Student Number: 252502

1. Abstract

This project aims to perform image classification on the Fashion MNIST dataset using deep learning techniques. To achieve this goal, we developed a deep neural network consisting of multiple convolutional and fully connected layers and explored different combinations of hyperparameters to optimize model performance. In model training, we use Cross-validation and compare accuracy of different parameter combinations to find the better hyperparameters combinations. Next, we apply the pre-trained ResNet-18 model on the Fashion MNIST dataset and compare the representation of the two models in the hidden layer. Then, we set up the correlation matrix. Through the analysis and understanding of the correlation matrix, we compare the performance differences of the two models, and analyze their advantages and disadvantages.

Finally, according to the understanding of the experiment, we provide some optimization suggestions for the custom model to make it have better generalization ability and performance.

2.Methodology

In this experiment, we will use a convolutional neural network (CNN) for image multi-classification of the MNIST-Fashion dataset.

2.1 Data Set Loading and Pre-processing

This experiment employed the Fashion-MNIST dataset, which contains 10 categories of clothing items, with a total of 60,000 training samples and 10,000 test samples. Each sample is a 28*28 pixel grayscale image. We used the Torchvision library to download and load the Fashion-MNIST dataset. As we loaded the dataset, we converted the image data into a Tensor format in the range [0,1] to ensure data compatibility with the PyTorch framework and to use a GPU for parallel processing of data. This facilitates calculations and operations, accelerating the computation process. Next, we packaged the training data and the test data separately with DataLoader, where the training data is randomly shuffled. Increasing the randomness during the model training process helps the model generalize better and prevents overfitting. It can also make the model learn the overall characteristics of the dataset in a short time, allowing the model to converge quickly.

2.2 Convolutional Neural Network Structure

In this experiment, we used a convolutional neural network (CNN) with five convolutional layers and three fully connected layer to classify the Fashionist dataset. The following is a structural introduction of the convolution layer and the fully connected layer: [1]

First Convolutional Layer: (Input channels: 1, Output channels: 96, Convolution kernel size: 3 x 3, step: 1, Padding: 1, Followed by batch normalization, ReLU activation function, and max-pooling layer (3x3, stride 2))

The second convolution layer: with the number of input channels set to 96the number of output channels is 256, the size of the convolution kernel is 3x3, the step size is 1, and the filling is 1. And then use batch normalization, ReLU activation function, and maximum pooling layer (3*3, step size 2).

The third convolutional layer: (with the number of input channels set to 256, the number of output channels is 384, the size of the convolutional kernel is 3*3, the step size is 1, and the filling is 1. And then use ReLU activation function.)

The fourth convolutional layer: (with the number of input channels set to 384 the number of output channels is 384, the size of the convolutional kernel is 3*3, the step size is 1, and the filling is 1. And then use ReLU activation function.)

The fifth convolutional layer: (with the number of input channels set to 384， the number of output channels is 256, the size of the convolutional kernel is 3*3, the step size is 1, and the filling is 1. And then use the ReLU activation function and the maximum pooling layer (3*3, step size 2).)

After the convolutional layers, the output is flattened to convert the multi-dimensional tensor into a one-dimensional tensor, which is then fed into the fully connected layer for classification. The fully connected layer consists of three parts:

The first fully connection layer: the input dimension is 256 * 2 * 2 (calculated based on the input image size) and the output dimension is 2048. With the ReLU activation function and the Dropout layer of 0.05.

The second fully connection layer: Input dimension is 2048, output dimension is 2048. With ReLU activation function and the Dropout layer of 0.05.

The third fully connection layer (output layer): input dimension is 2048, output dimension is 10 (number of output categories).[5]

2.3 Optimizer and Loss Function

We used the SGD optimizer and set the learning rate to 0.003. This learning rate ensures better convergence speed and accuracy of the model. We then called the nn.CrossEntropyLoss() function to use the cross-entropy loss function, which is often used for multi-classification problems. The cross-entropy loss function minimizes the difference between the predicted label and the real label, making the prediction result more accurate.

2.4 Overfitting

For solving the overfitting problem, we set the dropout layer in fully connected layer to prevent overfitting. We set the dropout probability to 0.05, this will improve the generalization ability of the model on the test set.

Then we used L2 regularization to prevent overfitting. The specific operation is to add the weight decay parameter to the optimizer to implement L2 regularization. In our model, we set the weight decay value to 0.001.

2.5 Hyperparameter Test Value

The following are some of the hyperparameter values we tested in our experiment, as shown as table:

Batch size	64	128	256
Learning rate	0.1	0.003	0.0005
Dropout	0.1	0.5	0.7

Through cross-validation, we explore the model performance in different hyperparameters to obtain the optimal combination of hyperparameters that can optimize model performance. The following is a summary of the experimental process:

Firstly, Load and preprocess the Fashion-MNIST dataset. Secondly, define a convolutional neural network (CNN) model with five convolutional layers and one fully connected layer. Thirdly, Set the hyperparameters, including learning rate, Dropout probability, regularization method, batch size, etc. Fourth, utilize the stochastic gradient descent optimizer and cross entropy loss function to train the model. (use Dropout layer and L2 regularization to avoid overfitting).And then test different hyperparameter to find the best combinations of hyperparameters. Finally, use the test dataset to evaluate the model performance and analyze the experimental results.

2.6 ResNet-18

In this experiment, we used the Fashion MNIST dataset and preprocessed the input image to make it compatible with the pre-trained ResNet-18 model. The basic unit of the ResNet is the residual unit, as shown in the figure 3.[3] We first resize the data from 28*28 to 224*224. And then copy the grayscale image onto the three channels, form the same RGB image as the original. Finally, normalize the image and convert it into a tensor. And then load the pre-trained ResNet-18 model, extract 100 images of neuron activation for each category, calculate a 10*10 correlation matrix where the i-j term represents the average correlation between the image representations of class i and class j.

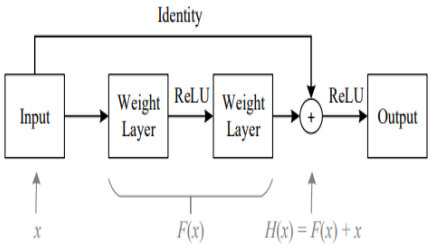

Fig3 structure of residual unit.

3. Result

3.1 CNN Model Result

In this section, we present the results of the hyperparameter tests conducted during the experiment. We used tables and figures to display the results clearly and understandably.

We started with the initial hyperparameter settings (Table 1) and then changed the value of each hyperparameter one by one to discuss the optimal parameter combination. (Table 2, Table 3, Table 4) Our analysis shows that the optimal learning rate is 0.05, which resulted in better accuracy and loss function performance in both the training and test sets. The optimal batch size is 128, which also produced better accuracy and loss function performance in both the training and test sets. Finally, the optimal dropout value is 0.5, leading to improved accuracy and loss function performance for both the training and test sets. In summary, the optimal hyperparameter combination for our custom CNN model is a learning rate of 0.05, a batch size of 128, and a dropout value of 0.5. This

combination results in the best performance on the Fashion MNIST dataset in terms of accuracy and loss function values in both the training and test sets. As shown as table2, table3, table4.[2]

After conducting the hyperparameter tests, the best-performing model achieved a test accuracy of 90.8% with a learning rate of 0.05, dropout probability of 0.5, and Batch size of 128. And we use the Dropout layer and L2 regularization to mitigate overfitting problem, that will improve the model's generalization ability. The performance gap between the training set and the verification set (the test set) is relatively small (93.2%and90.8%), indicating a low likelihood of overfitting(Fig 1).

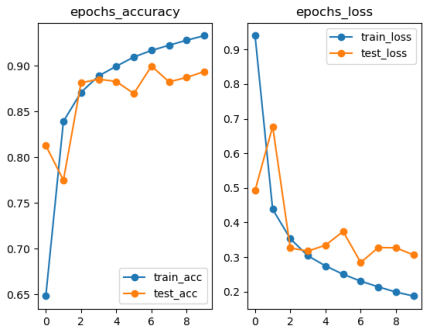

Fig 1 the accuracy and loss values of the training set and test set change within 10 epochs

In conclusion, the proposed CNN model successfully classified the Fashion-MNIST dataset with high accuracy. The hyperparameter tests and regularization strategies effectively optimized the model's performance and addressed potential overfitting issues.

3.2 ResNet-18 And CNN 10*10 Correlation Matrix

Observing the pre-trained ResNet-18 model, we find that its correlation matrix values are generally between 0.3 and 0.4, but the correlation values of correct classification

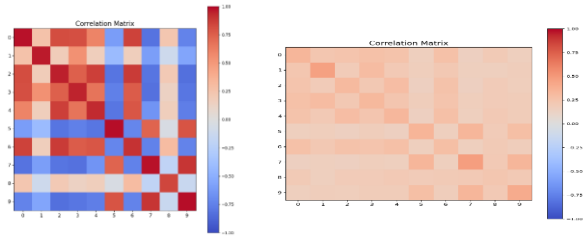

Fig2 the left is a CNN model 10*10 matrix, the right is ResNet-18 model 10*10 matrix.

are greater. Observing the customized CNN model, we observe that the value of the correlation matrix among all categories varies greatly, and the value on the main diagonal is close to 1.

4. Discussion

In this experiment, we observed that our custom CNN model successfully classified the Fashion-MNIST dataset with high accuracy. The optimal hyperparameter combination, which included a learning rate of 0.05, batch size of 128, dropout probability of 0.5, L2 regularization, SGD optimizer, and cross-entropy loss function, led to a test set accuracy of 90.8% and a training set accuracy of 93.4%. The loss function values were below 0.3, indicating a relatively small performance gap between the training and test sets and suggesting a reduced likelihood of overfitting.

To further improve our custom model, we could experiment with different regularization techniques (such as L1), regularization decay weights, optimizers (like Adam or use Adam and RmsProp together) [3], and loss functions. Additionally, we could increase the size of the convolution kernel, the number of convolution layers, or the number of epochs to optimize the network structure and enhance the model's performance. Using better hardware may also improve the prediction speed.

We also compared the performance of our custom CNN model with the pre-trained ResNet-18 model on the Fashion-MNIST dataset. We observed that the values of the correlation matrix for the ResNet-18 model were generally mainly in the range of 0.3 to 0.4, with the range indicating a moderate positive correlation, indicating some degree of correlation between activations of different classes of images in the last hidden layer. The correlation values for correct classification are higher than those for incorrect classification, proving that it is still by some degree capable of classification.

The custom model demonstrated better performance on the Fashion-MNIST dataset because it was specifically designed for this task. The custom CNN model performs well in terms of intra-class correlation, and the value on the diagonal is close to 1, which indicates that the model is effective in distinguishing images of the same class. The ResNet-18 model, despite not being trained on this dataset, still have some classify ability, proving its good generalization and transfer learning capabilities.

In conclusion, when selecting a model, it is essential to choose one that best suits the task at hand. The integration of the custom model and ResNet-18 could potentially enhance their generalization and robustness. For instance,

the prediction results of the two models could be combined through random acceptance or weighted calculations. This experiment provided valuable insights into the strengths and weaknesses of both the custom CNN model and the ResNet-18 model. In future studies, more complex models could be explored to improve the performance of image multi-classification tasks.

5.References

[1] Vijayaraj, A. et al. (2022) 'Deep Learning Image Classification for Fashion Design', Wireless communications and mobile computing, 2022, pp. 1–13. doi:10.1155/2022/7549397.

[2] K V, G. and K, S. (2019) 'Hyperparameter Optimization and Regularization on Fashion-MNIST Classification', International journal of recent technology and engineering, 8(2), pp. 3713–3719. doi:10.35940/ijrte.B3092.078219.

[3] Sumera, S. et al. (2022) 'Implementation of CNN and ANN for Fashion-MNIST-Dataset using Different Optimizers', Indian journal of science and technology, 15(47), pp. 2639–2645. doi:10.17485/IJST/v15i47.1821.

[4] Tang, Y., Cui, H. and Liu, S. (2020) 'Optimal Design of Deep Residual Network Based on Image Classification of Fashion-MNIST Dataset', Journal of physics. Conference series, 1624(5), p. 52011–. doi:10.1088/1742-6596/1624/5/052011.

[5] Chou, F.-I. et al. (2019) 'Optimizing Parameters of Multi-Layer Convolutional Neural Network by Modeling and Optimization Method', IEEE access, 7, pp. 68316–68330. doi:10.1109/ACCESS.2019.2918563.

6. Appendix

Lear n_ra te	Batc h_si ze	Dropo ut	Train _acc	Tra in_l oss	Test _acc	Test_loss
0.05	128	0.5	0.932	0.1 862	0.90 8	0.27

Table1

Learning rate	Train_acc	Train_loss	Test_acc	Test_loss
0.1	0.932	0.1883	0.907	0.261
0.05	0.932	0.1862	0.908	0.270
0.003	0.827	0.4754	0.827	0.475
0.0005	0.413	1.5857	0.534	1.378

Table2

Batch size	Train_acc	Train_loss	Test_acc	Test_loss
64	0.874	0.3517	0.867	0.366
128	0.932	0.1862	0.908	0.27
256	0.764	0.6228	0.771	0.592

Table3

Dropout	Train_acc	Train_loss	Test_acc	Test_loss
0.05	0.841	0.4402	0.830	0.470
0.1	0.838	0.4402	0.843	0.434
0.3	0.836	0.4485	0.832	0.464
0.5	0.932	0.1862	0.908	0.27
0.7	0.812	0.5102	0.817	0.490

Table4