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

陳芷韻 *Student ID:112066514*

1. The model developed process

In this report, we aim to train a CNN model for the classification of histological images of cancerous tissues. Each sample in the dataset is a 150x150 pixel RGB image. During the development, I initially employed a simple three-layer convolutional neural network. Surprisingly, the training accuracy easily reached 99%, but the validation accuracy remained below 50%. This indicates a clear case of overfitting. Thus, our focus in this report is on resolving the overfitting issue to improve the validation accuracy. The process of fine-tuning the model is illustrated in Figure 1.

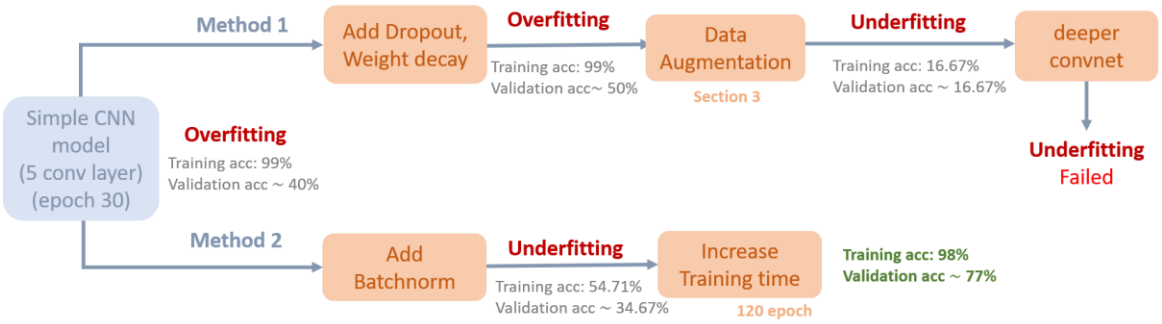


Figure 1. The model developed process

In the first approach, we initially incorporated Dropout and adjusted weight decay (increased regularization) while altering the learning rate scheduler. However, these modifications did not notably improve the validation accuracy. Therefore, we proceeded to attempt data augmentation to increase the size of dataset. However, upon training with the original model, not only did we fail to resolve the issue of overfitting, but instead encountered underfitting. The training accuracy kept at 16% and showed no signs of improvement.

In the end, we abandoned the approach of data augmentation and instead attempted another form of regularization. We integrated batch normalization into the original CNN structure to address the overfitting issue. The detailed architecture and parameters are outlined in Section 2. Despite observing a noticeable increase in validation accuracy from the results, the problem of overfitting persisted.

2. CNN model

2.1 Fixed parameters

The following are the hyperparameters that we fixed constant in the experiments. The nn.CrossEntropyLoss function were used in the classification of six groups. The difference in results between the two learning rate schedulers, CosineAnnealingLR and StepLR, was not significant.

Table 1. Fixed parameters in the development.

Parameters	Model	Loss	Epoch	Batch size	Learning rate scheduler
	CNN	CE	120	32	CosineAnnealingLR

2.2 Final CNN structure

After several attempts, we constructed a framework comprising five convolutional layers, each with a kernel size of 3, and MaxPooling layers with a kernel size of 2. Additionally, we incorporated Batchnorm layers between the convolutional and MaxPooling layers, as Figure 2 shows.

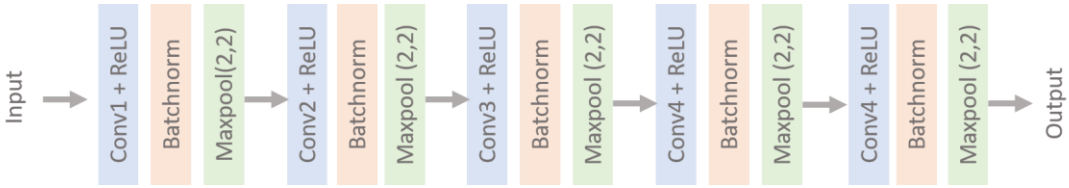
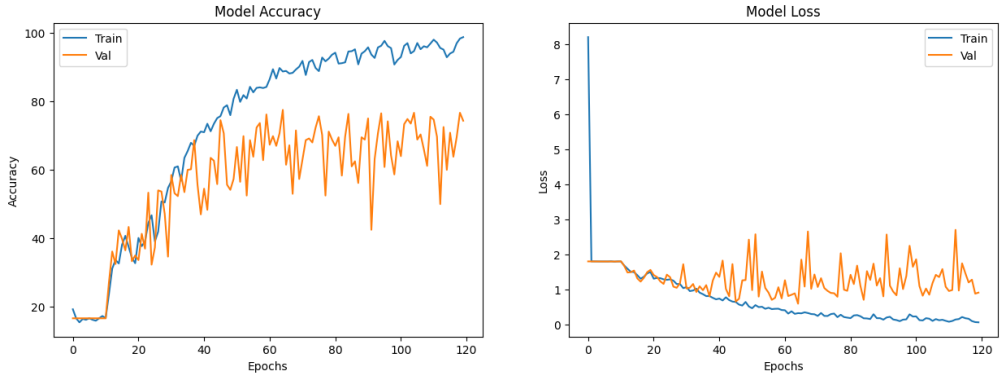


Figure 2. the structure of final CNN model.

The Figure 3 illustrates the evolution of training and validation loss and accuracy, while Table 2 records the final data.

Performance:



(a) Model accuracy evolution (b) Model loss evolution

Figure 3. Training and validation evolutions with ConvNet model.

Roughly examining the results from Figure 3, we observe a gap between the training and validation outcomes. It's evident that when the epoch reaches 100, the training accuracy surpasses 95%. However, the validation accuracy shows no upward trend beyond epoch 50 and exhibits significant oscillations. Clearly, we haven't entirely resolved the overfitting issue.

Table 2. Training results.

Model	Train Loss	Train Accuracy	Best Val Loss	Best Val Accuracy	Test Accuracy	Training time
ConvNet	0.0545	98.75%	0.5864	77.50%	80.17 %	24:05

From the table, it's apparent that training the convolutional network with five big layers alone requires over 20 minutes. Furthermore, the test accuracy stands at only 80%, which is consistent with the performance observed in validation.

3. Data Augmentation Test

Next, we attempted data augmentation using PyTorch's transforms.RandAugment() and transforms.RandomHorizontalFlip(). The image size was kept at 150x150. The randomly augmented data is depicted in the following figure.

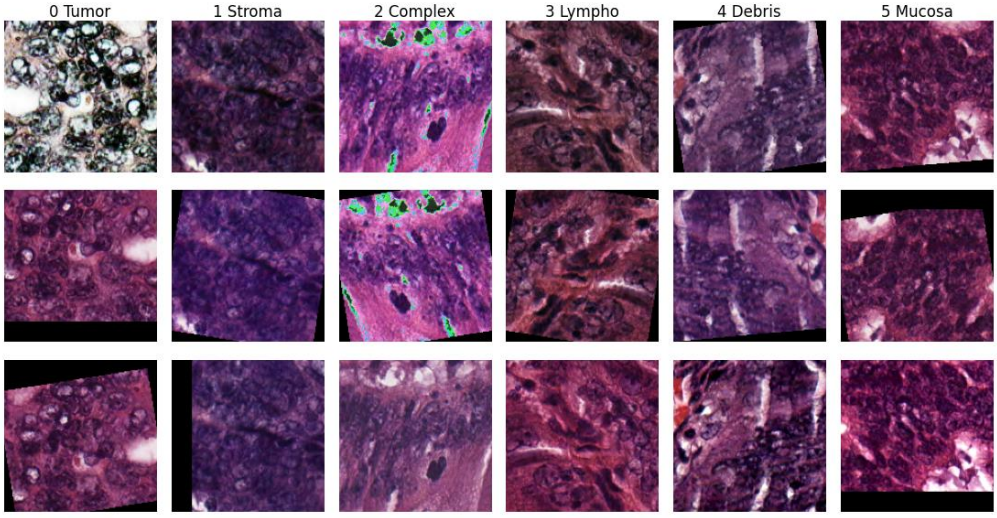


Figure 4. randomly augmented data for six groups.

After data augmentation, the complexity of the data increased, necessitating the creation of a deeper model to accommodate this complexity. Therefore, we progressed from a five-layer convolutional model to a ten-layer one. However, the training accuracy remained at 16.67%. Additionally, when attempting to resize the training data to 256x256, a 'CUDA out of memory' error message was encountered.

Due to unfamiliarity with data augmentation, we opted to abandon this commonly used method for addressing overfitting. Instead, we reverted to the original model and introduced batch normalization layers to tackle the issue.