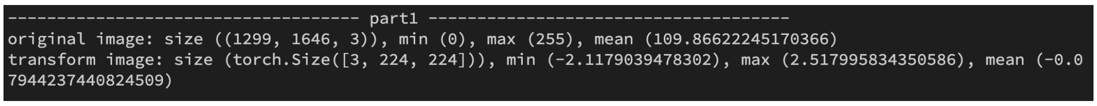
Assignment 2

Name: Yang Hao-Wei, Student ID: 109062526 05/01/2021

1 Part 1

1.1 Read all training data and apply preprocessing steps

I use the transform package in pytorch to process the image, including Resize((256, 256)), RandomHorizontalFlip (p=0.5, only in train time), and Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]). The following picture is the information of the pictures before and after the transform.



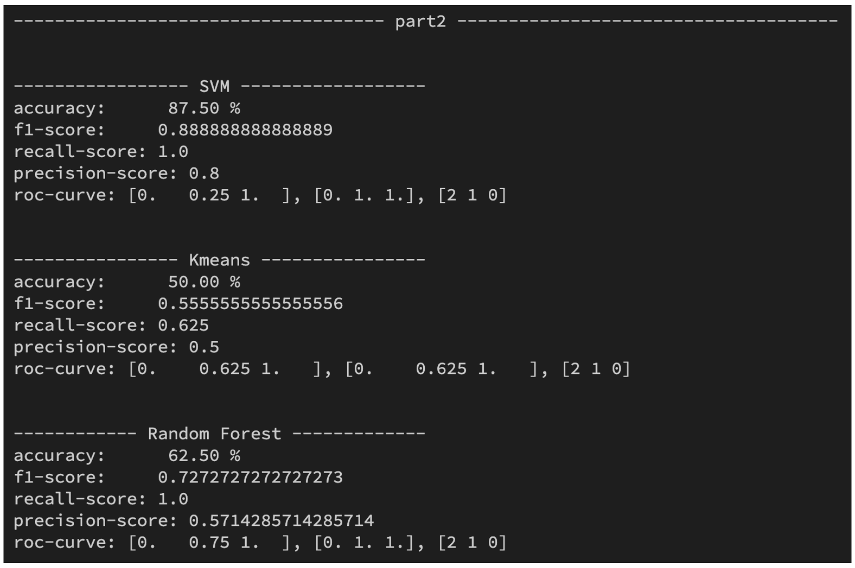
```
----------------------------------- part1 ------------------------------------
original image: size ((1299, 1646, 3)), min (0), max (255), mean (109.86622245170366)
transform image: size (torch.Size([3, 224, 224])), min (-2.1179039478302), max (2.517995834350586), mean (-0.0
7944237440824509)
```

Figure 1: image info before and after transformation

2 Part 2

2.1 Train and Validate whether the image is normal or pneumonia

In this section, I try 4 different models to train on the dataset. Firstly, I use the package in sklearn including SVM, Kmeans, and Random Foreset to classify images. To be notice, due to the fact that the input data dimension these algorithms required is two-dimensional, I first used the function "get-numpy-data2d-with-labels()" to converted the original four-dimensional (3 channels + 1 batch size) image into two dimensions by fixing the first dimension (batch size) and expanding the last three dimensions into one dimension. The results of the three algorithms are shown below.



```
----------------------------------- part2 -----------------------------------


---------------- SVM -----------------
accuracy:      87.50 %
f1-score:     0.888888888888889
recall-score: 1.0
precision-score: 0.8
roc-curve: [0.   0.25 1.  ], [0. 1. 1.], [2 1 0]


--------------- Kmeans ---------------
accuracy:      50.00 %
f1-score:     0.5555555555555556
recall-score: 0.625
precision-score: 0.5
roc-curve: [0.    0.625 1.   ], [0.    0.625 1.   ], [2 1 0]


----------- Random Forest ------------
accuracy:      62.50 %
f1-score:     0.7272727272727273
recall-score: 1.0
precision-score: 0.5714285714285714
roc-curve: [0.   0.75 1.  ], [0. 1. 1.], [2 1 0]
```

Figure 2: sklearn packages result

Next, I use the pre-trained models, resnet152, provided by pytorch and I freeze the convolution layers (extracting features), only update the classifier part. The results of the three algorithms are shown below. As for the metrics to judge the quality of the model, I use f1-score, precision, roc-curve, and recall, and they are all functions from sklearn.metrics.

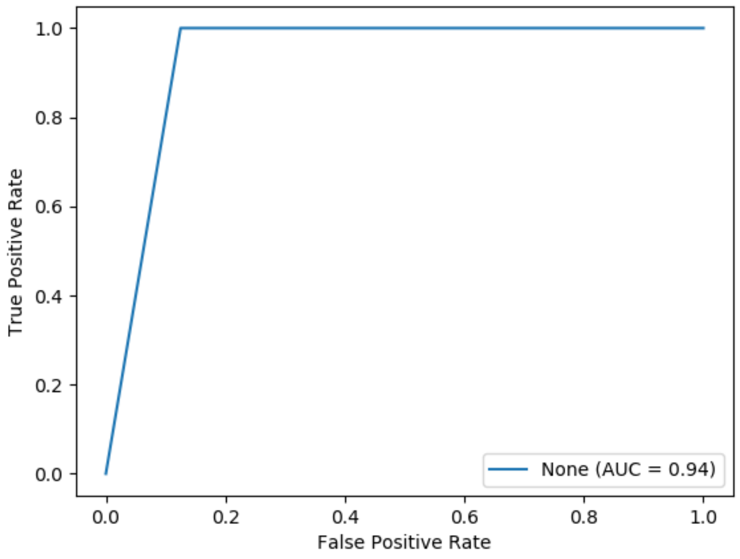
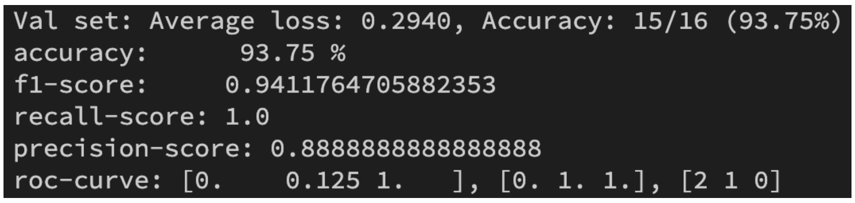


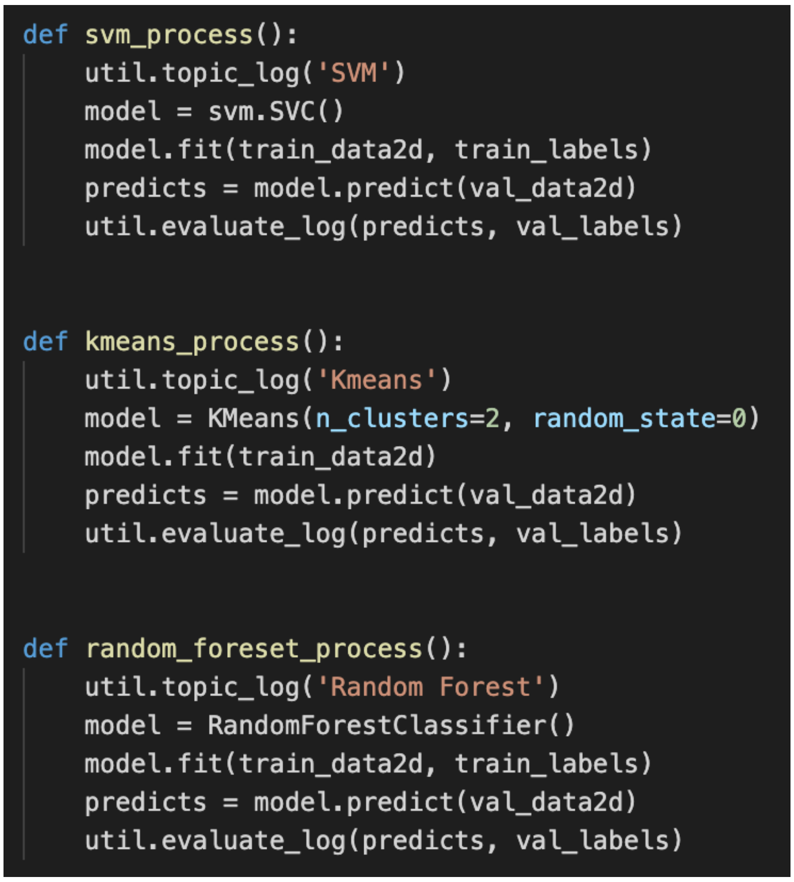
Figure 3: roc-curve plot by sklearn.metrics.RocCurveDisplay()



```
Val set: Average loss: 0.2940, Accuracy: 15/16 (93.75%)
accuracy:       93.75 %
f1-score:      0.9411764705882353
recall-score: 1.0
precision-score: 0.8888888888888888
roc-curve: [0.     0.125 1.    ], [0. 1. 1.], [2 1 0]
```

Figure 4: Resnet152 result

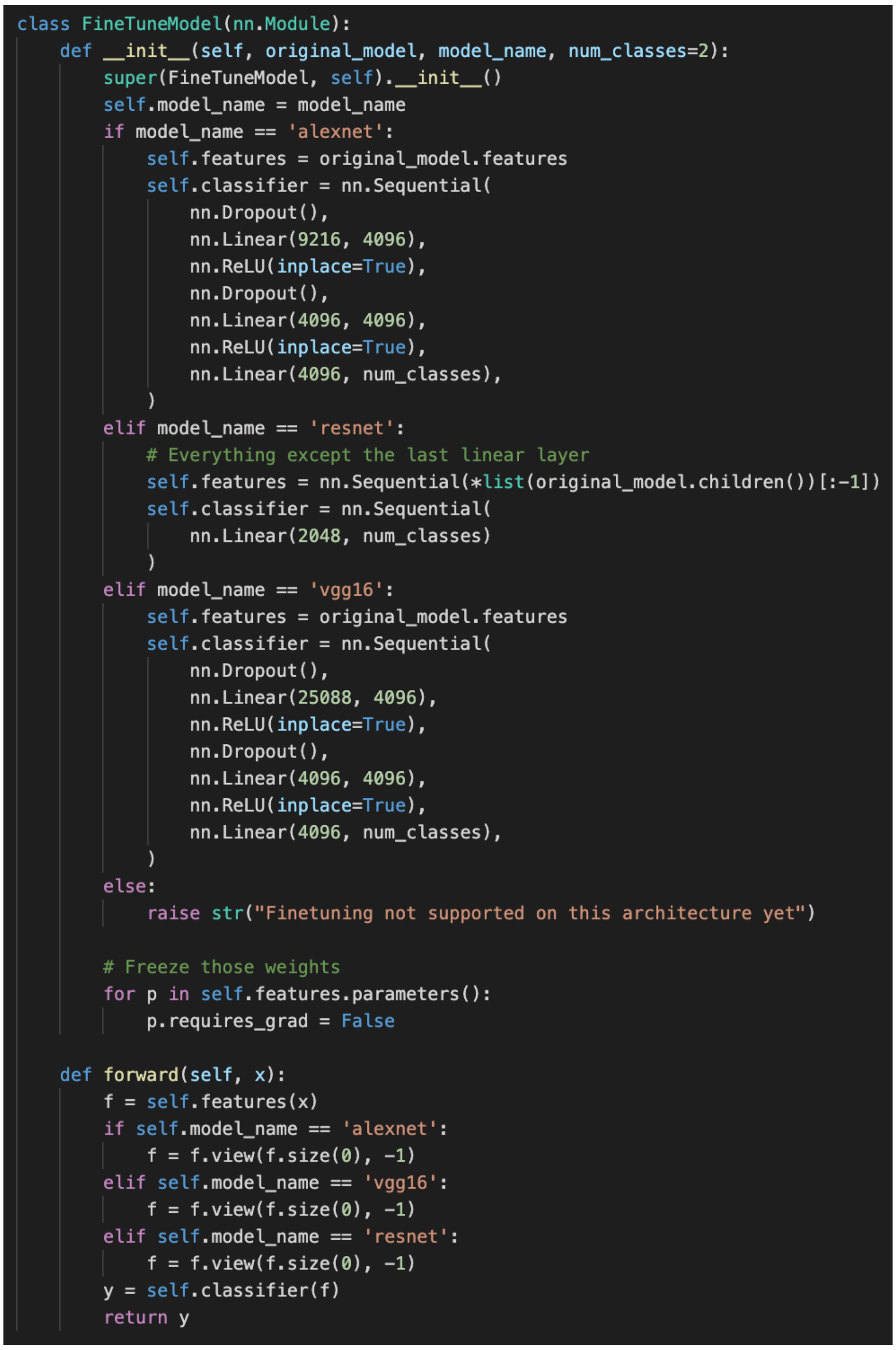
2.2 Detail Setting of Sklearn Models



```python
def svm_process():
    util.topic_log('SVM')
    model = svm.SVC()
    model.fit(train_data2d, train_labels)
    predicts = model.predict(val_data2d)
    util.evaluate_log(predicts, val_labels)


def kmeans_process():
    util.topic_log('Kmeans')
    model = KMeans(n_clusters=2, random_state=0)
    model.fit(train_data2d)
    predicts = model.predict(val_data2d)
    util.evaluate_log(predicts, val_labels)


def random_foreset_process():
    util.topic_log('Random Forest')
    model = RandomForestClassifier()
    model.fit(train_data2d, train_labels)
    predicts = model.predict(val_data2d)
    util.evaluate_log(predicts, val_labels)
```

Figure 5: simply load the model from sklearn, for kmeans, set the cluster num to 2. For the training data, use the output from "get-numpy-data2d-with-labels()"

2.3 Detail Setting of CNN Models

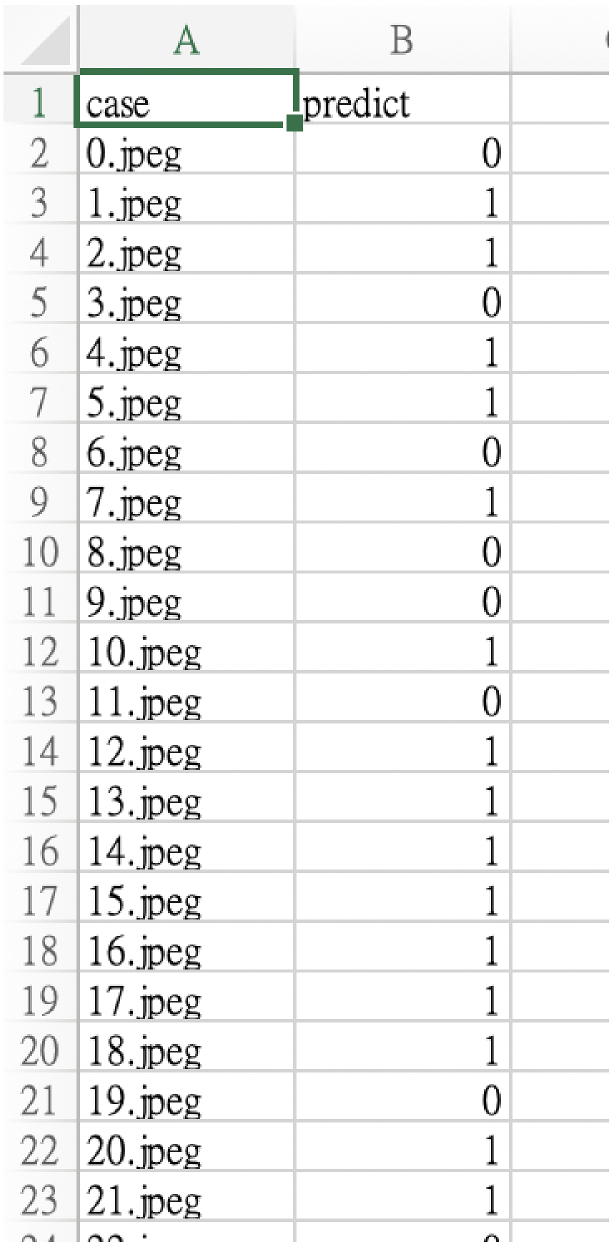


```
class FineTuneModel(nn.Module):
    def __init__(self, original_model, model_name, num_classes=2):
        super(FineTuneModel, self).__init__()
        self.model_name = model_name
        if model_name == 'alexnet':
            self.features = original_model.features
            self.classifier = nn.Sequential(
                nn.Dropout(),
                nn.Linear(9216, 4096),
                nn.ReLU(inplace=True),
                nn.Dropout(),
                nn.Linear(4096, 4096),
                nn.ReLU(inplace=True),
                nn.Linear(4096, num_classes),
            )
        elif model_name == 'resnet':
            # Everything except the last linear layer
            self.features = nn.Sequential(*list(original_model.children())[:-1])
            self.classifier = nn.Sequential(
                nn.Linear(2048, num_classes)
            )
        elif model_name == 'vgg16':
            self.features = original_model.features
            self.classifier = nn.Sequential(
                nn.Dropout(),
                nn.Linear(25088, 4096),
                nn.ReLU(inplace=True),
                nn.Dropout(),
                nn.Linear(4096, 4096),
                nn.ReLU(inplace=True),
                nn.Linear(4096, num_classes),
            )
        else:
            raise str("Finetuning not supported on this architecture yet")

        # Freeze those weights
        for p in self.features.parameters():
            p.requires_grad = False

    def forward(self, x):
        f = self.features(x)
        if self.model_name == 'alexnet':
            f = f.view(f.size(0), -1)
        elif self.model_name == 'vgg16':
            f = f.view(f.size(0), -1)
        elif self.model_name == 'resnet':
            f = f.view(f.size(0), -1)
        y = self.classifier(f)
        return y
```

Figure 6: use pretrained model from pytorch and freeze convolution layers (required gradient = false), only backpropagate on classifier (nn.linear(2048, 2))

3 Part 3: classification on testing dataset

Please refer to the csv file in the folder.



	A	B
1	case	predict
2	0.jpeg	0
3	1.jpeg	1
4	2.jpeg	1
5	3.jpeg	0
6	4.jpeg	1
7	5.jpeg	1
8	6.jpeg	0
9	7.jpeg	1
10	8.jpeg	0
11	9.jpeg	0
12	10.jpeg	1
13	11.jpeg	0
14	12.jpeg	1
15	13.jpeg	1
16	14.jpeg	1
17	15.jpeg	1
18	16.jpeg	1
19	17.jpeg	1
20	18.jpeg	1
21	19.jpeg	0
22	20.jpeg	1
23	21.jpeg	1

Figure 7: csv file snapshot

4 Question and Summary

In this assignment, I've learned a lot, from data processing, such as using transform package, dataset and dataloader in torch, building and using models from sklearn or pytorch pretrained models packages. I also learned to use transfer learning technique to train a convolution nerual network. Most of all, during this assignment, at first, I was not that familiar with CNN structure and the principle/theory behind this technique, so I search the Internet for more information on CNN. And now I think I have a better understand on how CNN work and what convolution, stride, channel, kernel, etc is.

5 Packages used

argparse
os
cv2
PIL.Image
numpy
pandas
matplotlib.pyplot
sklearn
sklearn.metrics
torch
torch.nn
torch.utils.data
torch.optim
torch.nn.functional
torchvision.transforms
torchvision.models