

Machine Learning 2

by Xueyang Hu

Submission date: 03-May-2023 03:43PM (UTC+0100)

Submission ID: 204535056

File name: 41181_Xueyang_Hu_Machine_Learning_2_498640_2072296616.docx (310.84K)

Word count: 4631

Character count: 27114



**ACCA7025 – APPLICATION OF MACHINE
LEARNING**

MACHINE LEARNING 2



<200549> - <Xueyang Hu>

UNIVERSITY OF WALES TRINITY SAINT DAVID - Wales Institute of Science and Arts - School
of Applied Computing

Table of Contents

1.0 Introduction 2

2.0 Work Review 2

2.1 DNN..... 2

2.2 CNN..... 4

3.0 Method 5

3.1 Dataset..... 5

3.2 Architecture..... 5

4.0 Evaluation 8

5.0 Conclusion 9

6.0 Appendix: Code..... 10

Bibliography 15

1.0 Introduction

Artificial intelligence is a field that studies how to make machines have human's work abilities. Russell [1] summarized the characteristics of artificial intelligence into four categories through multiple concepts: Thinking humanly and rationally, and acting humanly and rationally. Machine learning is a research branch of artificial intelligence that focuses on how to equip machines with human's learning and thinking abilities to handle more complex tasks. Bishop [2] believes that a major feature of machine learning is the ability of finding meaningful relationships and patterns from data, rather than rigidly executing tasks within a fixed framework of hard rules. Neural Network is a learning model in the field of machine learning, which learns and understands knowledge by simulating the transmission of information by neurons, and processes new tasks through the learning results. Its inspiration comes from American psychologist James' exposition of the basic principles of brain structure and function, which was then transformed into Neural Network model theory by McCulloch and Pitts [3]. Due to the fact that Neural Network model is closer to real human thinking model, they have become one of the main research directions in machine learning in recent years. It has a wide range of applications, such as automatically generating physical mesh models based on point cloud [4], and automatically generating images based on text descriptions [5].

This report is mainly divided into three parts. In the Work Review section, the concepts, principles, characteristics, and applications of DNN and CNN will be introduced. In the Method section, this report will introduce the dataset used in the experiment and provide a detailed explanation of the architecture, training process, and optimization process of the neural network constructed in the experiment. In the Evaluation section, this report will test the neural network constructed earlier, quantify the results through various indicators and explain them .

Code presentation video link:

https://uowtsd-my.sharepoint.com/:v:/g/personal/2200549_student_uowtsd_ac_uk/ETrNqYn96fxOpeStWsX34gABxYgchjxskvpQzg--hvhg?e=i5eu8I

2.0 Work Review

2.1 DNN

The most basic neural network can be seen as a model that simulates a mathematical function through weight parameter matrixs and corresponding operations [6]. The training process of a neural network is the process of adjusting the weight parameters to make the simulated mathematical function closer to the ideal function. The most basic neural network Multiple Layer Perceptron (MLP) is a hierarchical structure model. Each layer of this model contains multiple neurons, and each neuron represents a weight parameter matrix. The model finally outputs a result or several results depending on the last layer's neuron number. The connections between neurons in each layer represent the process of combining last layer's neurons output according to their weight in neurons of current layer. The final output result of MLP will be compared with the groundtruth, and the gap between it and the groundtruth will be quantified through the loss function. This gap will be transmitted backward in the form of gradient and used to adjust the parameters in each neuron of each layer. This process

is called back-propagation. In order to make the model have the ability to simulate nonlinear functions, the model will contain nonlinear conversion between some layers. These nonlinear conversion functions are called activation function. MLP can be divided into three parts: input layer, hidden layer, and output layer. When there are many hidden layers, the neural network architecture is called a Deep Neural Network(DNN). Deep neural networks have better function fitting ability, which enables them to handle more complex tasks. This advantage has been mathematically proven in the research of HN Mhaskar et al. [7]. However, the more hidden layers there are, the more problems they bring, such as gradient vanishing and gradient explosion.

The gradient value transfer between each layer during back-propagation is usually less than 1.0. As the depth of the network increases, the gradient change during back-propagation decreases layer by layer, and the gradient almost close to 0 when approaching the input layer. This phenomenon is called gradient vanishing [8]. This results in normal weight updates for hidden layers near the output layer, while slow or stationary weight updates for hidden layers near the input layer, which cause the trained DNN is only equivalent to a shallow neuron network. This usually happens when an inappropriate loss function is used. On the other hand, Gradient explosion is the opposite. When the gradient value transmitted between hidden layers is greater than 1.0, back-propagation will cause the gradient change to grow exponentially, leading to the overall instability of the network. This is usually because the weight initiate values are too large[9]. These problems can be suppressed by setting the gradient change threshold between each layer, regularization the weight, and replacing sigmoid and other activation function with relu activation function.

In addition, due to the difficulty in fitting identity maps, the fitting performance of deep neural networks may be worse than shallow networks in some cases, which is known as network degradation. K He et al.[10] demonstrated this through experiments and proposed a residual neural network (Res-Net), which avoids this problem through the skip connection structure, ensuring the ability of DNN to better fit with increasing layers.

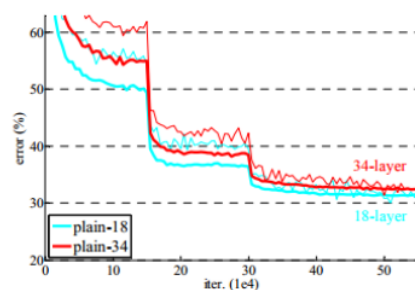


Figure 1 – The error rate change trend comparison between 18 layer network and 34 layer network with the increase of iteration number[10], It can be seen that the 34-layer's error rate is higher than 18-layer

DNN has a wide range of applications. Taking the research of O Kudashev et al. [11] as an example, they applied DNN to speaker recognition tasks, extracting voice features through DNN and modeling them to obtain the speaker's identity. In addition, DNN also has applications in text processing, image recognition, and other fields. But DNN also has its drawback, which is that the architecture can only handle fixed size sequence data. To address this issue, various neural network architectures have been proposed for processing different types of data, such as GNN for processing graph type data [12] and RNN for processing sequence data with pre and post correlation [13]. And CNN is one of them.

2.2 CNN

Convolutional Neural Network (CNN) is a type of neural network commonly used for processing 2D or higher dimensional data such as images. It is often used to execute tasks such as pattern recognition of images[14]. DNN is often used to process fixed size sequence data, and when it is used to process higher-dimensional data, these data are usually transformed to a sequence so they can be learned by DNN. This approach increases the computational complexity of DNN and disrupts the correlation between each adjacent data item in high-dimensional data. To solve this problem, CNN was proposed. The main difference between CNN and DNN is its ability to extract features from high-dimensional data, which preserves the relationships between data items while compressing input data, thereby reducing computing complexity. The following will introduce CNN based on image processing as a pre-condition.

CNN consists of three parts: convolutional layer, pooling layer, and fully connected layer. Convolutional layer is responsible for extracting features from images. It convolves the 2D matrix from the previous layer through convolution kernel, and passes the generated feature map to the next layer. The weight values in the convolution kernel can be trained, which enables the convolutional layer to extract image features in a more reasonable way. The Weight Sharing refers to the use of the same convolutional kernel for a 2D matrix in each layer during the convolution process. It is a way for convolutional layers to improve its efficiency while accelerating network convergence speed [15]. The Receptive Field refers to the size of the area one pixel of feature image corresponding to on the input original image. Generally, the larger the Receptive field of the feature map, the higher the degree of information retention for the original image. Researchers have proposed many ways to expand the Receptive field, and the Diluted Convolution is one of them [16]. It replaces the densely distributed convolution kernel with a sparsely distributed one, in order to expand the convolution range of the convolution kernel without increasing the training time too much.

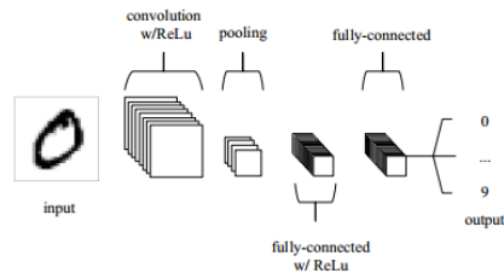


Figure 2 - The architecture of CNN[13]

The role of pooling layer is to compress feature maps while preserving as much information as possible. It compresses data by categorizing data from a certain region into a single data item. There are many ways to pool, such as the max- pooling that takes the max value of the region and average-pooling that takes the average value of a region. H Touvron et al. [17] used an attention based pooling method in their research, which uses a simple convolutional network to learn pooling operations and achieves the goal of maximizing information retention through weighted summation. The fully connected layer is responsible for receiving feature vectors from the pooling layer and outputting the desired results. Its architecture is like MLP.

CNN will also face the problem of overfitting. To solve this issue, the concept of drop-out layer has been proposed. Its key idea is to randomly select neurons with a certain probability to change their output to 0 during the training process. which means these neurons are not trained in this epoch round. N Srivastava et al. [18] proved through experiments that this training method significantly improves the generalization ability of CNN, enabling it to achieve better testing results after learning some medium to large datasets.

Under the same conditions, CNN generally outperforms DNN in performing tasks related to image data, as demonstrated by J Tan et al. [19] in comparing the accuracy of CNN and DNN in medical image detection tasks. CNN has developed into many improved variants, such as AlexNet, GoogleNet, ZFNet, and so on[20]. Its application range is also becoming increasingly widespread, not only for processing 2D data such as images, but also for processing 3D or higher dimensional data such as mesh models.

3.0 Method

3.1 Dataset

The benchmark dataset used in this experiment is FER-2013, which is one of the benchmarks for comparing facial expression recognition models. This dataset consists of 35887 facial grayscale jpg images with a size of 48 * 48, including 28709 training data, 3589 public test data, and 3589 private test data. The dataset contains 7 types of data, namely angry, disgust, fear, happy, sad, surprise, and neutral. Each type of data's number is different. The data source of this dataset is the results obtained from Google Image Engine's search for emotional keywords, so there is a certain degree of error label in the dataset. Due to significant differences in pose, age, and occlusion conditions among the faces in this dataset, it is highly challenging [21].



Figure 7 –some example of FER-2013

3.2 Architecture

The experiment use a simple CNN architecture to learn the dataset and test its learning ability. The original training set is divided into training set and verification set. The verification set includes the first 1000 items of data in the original training set, which is used to verify the learning result of the current architecture, and adjust its hyperparameter according to the accuracy of CNN's classification of verification set data. The training set includes the remaining data except for the validation set, which is used to train CNN.

CNN has two basic parameters: the size of the convolutional kernel and the number of filters. The former determines the size of the region for a convolution operation, while the latter determines the number of channels for the new feature map output by the convolutional layer after convolution. According to Abhinav et al.'s experiment [22], a combination of too large kernel size and too few filters can make CNN training unstable. Meanwhile, considering the efficiency of training, the feature extraction part of the neural network is divided into seven layers, including five convolutional layers and two pooling layers, in the order of 2 convolutional layers -1 pooling layer -3 convolutional layers -1 pooling layer. The filters number for the first convolutional layer is set to 8, the second is set to 16, the third is set to 32, the fourth is set to 64, and the fifth is set to 128. This setting method conforms to the architecture of traditional CNN, where the length and width of each layer's feature map gradually decrease while the number of features gradually increases. The kernel size is uniformly set to 4, except for the last layer which is set to 3. The step size of the convolutional movement is set to 1, so that each layer of the feature map contains more information. The activation function is set to relu, because it will not cause the problem of gradient vanishing. Do not fill the edges with 0 padding after each convolution to recover to the original image size, as this will result in a large number of invalid calculations in the training process and reduce computational efficiency. The pooling layer adopts max pooling, which has been proven to be excellent in multiple studies related to neural networks. The size of pooling is 4, and the step size is set to 4. At the same time, the batch normalization operation is added in front of each pooling layer, which pulls the distribution of feature value of this layer back to the standard normal distribution, and the feature value will fall in the area where the activation function is more sensitive to input. Small changes in input can lead to large changes in the loss function, making the gradient larger and avoiding gradient vanishing. It can also accelerating convergence. Finally, a $48 * 48 * 1$ size original image will be converted into a $6 * 6 * 48$ feature vector. Perform a one-dimensional operation on the feature vector and input this quantity into the fully connected layer.

The fully connected layer is divided into three layers, with 64 neurons in the first layer, 32 neurons in the second layer, and 16 neurons in the third layer. The activation function is relu. Finally, a drop out layer is added to improve the generalization ability of the neural network. In the fully connected layer, it will randomly select neurons with a probability of 0.5 to not be trained in this iteration. Finally, the output is normalized through the softmax function and outputs seven values, which represent the probability that the data belongs to seven image classes.

Kaiming initialization is adopted for weight initialization, which ensures that the initialization parameters of each layer present a Gaussian distribution when using the relu activation function. This initialization method can help suppress gradient explosion or vanishing. The optimizer uses SGD, which provides the best optimization performance compared to other optimizers, as demonstrated in the experiment [22]. The learning rate of the optimizer has attempted to use an exponentially decreasing method, However, after testing, it was found that the convergence speed of the neural network under this method is too slow and the effect is not good. Therefore, a fixed value of 0.01 was finally used as the learning rate. The loss function uses sparse categorical cross entropy, which is suitable for

multi-classification neural network models with number label. Due to insufficient computing power of the my computer and considering the experimental time, CNN stopped training after 60 epochs. But this also leads to a low accuracy of the training results. Preserved The model with the smallest loss value during the iteration process through checkpoint. Finally, in order to shorten the training time, the batch size of the model is set to 16, which means that the neural network will batch process 16 data items at once.

The comparison of training accuracy and validation accuracy changes during the training process under this architecture is as follows.



Figure 9 –The Training and Validation accuracy and loss value change trend comparasion

It can be seen that as the number of iterations increases, the training accuracy increases, but the validation accuracy shows a trend of continuous increase but becoming smoother. The former proves that the fitting result of the neural network on the training set will be better and better with the learning process going on, while the latter may be caused by overfitting in the late training period. This can also be seen from the changes in the loss value. The change in validation loss shows a trend of first decreasing and then increasing, which may be due to the large difference between various types of data's amount in the training set, resulting in poor or even regression of the classifier's learning performance for relatively small number of image types.

When the fully connected layer has one, two, and three layers, the comparison of the accuracy changes during the training process is as follows. For faster testing speed, only 20 epochs were conducted.

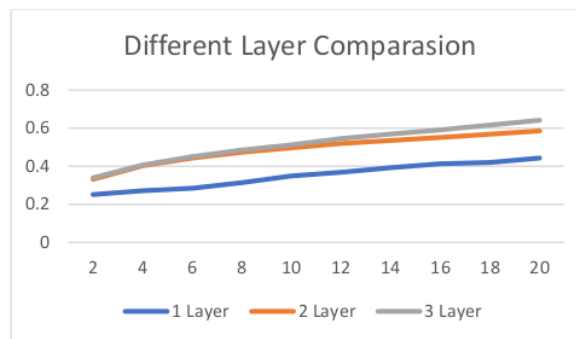


Figure 9 –Different Layer Comparasion

From the figure, it can be seen that as the number of fully connected layers increases, the convergence speed of the network becomes faster and faster. This is because deeper fully connected layers can represent more complex functions, enabling faster learning of classification tasks. Meanwhile, due to the use of drop out layers and the fact that the number of layers is not very deep, the impact of gradient vanishing can be almost ignored.

When the activation function used is respectively relu and sigmoid, the training process is compared as follows. For faster testing speed, only 20 epochs were conducted.

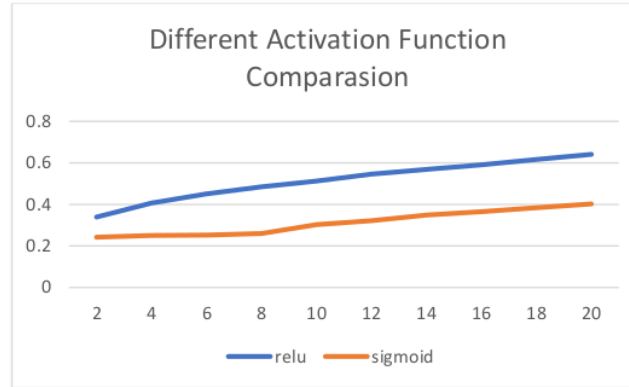


Figure 9 –Different Activation Function Comparasion

It can be seen that the effect of relu activation function is better than that of sigmoid. This is because the output gradient of the relu function is always 1 when the input is greater than 0, which suppresses the gradient vanishing problem.

4.0 Evaluation

The model with the highest accuracy of the verification set is used for testing. The confusion matrix, recall rate, precision and F1 Score results are shown in the following table.

predict \ real	Angry	Disgust	Fear	Happy	Neutral	Sad	Surprise
Angry	231	0	97	153	45	395	37
Disgust	40	0	15	11	2	41	2
Fear	117	0	142	186	73	356	150
Happy	25	0	38	1465	31	173	42
Neutral	52	0	81	258	341	462	39
Sad	95	0	82	227	129	686	28
Surprise	24	0	82	84	8	69	564

Table 1 –Confusion matrix of prediction result

	Precision	Recall	F1-score	Support
Angry	0.396	0.241	0.300	958
Disgust	0.000	0.000	0.000	111
Fear	0.264	0.139	0.182	1024
Happy	0.615	0.826	0.705	1774
Neutral	0.542	0.277	0.366	1233
Sad	0.314	0.550	0.400	1247
Surprise	0.654	0.679	0.666	831

Table 2 –Confusion matrix of prediction result

The overall accuracy is 0.478.

There is a very strange phenomenon in the above table, which is that no images are classified as Disgust. After modifying the hyperparameter of the neural network, the problem has still not been solved. By examining the classification probabilities of some samples, it was found that the classification probability output of the Disgust class was not 0, but it was much lower than the output probabilities of other classes. Therefore, it is speculated that this may be due to the low proportion of the data volume of the Disgust class in the training set. At the same time, through viewing the dataset, it is found that the Disgust type facial expression picture are relatively close to Angry or Sad, and they do not have distinctive features different from these two types of pictures, which can also be seen in the confusion matrix. Therefore, this results in images of the Disgust class being classified by the classifier into other classes. This also indicates that the current CNN architecture has poor ability in learning image details



Figure 11 – comparasion between disgust(left), sad(center), and angry(right) image example. It can be seen that tuesday are very similar

In addition, the Angry class is very similar to the Sad class in features, so the classifier is easy to confuse these two types of images, as can be seen in the confusion matrix. However, the proportion of Sad image is higher than that of Angry in training set, so the classifier tend to classify the images with this feature as Sad. The Fear class also has similar feature as Sad, so its Precision is also not very high. The Happy class has a high F1-Score due to its large number of data samples and distinct features, but images from other classes are also easily classified into the Happy class. The Surprise class has the highest Precision due to its most distinctive features. Due to the lack of obvious facial features, Neutral images are easily classified into other classes, as can be seen from their Recall value.

Overall, from the experimental results, it can be seen that CNN does have the ability to learn image features, but at the same time, it exposes two problems with the current CNN architecture: firstly, it is easily affected by the uneven proportion of various types of data in the training set, and tend to classify images with similar features into types with a higher data proportion in training set. Secondly, the ability to distinguish small features is still poor. The former may need to be solved by modifying the architecture, while the latter may be due to insufficient training time and insufficient depth of the neural network.

5.0 Conclusion

The theme of this report is an experiment on CNN. Firstly, the sources, basic knowledge, and related applications of DNN and CNN were introduced. Then, a CNN architecture was designed and trained to learn the FER2013 dataset. To achieve better training result, the hyper-parameters of this architecture are adjusted according to the validation result. Finally, the experiment use the test set to test the effectiveness of the architecture. Due to insufficient training epoch, the framework did not achieve a good training result.

6.0 Appendix: Code

```
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Conv2D, MaxPool2D, Activation, Dropout, Flatten, Dense
import random
import pathlib
import numpy as np
from sklearn import metrics
from matplotlib import pyplot as plt

def load_and_preprocess_from_path_label(path, label):
    image = tf.io.read_file(path) # read image
    image = tf.image.decode_jpeg(image, channels=1)
    image = tf.image.resize(image, [48, 48]) # reset the size of image
    #image /= 255.0 # reset the gray range
    return image, label

def load_path_get_image_and_label_set(path):
    data_path = pathlib.Path(path)
    all_image_paths = list(data_path.glob('*/*'))
    all_image_paths = [str(path) for path in all_image_paths] # the path of all image
    random.shuffle(all_image_paths) # random shuffle for better train
    image_count = len(all_image_paths)

    label_names = sorted(item.name for item in data_path.glob('*/*') if item.is_dir())# get the name of 7
types
    label_to_index = dict((name, index) for index, name in enumerate(label_names))#give each type a
index
    all_image_labels = [label_to_index[pathlib.Path(path).parent.name] for path in
all_image_paths]#attach index to image
    ds = tf.data.Dataset.from_tensor_slices((all_image_paths, all_image_labels))# transfer to dataset
    image_label_ds = ds.map(load_and_preprocess_from_path_label)
```

```

image_set = []#split them to image and label for train
label_set = []
for image, label in zip(all_image_paths, all_image_labels):
    r_image, r_label = load_and_preprocess_from_path_label(image, label)
    image_set.append(r_image)
    label_set.append(r_label)
returned_images = np.array(image_set)
returned_labels = np.array(label_set)
return returned_images, returned_labels

path = 'C:/Users/86139/Desktop/assignment(second semester)/ML/Assignment2/dataset/dataset/train';
[train_image, train_label] = load_path_get_image_and_label_set(path)
valid_image = train_image[:1000]
valid_label = train_label[:1000]
train_image = train_image[1000:]
train_label = train_label[1000:]

path = 'C:/Users/86139/Desktop/assignment(second semester)/ML/Assignment2/dataset/dataset/test';
[test_image, test_label] = load_path_get_image_and_label_set(path)
train_image = train_image/255.0
valid_image = valid_image/255.0
test_image = test_image/255.0

#build the model
model = Sequential()
initializer = tf.keras.initializers.HeNormal()
#first Layer
model.add(Conv2D(input_shape=(48, 48, 1), filters=8, kernel_size=4, padding='valid',
activation='relu', kernel_initializer=initializer))
model.add(Conv2D(filters=16, kernel_size=4, padding='valid',
activation='relu', kernel_initializer=initializer))
model.add(tf.keras.layers.BatchNormalization())

```

```

model.add(MaxPool2D(pool_size=2, strides=2))

# second Layer

model.add(Conv2D(filters=32, kernel_size=4, padding='valid',
activation='relu',kernel_initializer=initializer))

model.add(Conv2D(filters=64, kernel_size=4, padding='valid',
activation='relu',kernel_initializer=initializer))

model.add(Conv2D(filters=128, kernel_size=3, padding='valid',
activation='relu',kernel_initializer=initializer))

model.add(tf.keras.layers.BatchNormalization())

model.add(MaxPool2D(pool_size=2, strides=2))


model.add(Flatten())


# full connection layer

model.add(Dense(64, activation='relu',kernel_initializer=initializer))
model.add(Dense(32, activation='relu',kernel_initializer=initializer))
model.add(Dense(16, activation='relu',kernel_initializer=initializer ))
model.add(Dropout(0.5))
model.add(Dense(7, activation='softmax'))


#lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(initial_learning_rate=0.01)


model.compile(optimizer=tf.keras.optimizers.SGD( ),
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
              metrics=[tf.keras.metrics.sparse_categorical_accuracy])


checkpoint = tf.keras.callbacks.ModelCheckpoint('/model_save/best.h5', monitor='val_loss',
verbose=0,

              save_best_only=True,
              save_weights_only=False, mode='min',
              period=1)

callback_list = [checkpoint]

```

```

history = model.fit(train_image,train_label, epochs = 25,
batch_size=16,validation_data=(valid_image,valid_label),validation_freq=1,callbacks =
callback_list)

model.summary()

#show the training process

acc = history.history['sparse_categorical_accuracy']
val_acc = history.history['val_sparse_categorical_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
print(acc)
print(val_loss)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(loss, label='Training loss')
plt.plot(val_loss, label='Validation loss')
plt.title('Training and Validation loss')
plt.legend()
plt.show()

model = keras.models.load_model('./model_save/best.h5')
score = model.evaluate(test_image, test_label)
print(score)
predict_res = model.predict(test_image)
predict_res = np.argmax(predict_res,1)

```

```
#result
Confuse = metrics.confusion_matrix(test_label,predict_res)
report = metrics.classification_report(test_label,predict_res,digits=3);
print(Confuse)
print(report)
print(metrics.accuracy_score(test_label,predict_res))

#output
output = open('confuse_res.xls', 'w', encoding='gbk')
output.write('confuse_res\n')
for i in range(len(Confuse)):
    for j in range (len(Confuse[i])):
        output.write(str(Confuse[i][j]))
        output.write('\t')
    output.write('\n')
output.close()
```

Bibliography

- [1] S. J. Russell, P. Norvig, and E. Davis, *Artificial Intelligence: A modern approach*. Harlow, England: Pearson Educación, 2022.
- [2] C. M. Bishop, *Pattern recognition and machine learning*. New York, NY: Springer New York, 2016.
- [3] Carbonell J G, Song D, and Philipp G, "The exploding gradient problem demystified-definition, prevalence, impact, origin, tradeoffs, and solutions," *arXiv preprint*, 2017.
- [4] "A comparison study on the effect of false positive reduction in deep learning based detection for juxtapleural lung nodules: CNN vs DNN," *Modeling and Simulation in Medicine Symposium (MSM 2017)*, 2017.
- [5] F. Scarselli, M. Gori, Ah Chung Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.
- [6] Gu J, Wang Z, Kuen J, and et al, "Recent advances in convolutional neural networks," *Pattern recognition*, vol. 77, pp. 354–377, 2018.
- [7] H. Guo, S. Peng, H. Lin, Q. Wang, G. Zhang, H. Bao, and X. Zhou, "Neural 3D scene reconstruction with the Manhattan-World Assumption," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [8] H. N. Mhaskar and T. Poggio, "Deep vs. shallow networks: An approximation theory perspective," *Analysis and Applications*, vol. 14, no. 06, pp. 829–848, 2016.
- [9] J. Kolbusz, P. Rozycki, and B. M. Wilamowski, "The study of architecture MLP with linear neurons in order to eliminate the 'vanishing gradient' problem," *Artificial Intelligence and Soft Computing*, pp. 97–106, 2017.
- [10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [11] N. Milosevic, "Convolutions and Convolutional Neural Networks," *Introduction to Convolutional Neural Networks*, 2020.
- [12] O. Kudashev, S. Novoselov, T. Pekhovsky, K. Simonchik, and G. Lavrentyeva, "Usage of DNN in speaker recognition: Advantages and problems," *Advances in Neural Networks – ISNN 2016*, pp. 82–91, 2016.
- [13] P. Giannopoulos, I. Perikos, and I. Hatzilygeroudis, "Deep learning approaches for facial emotion recognition: A case study on fer-2013," *Advances in Hybridization of Intelligent Methods*, pp. 1–16, 2017.
- [14] Pham H, Guan M, Zoph B, and et al, "International conference on machine learning. PMLR," 2018, pp. 4095–4104.

- [15] R. C. Eberhart and R. W. Dobbins, "Early Neural Network Development history: The age of camelot," *IEEE Engineering in Medicine and Biology Magazine*, vol. 9, no. 3, pp. 15–18, 1990.
- [16] S. J. Russell, P. Norvig, and E. Davis, *Artificial Intelligence: A modern approach*. Harlow, England: Pearson Educación, 2022.
- [17] Srivastava N, Hinton G, Krizhevsky A, and et al, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.
- [18] T. Qiao, J. Zhang, D. Xu, and D. Tao, "Mirrorgan: Learning text-to-image generation by redescription," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [19] JTouvron H, Cord M, El-Nouby A, and et al, "Augmenting convolutional networks with attention-based aggregation," *arXiv preprint*, 2021.
- [20] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of Recurrent Neural Networks: LSTM cells and network architectures," *Neural Computation*, vol. 31, no. 7, pp. 1235–1270, 2019.
- [21] Yu F and Koltun V, "Multi-scale context aggregation by dilated convolutions[J]," *arXiv preprint*, 2015.
- [22] A. Agrawal and N. Mittal, "Using CNN for Facial Expression Recognition: A study of the effects of kernel size and number of filters on accuracy," *The Visual Computer*, vol. 36, no. 2, pp. 405–412, 2019.

Machine Learning 2

GRADEMARK REPORT

FINAL GRADE

69/100

GENERAL COMMENTS

Instructor

Please include an abstract.

You use italics throughout for your report. Avoid this. In general use italics only for emphasis.

You review related concepts and theory well and provide in-line citations. Good. You code improve on this review of related work by identifying authors who undertake similar work and/or use the same dataset.

You introduce the dataset in 3.1. Good but please review its pros. and cons. here, although I note you discuss the limitations of the dataset later on.

Your experimental design/method identifies and justifies parameters used to define your architecture, etc. Good. You also explain that you modified your architecture to improve results and document effects of these modifications. Good. It is often helpful to summarize these experiments in a table or using bar plot graphs (preferred).

A confusion matrix is provided and analysed well. Good. You refer back to the image classes and reasonable suggestions are provided for poor classification/misclassifications. Again, good.

Regarding structure try placing the above in a section called "Method and Experiments" as this better reflects your content, here.

Overall a good effort but please take note of the comments above.

PAGE 1

PAGE 2

PAGE 3

PAGE 4



Comment 1

stagnate

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

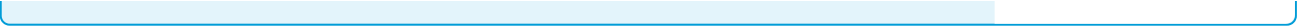
PAGE 16

PAGE 17

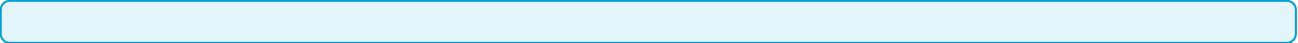
GRADING FORM: SMU GENERIC_GJ

XUEYANG HU

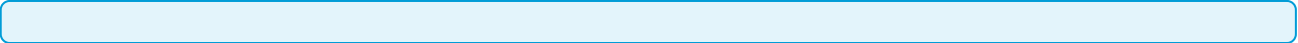
0



RESEARCH-INFORMED LITERATURE



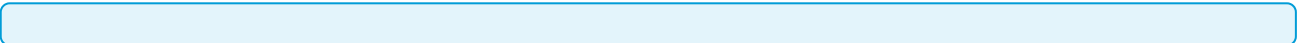
KNOWLEDGE AND UNDERSTANDING



ANALYSIS



PRACTICAL APPLICATION AND DEP.



SKILLS FOR PROFESSIONAL PRAC.

