

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

Name: 吳欣鴻

Student ID: 112066523

Task A: Reduce Overfitting

1. Analyze the observed effects of the implemented technique.

The original outcome I discovered before optimizing was not regular and I couldn't watch a steady trend in both 'Train Accuracy v.s. Validation Accuracy' and 'Train Loss v.s. Validation Loss' diagrams. This means there is overfitting and we should do some adjustment in parameters. There are several techniques and methods that can be implemented to mitigate overfitting:

(1) Dropout

Dropout is a regularization technique that randomly sets a fraction of neurons to zero during each training iteration. This prevents the network from relying too heavily on specific neurons and forces it to learn more robust features, which include reduced overfitting, slight increase in training loss, improved validation accuracy.

(2) Data Augmentation

Data augmentation involves applying random transformations (e.g., rotation, scaling, flipping) to the training data. This increases the diversity of the training dataset.

(3) Weight Regularization

Weight regularization adds a penalty term to the loss function based on the magnitude of model weights. This discourages overly large weights, which can lead to overfitting.

(4) Early Stopping

Early stopping involves monitoring the validation loss during training and stopping when it starts to increase, including better generalization, and decreased training accuracy.

When training a deep neural network, we can also consider the following two ways:

(1) Focus on training one model

(2) Use a lot of calculations to train a large number of models at the same time

2. Implementation Visualization

```
epochs = 50 #  
  
optimizer = optim.Adam(model.parameters(), lr=1e-5) #  
# lr_scheduler = CosineAnnealingLR(optimizer, T_max=len(train_loader)*epochs, eta_min=0)  
lr_scheduler = StepLR(optimizer, step_size=15, gamma=0.1) #
```

(1) Epochs:

An epoch is one complete pass through the entire training dataset during the training of a neural network.

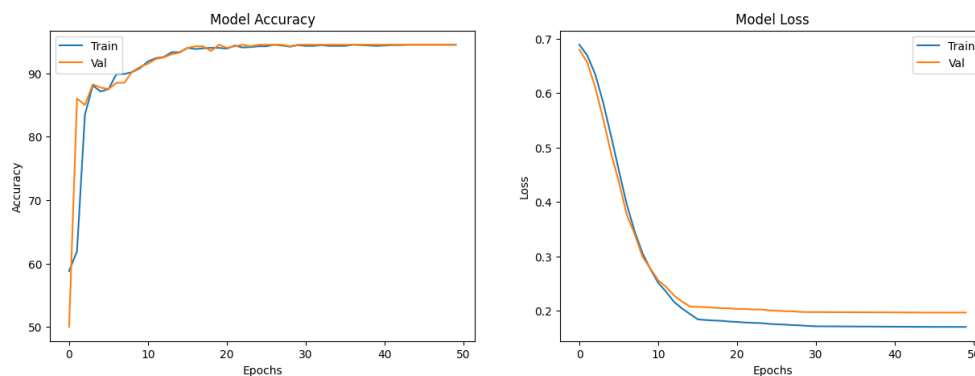
(2) Optimizer:

The optimizer is an algorithm that adjusts the model's parameters (weights and biases) during training in order to minimize the loss function.

(3) Step Size:

The 'step_size' in the context of your code is used for a learning rate scheduler.

I improved the accuracy and loss through change these three variables, and then I can get a better result.



Train accuracy: 85.25% ; Validation accuracy: 85.75%

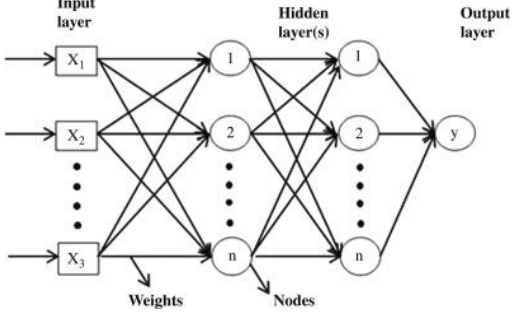
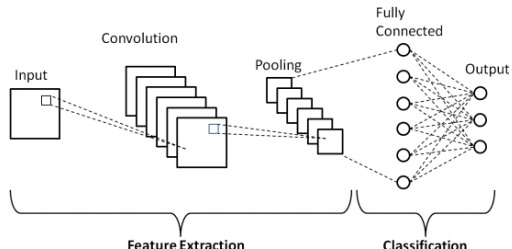
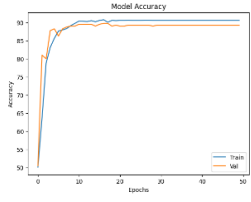
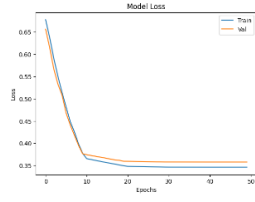
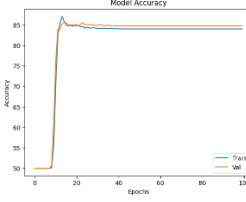
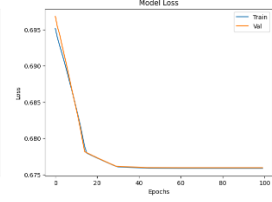
Train loss: 0.3648 ; Validation loss: 0.3391

Additionally, the result of test accuracy of trained model:

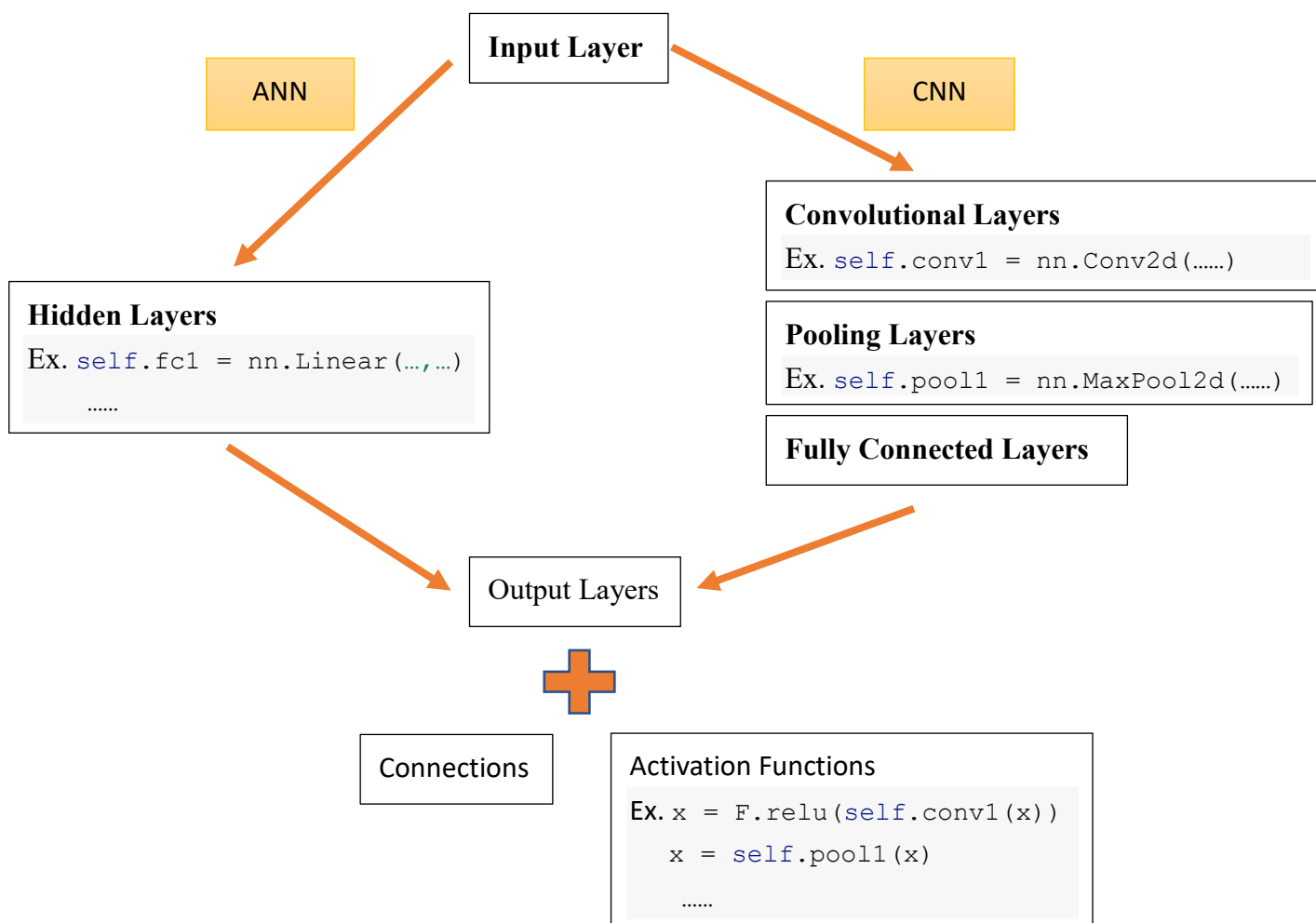
Test accuracy is 78.75%

Task B: Creating an Evaluation Code

1. The differences between CNN and ANN

	ANN	CNN
Feature Extraction Capabilities	more general-purpose and require manual feature engineering	highly specialized for grid-like data and designed to automatically learn hierarchical features
Training Speed on Image Data	slower due to their larger number of parameters and the absence of weight sharing.	faster due to their architecture's weight sharing, which reduces the number of parameters
Model Performance	(1) less effective on image data due to their inability to capture spatial features (2) more suitable for non-image data	(1) tend to outperform ANNs on image-related tasks due to their superior feature extraction capabilities. (2) achieve state-of-the-art performance
Schematic diagram of architecture		
Accuracy & Loss	Train accuracy: 89.75% ; Validation accuracy: 89.0% Train loss: 0.3898 ; Validation loss: 0.3990 <div style="display: flex; justify-content: space-around;">   </div>	Train accuracy: 84.0% ; Validation accuracy: 85.5% Train loss: 0.6759 ; Validation loss: 0.6760 <div style="display: flex; justify-content: space-around;">   </div>

2. The architectures of ANN and CNN models



Task C: Global Average Pooling in CNNs

1. The role of GAP

Global Average Pooling (GAP) is a technique commonly used in convolutional neural networks (CNNs) to reduce the spatial dimensions of feature maps in the final convolutional layer and prepare the model for classification. In short, it's a method that can replace FC.

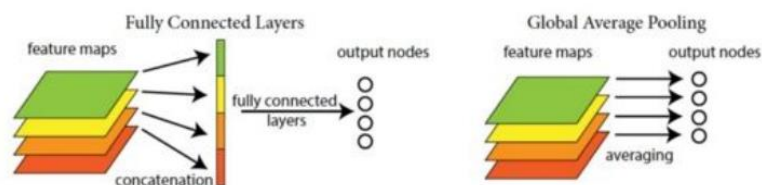


Fig. The schematic diagram of two different process.

With GAP, there is no need for manual dimension calculations. The number of neurons in the fully connected layers is determined directly from the number of feature maps produced by the last convolutional layer. The main thing is to reduce parameters and reduce overfitting.

2. Performance

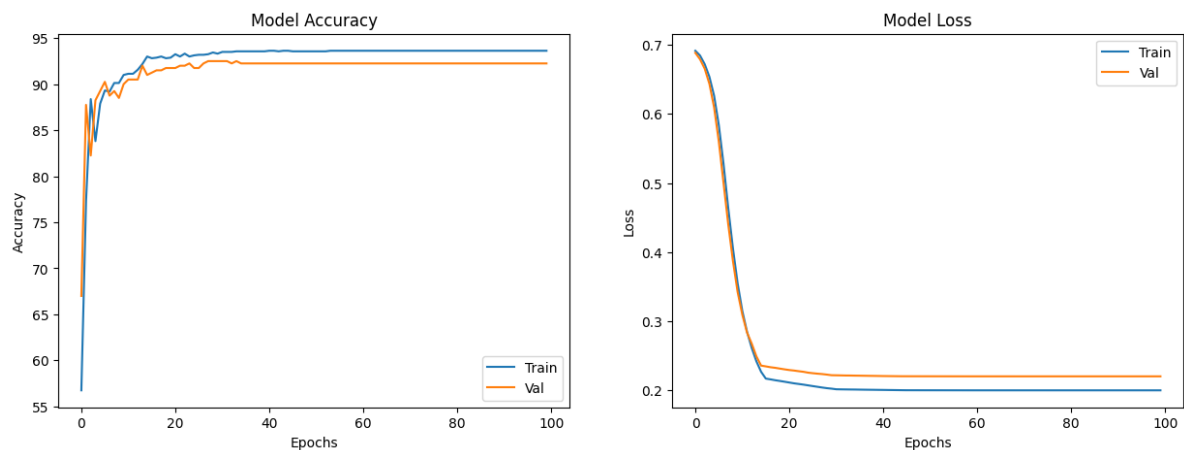
To improve the result of GAP, I tried to change the values inside the 'epoch', 'optimizer', and 'lr_scheduler'. As a result, we can get a better outcome and we can successfully improve the test accuracy.

(1) The modification of parameters:

```
epochs = 100 #
optimizer = optim.Adam(model.parameters(), lr=1e-5)
# lr_scheduler = CosineAnnealingLR(optimizer, T_max=len(train_loader)*epochs, eta_min=0)
lr_scheduler = StepLR(optimizer, step_size=15, gamma=0.1)
```

Epoch = 100 ; optimizer_lr = 10^{-5} ; step_size = 15

(2) The result of model accuracy and model loss:



Train accuracy: 93.62% ; Validation accuracy: 92.50%

Train loss: 0.1999 ; Validation loss: 0.2201

(3) The result of test accuracy of trained model:

Test accuracy is 80.0%

Apparently, the result I've tried couldn't be near our most ideal condition.

Moreover, I discovered that every time I tried to run it again, the validation loss would increase a little gradually.