Image classification by Convolutional Neural Network with two-stage training

Yi Zhou
Advisor: Professor Qiang Wu

Abstract:
Convolutional neural networks are an important class of deep learning. It plays a dominant role in computer vision tasks, especially in image classification, object detection, image recognition, etc. In this article, we will focus on the image classification of CNN. Furthermore, in previous literature, it was shown that the two-stage algorithms of feed-forward neural networks (FNNs) and Long Short-Term Memory (LSTM) are more powerful than their one-stage counterpart, but CNNs were not explored. In this work, we will build different CNN architectures based on different datasets. The main idea of the two-stage algorithms is to extract features from the pre-trained model and retrain a new model with only the same last layer as the previous model. We will explore whether two-stage algorithms are more effective than one-stage training. The result shows the degree of accuracy improvement of the two-stage algorithm is related to the volatility of the pre-trained model in different image datasets.
Keywords: Convolutional neural networks, image classification, data augmentation, fashion-MNIST, cifar10, dogs & cats, face mask detection.

1. INTRODUCTION

In recent years, computer vision as a part of artificial intelligence (AI) has developed rapidly. Computer vision includes object detection, image generation, semantic segmentation, video analysis, etc. They enable computers and systems to obtain corresponding information from the visual data for learning and understanding to simulate human behaviors. Image classification discussed in this article is a fundamental task in the field of computer vision, it has great significance in our life. The purpose of image classification is mainly to classify or label images using established categories based on the content of image datasets. It involves the procedures such as preprocessing images, extracting features, and recognizing matches, and the built model will learn from the training data to distinguish between different classes. Image classification is used in many fields, including scene classification in self-driving cars, medical imaging applications and face recognition, and other fields [1]-[3]. For example, the accuracy improvement of image classification in self-driving cars can help the car better understand the traffic scene, including traffic lights, road signs, and markings, so it can navigate more safely. In addition, in the field of medical imaging applications, image classification can better help identify the diseases for some diseases that require X-rays, CT scans, and so on.

The convolutional neural network is widely used in image classification. It can automatically extract features from images and has relatively fewer parameters to be trained, which has advantages over other traditional machine learning methods. The architecture development of CNN has undergone great changes, from a simple convolutional neural network to a deep convolutional neural network. The first CNN architecture - LeNet-5 was proposed by LeCun Y in 1998 and applied to MNIST datasets [4]. The LeNet-5 has a total of five learned layers, two convolutional layers, and three fully connected layers. Due to the influence of system hardware devices, there is no GPU acceleration. Compared with traditional algorithms, the efficiency is not well much. With the development of GPU, the AlexNet was proposed by Krizhevsky in 2012 [5].

The AlexNet has a total of eight learned layers and uses two GPUs to accelerate, and Krizhevsky proposed some creative methods such as the ReLU activation method to reduce the problem of gradient vanishing, data augmentation, and dropout to reduce data overfitting. In addition, VGG was proposed in 2014 as one of the representatives of deep convolutional neural networks [6]. After that, deep CNN developed rapidly, such as GoogLeNet [7], ResNet [8], Xception [9], and so on. Overall, CNN has shown significant success in various image tasks. However, it is important to research and explore CNN's ability to maintain accurate predictions under various circumstances. To ensure the dependability and practicality of CNN models, robustness must be guaranteed.

In general, the robustness of CNN models can be affected by some factors. For example, when there is noise in the data, the data quality is low, the amount of dataset is small, or the distribution of data classes is imbalanced, these can affect the robustness of the CNN. Hence, it is essential to enhance the robustness of CNN. In this literature [10], the authors studied how robust loss works in the construction of the neural network. It is shown that two-stage algorithms are always more powerful than one-stage approaches in feed-forward neural network (FNN) and the Long Short-Term Memory (LSTM) for regression analysis but didn't consider the study of robust CNN.

In this article, we will study how robust accuracy plays a role in the convolutional neural network. First, we propose a simple convolutional neural network for image classification. Based on this convolutional neural network, we design a two-stage algorithm and implement a comparative study on four real-world applications. The results indicate two findings: (1) When the dataset is small and the model fitting degree fluctuates greatly during the data training process, the two-stage algorithm can improve the robustness of the model. (2) When the dataset is large, the data quality is high, and the volatility of the model is stable during the data training process, although the effect of the two-stage algorithm on improving the accuracy of image classification is not obvious, it also makes the model more robust.

2. CONVOLUTIONAL NEURAL NETWORK

This section mainly analyzes the role of each layer of the structure and methods used in the CNN model in detail. CNN has several layers, mainly convolutional, pooling, and fully connected layers (see Figure 1). Moreover, there are also other techniques used in the construction of convolutional neural network to improve the model's robustness.

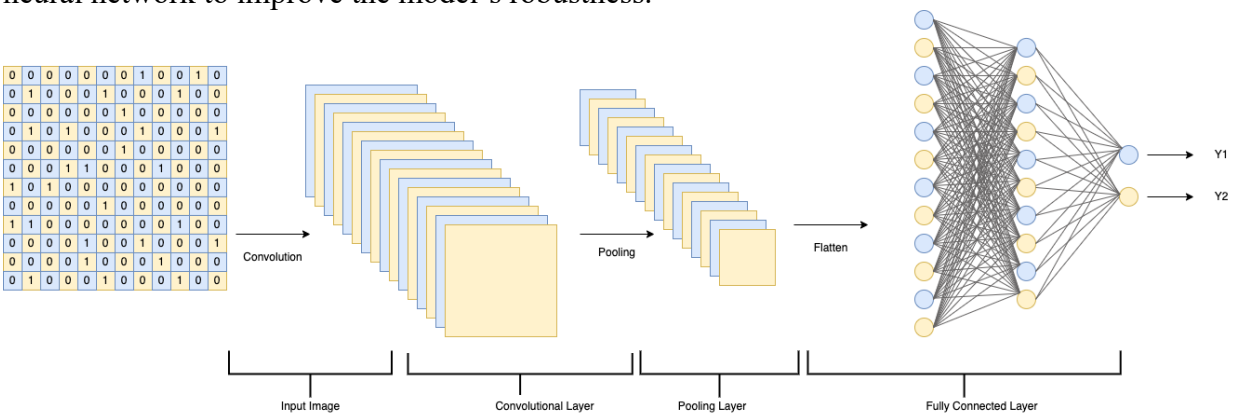


Figure 1. The network structure of convolutional neural network

2.1 Convolutional layer

The convolutional layer is the most important layer for building a convolutional neural network. The purpose of this layer is to sequentially convolve the local receptive field from the left top to the right button in the input image according to the convolution kernel (filter) matrix which is used to extract features from the images. In the whole network, it produces most part of the computations compared with other layers.

The input of each layer in the convolutional neural network has three dimensions: width, height, and depth. The values for width and height are usually the same, and each convolution kernel (filter) is relatively small in space (width and height), generally set to 3×3, or 5×5, but the depth is consistent with the input depth. Each convolutional layer will have multiple depth slices, but the neurons in each depth slice use the same kernel weights. Hence, the total number of weights is the multiplication of the number of depth slices and the kernel size, and the number of channels, which can decrease the number of parameters in the network. This is the concept of parameter sharing [4]. When the kernel scans the entire input, it generates a feature map each time, and all feature maps are stacked to form the entire output result. From Figure 2, we can figure out how the kernel computes based on the input data. The kernel weights are updated through learning during the process of model training, mainly during backpropagation.

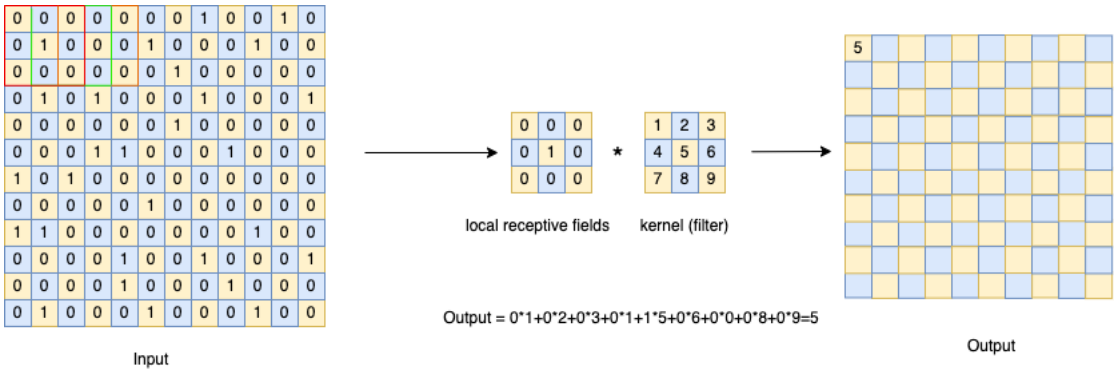


Figure 2. The operation of single depth slice in a convolutional layer

If the stride is set, then the horizontal and vertical local receptive field movement steps will be adjusted according to the set stride value. For example, when the stride is one, then the filter moves one pixel at a time, and other values are the same. In practice, the stride is generally set to one or two. The size of the zero-padding is also a factor to consider. Let's assume the input size is $W \times W$, the kernel (filter) size is $F \times F$, the stride is S, and the amount of zero-padding is P, then we can obtain the output size by the given $(W - F + 2P) / S + 1$ formula.

In the convolutional layer, in addition to the kernel, there is another important part, which is the nonlinear activation function. The linear activation function is not effective for multi-layer networks, and many problems are nonlinear in the real world. The nonlinear activation function can better handle and fit various complex data in CNN. After the convolution operation, the feature maps will be passed through a nonlinear activation function. There are some common activation functions such as Sigmoid, Tanh, and ReLU (see Equation (1)). The most used function is ReLU

in deep neural networks because ReLU can converge faster and does not suffer from the problem of gradient vanishing.

$$\begin{cases} \text{Sigmoid: } f(x) = \frac{1}{1+e^{-x}} \\ \text{Tanh: } f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \\ \text{ReLU: } f(x) = \max\{0, x\} \end{cases} \tag{1}$$

2.2 Pooling layer

The pooling layer is generally placed between successive convolutional layers. Its main purpose is to reduce the spatial dimensions (width and height) of the feature maps from the convolutional layer, but the depth is kept the same. This can help to decrease the number of parameters in the network and lower the computational resource consumption.

The pooling layer is to downsample upon feature maps, which can reduce the amount of data while retaining useful information. There are two common types of pooling: Max Pooling and Average Pooling. The Max Pooling is calculating the max value from each region of the feature map, that is, the area covered by the kernel (filter). The kernel size is always smaller than the size of the feature map, it is almost set to 2×2. Through sliding the kernel, all max values are aggregated to decrease the size of the feature map. The Average Pooling is calculating the average value from each region of the feature map covered by the kernel. In general, the effect of Max Pooling is better than the Average Pooling. The Max Pooling can keep more texture information while the Average Pooling keeps more background information. We use the former more than the latter in image classification.

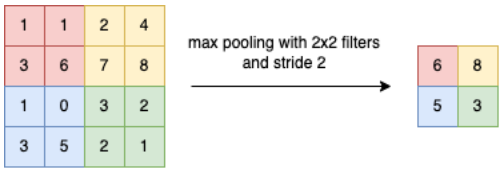


Figure 3. The Max Pooling operation of single depth slice

2.3 Fully connected layer

The fully connected (FC) layers are the last part of the convolutional neural network architecture, which includes multiple dense layers which form a feed-forward neural network. The purpose of this part is to convert the high-level feature maps from the final convolutional layer or pooling layer into category outputs. The feature maps will initially be flattened in this procedure, expanding all their values into a 1D vector to better feed the data into the feed-forward neural network. This also means that all operations in the fully connected layer can be calculated by the matrix.

There are usually two types of activation functions used in this part, ReLU for the hidden dense layer and SoftMax for the last classification dense layer. For details on ReLU, see the introduction in the convolutional layer section. The SoftMax function is used in the multiple classes image classification problem to convert the raw output values from the last dense layer to category probabilities. The outputs of SoftMax are interrelated and the range of each value is from 0 to 1 and all values add up to 1. The formula is as follows Equation (2). For example, raw values are: -

0.3, 1.2, -0.6, 2.4, then after being passed to SoftMax activation the values are: 0.047, 0.212, 0.035, 0.705.

$$S(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{N} e^{y_j}} \tag{2}$$

2.4 Batch Normalization layer

During the training process, the parameters of each layer adjust causing the input of the latter layer to adjust, so it also changes the distribution of each batch of training data, which also can be said as a covariate shift. In each iteration, the network has to fit different parameters. The various distribution of data increases the training complexity and raise the possibility of overfitting.

In convolutional neural networks, batch normalization is a standard technique that can be applied in forward and backward propagation due to its differentiability, and it can help address the "internal covariate shift" problem [11]. The Batch Normalization layer is usually inserted between the convolutional and pooling layers and is also used in the fully connected layer. It can effectively improve the robustness of bad initialization, enhance network performance, and speed up the convergence of the network by normalizing the input to the next layer.

2.5 Dropout layer

The Dropout layer is a simple and effective regularization method for convolutional neural networks to reduce data overfitting. Its working principle is to randomly discard a portion of the neurons by a set probability while retaining the functions of the remaining neurons, and the discarded neurons are not considered in the forward and backward propagation. In the next iteration, randomly discard some neurons again until the end of training. As a result, the network needs to learn more robust features of unseen data and the neurons are not excessively dependent on specific neurons. This can reduce training data overfitting problems and improve the generalization of the networks.

2.6 Model Compilation

Model Compilation is an operation after defining the architecture of convolutional neural networks. It includes defining the loss function, optimizer, and metrics.

The loss function is to measure the performance of the model through the difference between the predicted output and the true labels. The cross-entropy loss function is commonly used in image classification, which includes categorical cross-entropy loss and sparse categorical cross-entropy loss for multi-class classification and binary cross-entropy loss for binary classification. Categorical cross-entropy loss is the most used. Its principle is to combine the SoftMax activation function and cross-entropy loss.

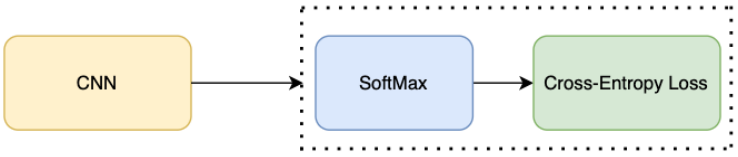


Figure 3. The workflow of model compilation

When we use categorical cross-entropy loss, it required the true labels to be one-hot before model compilation, which means the model only considers the term for the positive class and omits the term for the negative class. The formula of cross-entropy loss is Equation (3). After discarding the target labels that cause the summation to be zero, the formula can be written as Equation (4).

$$CE = -\sum_i^C T_i \cdot \log\left(S(y_i)\right) \tag{3}$$

$$CE = -\log\left(S(y_p)\right) \tag{4}$$

where C is the total number of classes, T is the set of targets, S is the SoftMax output, and y_p denotes the positive class.

The optimizer is to determine how to update the weights of the network during training the model to minimize the loss function. There are several common optimizers such as Adam, SGD with momentum, RMSprop, etc. In image classification tasks, the Adam optimizer is used widely because it provides faster convergence, lower memory requirements, and better performance since it can tune the learning rate for each parameter separately.

The Adam optimizer formula can be represented from Equation (5) to Equation (10) [12], these procedures will be repeated until θ_t converged.

$$\theta_t = \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t + \epsilon}) \tag{5}$$

$$\hat{m}_t = m_t / (1 - \beta_1^t) \tag{6}$$

$$\hat{v}_t = v_t / (1 - \beta_2^t) \tag{7}$$

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \tag{8}$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \tag{9}$$

$$g_t = \nabla_\theta f_t(\theta_{t-1}) \tag{10}$$

In the above formulas, equation (5) is the basic version that updates the parameters, Equation (6) and Equation (7) compute the estimation of the first-order moment and second-order raw moment of bias-corrected, Equation (8) and Equation (9) update the estimation of the first-order moment and second-order raw moment of the bias-correction, and Equation (10) computes gradients. α is the learning rate, β_1 and β_2 are the decay rates for the moment estimates.

The metrics are used to evaluate the performance of the model during training and testing. In this article, we mainly use the Keras module to train and test the model. There are some metrics available in Keras such as accuracy, precision, F1 score, AUC, Mean Absolute Error (MAE), etc. Since accuracy represents the proportion of samples that are correctly predicted to all samples, we simply utilize it as a measurement when training the model.

3. TWO-STAGE TRAINING METHOD

Robustness, which is essential for real-world application, refers to the model's ability to perform well or generalize effectively. In the introduction section, we have briefly introduced some factors that affect the robustness of the convolutional neural network. In this section, we will explain in detail a few typical influencing factors and how they affect the model. (1) Data quantity. It refers to the volume of the dataset. If there is more data, it can help train more complex and robust models, but more data also means more time to process. When the data quantity exceeds a certain point, the ability to improve the model performance is limited. (2) Data quality. It generally refers to

accuracy, relevance, consistency, and integrity of data. If the data quality is low, it will influence the model to learn more generalized patterns, which may cause some errors or biases when training the model, and data overfitting may occur. (3) Class distribution. It refers to the volume of data of different classes. If the class distribution of the data is unbalanced, it will also affect the model learning from the unseen data, however, in this article, the class distribution is balanced. (4) Noisy data. If the data has noise and outliers, it may skew the model to learn and affect its robustness which causes the model to overfit. (5) Data Augmentation. This method can make the model more robust by increasing the variety of the data when the dataset is small. In addition to these, there are also other factors, which will not be listed one by one. It's important to consider these factors during data preprocessing, and model training. Furthermore, designing a robust algorithm might also improve the model's performance and generalization abilities.

In the previous section, we introduced the architecture of the convolutional neural network. Its main idea is to extract the local patterns and features from the input data, and after some processing, flatten the high-level feature maps and classify them through the last fully connected layer which is also called the SoftMax layer. Hence, these procedures can be split into two parts: a feature extraction part and a classification decision part, but these two parts are as a whole and trained together in the model. Our idea is to extract the features from the already-trained model after the entire training of these two parts.

In the previous literature [10], the authors have proven two-stage algorithm can improve the robustness and reduce the loss of FNN and LSTM models used on the data contaminated by outliers by using different regression methods in the second part which is called "a function evaluation part". However, what the authors used is the data, the images are different from the intuitive data, it needs to extract and refine the information, and then convert it into data for analysis. This means we do not know whether the two-stage algorithm can improve the robustness of convolutional neural network image classification. In this article, we will mainly discuss when the two-stage algorithm performs well and when it does not.

Based on the previous literature, we design a comparable but slightly different two-stage algorithm structure for convolutional neural networks. For our two-stage training algorithm (see Figure 4), the first stage is that we run all the steps of the original CNN model, and the second stage is that we will build another new model, which only includes the last dense layer (SoftMax layer) with the same classified numbers. Before running the second stage, we need to obtain the feature values trained in the first stage. We regard the process of feeding the raw input datasets into the last hidden layer before the SoftMax layer in the part of the fully connected layer as the feature extraction process. Then use the extracted feature values as the original input data of the second model and train the new model again. This process is to make decisions on the classification again. The two-stage training can improve the robustness of some datasets for which the robustness of the model is not strong.

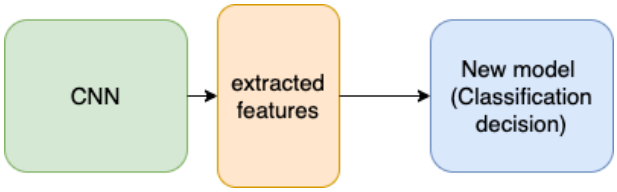


Figure 4. The workflow of the two-stage training algorithm

4. APPLICATIONS

In this section, we will introduce four real-world image datasets, namely Fashion-MNIST, Cifar-10, dogs & cats, and face mask detection, and explain in detail their different CNN architecture and the data preprocessing method used in the first stage of the algorithm. Then we will apply the second stage of the algorithm to these four datasets.

4.1 Fashion-MNIST Dataset

The Fashion-MINIST is a benchmark dataset that is widely used for evaluating the performance of various deep-learning models in image classification. This dataset was released in 2017 as a replacement for the MNIST dataset, which is more complex than MNIST. It has a total of 70,000 fashion products, which can be divided into 10 fashion product types, each category has 7,000 images, which indicates that the distribution of data classes is balanced. Among them, 60,000 images of them are training data, 10,000 images of them are test data, and these data are automatically set in the Keras dataset. To verify the fitting performance of the model training process, we also need to randomly extract 20% of the training data as validation data. This ratio can also be set to other values. The spatial size of each image in this dataset is 28×28 grayscale pixels by default, and the channel is 1. Figure 5 shows some samples of the Fashion-MNIST dataset, we can find out what the categories of these 10 fashion products are, respectively, ankle boot, T-shirt/top, dress, pullover, shirt, sandal, sneaker, trouser, bag, and coat, and each category has 4 sample images.

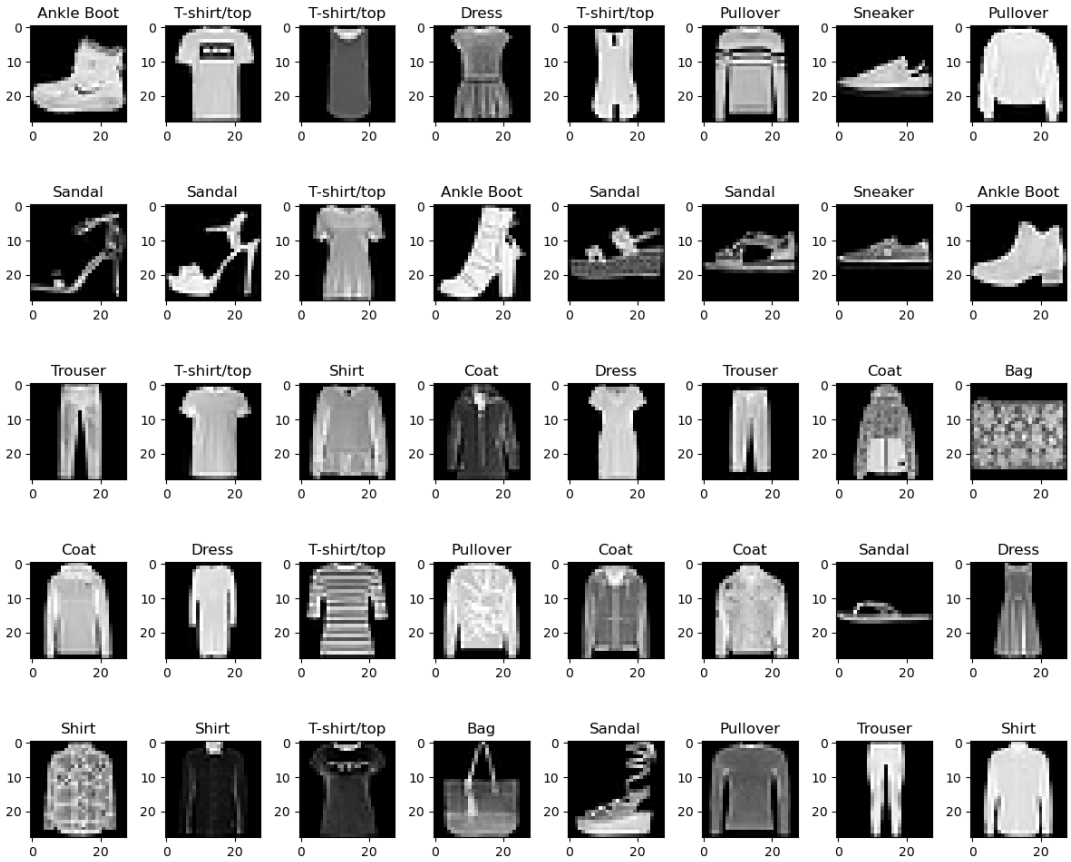


Figure 5. The sample of the Fashion-MNIST dataset

In the first stage, the architecture of the convolutional neural network model is designed with reference to Architecture 3 (see Figure 6) in the paper [13], because Architecture 3 (3 convolutional layers and 2 fully connected layers) has a better testing accuracy than other models. It has three convolutional layers, two Max Pooling layers, two dense layers in fully connected layers, and three Dropout layers. Among them, each convolutional layer has 64 kernels whose size is 2×2 and uses the ReLU nonlinear activation function, the size of each Max Pooling layer is set to 2×2, the value in each Dropout layer is set to 0.25, and the first hidden layer has 64 hidden neurons and uses the ReLU activation function, and the last dense layer use the SoftMax activation function, the output categories must set to 10.

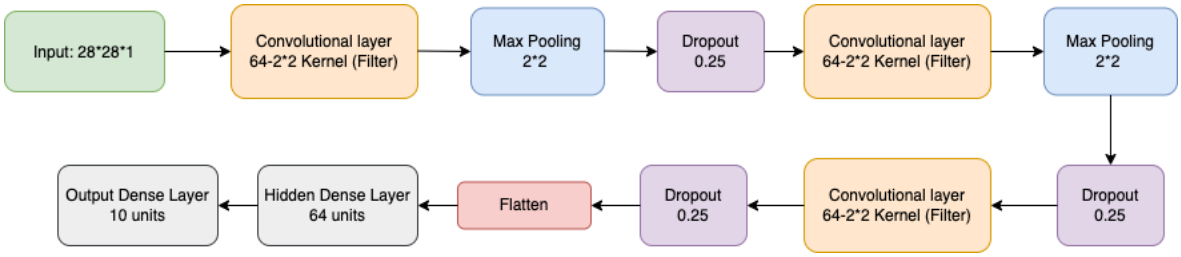


Figure 6. The architecture of CNN for Fashion-MNIST

Before training the model, we need to normalize the data which includes training data and test data. A common operation for normalization is to divide the data matrix by 255, because the original images pixel values are in the range of [0,255], the pixel values can be rescaled to the range of [0,1] by simple division. For the training label and test label, since we use the categorical cross-entropy loss function when compiling the model, we must convert the label into one-hot by using the "to_categorical" function in the Keras library.

During the model training process, we may encounter data overfitting problems, that is, the training accuracy is improving while the validation accuracy is decreasing, and this situation is not getting better. To address this problem, we add the EarlyStopping and ModelCheckPoint methods to avoid overfitting and store the best model weights when training the dataset. The main function of EarlyStopping is to check whether the metric set in the "monitor" was better than the best value found so far after each epoch finished. If there is no improvement, it determines when to stop the training process according to the parameter set in the "patience. For example, the patience is set to 10, if there is no improvement for 10 continuous times, this command can shut down the entire epoch loop. If it improves, the patience count will be reset and restarted from 0. The main function of ModelCheckPoint is to save the best model automatically in terms of the metric set in the "monitor". Typically, it works together with the EarlyStopping method. For the Fashion-MNIST dataset, the epoch is 50 which means the training times repeat 50 times, the metrics of the monitor of both methods are set to "val_accuracy", and the patience is set to 10.

After training this model, we will evaluate the accuracy of the test data and save this model for the second stage of training. The new model of the second stage is rebuilt based on the last layer of this model. The model compilation is the same as the first model. The training epoch is set to 50 for the new model, and the EarlyStopping and ModelCheckPoint methods are also used.

4.2 Cifar-10

The Cifar-10 is also a benchmark dataset generally applied to compare different models' accuracy. This dataset was published in 2009 and is a subset of the 80 million image datasets from 2008. It also can be downloaded from the Keras datasets. It consists of 60,000 images with 10 categories, each category has 6,000 images, which means the class distribution is balanced. The dataset has already been split into training data, training labels, test data, and test labels, where the training data is 50,000 images and the testing data is 10,000 images. As in the Fashion-MNIST dataset, we set 20% of the training data as validation data. The image size is 32×32 color pixels, and the channel is 3. Figure 7 has shown some samples of the Cifar-10 dataset, there are 10 item classes that include 6 kinds of animals (frog, deer, bird, horse, cat, dog) and 4 kinds of transportation vehicles (truck, automobile, ship, airplane), and each class has 4 sample images. Before training and splitting the data, we also need to do some simple processes for the data such as data normalization and label encoding.

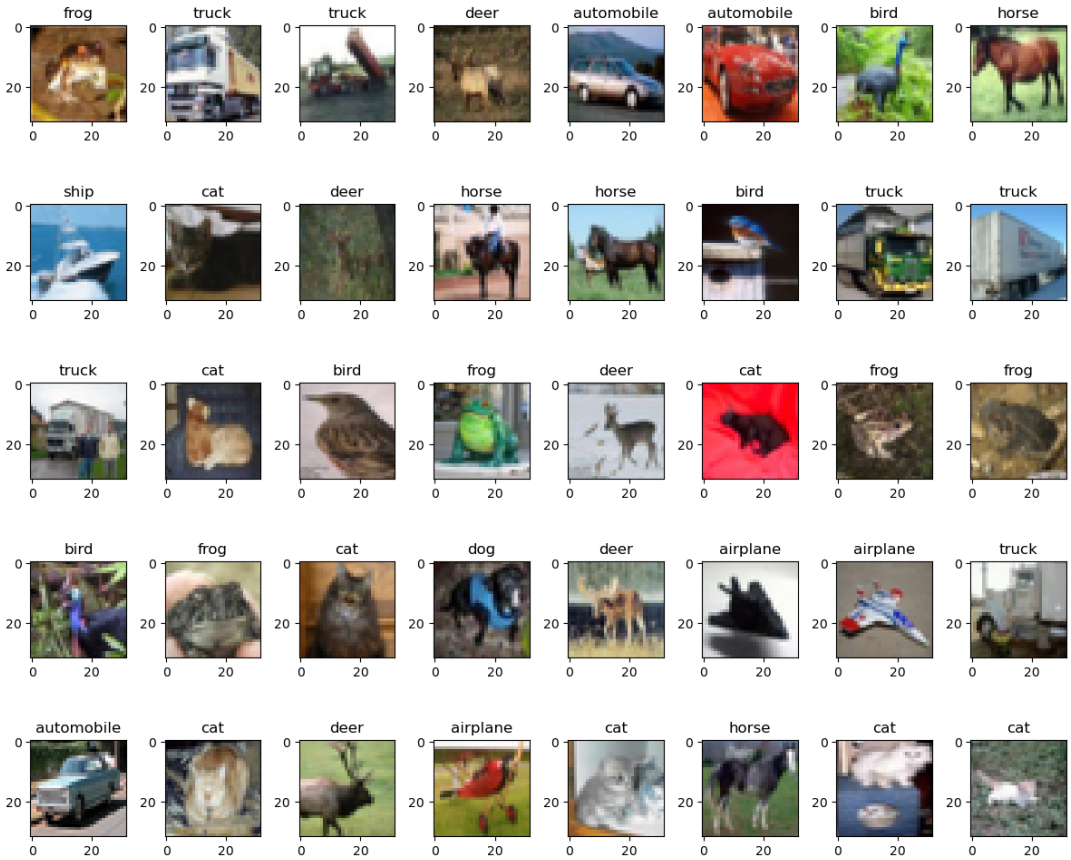


Figure 7. The sample of the Cifar-10 dataset

The architecture of the convolutional neural network for Cifar-10 (see Figure 8) of the first stage refers to the Model 4 designs on Kaggle [14]. This model is obviously deeper than the previous model because the Cifar-10 dataset is more complex than Fashion-MNIST, and it can be seen from Figure 7 that Cifar-10 has more interference information. It has a total of 6 convolutional layers, 7 Batch Normalization layers, 3 Max Pooling layers, 4 Dropout layers, 1 Flatten layer, and 2 Dense layers. The parameter configuration of the convolution kernel is also different from the previous

model, the kernel size is 3×3 and its quantity varies in different layers, from 32 to 64 to 128. The Dropout parameter increases each time, from 0.2 at the beginning to 0.5 at the end. And the unit of the first dense layer in this model is set to 128, and the unit of the output layer must be set to 10 because this dataset has 10 classes.

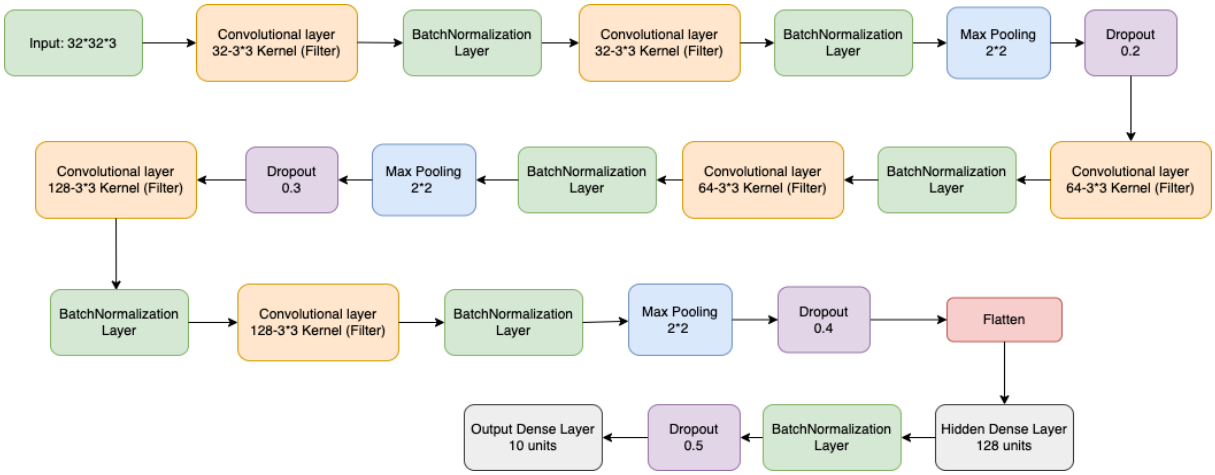


Figure 8. The architecture of CNN for Cifar-10

For this instance, the model compilation is the same as the previous dataset that uses the categorical cross-entropy loss function, the Adam optimizer, and the accuracy metric. We still use the EarlyStopping and ModelCheckPoint methods in the training process, both monitors use the "val_accuracy" metric, and the patience in EarlyStopping is also set to 10. The training epoch is still set to 50. These functions and parameters set are also used in the second stage to better observe the algorithm.

4.3 Dogs & Cats

The Dogs & Cats is a classic image dataset found from the database on Kaggle. The original dog & cat Kaggle dataset consists of 25,000 images, however, the data we found was a subset of these images, about more than 10,000 images, of which more than 8,000 are training data and above 2,000 are test data. Among them, each data folder is half cats and half dogs (see Figure 9), and the distribution of data categories is balanced. We set 20% of training data as validation data and set the shuffle as True which means the image order is randomly before each regrouping.

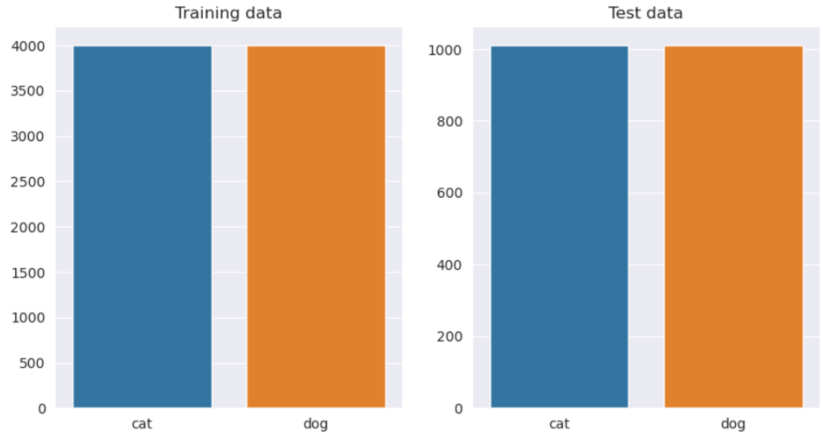


Figure 9. The distribution of dogs & cats in training data and test data

In this part, since the dataset is not downloaded from the Keras database, it is stored in folders, which means that the data has not been processed in advance. So compared with the previous two datasets, we need to perform data preprocessing for this dataset. Here, we use OpenCV or CV2, a popular open-source computer vision library, to process the images. This library has many powerful functions and tools not limited to images but other computer vision tasks as well, we mainly use CV2 to read and resize the images, and the Dogs & Cats images are resized to 128×128 with 3 channels. Figure 10 has shown some samples of this dataset.

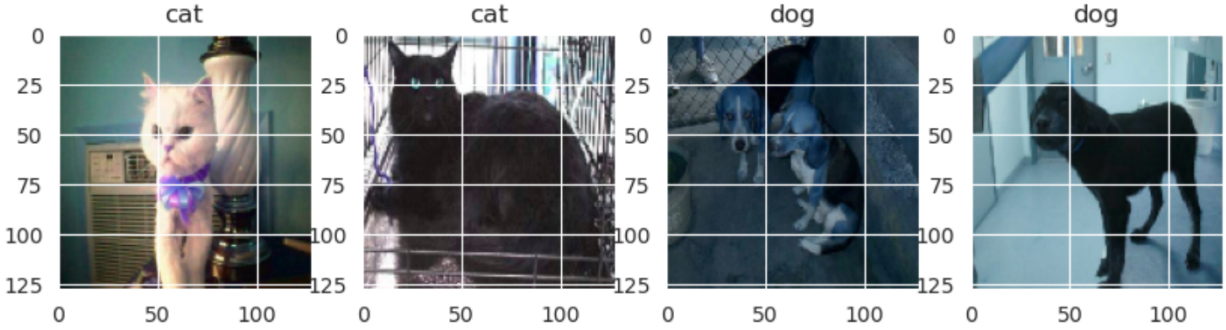


Figure 10. The sample of the Dogs & Cats dataset

Data Augmentation is applied to this dataset by using the ImageDataGenerator function from the Keras preprocessing image package, because the amount of the Dogs & Cats dataset is not large compared with the previous two datasets, but the Data Augmentation method can expand its size and diversity of this dataset by implementing some transformations to the original training data. Because a large dataset can help the model learn better from more samples, reduce overfitting to the data, and improve the robustness of the model. Here, we mainly rotate the image randomly, shift the image horizontally and vertically, and flip the image horizontally. The rotation range is set to 10, the zoom range is set to 0.1, and the width and height shift ranges are set to 0.2 each. And the batch size is set to 32 when training the model.

The architecture of the convolutional neural network for Dogs & Cats of the first stage refers to the model (see Figure 11) designs on Kaggle [15]. This model has 3 convolutional layers, 3 Max Pooling layers, 4 Batch Normalization layers, 4 Dropout layers, and 2 fully connected dense layers. The number of kernels on each convolutional layer increase layer by layer, from 32 to 64 to 128. The parameters of the first three Dropout layers are 0.25 and the last one inside the fully connected layers is 0.5. And there are 512 units in the first dense layer, and 2 units in the last dense layer.

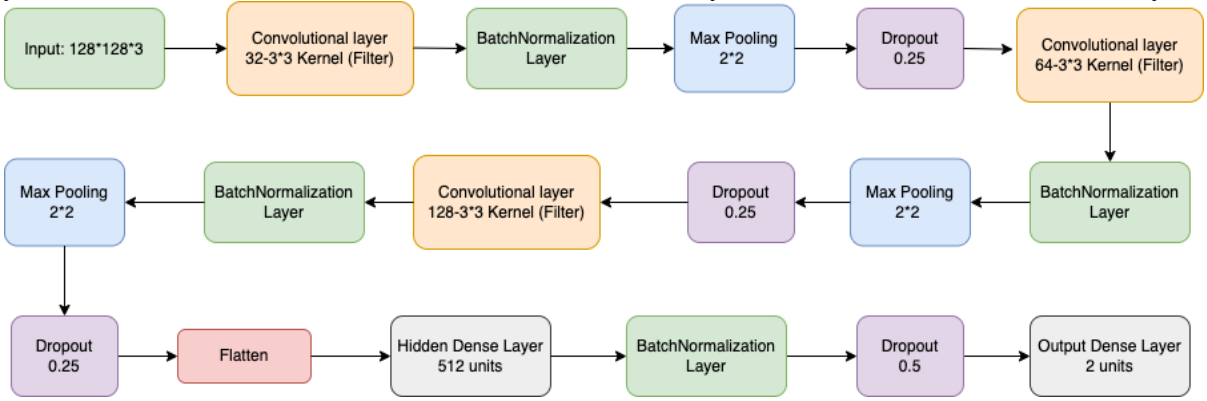


Figure 11. The architecture of CNN for Dogs & Cats

The model compilation for this dataset is the same as the previous two datasets. And we also use the EarlyStopping and ModelCheckpoint methods to monitor the accuracy of the training data and validation data. The patience of the EarlyStopping is also set to 10 and the epoch is 50. All of these are also used in the second stage of training.

4.4 Face Mask Detection

The Face Mask Detection dataset is a relatively new dataset compared with the previous three datasets and is found in Kaggle [16]. This dataset focuses on whether people wear masks and whether they wear masks correctly during the Covid-19 epidemic because the coronavirus is mainly transmitted through the respiratory droplets produced by infected patients when coughing or sneezing, so wearing masks correctly is conducive to reducing the spread of the coronavirus. This dataset has 3 categories: with mask, without mask, and mask worn incorrectly, with 2994 images for each category (see Figure 12). The data shows that the data class distribution is balanced.

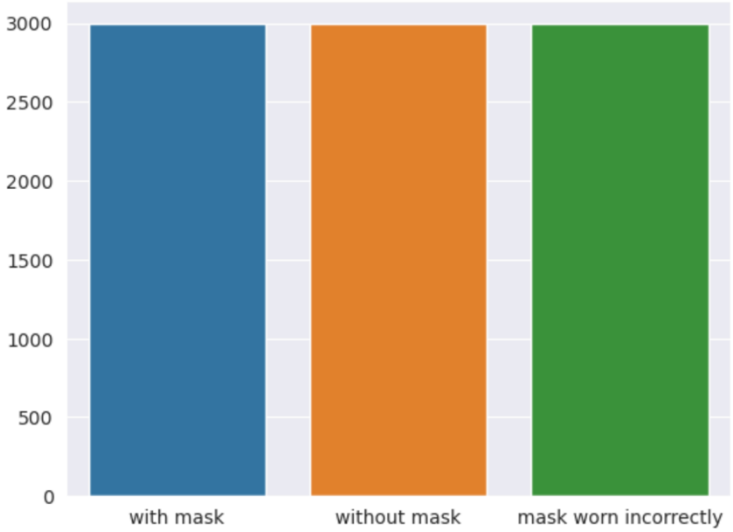


Figure 12. The distribution of Face Mask Detection in each category

For this dataset, since the images are stored in three different folders, we also need to use OpenCV or cv2 image processing tools to import the images. The size of the images is reshaped to 128×128 with 3 channels. Because the data is not divided into training data and test data in advance, we need to use the train_test_split function twice, the first time all data is split into training data and test data, and the second time the training data is split into training data and validation data. The test ratio is 0.2 for each. The shuffle in this function is set to True, which means the data will be shuffled before each split.

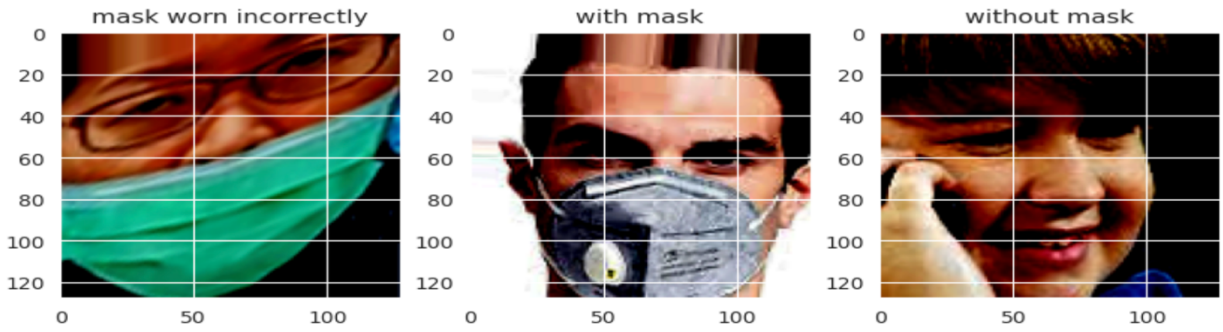


Figure 13. The sample of the Face Mask Detection dataset

Data Augmentation is used by implementing the ImageDataGenerator function since the number of this dataset is not large. The image rotation range is 10, the randomly zoom image range is 0.1, the width and height shift ranges are 0.2, and the horizontal flip is set to True.

The architecture of CNN for Face Mask Detection in the first stage is a reference to model 2 designed in Kaggle [17]. This model has 3 convolutional layers, 3 Max Pooling layers, 1 Batch Normalization layer, 3 Dropout layers, and 2 fully connected layers. Figure 14 has shown the parameters and order defined by each layer.

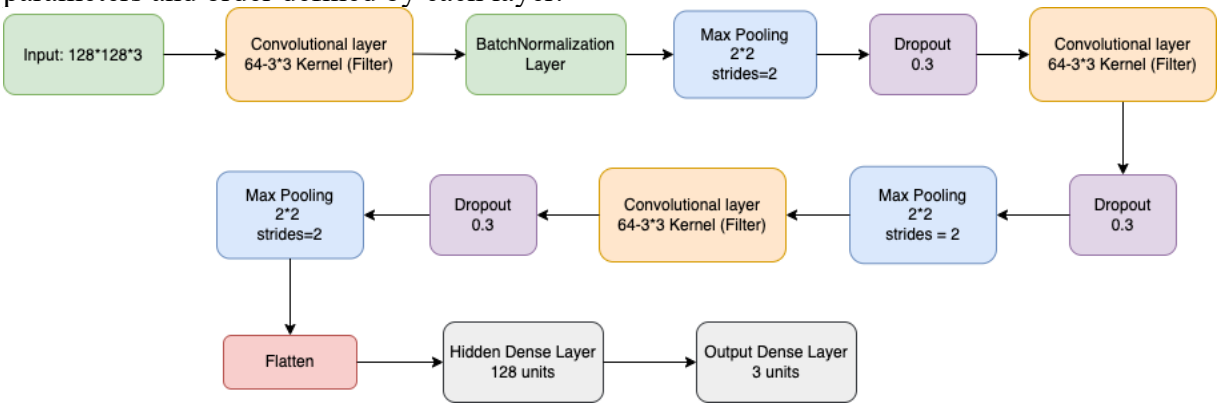


Figure 14. The architecture of CNN for Face Mask Detection

For this dataset, two versions are designed for the same CNN architecture. The first version still uses the EarlyStopping and ModelCheckPoint methods that keep the same as the previous three datasets, the patience is also set to 10. But because the data fitting degree of this model fluctuates greatly (details will be introduced in the next section), we design another version that combines the EarlyStopping and Learning Rate Reduction methods to improve the performance of this model. The Learning Rate Reduction method is to allow the model to dynamically reduce the learning rate when the metric monitored stops improving, which can help the model find a better solution and avoid falling into a suboptimal solution. For the second version, the patience of EarlyStopping is set to 5, and in the Learning Rate Reduction method, the factor is 0.5, the patience is 3, and the minimal learning rate is 0.00001. And the epoch in the model training process is 50, the batch size is 32.

It can be seen from the above four figures of the architectures of the CNN model for each dataset is different. The purpose of this article is not to focus on designing the architecture of the CNN

model in the first stage, but to run the second stage of the algorithm based on the model designed by others, to compare the two-stage algorithm and one-stage CNN model which performs better.

5. RESULTS AND ANALYSIS

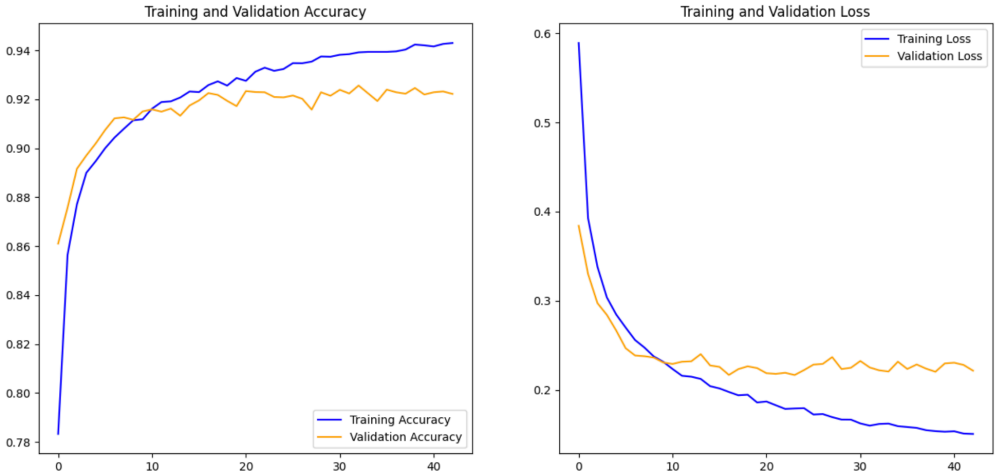
In this section, we will show the result of two-stage training for each dataset and explain the algorithm's impact on the model's robustness. These results were obtained by running on a GPU with the hardware name NVIDIA GeForce RTX 2080 Ti.

The experiments on each dataset are repeated at least 20 times, and the results of all experiments are averaged. The results are shown in Table 1. The second stage of two-stage training is trained by two different optimizers, the first one uses the same Adam optimizer as the first stage, and the second uses the SGD with momentum optimizer to analyze the impact of different optimizers on retraining. Figure 15 shows the training process of the first stage of an experience in each dataset.

Table 1. Four Datasets Testing Accuracy Results

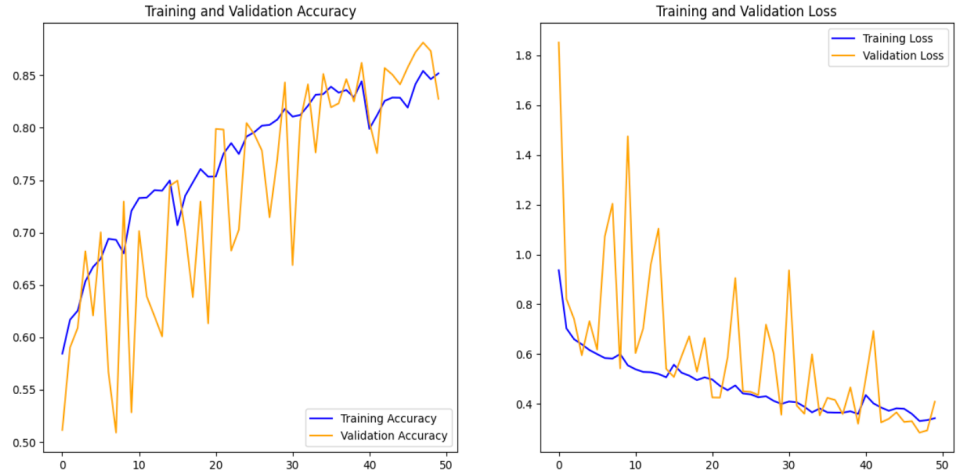
Method \ Dataset		Fashion-MNIST	Cifar-10	Dogs & Cats	Face Mask Detection	
					Version 1	Version 2
CNN (Adam)	Training accuracy	97.03%	98.77%	82.47%	92.27%	97.28% (99.18%)
	Validation accuracy	92.10%	85.91%	80.64%	90.99%	96.11% (98.07%)
	Test accuracy	91.75%	85.66%	81.40%	90.88%	95.99% (97.87%)
CNN + second stage (Adam)	Training accuracy	97.64%	99.24%	91.62%	99.06%	99.30% (99.59%)
	Validation accuracy	92.25%	86.44%	86.75%	97.78%	98.31% (98.57%)
	Test accuracy	91.86%	86.07%	87.32%	97.59%	98.13% (98.44%)
CNN + second stage (SGD)	Training accuracy	97.49%	98.88%	91.18%	98.65%	99.03% (99.34%)
	Validation accuracy	92.30%	86.38%	86.80%	97.51%	98.00% (98.28%)
	Test accuracy	91.91%	86.06%	87.65%	97.19%	97.75% (98.10%)

(In Table 1, version 2 of Face Mask Detection has two results, the first is the averaged result includes one outlier experiment and the second inside the parenthesis excludes one outlier experiment. The reason for this outlier will be explained later)

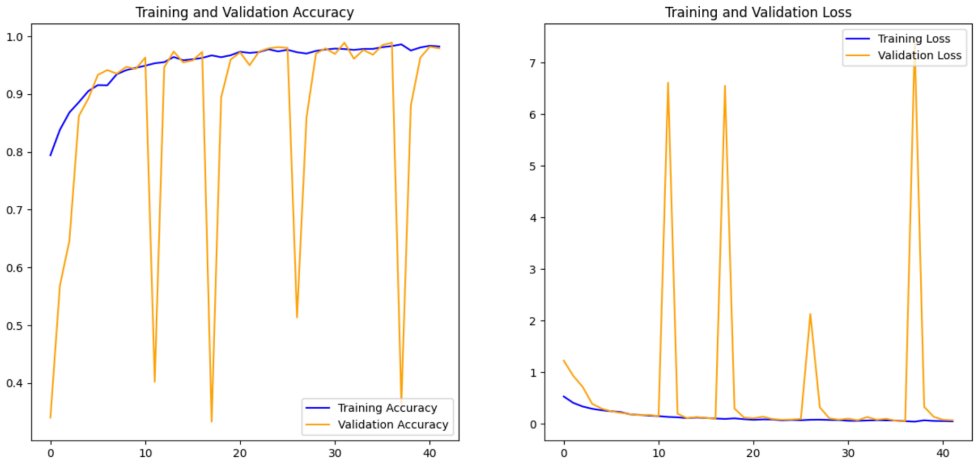


(a) Fashion-MNIST

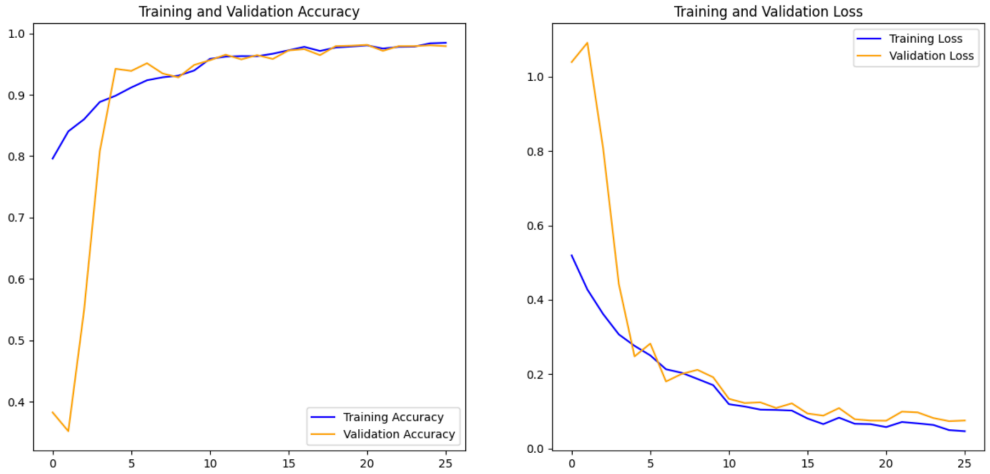
(b) Cifar-10



(c) Dogs & Cats



(d) The EarlyStopping Method for Face Mask Detection



(e) The EarlyStopping and LearningRateReduction methods combined for Face Mask Detection

Figure 15. A random experiment of training and validation accuracy and loss of each dataset in the first stage of two-stage training

From Table 1, we can figure out that the two-stage training method can improve the image classification accuracy again compared with the one-stage training method. However, some datasets are not significantly improved, and some datasets are significantly improved. Based on these results, we want to understand what causes this result. By analyzing Figure 15, the validation data in Fashion-MNIST (Figure 15 (a)) and Cifar-10 (Figure 15 (b)) fluctuate less during the training process, while the validation data in Dogs & Cats (Figure 15 (c)) and Face Mask Detection in version 1 (Figure 15 (e)) fluctuate more. The reason for this result is firstly related to the dataset itself. The latter two datasets have less data quantity, lower data quality, and more noise than the first two datasets. Secondly, it is related to the designed model or the learning rate of the model. Since the two-stage training does not consider the design of the model, we only briefly analyze the influence of the learning rate of the model on two-stage training. For example, after adding the Learning Rate Reduction method (Figure 15 (e)), the volatility of the validation data during the training process is reduced. The results with outlier experiments are much lower than version 1 of Face Mask Detection. That's why the two results of version 2 are shown in Table 1. The second results show that the two-stage algorithm still improves the robustness of the model in the absence of this outlier.

Combining Table 1 with Figure 15 shows how the stability of the first-stage model training affects the improvement level. For instance, from Figure 15 (c) and (d), the validation data fitting of Dogs & Cats and Face Mask Detection datasets fluctuates greatly. This leads to the fact that after the training is stopped, the weights of the model for the last epoch may not be able to predict the image category well, resulting in the accuracy prediction value of the first-stage model for the test data being lower than the best model during the training process.

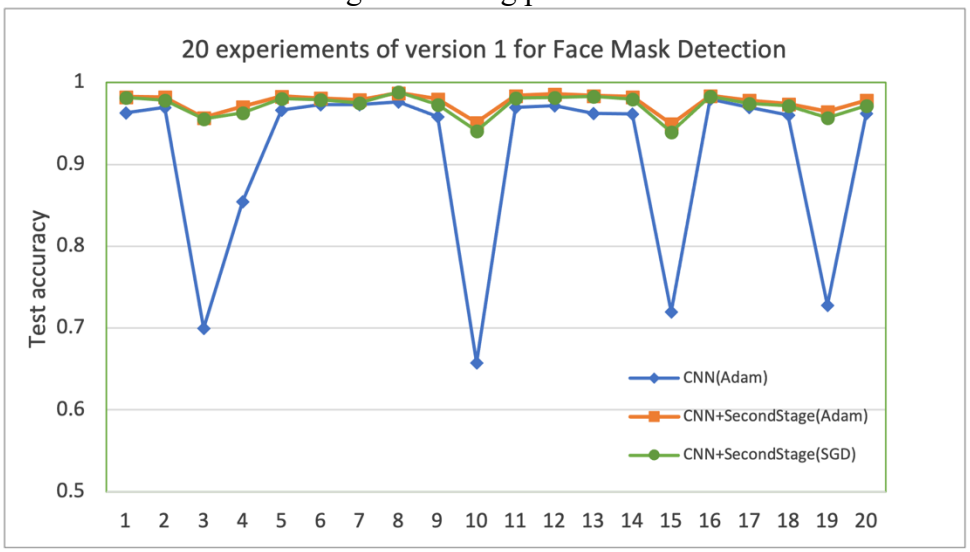


Figure 16. The test data accuracy across 20 experiments for version 1 of the Face Mask Detection dataset

As we can see from Figure 16, the test accuracy of the CNN model in version 1 for the Face Mask Detection dataset for each experiment, the large volatility of the model resulted in underpredictions in some experiments, which reduces the overall average prediction. However, the second stage allows the new model to redefine the image classification based on the first-stage model, helping this model recover and exceed the prediction value of the previous model.

From Table 1, the average test accuracy of Dogs & Cats improves from 81.40% to 87.32% with the Adam optimizer, and 87.65% with the SGD optimizer. The average Face Mask detection test accuracy in version 1 improves from 90.88% to 97.59% with the Adam optimizer and 97.19% with the SGD optimizer. It improves from 95.99% to 98.13% with the Adam optimizer and 97.75% with the SGD optimizer in version 2. The results after deleting one outlier experiment, the test accuracy improves from 97.87% to 98.43% with the Adam optimizer and 98.10% with the SGD optimizer. These results show that this method can improve the robustness of the CNN model regardless of the optimizer chosen. Even though the volatility of the model is decreased, the two-stage training can still improve the model's performance for smaller datasets.

Table 2. t-Test for Assessing the Statistical Significance of Two-Stage Algorithm Improvements

Datasets	Mean of first-stage test accuracy	Mean of second-stage test accuracy	Significance Level	P(T<=t) two-tail

Fashion-MNIST	0.9175	0.9186	0.05	0.026120
Cifar-10	0.8566	0.8607	0.05	0.000320
Dogs & Cats	0.8140	0.8732	0.05	0.000016
FacemaskDetection (version1)	0.9088	0.9759	0.05	0.007079
FacemaskDetection (version2)	0.9599	0.9813	0.05	0.189317
FacemaskDetection (version2 exclude one outlier)	0.9787	0.9844	0.05	0.000004

For these 20 experiments, we use the t-Test method to obtain their P-Values to access the statistical significance. Even for the Fashion-MNIST and Cifar-10 datasets, the two-stage training has little effect on the improvement of the test accuracy, but the P-values in Table 2 demonstrate that the improvement of test accuracy by the two-stage algorithm is significant. For the other two datasets, the P-values also can prove this algorithm is significant, except for version 2 of Face Mask Detection. Even a P-value greater than 0.05 doesn't prove that the result is completely meaningless or worthless. This may be caused by the randomness of the experiment and the insufficient number of experiments. Therefore, there is one experiment in the first stage whose results are quite different from other experiments, but the second-stage training does improve the test accuracy of the first stage. At the same time, after we try to exclude this outlier experiment, it can be clearly seen that the P-value is very significant, and the accuracy is also improved.

6. CONCLUSIONS

In this article, we proposed to implement a two-stage training method based on the convolutional neural network for image classification. This paper first introduced the development trend of CNN architectures in recent years. Then the most common components of CNN architectures used in these four datasets were described. Furthermore, four real-world application datasets using CNN are elaborated. Among them, different CNN architectures are used for different datasets, and for two of the datasets that were not downloaded from Keras databases, we used the OpenCV method to read the images. Finally, the results of each dataset in the two-stage training method were analyzed.

The original motivation of the two-stage algorithm was to study whether two-stage training is feasible in convolutional neural networks. The results demonstrated that when the model training process in the one-stage algorithm is stable, the use of the two-stage algorithm has a limited ability to improve the prediction accuracy of the original model, which means that the disadvantage of this algorithm is that it cannot greatly enhance the accuracy of a well-trained model. But for the model with an unstable training process, the two-stage training can increase the robustness of the model.

REFERENCES

[1] Ni, Jianjun, Yinan Chen, Yan Chen, Jinxiu Zhu, Deena Ali, and Weidong Cao. 2020. "A Survey on Theories and Applications for Self-Driving Cars Based on Deep Learning Methods" *Applied Sciences* 10, no. 8: 2749. https://doi.org/10.3390/app10082749.

[2] Li, Qing, Weidong Cai, Xiaogang Wang, Yun Zhou, David Dagan Feng, and Mei Chen. "Medical image classification with convolutional neural network." In *2014 13th international conference on control automation robotics & vision (ICARCV)*, pp. 844-848. IEEE, 2014.

[3] Coşkun, Musab, Ayşegül Uçar, Özal Yildirim, and Yakup Demir. "Face recognition based on convolutional neural network." In 2017 International Conference on Modern Electrical and Energy Systems (MEES), pp. 376-379. IEEE, 2017.

[4] LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86, no. 11 (1998): 2278-2324.

[5] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Communications of the ACM 60, no. 6 (2017): 84-90.

[6] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

[7] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1-9. 2015.

[8] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770-778. 2016.

[9] Chollet, François. "Xception: Deep learning with depthwise separable convolutions." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251-1258. 2017.

[10] Liu, Shu, and Qiang Wu. "Robust Representations in Deep Learning." DBKDA 2023 (2023): 34.

[11] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." In *International conference on machine learning*, pp. 448-456. pmlr, 2015.

[12] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[13] Kadam, Shivam S., Amol C. Adamuthe, and Ashwini B. Patil. "CNN model for image classification on MNIST and fashion-MNIST dataset." Journal of scientific research 64, no. 2 (2020): 374-384.

[14] https://www.kaggle.com/code/kedarsai/cifar-10-88-accuracy-using-keras

[15] https://www.kaggle.com/code/uysimty/keras-cnn-dog-or-cat-classification

[16] Face Mask Detection Dataset. https://www.kaggle.com/datasets/vijaykumar1799/face-mask-detection

[17] https://www.kaggle.com/code/muhammetzahitaydn/mask-detection-cnn-resnet50-efficientnet/notebook