Report

CS542 Class Challenge: Image Classification of COVID-19 X-rays

Bofeng Liu
bofeng96@bu.edu
U47945506

0. Abstract

This report is about summarizing what I have done to try to classify X-ray images with high accuracy. The data I use is collected by Adrian Xu, combining the Kaggle Chest X-ray dataset with the COVID-19 Chest X-ray dataset collected by Dr. Joseph Paul Cohen.

1. Architecture

Task1

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 7, 7, 512)         14714688
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dropout (Dropout)            (None, 25088)             0
_________________________________________________________________
dense_feature (Dense)        (None, 256)               6422784
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 257
=================================================================
Total params: 21,137,729
Trainable params: 6,423,041
Non-trainable params: 14,714,688
_________________________________________________________________
```

Figure 1

Architecture: Input → vgg16 → Flatten layer → Dropout → Dense layer → Dropout → Sigmoid classifier

The layer dimensions and the number of parameters are listed in Figure 1.

Task2

In task 2, I trained ***three*** deep neural network models.

```
Model: "Architecture_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 7, 7, 512)         14714688
_________________________________________________________________
pooling (GlobalAveragePoolin (None, 512)               0
_________________________________________________________________
dense_feature_1 (Dense)      (None, 512)               262656
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_feature_2 (Dense)      (None, 512)               262656
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 4)                 2052
=================================================================
Total params: 15,242,052
Trainable params: 527,364
Non-trainable params: 14,714,688
_________________________________________________________________
```

Figure 2

Architecture: Input → vgg16 → GAP → Dense layer → Dropout → Dense layer → Dropout → Softmax classifier

The layer dimensions and the number of parameters are listed in Figure 2

```
Model: "Architecture_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg19 (Model)                (None, 7, 7, 512)         20024384
_________________________________________________________________
pooling (GlobalAveragePoolin (None, 512)               0
_________________________________________________________________
dense_feature_1 (Dense)      (None, 512)               262656
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_feature_2 (Dense)      (None, 512)               262656
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 4)                 2052
=================================================================
Total params: 20,551,748
Trainable params: 527,364
Non-trainable params: 20,024,384
_________________________________________________________________
```

Figure 3

Architecture: Input → vgg19 → GAP → Dense layer → Dropout → Dense layer → Dropout → Softmax classifier

The layer dimensions and the number of parameters are listed in Figure 3

```
Model: "Architecture_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Model)             (None, 7, 7, 2048)        23587712
_________________________________________________________________
BN (BatchNormalization)      (None, 7, 7, 2048)        8192
_________________________________________________________________
pooling (GlobalAveragePoolin (None, 2048)              0
_________________________________________________________________
dense_feature (Dense)        (None, 512)               1049088
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 4)                 2052
=================================================================
Total params: 24,647,044
Trainable params: 1,055,236
Non-trainable params: 23,591,808
_________________________________________________________________
```

Figure 4

Architecture: Input → ResNet50 → Batch Normalization → GAP → Dense layer → Dropout → Softmax classifier

The layer dimensions and the number of parameters are listed in Figure 4.

2. Optimizers, loss functions, etc

		Optimizers	Loss function	Parameters	Regularization
Task1		adam	binary crossentropy	accuracy	L2
Task2	model 1 vgg16	adam	categorical crossentropy	accuracy	N/A
	model 2 vgg19	adam	categorical crossentropy	accuracy	L2
	model 3 ResNet	adam	categorical crossentropy	accuracy	L2

3. Comparison

VGG16 vs VGG19

Basically, the difference between model_1 and model_2 is that I reuse two different trained models, VGG16 and VGG19. Those models are used to accelerate the training of neural networks as a weight initialization scheme.

VGGNet is a deep convolutional neural network used for image classification. Unlike AlexNet which uses large receptive fields, VGGNet stacks several small filters (3x3 conv layers) which make the architecture deeper but has more non-linearities. 16 and 19 stands for the number of weight layers in the network. VGG19 has a slightly better performance compare to VGG16, but it costs more memory and has a slower runtime. To be specific, VGG16 is over 533MB and VGG19 is over 574MB.

In model 1, I add two fully-connected layers after VGG16 which followed by a Softmax classifier.

In model 2, I add regularization terms to the two fully-connected layers and I replace the flatten layer with a global average pooling layer.

VGGNet vs ResNet

As I mentioned above, VGG only stacks 3x3 convolutional layers on top of each other to make networks deeper. It reduces dimensionality by maxpooling. And at the end, VGG has two fully-connected layers which followed by a softmax classifier.

ResNet50 is a residual neural network that has 50 weight layers. Although the depth of ResNet is much deeper than VGGNet, the space usage of ResNet is significantly smaller. The main reason is that ResNet uses a global average pooling layer which is similar to max polling rather than fully-connected layers.

In model 3, I add batch normalization before GAP and increase the dropout rate to 0.5. I only use one fully-connected layer with regularization which seems to reduce the overfitting too.

4. Accuracy and Loss

Task1

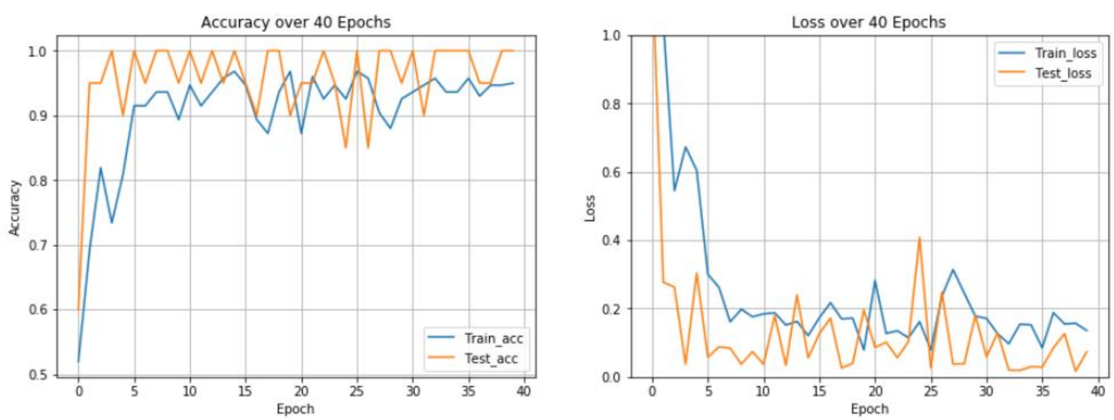


Figure 5

From these two curves, we learn that the model I trained is relatively good. It has an accuracy of ~95% on both the training set and the validation set. This means that my model is expected to perform with a high accuracy on new data.

Task2

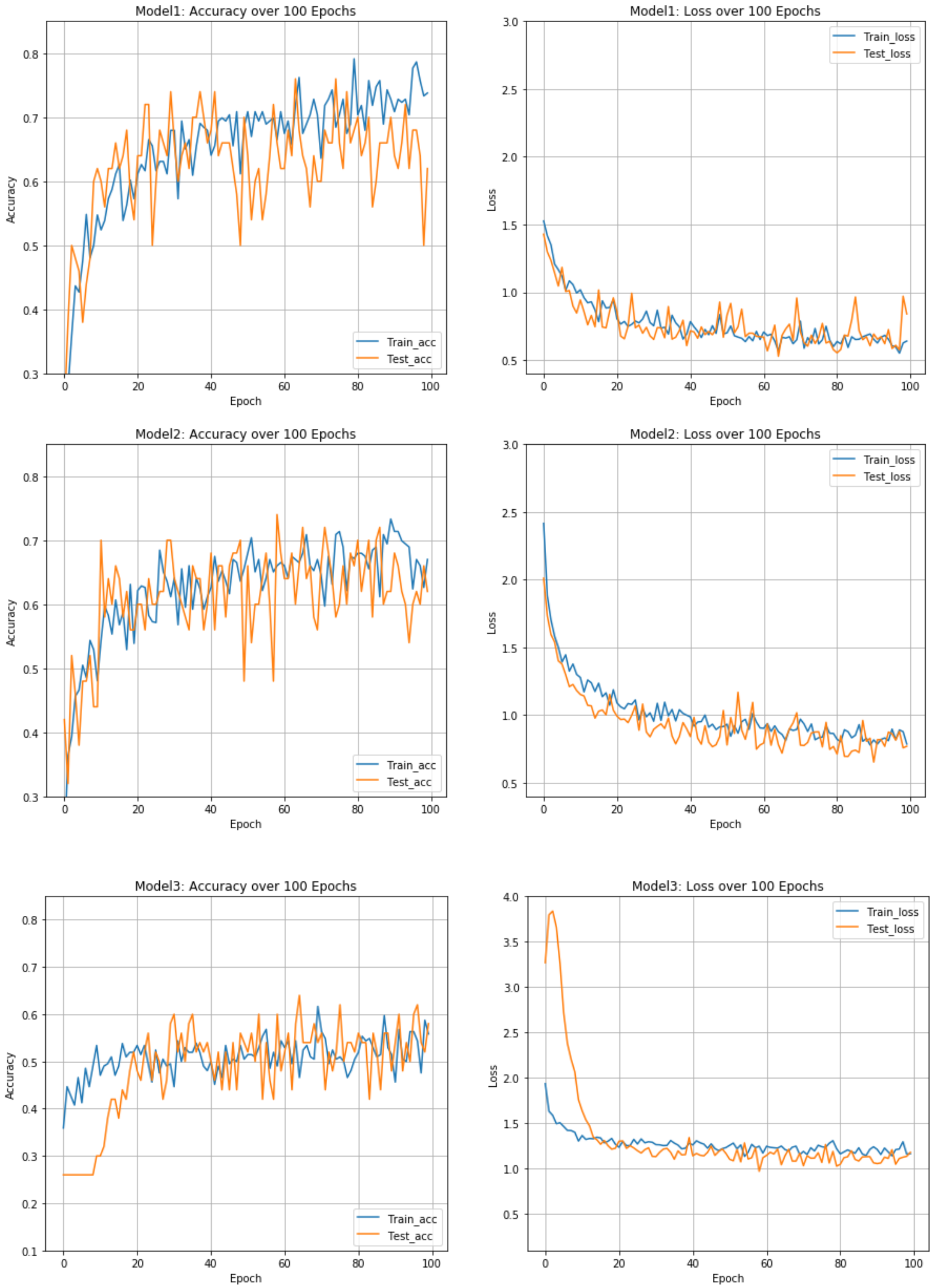


Figure 6

After training model 2 and model 3 for the first time, I see that the model 2 and model 3 has a higher accuracy on training set, but they perform really bad on validation set (~26%). So, I use some strategies to reduce overfitting like *increase dropout rate*, *redesign network architecture*, *add regularization, add batch normalization and replace FC layers with GAP layers*. It appears that those techniques make sense because the difference between Train_acc and Valid_acc becomes smaller.

Basically, **model 1** and **model 2** have an accuracy of ~70% on both training set and validation set. In terms of testing models on new data, model 1 has an accuracy of ~70%, meanwhile model 2 has an accuracy of ~50%.

Model 3 has a lower accuracy of ~55% on both training set and validation set and the testing on new data shows an accuracy of ~30%.

5. t-SNE visualizations

Task1

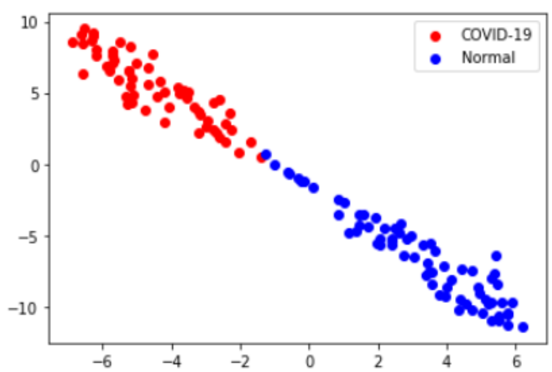
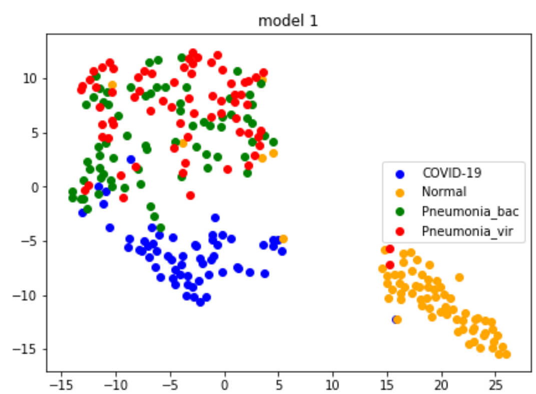


Figure 7

The graph generated by t-SNE algorithm give me an intuition that my model classifies normal and COVID-19 rays correctly.

Task2



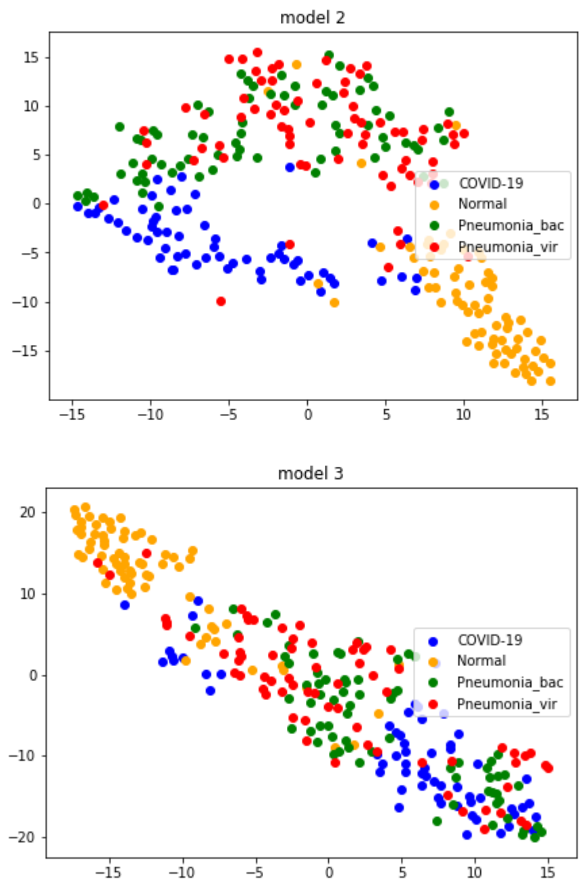
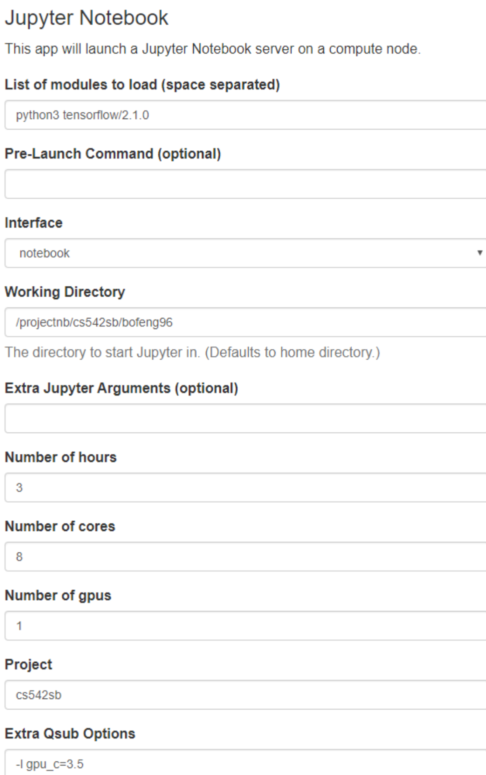


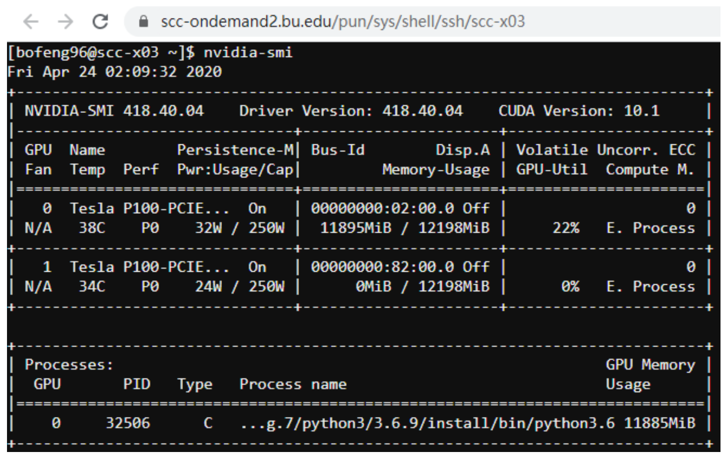
Figure 8

As we can see, the three models achieve classification on normal and COVID-19 X-rays with high performance. The models based on VGG can differentiate COVID-19 with Pneumonia-Bacterial or Pneumonia-Viral but the model based on ResNet has a relative bad performance.

6. SCC snapshots

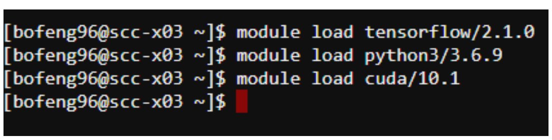
I requested 8 CPUs and 1 GPU.






```
[bofeng96@scc-x03 ~]$ nvidia-smi
Fri Apr 24 02:09:32 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.40.04    Driver Version: 418.40.04    CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla P100-PCIE...  On   | 00000000:02:00.0 Off |                    0 |
| N/A   38C    P0    32W / 250W |  11895MiB / 12198MiB |     22%   E. Process |
+-------------------------------+----------------------+----------------------+
|   1  Tesla P100-PCIE...  On   | 00000000:82:00.0 Off |                    0 |
| N/A   34C    P0    24W / 250W |      0MiB / 12198MiB |      0%   E. Process |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0     32506      C   ...g.7/python3/3.6.9/install/bin/python3.6 11885MiB |
+-----------------------------------------------------------------------------+
```



```
[bofeng96@scc-x03 ~]$ module load tensorflow/2.1.0
[bofeng96@scc-x03 ~]$ module load python3/3.6.9
[bofeng96@scc-x03 ~]$ module load cuda/10.1
[bofeng96@scc-x03 ~]$
```