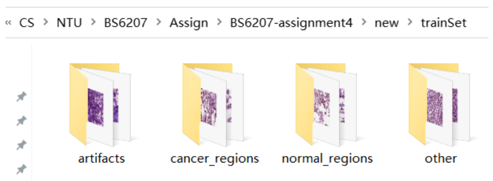
1. Preprocess Data

1.1 Split data

I use 'glob' and 'os' packages to split the data into "trainSet", "validSet","testSet" and "train_valid_set" and the size ratio between train set and valid set is 4:1. In each set, there will be four files shows below. The 'ImageFolder' package will consider the images in the single file as the same label and the label name of them are the name of the file.


« CS › NTU › BS6207 › Assign › BS6207-assignment4 › new › trainSet
artifacts
cancer_regions
normal_regions
other

1.2 Data Normalization

Then, the means and standard deviation(std) of each RGB channels are calculated. The means and stds for these 3 channels are [0.6341895, 0.4980002, 0.64840436], [0.22627775, 0.27285302, 0.22700454], respectively.

1.3 Transform

There are many different methods in pytorch to do the transforms with the train data. 'RandomResizedCrop', 'RandomHorizontalFlip', 'ColorJitter' are used for train data, and 'ToTensor','Normalize(mean,std)' are used in every data set.

RandomResizedCrop was used to randomly crop graph to the scale and resize the scaled graph to the set size.

RandomHorizontalFlip was used to filp the praph horizontally in some probability.

ColorJiter was use to change the RGB channel numbers with different light, saturation and so on.

2. Model Establishment

Two different models are established for selection of different model architectures.

One is the CNN with three residual blocks, each residual block contains 2 residual nets and then followed by a classifier with 2 linear layers with [1024,512,4] dimensions.

The other is the normal CNN with three CNN classical networks (Con2d+BatchNorm+Relu+MaxPooling) followed by a classifier with two linear layers with the same dimension above.

3. Tuning of parameters

3.1 Model Architectures Tuning

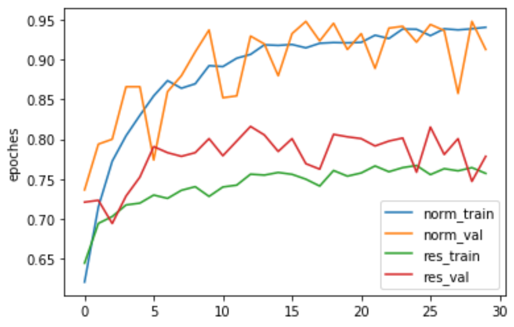


Fig 1

Default Model Setting:

Loss=CrossEntropyLoss

Optimizer=Adam optimizer with 0.01 learning rate.

Early stop is TRUE with 10 patience.

Fig 1 shows that performances of NN with normal Architectures are much better than that of the NN with residual nets. The accuracy of normal network is nearly 0.95 but the accuracy of residual networks only nearly 0.8, which is 15.7% less than the normal model. The reason about this maybe is that the residual network is stuck at the local minima and can not escape or the model is much deeper that cannot be trained effectively.

In this case, residual network is abandoned and the normal network was select to do further study.

3.2 Learning rate tuning

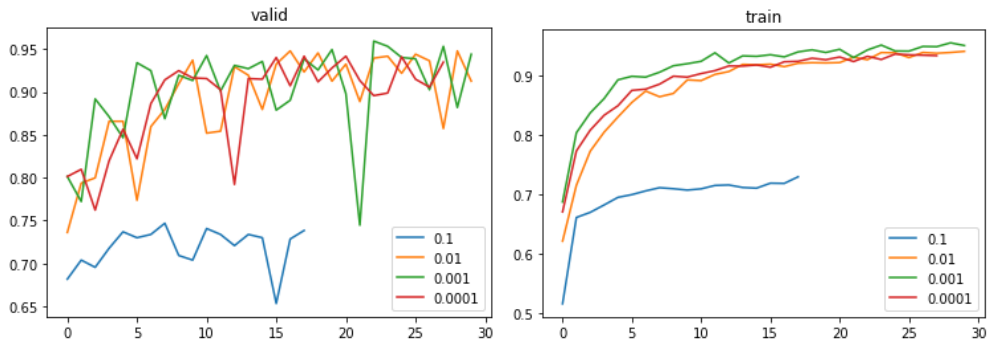


Fig 2

Fig 2 shows the learning rate tuning with 0.1,0.01,0.001,0.0001. It can be seen that learning rate with 0.1 is the worst. Model can hardly trained and the accuracy only reach near 0.75. However, other learning rates are good and 0.001 is the best in those 4 different learning rates. Then 0.001 was chosen for the further tuning.

3.3 Pooling Methods Tuning

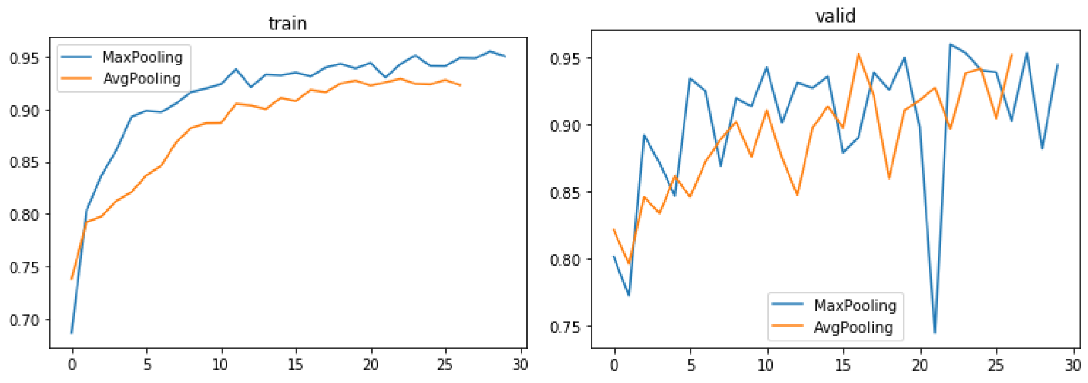


Fig 3

Fig 3 shows the performance of different pooling methods, strides, padding and kernel size between those two are totally same. It shows that max pooing is better than average pooling in this data set. The train accuracy of average pooling only reaches 0.92 but over 0.95 was achieved by max pooling. The performance in valid data set also shows the same result. Then the Max pooling method is chosen for the NN.

3.4 Dropout tuning

Fig 4 shows the performance of the model with different dropout values. It can be seen that when the value of dropout increase, the accuracy of the mode in training data set gradually decrease. Performance of no dropout layer could reach near 0.95. The performance of those 4 in valid dataset is unclear. They all reach the good accuracies.

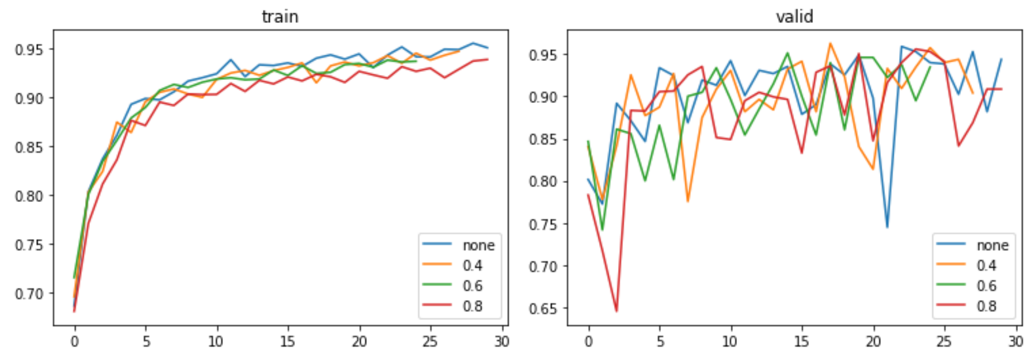
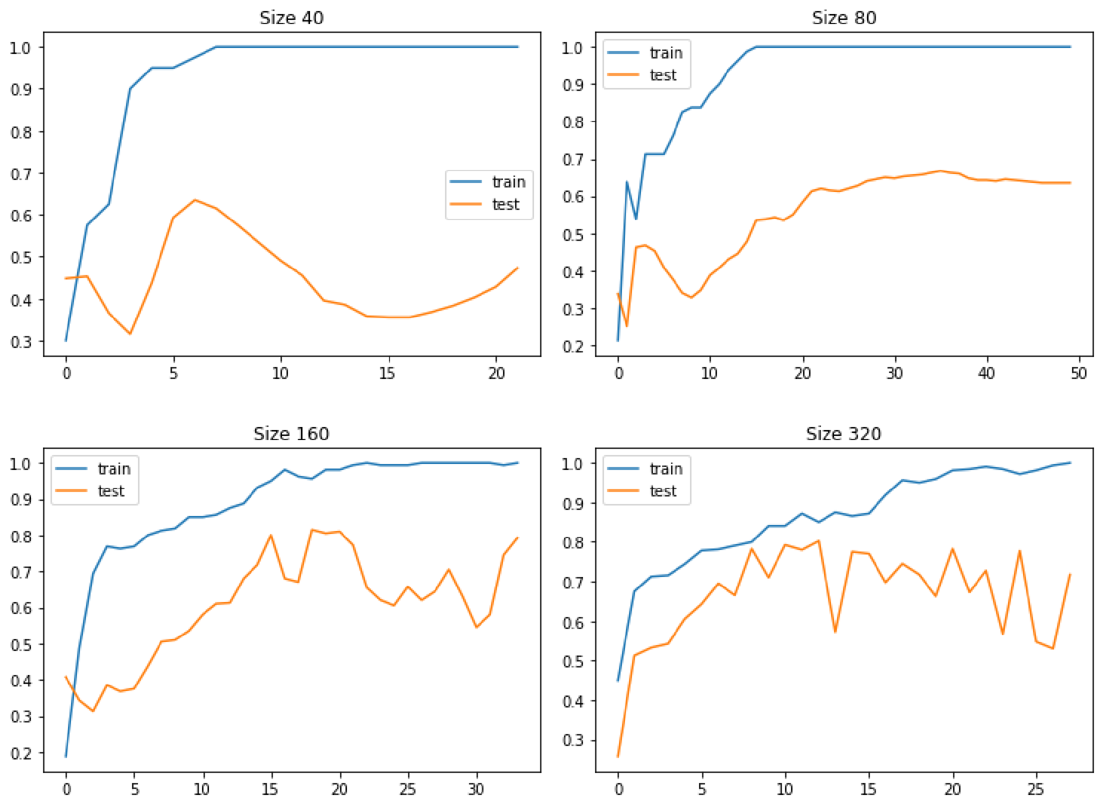


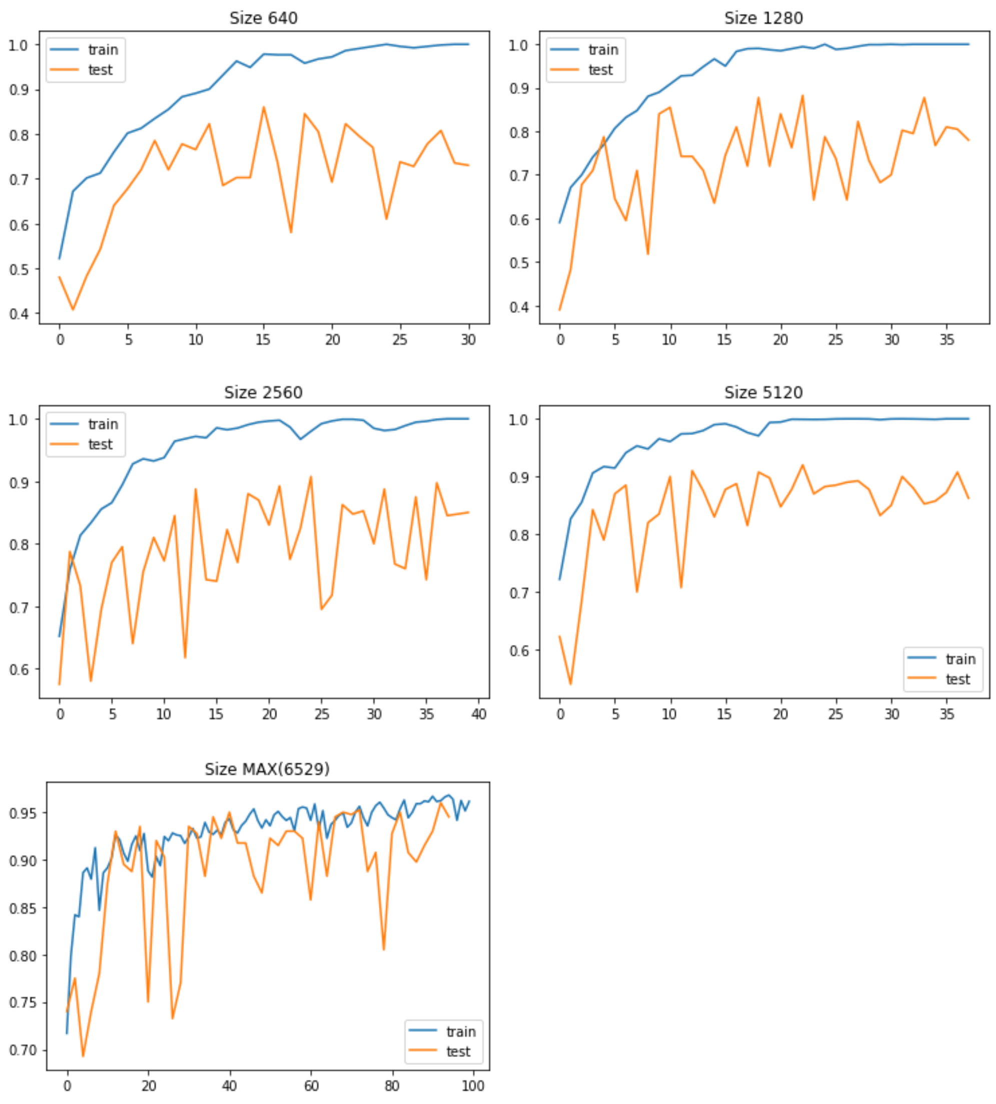
Fig 4

In this case, dropout was not set in this NN model. The potential reason for this is because the size of the data set is large enough for this net work, and transformation was also used to decrease the probability of over-fitting.

4. Test with different size of train set

Different train size was used to test the performance of final model which was tuned above. The train sizes are [40, 80,160,320,640,1280,2560,5120,6529]. The data set with size 6529 is the maximum size with all data except the 400 test data. And other sizes were got exponentially start from 40. Except the maximum size, all the network suffers the over fitting problem. The reason about that is I randomly choose the data with the size 40 to 5120 from the maximum 6529 data set. In this case, they are not generated by 'DataLoader' package in the second epoch, and then they do not get the 'Transforms' functions in the second epoch. This can also show that the 'transforms' package is a good tool to avoid over fitting.





The other reason about this is also I randomly chose the data in total 6529 data set. In this case, the data may not contain all 4 different labels. This is obvious and the result will be corrected later.

Fig 5 shows the best accuracy of different train set size. The result shows that performance will increase rapidly along with the increasing of the train set size when the data set size is small. Then the performance will reach a platform with the increasing of train set size when the data size is already large enough.

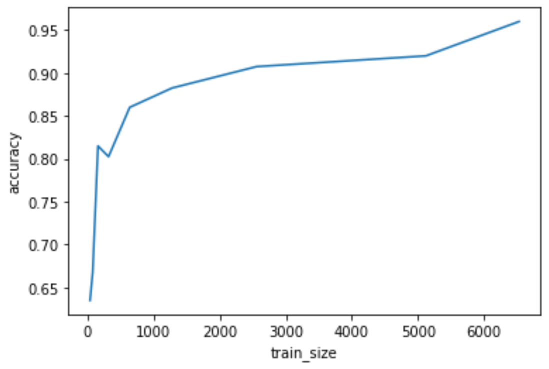


Fig 5