National Tsing Hua University
Fall 2023 11210IPT 553000
Deep Learning in Biomedical Optical Imaging
Homework 3

CHIH-YING,LIEN[1]

[1] *Institute of Photonics Technologies, National Tsing Hua University, Hsinchu 30013, Taiwan*
Student ID:110066520

1. Introduction

In this report, we introduced the Convolutional Neural Networks (CNN) model, it is a specialized neural network designed for tasks involving data with localized features, which leverages techniques like weight sharing and pooling to enhance efficiency and generalization.

2. Task A: *Reduce Overfitting*

2.1 Introduction

First, I used the initial code of Conv model and then tested 3 times, the model accuracy and loss show in the Fig. 1 (a) and Fig. 1 (b), respectively. During these tests, the training accuracy consistently reached 100% after approximately 10 epochs. However, the validation accuracy all remained 97 %. Furthermore, the validation loss consistently exceeded 0.1, which is higher than the train loss. Subsequently, I processed the test dataset, the test accuracy remained at around 73% for all tests. Based on these results, it is evident that this model is experiencing overfitting, because overfitting occurs when a model performs exceptionally well on the training data but struggles to generalize to unseen data.

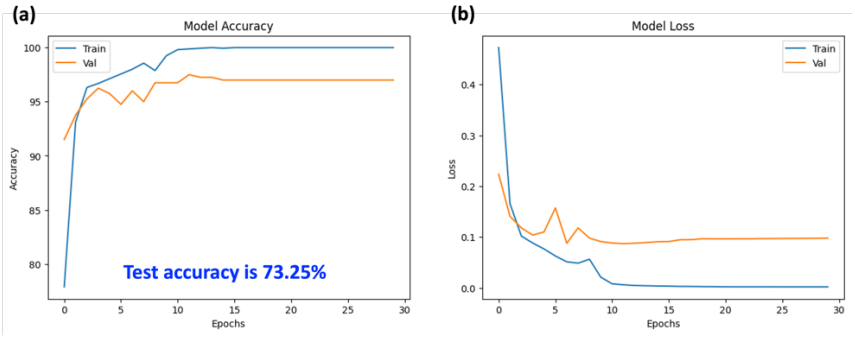

Fig. 1 (a) and (b) model accuracy and loss of initial Conv model, respectively.

2.2 Results and Analysis

In this task, I experimented with techniques such as introducing weight decay, extending the number of training epochs, adjusting the step size, and implementing batch normalization. However, these adjustments did not yield significant improvements in the results. Finally, I added the dropout in the model. The results of model accuracy and loss are shown in Fig. 2 (a) and (b) for dropout 0.5, and Fig. 2 (c) and (d) for a dropout of 0.3. I found that the train accuracy become lower in both cases, with training accuracy reaching 98% when applying a dropout of 0.5 and 99.12% when using a dropout of 0.3.

This is because there only a subset of neurons is active for each training iteration, preventing the model from overfitting. As the results, the training loss increases when compared to models without dropout due to not overfitting. Regarding test accuracy, we obtained results of 77.75% with a dropout 0.5 and 80% with a dropout 0.3. This is once again proved that dropout in preventing overfitting and enhancing the model's

generalization, allowing it to better accommodate unseen data. Consequently, these improvements lead to better test accuracy when adding dropout 0.3. To verify this, I repeated the process 4 more times, shown in Fig. 3 (a) to (d), test accuracies consistently exceeding 78.25%. Fig. (4) is the sections of code adding dropout.

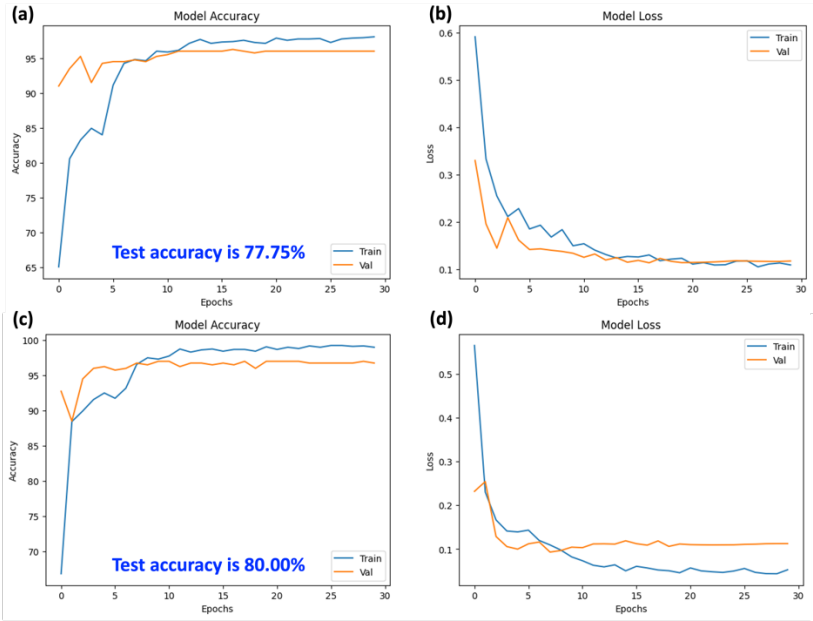

Fig. 2 (a) and (b) model accuracy and loss of adding dropout 0.5, respectively; (c) and (d) model accuracy and loss of adding dropout 0.3, respectively.

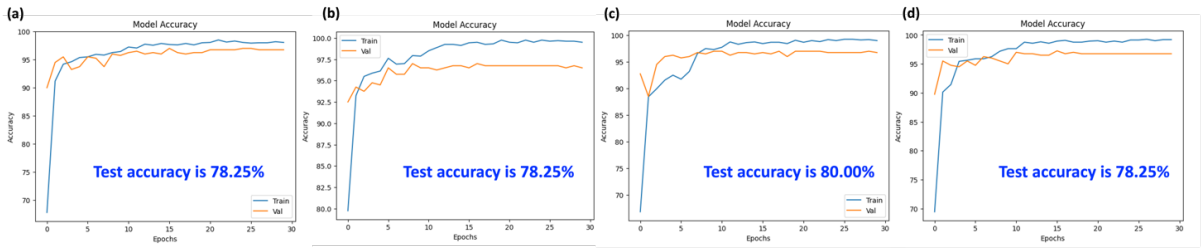

Fig. 3 (a) to (d) Results of four tests for model accuracy and loss of adding dropout 0.3

```python
class ConvModel(nn.Module):
    def __init__(self):
        super().__init__()

        # 1 channel, and using 3x3 kernels for simplicity, 256*256
        self.conv1 = nn.Conv2d(1, 32, kernel_size=3, stride=1, padding='same')
        self.pool1 = nn.MaxPool2d(kernel_size=2, stride=2) # 128*128

        self.conv2 = nn.Conv2d(32, 32, kernel_size=3, stride=1, padding='same') # 128*128
        self.pool2 = nn.MaxPool2d(kernel_size=2, stride=2) # 64*64

        self.conv3 = nn.Conv2d(32, 32, kernel_size=3, stride=1, padding='same') # 64*64
        self.pool3 = nn.MaxPool2d(kernel_size=2, stride=2) # 32*32

        # Adjust flattened dimensions based on the output size of your last pooling layer
        flattened_dim = 32 * 32 * 32

        self.fc1 = nn.Linear(flattened_dim, 32)
        self.dropout1 = nn.Dropout(0.3)
        self.fc2 = nn.Linear(32, 1)
        self.dropout2 = nn.Dropout(0.3)
```

```python
def forward(self, x):
    x = F.relu(self.conv1(x))
    x = self.pool1(x)

    x = F.relu(self.conv2(x))
    x = self.pool2(x)

    x = F.relu(self.conv3(x))
    x = self.pool3(x)

    # Flatten the output for the fully connected layers
    x = x.reshape(x.size(0), -1)  # x.size(0) is the batch size
    x = F.relu(self.fc1(x))
    x = self.dropout1(x)

    return self.fc2(x)
```

Fig. 4 Code of ConvModel adding dropout layer

3. Task B: Performance Comparison between CNN and ANN

3.1 Introduction

In this task B, we need to compare the difference between ANN and CNN. The Fig. 5 (a) and (b) are the architectural diagrams of ANN and CNN, respectively. From the architectural diagrams, we can find that

ANN typically consists of an input layer, hidden layers, and an output layer and CNN comprises convolutional layers, pooling layers, and fully connected layers. Moreover, convolutional layers are used to automatically extract features from images, pooling layers are employed to reduce spatial dimensions of data and decrease computational complexity, finally, the fully connected layers are utilized to map the extracted features to output categories. Therefore, in my point of view, CNN could perform well than ANN, especially when the large-scale data.

3.2 Results and Analysis

Fig. 6 (a) and (b) are the model accuracy and loss of the ANN and Fig. 6 (c) and (d) are the model accuracy and loss of the CNN. From the train and validation accuracy, CNN shows better performance than ANN, the train and validation accuracy of CNN are 4 % higher than ANN, as well as train loss of CNN is around 0.002 and ANN is around 0.01. Furthermore, the test accuracy of CNN is 2 % higher than ANN. This is because CNN use convolutional layers to process input data, meaning that each neuron is only connected to a small part of the input, which is able to capture local features in the input data, which is advantageous for our task of detecting pneumonia.

However, in our model, the training speed of CNN is slower than ANN, CNN takes 72 seconds ANN takes only 11 seconds. I think this is because our CNN model includes 32 convolution and pooling layers, which is more are complex than ANN and require more computational resources to process image data.

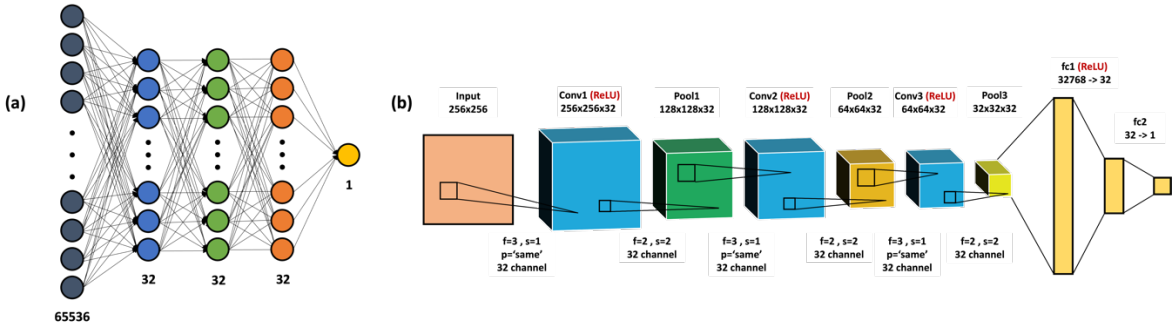

Fig. 5 Architectural diagrams of (a) ANN and (b) CNN

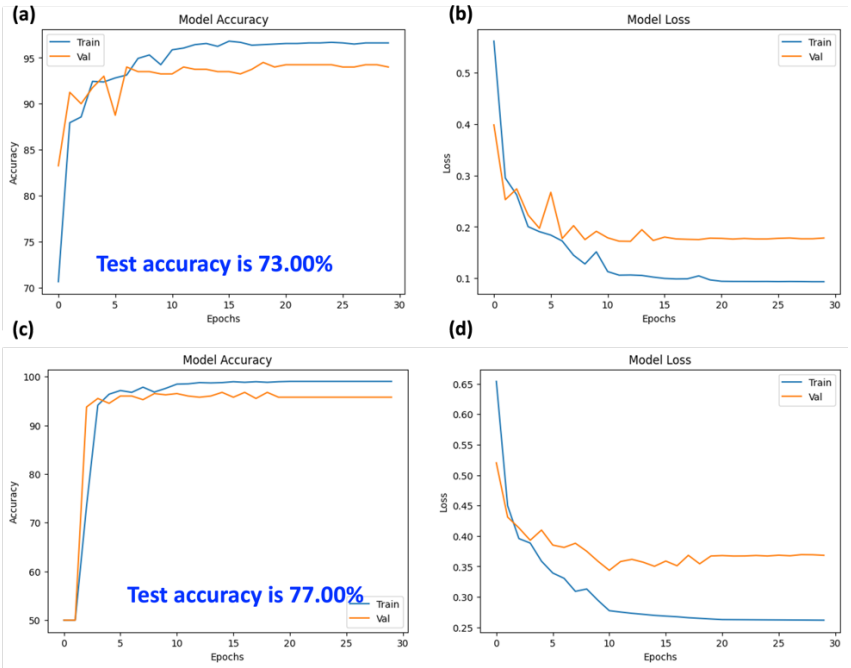

Fig. 6 (a) and (b) Model accuracy and loss of ANN, respectively; (c) and (d) Model accuracy and loss of CNN, respectively

4. Task C: Global Average Pooling in CNNs

4.1 Introduction

Global Average Pooling (GAP) is a pooling layer that computes the average of values in each channel of a feature map, effectively transforming the entire feature map into a fixed-size vector. Therefore, the fixed-size output consistent regardless of the input size or shape, eliminating the need for manual dimension calculations.

4.2 Results and Analysis

In my point of view, calculating the average could diminish the details of features. Therefore, my first consideration is to decrease the convolution kernel size from 3x3 to 2x2, making the train model more sensitive to details. The Fig. 7 (a) and (b) show the initial GAP (convolution kernel size: 3x3) result of the model accuracy and loss, respectively. Then we changed the convolution kernel size to 2x2 and I did it twice, the model accuracy are shown in Fig. 8 (a) and (c), the model loss are shown in Fig. 8 (b) and (d). Although the results of test/validation accuracy and loss didn't have obviously improvement, the test accuracy is improved from 66% to 78%, I think this is because the smaller convolution kernel size can better capture local details of features since they only cover a smaller area, enhancing the model's sensitivity and generalization to subtle features in images.

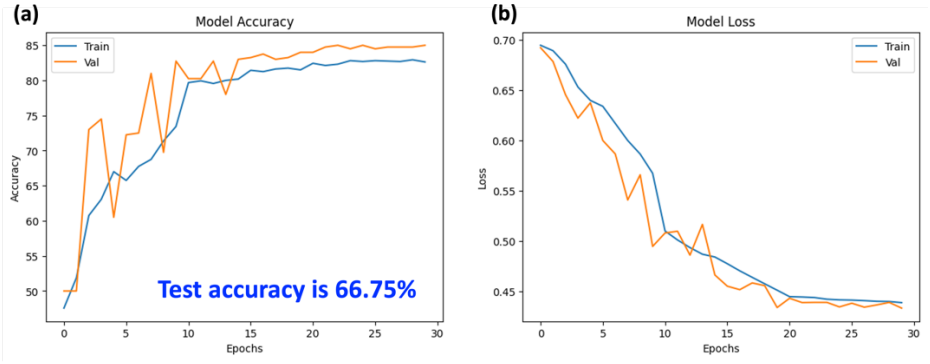

Fig. 7 (a) and (b) model accuracy and loss of initial Conv model, respectively.

Besides, I think adding batch normalization could enhance the stability of the training and validation processes because the data entering the GAP layer exhibits a more consistent distribution, ensuring that the processing of different features occurs on a similar scale. This helps prevent the neglect of certain features during the GAP process, as they have all undergone standardized treatment. The Fig. 9 shows the results of changing the convolution kernel size to 2x2 and applying batch normalization. As shown in Fig. 9 (a) and (c), we observe substantial enhancements in both training and validation accuracy, both increasing from approximately 84% to 96%. Furthermore, the training and validation losses have also improved noticeably, decreasing from around 0.4 to 0.1, as shown in Fig. 9 (b) and (d). Additionally, the test accuracy has seen a marginal improvement, rising from approximately 78% to 79%. The results shows that the batch normalization indeed could enhance the stability of the training and validation processes.

Overall, in our dataset, I believe that using GAP might not be the most suitable choice for our task, which involves classifying pneumonia. In this task, the goal is to identify a specific region within the images, whereas GAP is designed to capture overall features of feature maps. When it comes to discerning local features, it is more appropriate to employ GMP because it is preferably used to emphasize the most prominent feature locations.

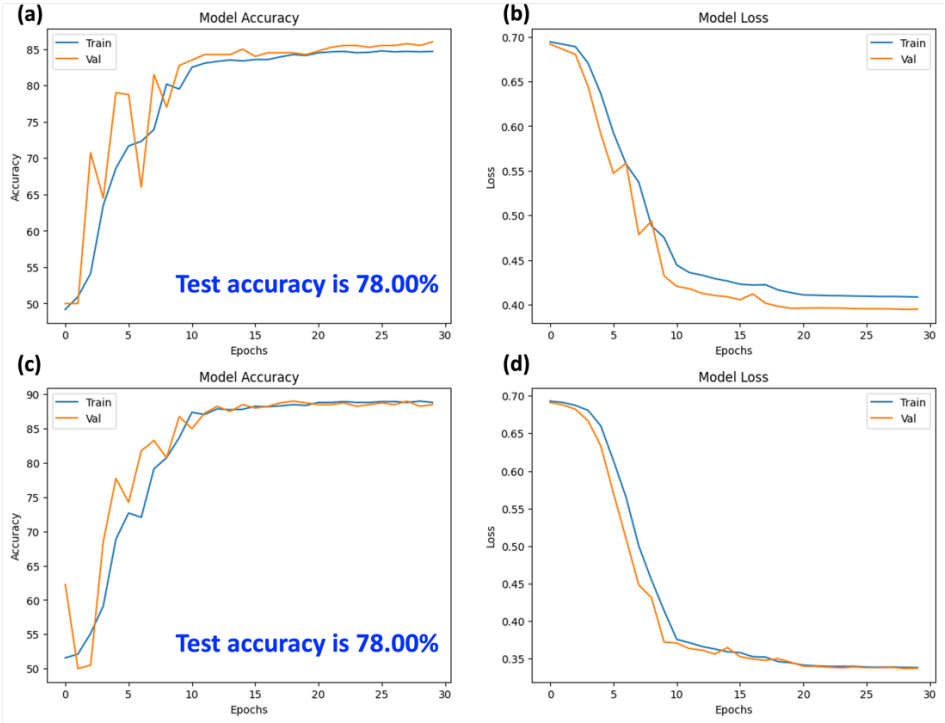

Fig. 8 Results of two tests for changing kernel size to 2x2, (a) and (c) are model accuracy, and (b) and (d) are model loss.

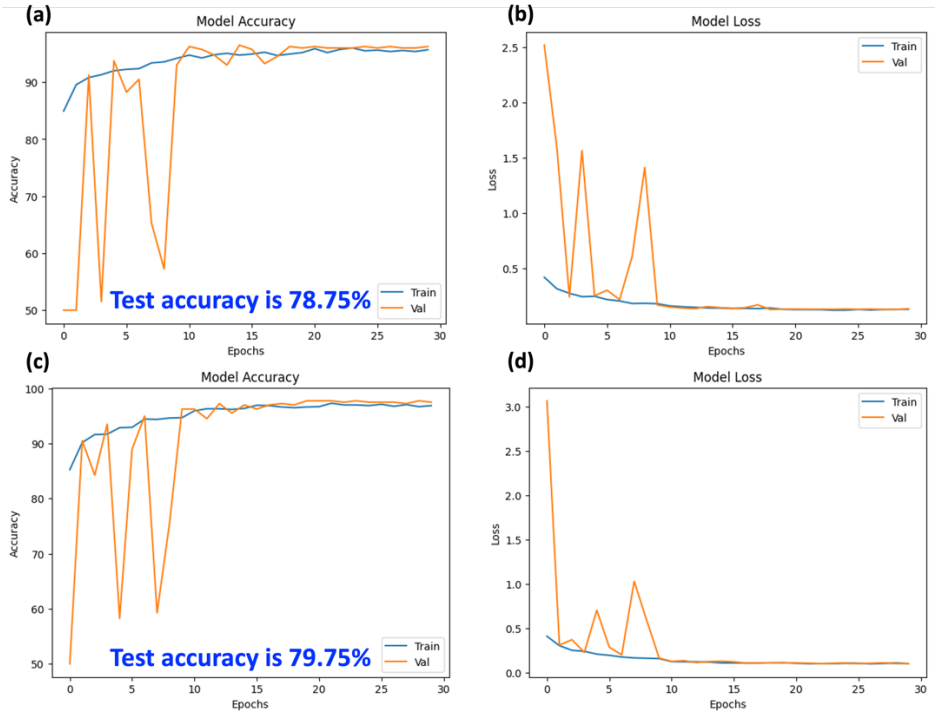

Fig. 9 Results of two tests for adding batch normalization and changing kernel size to 2x2, (a) and (c) are model accuracy, and (b) and (d) are model loss.

5. Conclusion

In this report, we utilize the CNN model. In task A, we found that the model is overfitting because the model performs exceptionally well on the training data. Therefore, I introduced dropout in the model, the results showed that the test accuracy increased from 73.25% to 80% when adding dropout 0.3, demonstrating dropout's effectiveness in preventing overfitting and improving generalization.

In task B, we compared the difference between ANN and CNN, in our results showed that CNN had better performance than ANN, model accuracy as well as loss. This is because CNN use convolutional layers to process input data, which can capture local features in the input data, which is advantageous for our task of detecting pneumonia. However, I think the training speed of CNN is slower than ANN is because our CNN model includes 32 convolution and pooling layers, which is more are complex than ANN.

In task C, we utilized the GAP in our CNN model. However, I think the concept of calculating the average could diminish the details of features. Therefore, I first changing the kernel size from 3x3 to 2x2, although test/validation accuracy and loss didn't have obviously improvement, the test accuracy is improved from 66% to 78%. I think smaller convolution kernel size would better capture local details of features since they only cover a smaller area. Moreover, I added batch normalization because I want the data entering the GAP layer exhibits a more consistent distribution to prevent the neglect of certain features. The test/validation accuracy and loss have incredibly improvement as well as test accuracy. Overall, I think GAP is not suitable to use in our task because we need to classifying pneumonia which is to identify a specific region within the images, whereas GAP is designed to capture overall features of feature maps. Therefore, employing GMP is preferably which can emphasize the most prominent feature locations.