National Tsing Hua University
Fall 2023 11210IPT 553000
Deep Learning in Biomedical Optical Imaging
Report: Cancer Histology Image Classification

WEN CHI CHEN[1]

[1]*NTHU IPT*

Student ID: 110066515

1. Samples to be classified

The samples contain 3750 colorectal cancer histopathology images cropped into patches of 150 × 150 pixels labeled into to 6 structure types: tumor, stroma, complex stroma, lympho, debris, and mucosa. The sample dataset has 625 images for each of the 6 structure types, describing a distinct textural feature. For instance, few shape characteristics can be found in the formation of cell nuclei (Fig 1).

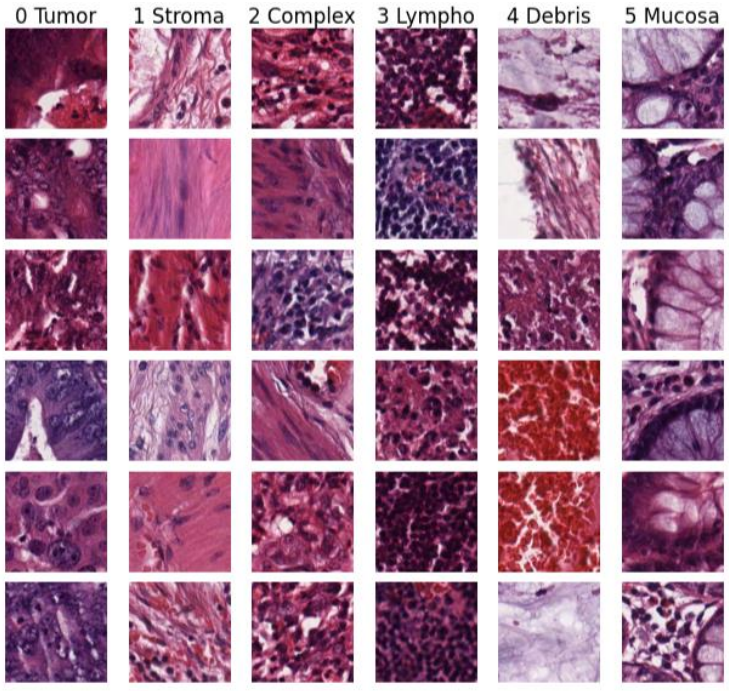

Figure 1. Image from the sample dataset. There are 6 images for each classification: tumor, stroma, complex, lympho, debris, mucosa.

2. Model architecture

The model I used in this report is GooLeNet, based on transfer learning & fine-tuning we have learned in HW4. The model is fine-tuning by adapting the code: "param.requires_grad = True", so the pre-trained GooLeNet model will be slightly trained with my additional fully-connected layer: 2 hidden layers with 1024 ->256 nodes and 256 ->64 nodes, connected with

ReLU() function before the batch normalizations. Note that the output of the entire model should be 6 classes, corresponding to the number of sample classifications. The batch size is set at 64, and the learning rate starts from 0.2e-3 with CosineAnnealingLR with the period that learning rate decays to 0 being 1.2*epoch. The model architecture and the training setting are shown in figure 2.

The hyperparameters I mainly tuned are the learning rate value and its reducing trend, which are the "lr = 0.2e-3" and "T_max = epoch*1.2", respectively. Since this is my first time using CosineAnealingLR in homework/ report, I additionally attach the operation principle of CosineAnealingLR in figure 3

```
import torch.nn as nn
import torch.nn.functional as F
from torchvision import models


model = models.googlenet(weights='IMAGENET1K_V1')

num_ftrs = model.fc.in_features

model.fc = nn.Sequential()

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

model.fc = nn.Sequential(

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

        nn.Linear(256, 64),
        nn.BatchNorm1d(64),
        nn.ReLU(),
        nn.Dropout(0.5),

        nn.Linear(64, 6),
)

print(model)
model = model.cuda()
```

```
epochs = 30
best_val_loss = float('inf')

# Criterion and Optimizer
criterion = nn.CrossEntropyLoss()     # for CE

optimizer = optim.AdamW(model.parameters(), lr=0.2e-3)
lr_scheduler = CosineAnnealingLR(optimizer, T_max=epochs*1.2, eta_min=0)
```

Figure 2. model architecture and the training setting. The model is fine-tuning GoogLeNet with addition FC layers. And the training setting is AdamW optimizer with CosineAnnealingLR learning rate schedule.

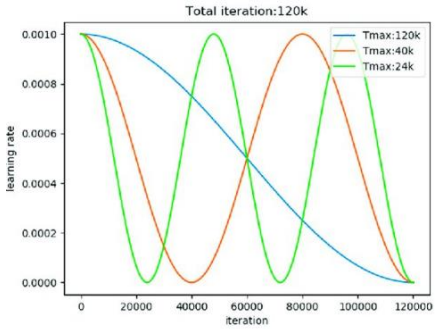

Figure 3. The cosine annealing leaning rate in different Tmax.[2]

3. Training model performance

Figure 4 shows the evaluation metrics and the test result of the model. Due to intentionally reduce the initial learning rate, the evaluation curve is smooth and gradually decreasing in loss function. If the learning rate is too high, the model will converge in very short epoch but with no improvement in validation accuracy, the validation accuracy value oscillating around 90%. However, if the learning rate is too high, model will be stuck in local minimum and finally get ~88% test accuracy. The same idea can be applied to the other hyperparameter: T_max in CosineAnealingLR to reduce either overfitting or underfitting.

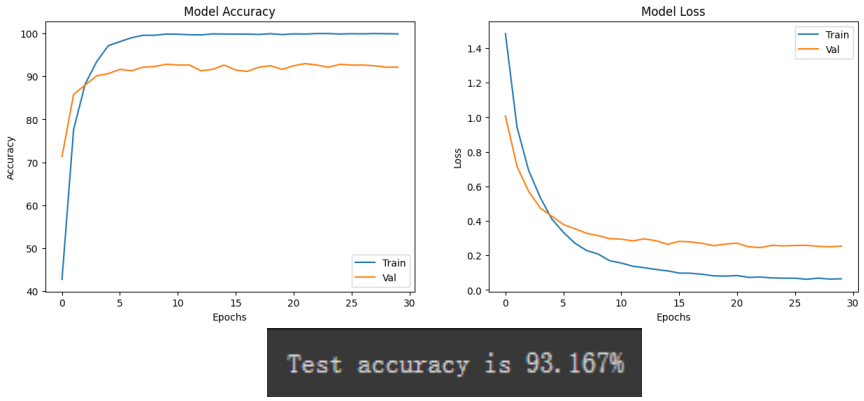

Figure 4. evaluation metrics and the test result of the model

4. Comparison

Here I show some comparisons with tuning the hyperparameters and their results.

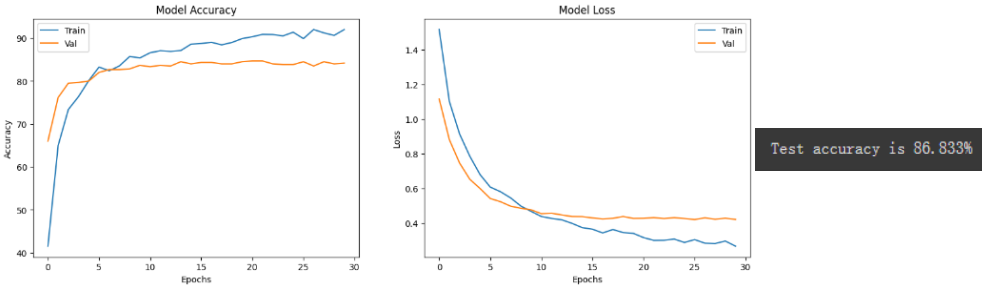

Figure 5. evaluation metrics and the test result of the Fixed Feature Extractor model, lr=0.2e-3

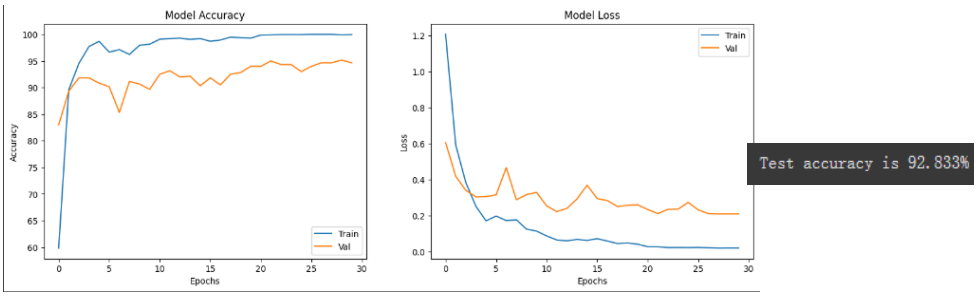

Figure 6. evaluation metrics and the test result of the Fine-Tuning model, lr=0.5e-3

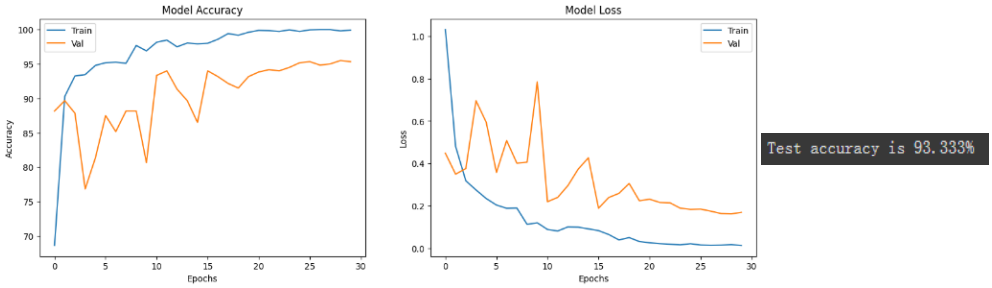

Figure 7. evaluation metrics and the test result of the Fine-Tuning model, lr=1e-3

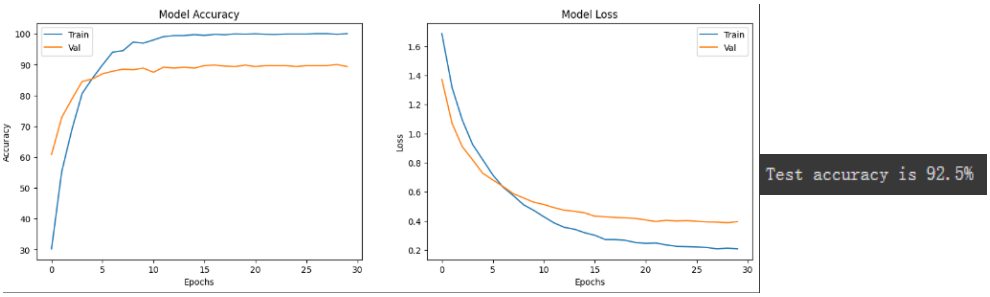

Figure 8. evaluation metrics and the test result of the Fine-Tuning model, lr=0.08e-3

References

1. E-BiT: Extended Bio-Inspired Texture Descriptor for 2D Texture Analysis and Characterization, Steve Ataky Alessandro L Koerich
2. A Robust Fabric Defect Detection Method Based on Improved RefineDet, Huosheng XieZesen Wu