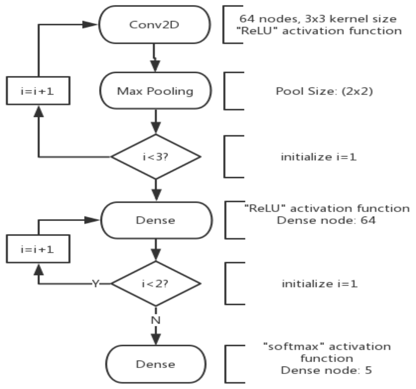
Conv2D
64 nodes, 3x3 kernel size "ReLU" activation function
Max Pooling
Pool Size: (2x2)
i=i+1
i<3?
initialize i=1
Dense
"ReLU" activation function Dense node: 64
i=i+1
i<2?
Y
N
initialize i=1
Dense
"softmax" activation function Dense node: 5

Fig. 10. Flowchart of the best fit model of task B2

From Figure 10, it can be seen that this model have 3 convolution layers, 3 max pooling layers and 3 dense layers. The final dense layer has 5 nodes and outputs the probability of test image belongs to each class.

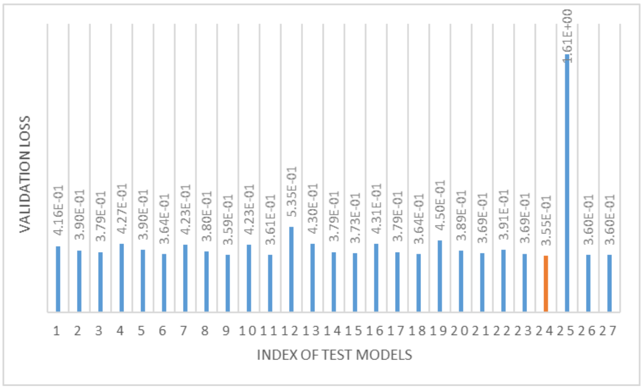

Fig. 11 Histogram of lowest validation loss for the 27 different test models of task B2

Test model 24 has the lowest validation loss of 0.355 among all the 27 test models, shown on Figure 11. Since the validation loss of test model 24 is low enough to make accurate predictions, this is chosen to be the best fit model for task B2.

4. Implementation

Since these 4 tasks use the same learning algorithm, some of the implementation details are similar. These similar implementation details are external libraries, key functions, dataset, hyper parameters, key functions, datasets, data pre-processing, dataset separation and execution of the models.

4.1. External libraries

The external libraries used to implement the CNN models are OpenCV, NumPy and TensorFlow. Starting from OpenCV, this external library is used to convert the color of images, convert images to array and resize images. The next external library is NumPy, which can be used to manipulate arrays and this includes creating arrays, resizing arrays and others. Lastly, TensorFlow is an extremely powerful external library. With the help of TensorFlow, the CNN model can be built and trained easily by calling layer class and models class function form library.

4.2. Hyper parameter selections

Hyper parameters of the model training include activation function of layers, loss function, learning rate and batch size.

4.2.1 Activation function

To begin with activation function, 3 activation functions are used to activate different layers. These functions are Rectified Linear Unit (ReLU), sigmoid and softmax.

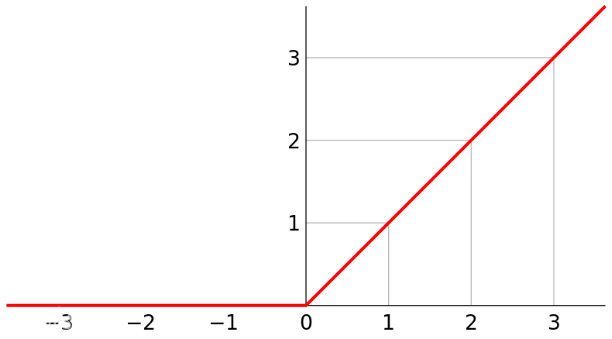

Fig. 12. ReLU function

ReLU is a linear activation function used for most of layers except the final dense layer. It is also known as ramp function given by

$$f(x) = \max(0, x) \tag{3}$$

From Figure 12, ReLU is a half rectified function with an output ranging from 0 to infinity. The use of the ReLU activation function can increase sparsity and reduce computational complexity by putting negative inputs of neurons into 0.

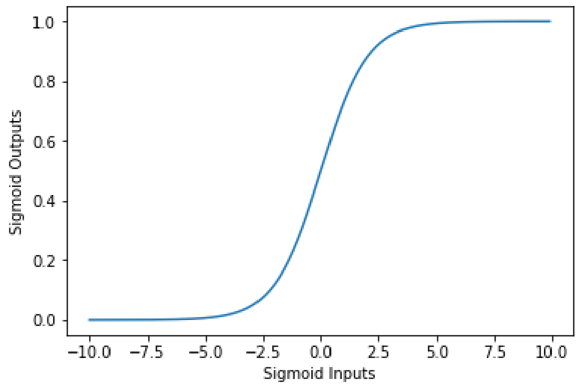

Fig. 13. Sigmoid function

Next, the sigmoid function is a non-linear activation function. This function is mainly used in the final dense layer of the binary classification model to help identify which class a test image belongs to. To elaborate, an output number smaller than 0.5 suggests that the test image belongs to class 0 and vice versa.

$$S(x) = \frac{1}{1+e^{-x}} \tag{4}$$

Equation (4) shows the expression of the sigmoid function. From Figure 13, it can be seen that the sigmoid function outputs numbers bigger than 0.5 but smaller than 1 if inputs bigger than 0. On the contrary, inputs smaller than 0 will have outputs smaller than 0.5 but bigger than 0.

Lastly, the softmax function is similar to the sigmoid function but can be used in multi-class classification task.

$$\sigma(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \; for\ i = 1, \dots K \tag{5}$$

From equation (5), this softmax function first takes standard exponential to each neuron inputs and normalizes it by dividing the sum of all exponential inputs. Using this function as an activation function can help to identify the probability of a test image belongs to each class.

4.2.2 Loss function

Cross entropy loss function is used in this assignment because it takes probability into calculation of distance between predictions and labels. Therefore, cross entropy is a good loss function to be used in classification problems.

There are two variations of cross entropy used in this assignment, namely, the binary cross entropy and the categorical cross entropy. The binary cross entropy loss is used in tasks A1 and A2 whereas the categorical cross entropy loss is used in tasks B1 and B2.

The binary cross entropy loss is often used together with the sigmoid function to deal with the binary classification problem. In equation (6), C is the number of class, y_i is true label, $S(i)$ is the sigmoid function shown in equation (4).

$$BCE = -\sum_{i=1}^{C=2} y_i \log(S(i)) \tag{6}$$

The categorical cross entropy loss is used with the softmax function to deal with the multi-class classification problem. In equation (7), $\sigma(i)$ is the softmax function shown in equation (5).

$$CCE = -\sum_{i=1}^{C=K} y_i \log(\sigma(i)) \tag{7}$$

4.2.2 Learning rate and batch size

Leaning rate is chosen to be the default value of 0.01 in all the 4 tasks because lower learning rate cannot further reduce loss but increases the training time and converge slower. Higher learning rate causes an increase in the validation loss. The batch size is chosen to be 32 because a lower batch size increases the training time and higher batch size increases the loss.

4.3. Key functions

Key functions can be generally split into two parts, namely, self-created functions and imported functions. To begin with, self-created functions are also called Data_Preprocessing() and Model(). Key imported functions used in this project are cv2.imread(), cv2.resize(), model.add(), model.compile and model.fit().

The Data_Preprocessing function is used to pre-process data by converting images to pixel level arrays and extracting csv file labels for each images. This function takes 3 parameters (resized image size, image path and label file path) and returns 6 dataset arrays. These 6 arrays are training set feature arrays and label arrays, validation set feature arrays and label arrays, test set feature arrays and label arrays.

The model function is used to train the models with the selected hyper parameters. This function takes 8 parameters (the number of convolution layers, the number of dense layers, nodes in layers, epoch numbers, training set feature arrays and label arrays, validation set feature arrays and label arrays) and returns the trained model.

The Cv2.imread and cv2.resize functions are imported from the OpenCV library. The conversion between images and arrays is executed using the cv2.imread function. On the other, the cv2.resize function resizes image array into the wanted array size.

Model.add, model.compile and model.fit are imported from the TensorFlow library. Model.add function is used to add hidden layers in sequence, including convolution layer, max pooling layer, activation layer and dense layer. The Model.compile function sets loss function and the learning rate of the trained model. Furthermore, the model.fit function defines the batch size, epochs, training set data and the validation set data.

4.4. Datasets

The tasks A1 and A2 use the same image datasets which contains 5000 images of celebrities' faces. This dataset is a subset of CelebA (CelebFaces Attributes Dataset), which is "a large-scale face attribute dataset with more than 200K celebrity images. The images in this dataset cover a large pose variations and background clutter. CelebA has large diversities, large quantities and rich annotations" [4]. To store the corresponding label of each image, a csv (comma separated values) file is used. To be specific, female celebrities' pictures are labelled as -1 and male celebrities' pictures are labelled as 1 in the csv file.

On the other hand, the tasks B1 and B2 use another image dataset which contains 10000 images of cartoon faces. This dataset is a subset of the Cartoon Set, which is "a collection of random, 2D cartoon avatar images. The cartoons vary in 10 artwork categories, 4 colour categories and 4 proportion categories, with a total of ~1013 possible combinations" [5]. The multi-class attributes of this cartoon face dataset is also stored in a csv file, but with labels ranging from 0 to 5 with each number representing a category.

4.5. Data preprocessing

The data pre-processing can be described using the flowchart illustrated in Figure 14.

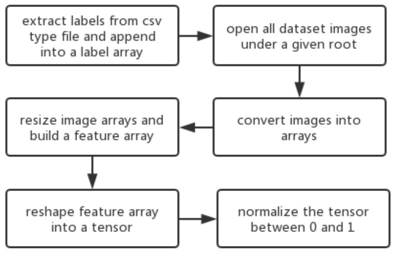
extract labels from csv type file and append into a label array
open all dataset images under a given root
convert images into arrays
resize image arrays and build a feature array
reshape feature array into a tensor
normalize the tensor between 0 and 1

Fig. 14. Flowchart describing data pre-processing

The conversion of images into arrays is implemented by calling the cv2.imread function. For example, an RGB image with a resolution of 500x500 is converted into an array with a shape of 500x500x3. If an image with the same resolution is converted into grey-scale, then the converted array has a shape of 500x500x1.

Next, the image array is resized to reduce the computational complexity. An appropriate image resolution should be selected to save the training time and resources. For example, a cartoon face image has 500x500 image resolution but to recognize the face shape, only 70x70 image resolution is required. Therefore, image arrays need to be resized using the cv2.resize function and resize arrays into the preferred size.

Furthermore, a reshape process is required to build a tensor prepared for the training process. The tensor is a 4D array, contains the number of images, the image resolution and the number of channels. The image resolution is a 2D array and the number of channels is defined by images' colour scale. For example, an RGB image gives 3 channels and grey-scale image gives 1 channel.

Lastly, normalizing the feature array from 0-255 into 0-1. This process can also reduce the computational complexity which saves the training time and resources.

4.6. Dataset separation

The dataset needs to be separated into 3 different sets, namely, training set, validation set and test set. The training set contains images used to train the model and the validation set is used to develop a score to measure how good the model to images not in training set. Lastly, the test set is used to examine how good the trained model is by showing it some images never seen before.

The separation of the dataset used in these 4 tasks is 70% for the training set, 20% for the validation set and 10% for the test set. In this arrangement, the trained model is more likely to be trained with images of large diversity and large quantities. Also, the 20% validation set and 10% test set can better reflect the actual performance of the trained model because the trained model can be tested with different classes of pictures.

4.7. Task A1: Gender detection: male or female

Figure 15 shows the training and validation loss curves of best model of task A1. It can be seen from the graph that the training loss decreases constantly until the 20th epochs, but the validation loss stop decreasing after the 8th epoch. The decreasing of the training loss is a consequence of overfitting. The model tries to remember the training data, but that is only helpful to the training set images.

From the validation loss curve, the trained model is converged after 8th epoch and varies slightly. After the 15th epoch, the model suffers severely from overfitting and can be seen from a big jump in the validation loss. Therefore, the stopping criterion of this model is to stop training the model at the lowest validation loss, which is the 8th epoch. Since the validation loss (0.2065) at stopping step is small, the trained model is a good model for task A1.

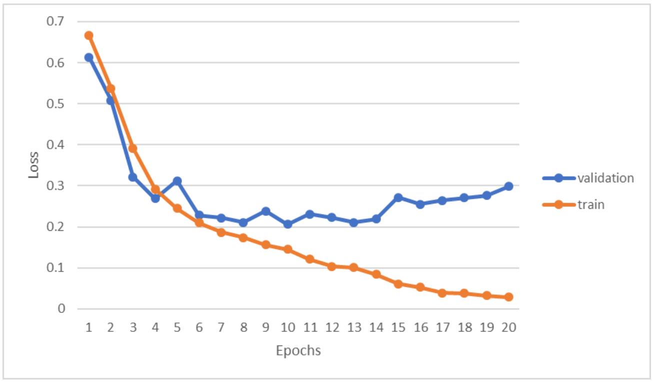

Fig. 15. Training and validation loss of task A1 model

4.8. Task A2: Emotion detection: smiling or not smiling

Figure 16 contains plots of the validation and training losses of task A2. Comparing Figures 15 and 16, it is apparent that the plots of tasks A1 and A2 share similar patterns. This is due to the similarity in the two tasks (detecting real person facial features). It is also clear that the training loss decreases constantly until the 13th epoch whereas the validation loss stops decreasing after the 6th epoch. In addition, the validation loss starts increasing again after 6th epoch. This trend clearly shows that the trained model converges at the 6th epoch and begins to suffer from overfitting after that. Therefore, the 6th epoch is chosen to be the stopping step with a validation loss of 0.3119. Since this validation loss is small, the trained model of task A2 can make reasonable predictions.

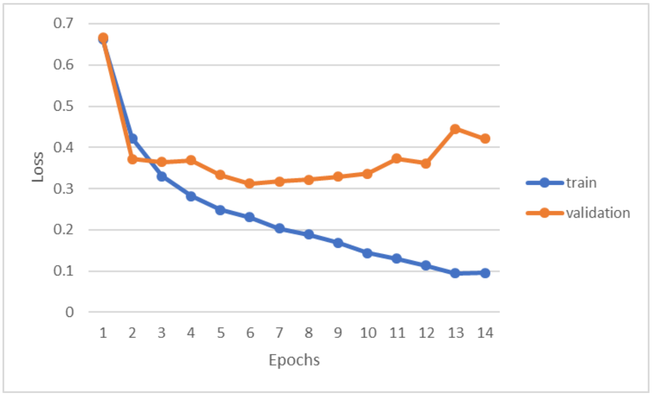

Fig. 16. Training and validation losses of task A2 model

4.9. Task B1: Face shape recognition (5 types)

Figure 17 shows the training and validation loss curves of the task B1 model. It is noticeable that the training loss stops decreasing after the 5th epoch, and the validation loss becomes stable after 12th epoch. Since the validation loss curve of task B1 model has extremely low value (smaller than 0.1), the trained model can make good predictions after running 2 epochs of the dataset. However, to achieve the best fit model, this model is chosen to stop running after 17th epoch. The reason for running 17th epoch is that because this validation loss curve becomes stable and with the lowest validation loss (6.03e-4) at 17th epoch.

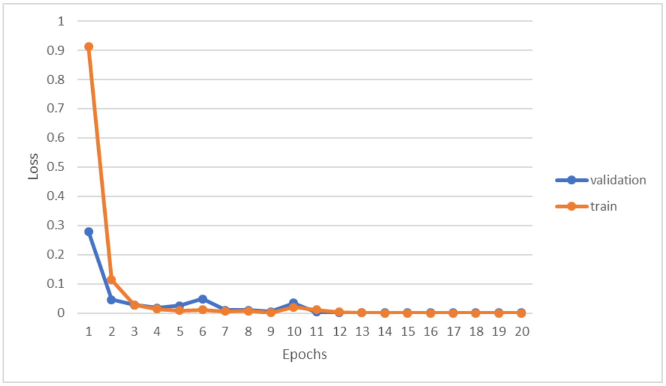

Fig. 17. Training and validation losses of the task B1 model

4.10. Task B2: Eye color recognition (5 types)

Figure 18 shows how the validation and training losses of the task B2 model varies. It is obvious that the training loss stops decreasing significantly after the 7th epoch because of the nature of this task. The eye color of cartoon faces with sunglasses cannot be identified. Therefore, the best that can be achieved is loss of around 0.3. The validation loss curve also stops decreasing significantly after the 6th epoch, and this means the trained model converges after 6th epoch. However, this model is stopped at 17th epoch because this is where the lowest validation loss happens for the trained model. Since the validation loss of 0.355 is reasonably small, the trained model of task B2 can make good predictions.

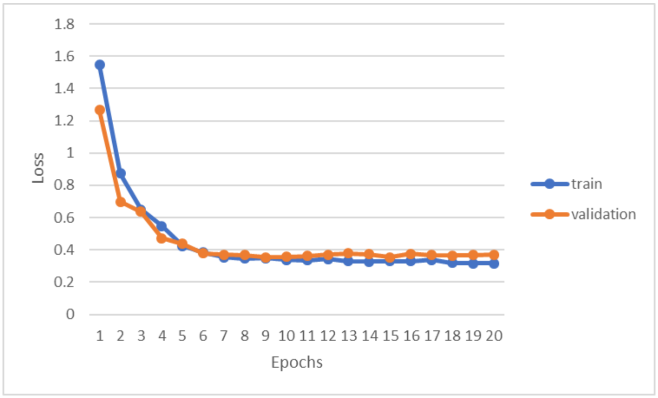

Fig. 18. Training and validation loss of task B2 model

5. Experimental Results and Analysis

Table 1 shows the accuracy score of the training sets, validation sets and test sets for each task. Training accuracy for these four tasks are the highest among all three sets because the trained model may remember some of images during training. On the other side, the validation set and the test set have similar accuracy scores because the model is not trained base on these two sets. The difference in usages of two sets is that the validation sets are used to determine how good the model is during the process of training and when to stop. On the contrary, the test sets are used to test how good the trained model is by showing some images this model has never seen.

Task	Model	Train Acc	Val Acc	Test Acc
A1	CNN	0.97	0.90	0.90
A2	CNN	0.92	0.88	0.87
B1	CNN	1.00	1.00	1.00
B2	CNN	0.85	0.81	0.84

Table 1. Training, validation and test accuracy of each task

These four tasks use the same learning algorithm, i.e., CNN. Among all the four tasks, the B1 task model has the highest test accuracy of 100%. On the other hand, the B2 task model has the lowest test accuracy of 84%.

5.1. Task A1: Gender detection: male or female

For the task A1, the accuracy score of the test set and the validation set are the same, 0.90. In other words, this trained model can detect 90% of the gender of real persons correctly. The real person images varies significantly, even human eyes can sometimes falsely recognize the gender of another one. Therefore, this trained model is a good model for the task A1 with 90% accuracy.

The training accuracy of 0.97 of this task is high because the CNN trained model starts to remember the training set. In other words, the trained model may run too many epochs and starts to overfit to the training set. However, the degree of overfitting is not severe and does not affect detection.

5.2. Task A2: Emotion detection: smiling or not smiling

The nature of task A2 is similar to that of task A1, since they share similar values of accuracy scores. However, the accuracy scores of each datasets in task A2 are lower than that in task A1. The lower accuracy score is due to the task complexity, determining a real person is smiling or not is harder than recognizing the gender of another person.

The training set accuracy of 0.92 is about 0.05 higher than both the training set accuracy of 0.88 and the test set accuracy of 0.87. This 0.05 difference in these datasets can also be explained by overfitting of the model. However, the degree of overfitting is not severe and does not affect detection.

5.3. Task B1: Face shape recognition (5 types)

The task B1 model is a perfect model to identify the face shape of "Cartoon Set" generated cartoon face because of 100% accuracy for all three datasets. This high accuracy score is because these three datasets all use "Cartoon Set" generated cartoon faces. To elaborate, the trained model has learnt how each type of face shape generated by "Cartoon Set" is curved. Although some cartoon faces are with beard, the model is still able to identify the face shape through analyzing parts of the face not being covered by beard.

5.4. Task B2: Eye color recognition (5 types)

The nature of task B2 is similar to that of task B1, identifying facial landmarks on "Cartoon Set" generated cartoon faces. Therefore, the task B2 model should have similar accuracy score as the task B1 model. However, the fact is that the training accuracy of B2 model is 0.85, the validation accuracy is 0.81 and the test accuracy is 0.84. Such low accuracy scores are caused because of the fact that some cartoon faces have sunglasses. Since this task aims to identify eye colors of cartoon faces, sunglasses will interfere the recognition of eye color. Furthermore, the difference in the validation accuracy and the test accuracy may be the reason of more cartoon face wearing sunglasses appear in the the validation set. However, overall, this model is still a good model to detect the eye color of "Cartoon face" generated cartoon face because the color of eye behind sunglasses cannot be identified by anyone.

6. Conclusion

In conclusion, the CNN algorithm has performed well and trained good models for each image classification task. However, overfitting is a big issue for the CNN model training. Running epochs and other parameters need to be carefully selected to achieve a good model. Besides that, the models trained with bigger datasets can produce better results.

Some improvements can be made to achieve better results for these tasks and future tasks. If the eye colour of cartoon face with sunglasses can be divided into the 6th type of eye colours, results of task B2 model will definitely improve considerably. For future tasks, increasing the dataset size can result in a better fit model because more samples can be learnt. Furthermore, stopping criterion can be chosen more carefully to fit to the specific task to avoid overfitting.

7. References

[1] P. Y. Andreopoulos, "Support Vector Machines Slides," 2019.

[2] P. M. Rodrigues, "Applied Machine Learning System 1," 2019.

[3] P. Y. Andreopoulos, "Neural Networks," 2019.

[4] Mmlab.ie.cuhk.edu.hk. (2019). Large-scale CelebFaces Attributes (CelebA) Dataset. [online] Available at: http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html [Accessed 17 Dec. 2019].

[5] Google.github.io. (2019). Cartoon Set: An Image Dataset of Random Cartoons. [online] Available at: https://google.github.io/cartoonset/ [Accessed 17 Dec. 2019].

8. Supplement Materials

1. code in folder with link *https://www.dropbox.com/sh/2ar8riy4yvqodp8/AAC6GW_fnQG8O8aeVJhT5ePda?dl=0*