Deep Learning Project Report

workload

胡圣然 11711011

Package and preprocess the data. Contribute to the training and testing framework. ROC plot drawing code.

朱以辰 11711512

Efficient Net model, which achieve highest accuracy in our group. Model ensembling. Testing the model.

喻家骏 11711102

CNN + Perceptron model. Contribute to handling the meta data.

吴江宁 11710217

DenseNet model. Contribute to training framework.

向昕昊 11712617

GoogLeNet model. Contribute to training framework.

models

Efficient Net

EfficientNet-B0

Structure

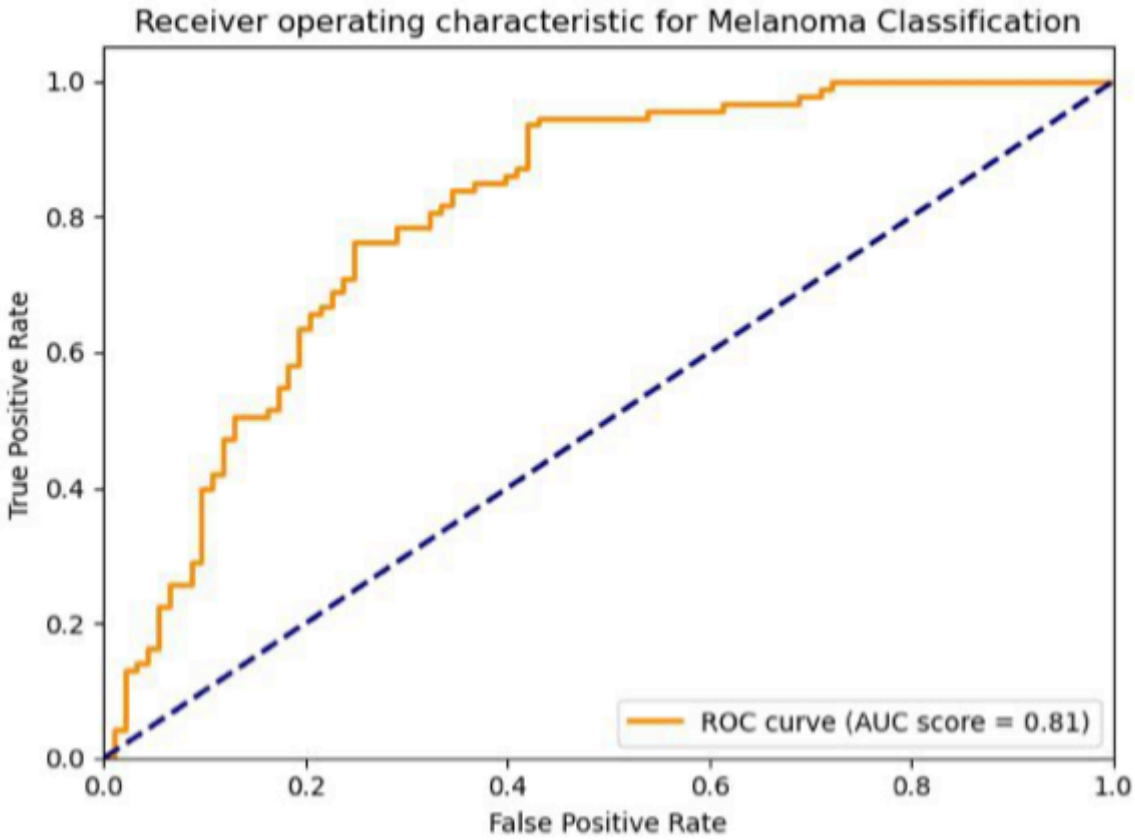
Conv2d
Batch Normalization
Swish
Blocks with expansion + depthwise + pointwise + squeeze-excitation
Conv2d
Batch Normalization
Swish
Global Average Pooling
Dropout
FC
SoftMax

Settings:

Learning rate = 0.01 with CosineAnnealingLR scheduler SGD optimizer with momentum=0.9 and weight decay=5e-4

Result Accuracy on train set : 87% Accuracy on validation set : 84%

ROC & AUC :

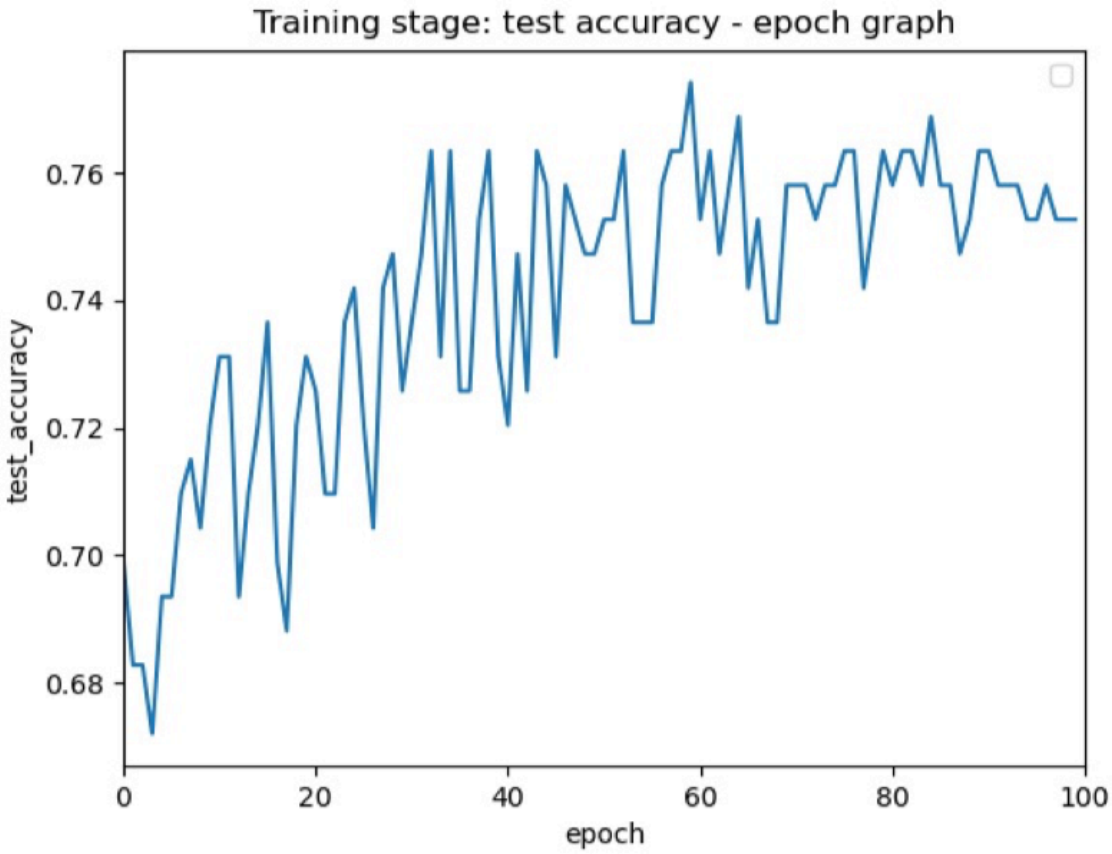


CNN + Perceptron

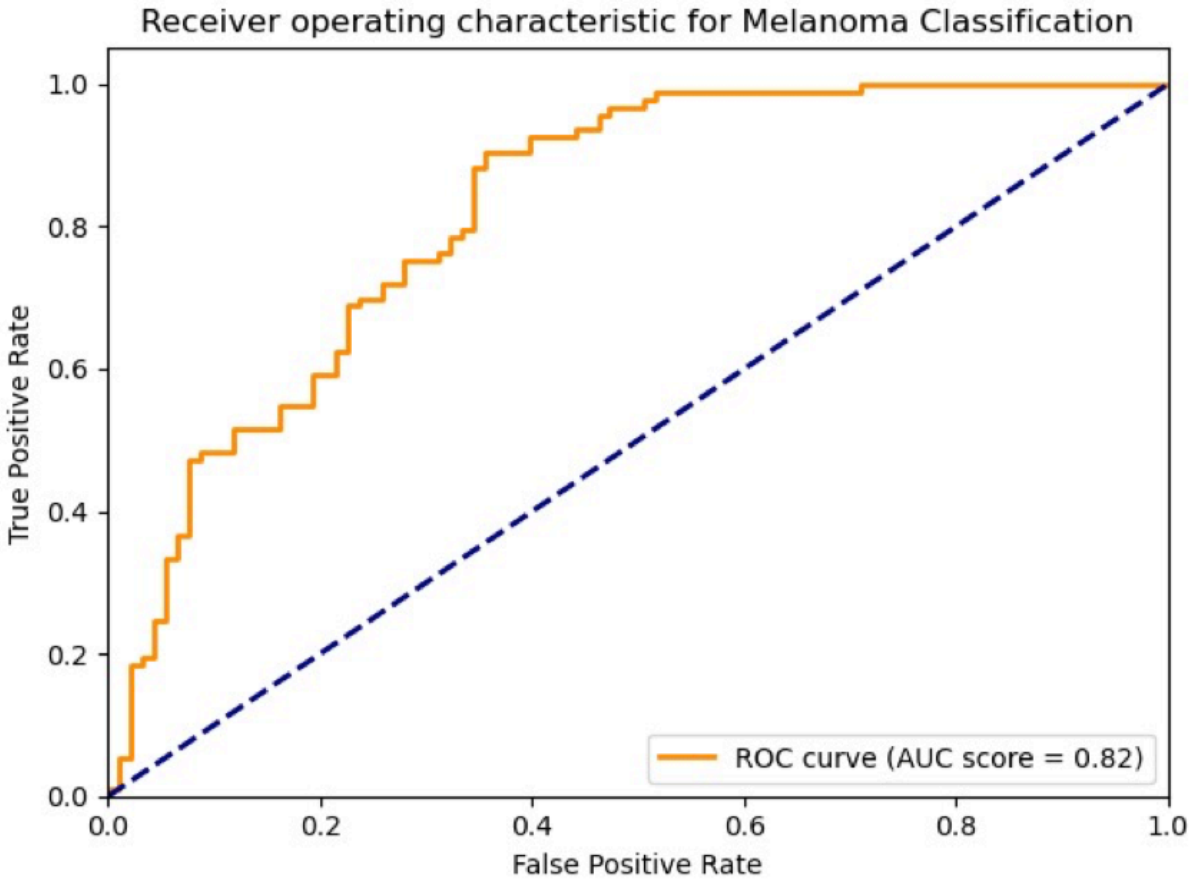
By jiajun.

To design the CNN, I referred to the CNN assignment of this semester and added some MaxPool layers, and I also referred to the mobilenet model and added a Dropout layer before the Linear model. To design the MLP, I first added a hidden layer with a dimension of 20, and then added more hidden layers. Also I added some code to convert the meta data from the form of dictionary to torch tensor. However, when I tested the model, I found that CNN itself could reach an accuracy as 80%, nonetheless, the CNN with MLP model can only reach an accuracy at around 75%. My speculate is that the MLP structure should be modified. Also there was a little bit overfitting when training, and I thought that this might because of the limited amount of the data set. However, an interesting thing is that the AUC score is 0.82, which is greater than the EfficientNet model implemented by one of my groupmates (but the accuracy of my model is lower).

Curve during training :



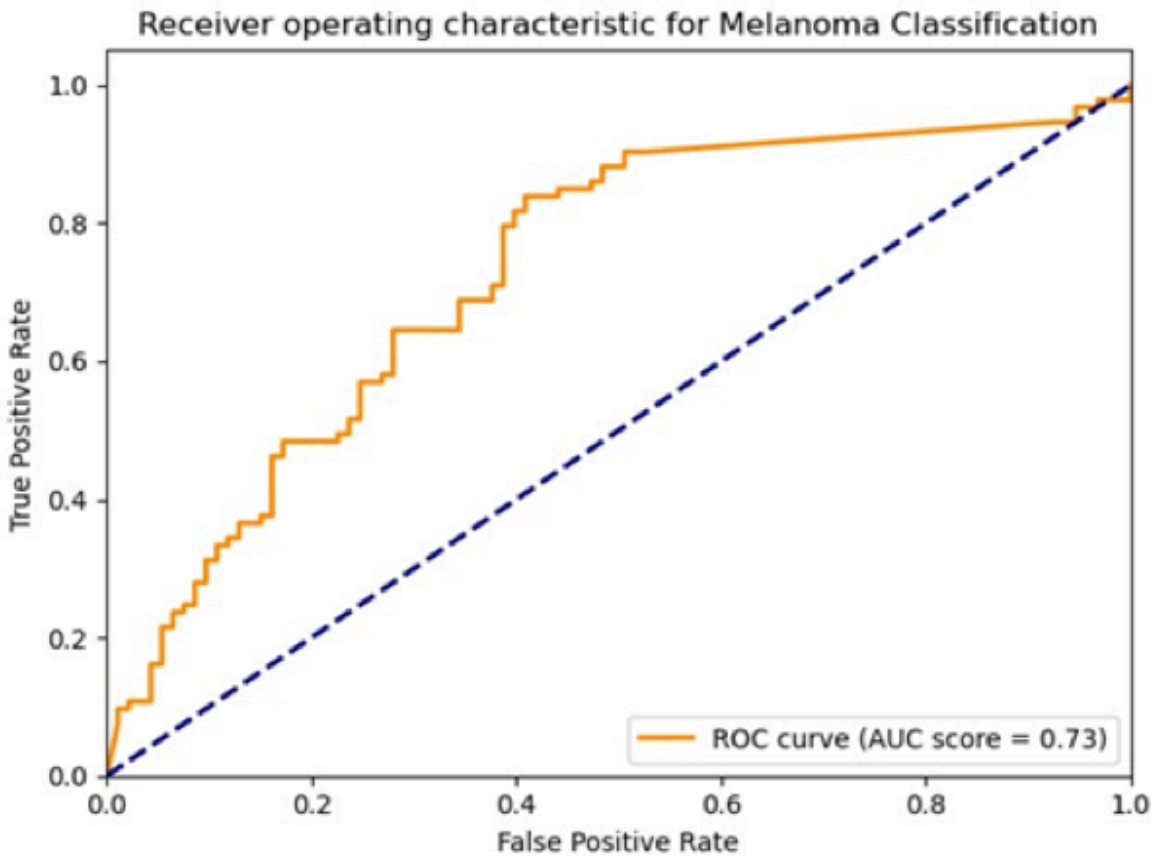
ROC & AUC :



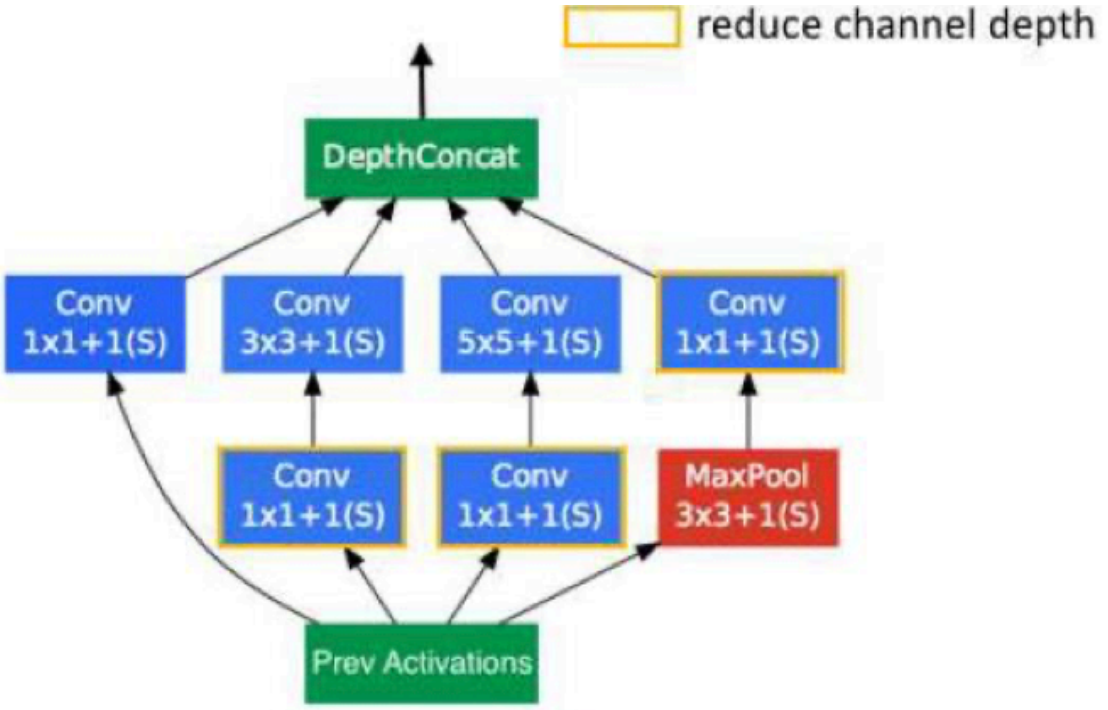
DenseNet

Problems:

- Costs a lot of memory.
  - When batch size is 2, it costs about 10GB GPU memory.
- Slow
  - Step: 283ms Tot: 1m48s for an epoch for 748 images Poor accuracy
- Overfitting
  - 71% acc on training set but 56% acc on testing set.
- ROC

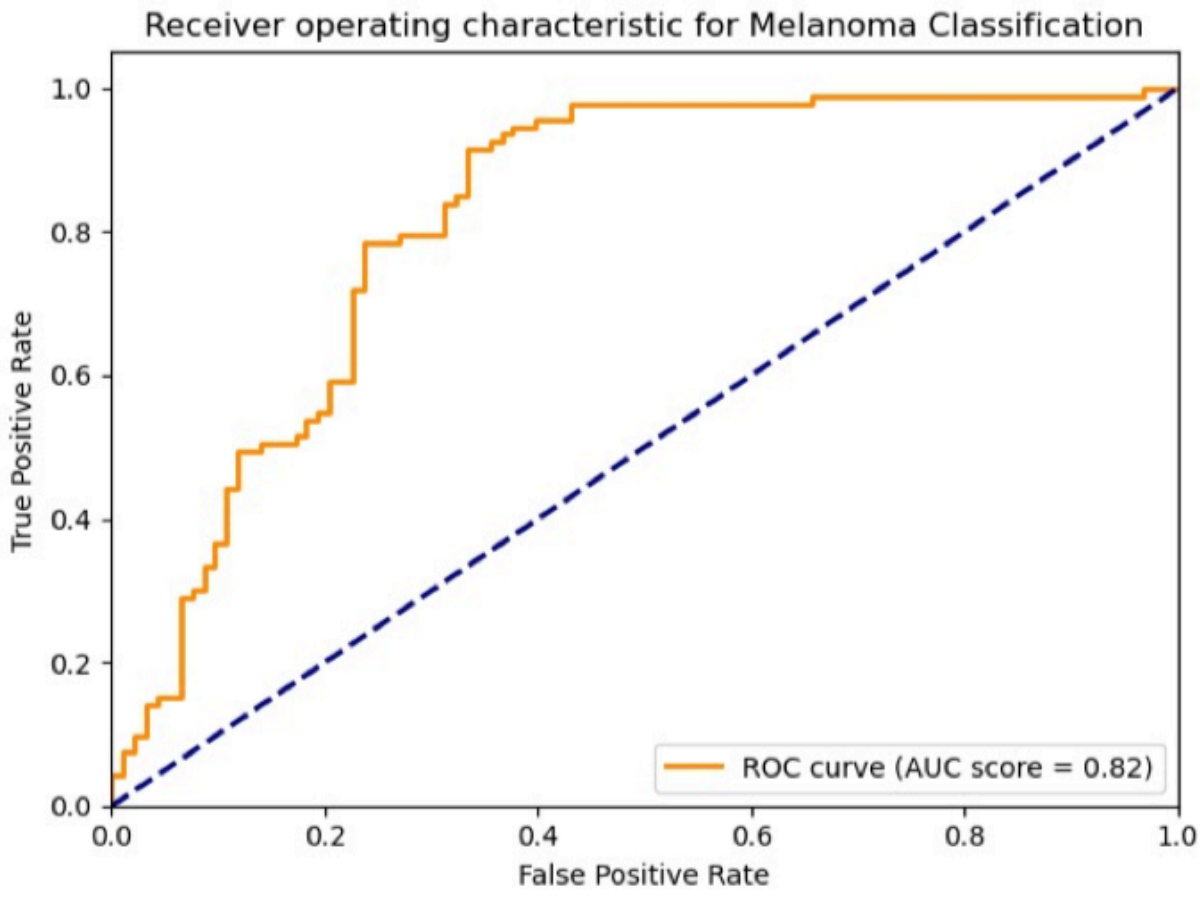


GoogLeNet



the learning rate equals 0.025, the momentum equals 0.9, the weight dacay equals 3e-4, the total epochs equal 200 and the batch size equals 2. Under this setting, the overall performance can be relatively high. The best accuracy is around 79%- 80% within 200 epochs.

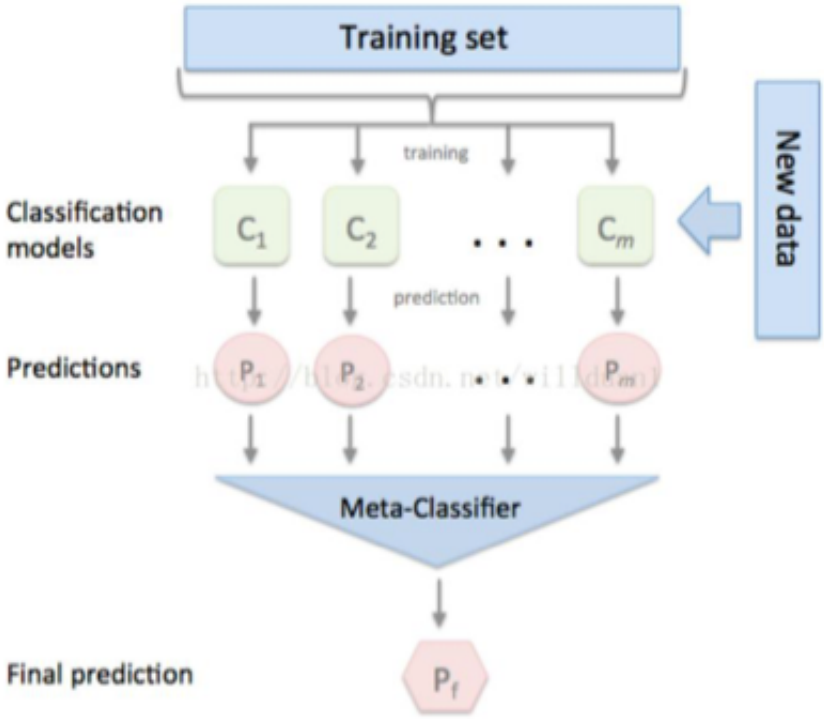
ROC:



Ensemble

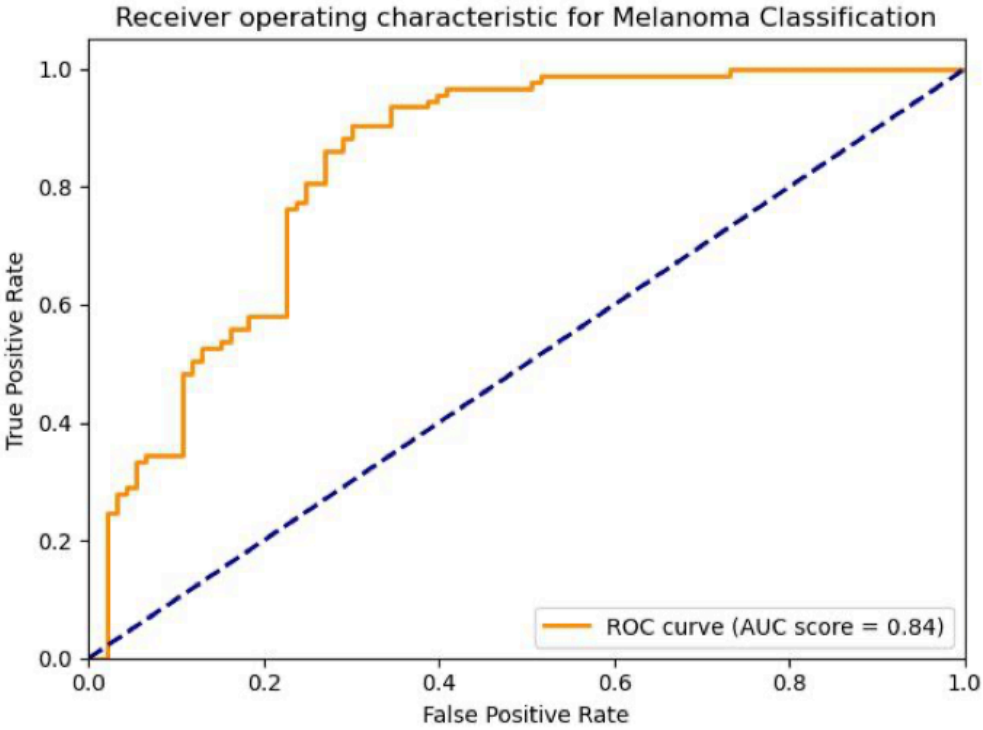
We join our efforts by applying bagging to combine all weak classifiers to make a strong one.

Bootstrap Aggregation (or Bagging for short), is a simple and very powerful ensemble method. Bagging is the application of the Bootstrap procedure to a high-variance machine learning algorithm.



Performance

```
(base) group3@nb1421:~/zhuyichen$ python test.py
Data root: /home/group3/DataSet
CSV file: validation_set.csv
Image folder: Validation_set
Load effnet-b0 with acc = 83.87096774193549
Load effnet-b0-2 with acc = 80.64516129032258
Load cnn with acc = 77.41935483870968
test.py:88: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone
 than torch.tensor(sourceTensor).
  meta['age_approx'] = torch.tensor(meta['age_approx']).unsqueeze(0)
 [========================== 47/47 ==========================>.]  Step: 4s14ms | Tot: 1m24s | Loss: 0.495 | Acc: 80.108% (149/186)
```

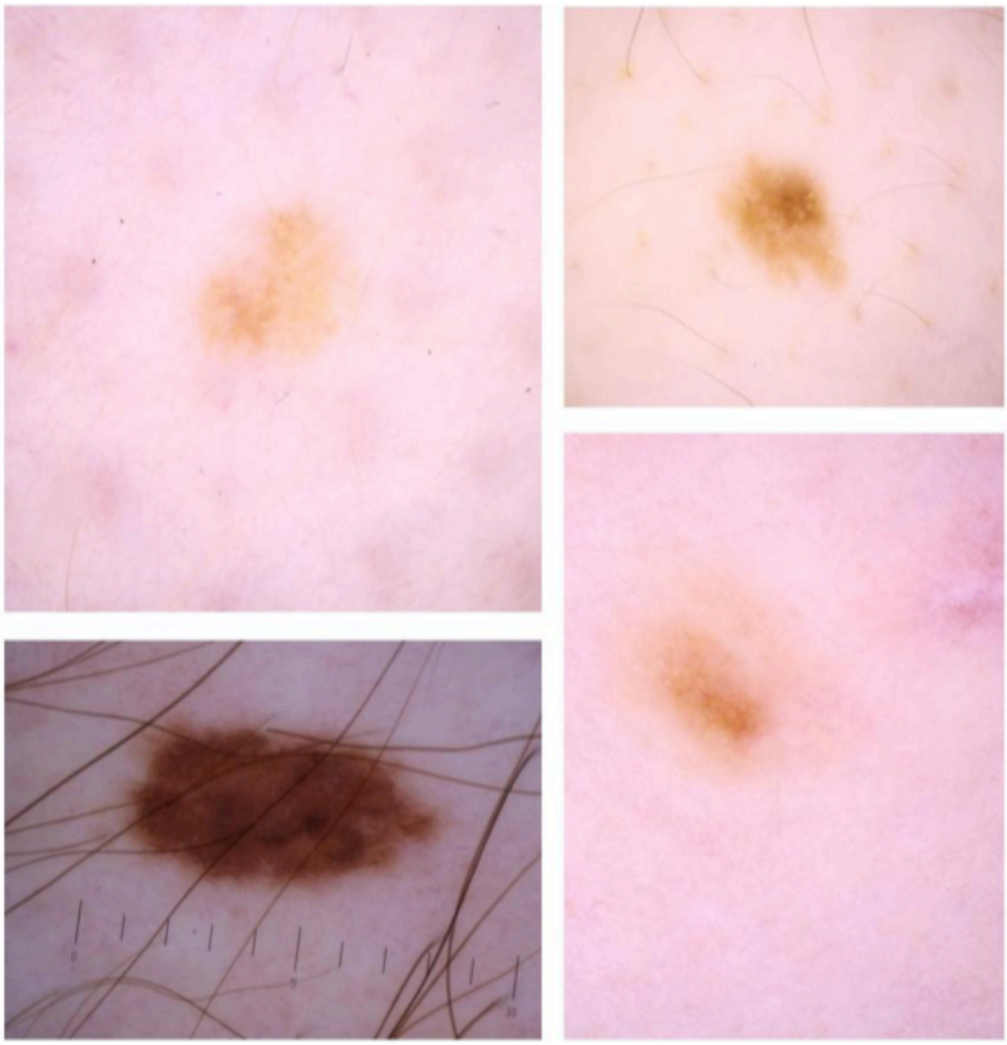


Although the AUC score of the ensemble model is higher than any single model, its accuracy on the validation set is just around the average. So eventually we decide not to use it but choose the EffNetB0 as the best model.

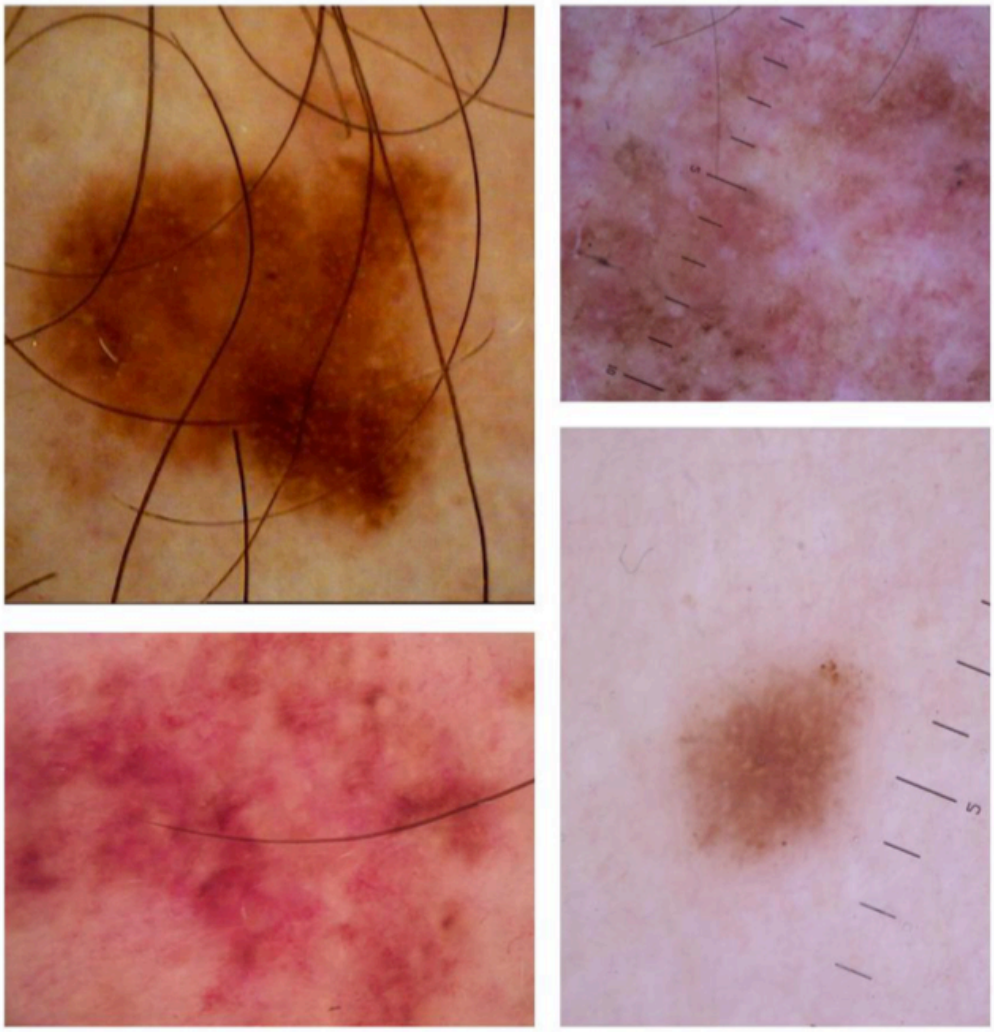
Misclassification

Misclassified samples:

0 → 1:



1 → 0:



What can be improved

◦The ensemble result does not have significant improvement. Maybe the differences between models are too large and the number of models is too small. Additionally, the number of training samples is also too small to do random sampling with replacement.

◦The meta data might get better use. We now simply convert it into one hot and put it into an MLP.