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

YOUHAN-WEN[1]

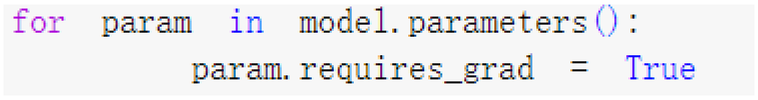
[1] *Institute of Photonics Technologies, National Tsing Hua University, Hsinchu 30013, Taiwan*

Student ID:110066554

1. Abstract

In this final report, we will build a deep-learning model by classifying histological images of cancerous tissues, and the output will be separated into six different tissues that are normally used in identifying cancer histology. First, we will build a simple CNN model with a ResNet18 pre-trained model and a cosine annealing schedule, the output test accuracy reach 83.83%, second, we will implant data augmentation which can help us to reduce the overfitting and increase the diversity of the dataset in the same time, in the result I think I have failed to input the data augmentation into train dataset so the accuracy shows no different.

It is interesting that with the learning rate scheduling technique changing to StepLR because I want to fine-tune the hyperparameter, the result being worse, the test accuracy drops under 75%. So the next change in this report is to swap the pre-trained model to GoogLeNet and continue with the cosine annealing schedule, though we get the test accuracy reach to 93.5%, and the main change we figured out is to set the param.requires_grad to true, which means neural network will keep track of those operations to compute gradients during backpropagation. With this key adjustment, the accuracy can easily break through the 90% chock point.

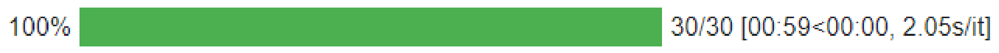


```
for param in model.parameters():
        param.requires_grad = True
```

Fig.1. The key parameter in this report, when the value is false PyTorch will not track operations involving this tensor for gradient computation, but when the value is true PyTorch will keep track of those operations to compute gradients during backpropagation.

2. First version of CNN

With the first version of the CNN, I built with ResNet18 pre-trained model and a cosine annealing schedule with no hyperparameters changing, the test accuracy is 83.83% and the best train and validation accuracy is 92.16% and 84.33%, from fig.2 we can that the overall training time is less than one minute, and from fig.3 it shows the train and validation accuracy has the gap which is around 8%, but there is no overfitting which means it's still a good model.



```
100% 30/30 [00:59<00:00, 2.05s/it]
```

Fig. 2. ResNet18 model training time.

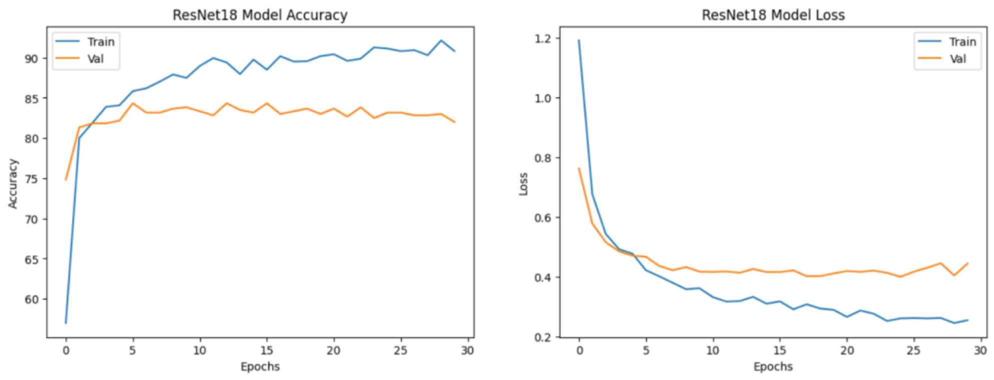
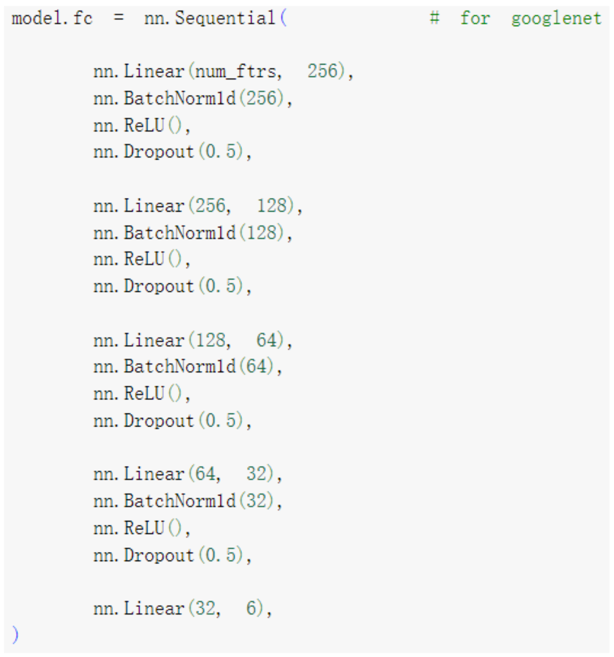


Fig. 3. ResNet18 model, test accuracy is 83.83% and the best train and validation accuracy is 92.16% and 84.33%.

3. Final version of CNN

At the beginning of the try, I added four layers of fully connected layers and used the step scheduler instead cosine scheduler because I wanted to tune the hyperparameter that I was familiar with, but the result was worse than before which test accuracy was under 75%, so next I change the pre-trained model to GoogLeNet instead ResNet18, but the result is still unacceptable, so I realize that neither the problem is pre-trained model nor scheduler I used.

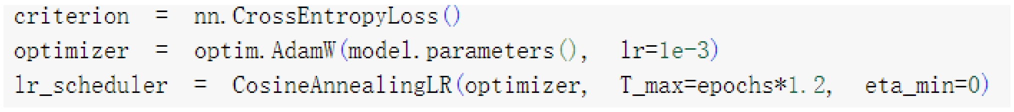
After discussion with classmates, we discovered the key point of this report, that we have to make sure the PyTorch will keep track of tensor operations to compute gradients during backpropagation.



```
model.fc = nn.Sequential(            # for googlenet

        nn.Linear(num_ftrs, 256),
        nn.BatchNorm1d(256),
        nn.ReLU(),
        nn.Dropout(0.5),

        nn.Linear(256, 128),
        nn.BatchNorm1d(128),
        nn.ReLU(),
        nn.Dropout(0.5),

        nn.Linear(128, 64),
        nn.BatchNorm1d(64),
        nn.ReLU(),
        nn.Dropout(0.5),

        nn.Linear(64, 32),
        nn.BatchNorm1d(32),
        nn.ReLU(),
        nn.Dropout(0.5),

        nn.Linear(32, 6),
)
```

Fig. 4. Four layers of fully connected layers, with six outputs of different tissues.

In the final version of this CNN model, I used GoogLeNet as a pre-trained model and tuned the hyperparameter of the cosine scheduler, here the learning rate we fixed at 1e-3, and the hyperparameter of the cosine scheduler we set the T_max at 1.2 times of the epochs, this hyperparameter control how many epochs the cosine annealing schedule should take to complete one cycle, and it can influence the learning rate dynamics during training. This training time took almost three minutes and the test accuracy reached 94.16% and the best train and validation accuracy was 98.24% and 94.17%, here the gap between train and validation was decreased, but the validation loss changed dramatically. So I changed the learning rate to 1e-4, and it became smother, but the test accuracy dropped to 90.33%



```
criterion  =  nn.CrossEntropyLoss()
optimizer  =  optim.AdamW(model.parameters(),  lr=1e-3)
lr_scheduler  =  CosineAnnealingLR(optimizer,  T_max=epochs*1.2,  eta_min=0)
```

Fig. 5. GoogLeNet model hyperparameter setting.

```
100% 30/30 [02:47<00:00, 5.57s/it]
```

Fig. 6. GoogLeNet model training time.

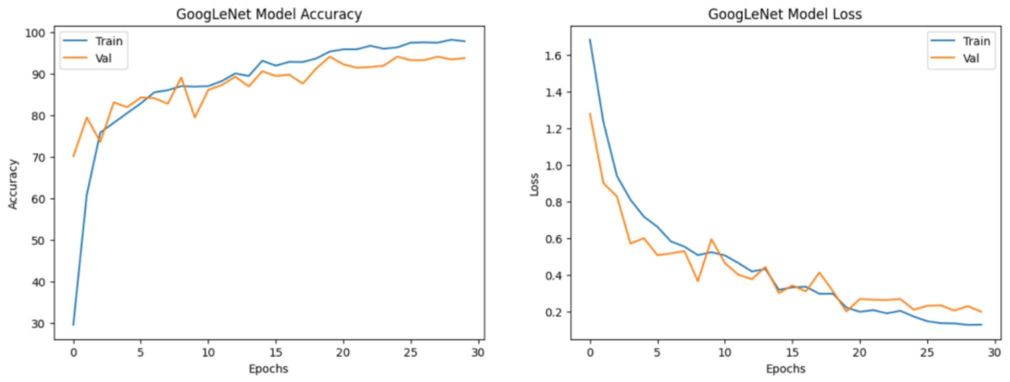


Fig. 7. GoogLeNet model the test accuracy is 94.16% and the best train and validation accuracy is 98.24% and 94.17%.

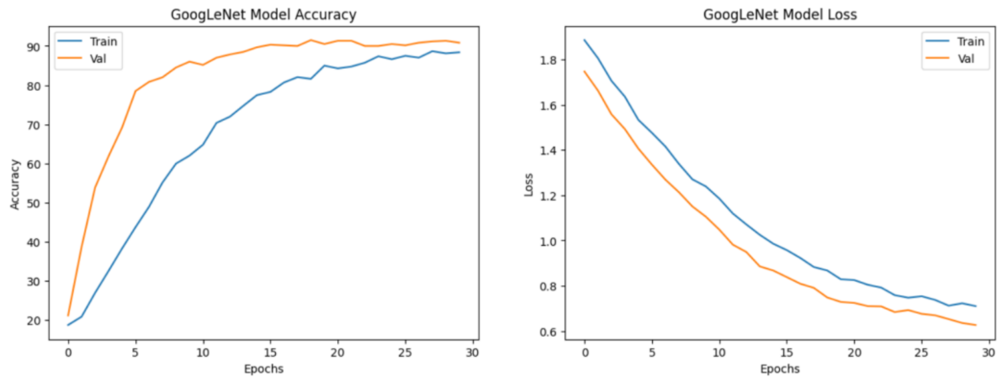


Fig. 8. GoogLeNet model the test accuracy is 90.33%, the vibration decreases but the accuracy also decreases.

In conclusion, this time we get new data on cancer histology to train this time, with the large quantity of the dataset, the accuracy easily reaches higher accuracy than before, and it is joyful to figure out the key point of this report, and learn new scheduler tuning.