COVID-19 Detection

From X-Ray Images

Group 10

The goals:

- Training a CNN network to help predict infection.
- Optimising the model for better performance.
- etc.

…And the difficulties?

- Extracting data (image - > usable digits)
- Optimisation (choosing the best parameter set)
- etc.

Here's what we do…

Preprocessing!

Label form is the key of supervised learning,
which makes preprocessing just two main parts:

data extraction and labelling.

```
print(d_covid19)
```

```
{'ARDS': {0.0: 516, 1.0: 14},
 'Bacterial': {0.0: 489, 1.0: 41},
 'COVID-19': {0.0: 198, 1.0: 332},
 'Chlamydophila': {0.0: 529, 1.0: 1},
 'E.Coli': {0.0: 530},
 'Fungal': {0.0: 508, 1.0: 22},
 'Influenza': {0.0: 526, 1.0: 4},
 'Klebsiella': {0.0: 521, 1.0: 9},
 'Legionella': {0.0: 523, 1.0: 7},
 'Lipoid': {0.0: 522, 1.0: 8},
 'MERS': {0.0: 522, 1.0: 8},
 'Mycoplasma': {0.0: 525, 1.0: 5},
 'No Finding': {0.0: 516, 1.0: 14},
 'Pneumocystis': {0.0: 508, 1.0: 22},
 'Pneumonia': {0.0: 54, 1.0: 476},
 'SARS': {0.0: 514, 1.0: 16},
 'Streptococcus': {0.0: 513, 1.0: 17},
 'Varicella': {0.0: 525, 1.0: 5},
 'Viral': {0.0: 165, 1.0: 365}}
COVID19_Dataset num_samples=530 views=('PA', 'AP')
```

```
import torchxrayvision as xrv
```

```
imgpath = "./covid-chestxray-dataset-master/images"
csvpath = "./covid-chestxray-dataset-master/metadata.csv"
views = 'PA', 'AP'
transform = torchvision.transforms.Compose([xrv.datasets.XRayCenterCrop(),
                                            xrv.datasets.XRayResizer(224)])
d_covid19 = xrv.datasets.COVID19_Dataset(imgpath=imgpath, csvpath=csvpath,
                                         views=views, transform=transform)
```

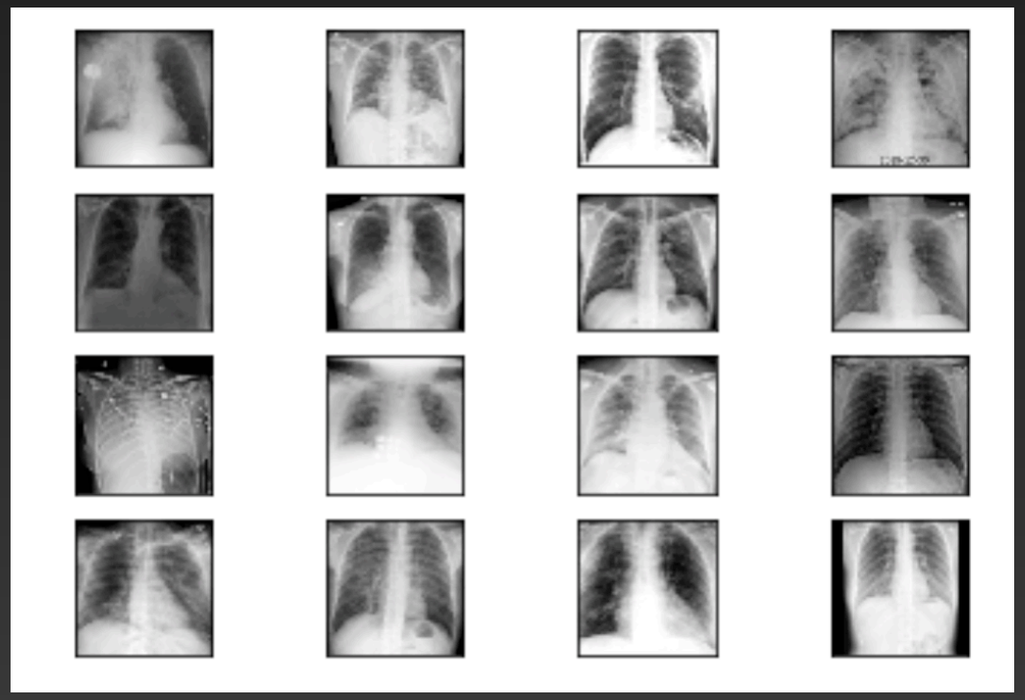
Labeling:

As we only focus on Covid-19, which is a dichotomous problem,
we need to turn the 19-class data Y into binary label.

```
label = []
for lab in Y:
    if lab[2] == 1.:
        label.append(1)
    else:
        label.append(0)
```

```
for i in range(1,17):
    plt.subplot(4,4,i)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(X[random.randint(0,529)]
       .reshape(224,224),plt.cm.gray)

plt.show()
```



Build CNN Network

In the beginning, we used the ResNet model, but the accuracy stays low.

So we turned to VGG16, which is a recommended model by our professors.

```
class VGG16(Model):
    def __init__(self):
        super(VGG16, self).__init__()
        self.c1 = Conv2D(filters=64, kernel_size=(3, 3), padding='same')
        self.b1 = BatchNormalization()
        self.a1 = Activation('relu')
        self.c2 = Conv2D(filters=64, kernel_size=(3, 3), padding='same', )
        self.b2 = BatchNormalization()
        self.a2 = Activation('relu')
        self.p1 = MaxPool2D(pool_size=(2, 2), strides=2, padding='same')
        self.d1 = Dropout(0.2)
```

Result: VGG16 outperforms other models in our experiment.

"Limited dataset in size?"

This can cause the problem of overfitting due to the huge amount of parameters learnt in the training process.

```
# < The following layers are not used during training >

#         self.c3 = Conv2D(filters=128, kernel_size=(3, 3), padding='same')
#         self.b3 = BatchNormalization()
#         self.a3 = Activation('relu')

# ...

#         self.c13 = Conv2D(filters=512, kernel_size=(3, 3), padding='same')
#         self.b13 = BatchNormalization()
#         self.a13 = Activation('relu')
#         self.p5 = MaxPool2D(pool_size=(2, 2), strides=2, padding='same')
#         self.d5 = Dropout(0.2)
```

So...

We simplified the architecture of VGG16 network.

Compile and training

The shown code presents our learning parameters corresponding to our final result

```
model = VGG16()

model.compile(optimizer=tf.keras.optimizers.Adam(lr=0.00001),
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
              metrics=['sparse_categorical_accuracy'])

history = model.fit(X, Y, batch_size=32, epochs=30, validation_split = 0.3,
                    validation_freq = 1)
```

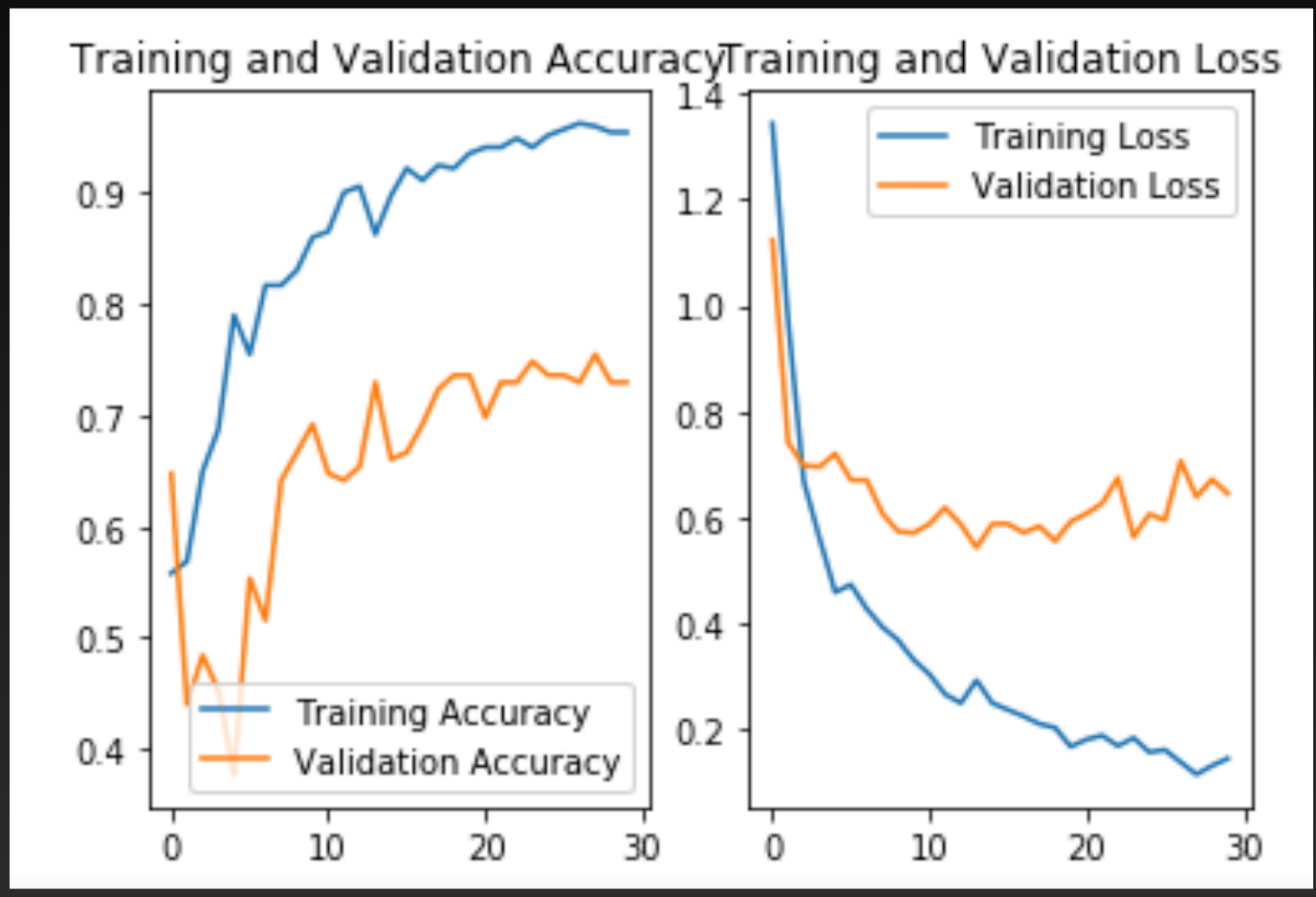
Adjusts which can affect the accuracy :

- VGG16 architecture
- Image size (224x224 for VGG16)
- Learning rate
- Batch size and epochs

Result

1. Epoch > 30?

Training loss keeps decreasing
Validation loss begins to rise
[overfitting]



2. Batch size > 32 ?

Validation accuracy drops.

3. Use original VGG16?

Could easily become overfitted.

4. Image resizing?

224 x 224 RGB is the fixed size for using VGG16.
However, using smaller size images can help us
speed up the training process.

Further thoughts.

3.2 Severity prediction

In order to predict the severity, we first need to pick out our training data from the label 1 sample(patients that have got covid-19), and we need to build a measurement to evaluate the severity.

Considering that there is usable information in the .csv file that is related to disease severity(such as survival, intubated), we can set different weights for these indicators, like ICU-related patients are more severe, and calculate a score for each patient that reflects the severity. make the score number the output of the network.

so our prediction is no longer a label, instead it's a numerical number.

3.4 Multimodal inputs

In this question, we are asked to combine the natural language clinical notes and the image, and we think that using NLP methods may be a good idea. First, we need to separate the words in a long sentence and get rid of some useless words(such as "and", "is" and other words with no actual meaning), so usually the words remaining should be adjective or verb. The next step is to build a word vector for the rest of the words. The length of the vector should be the total number of the remaining words, and each dimension of the vector represents a certain word. In order to build these vectors, we count how many times a word shows up in a note and make it the value in the certain dimension.

In this way, the notes in words are transformed into vectors that can be directly used as input of the neural network.

We can also use python libraries that can evaluation a word's emotion(positive or negative) to improve prediction.

Contribution

Main contributor:

D10-03 Chunxu Liu:

Data preprocessing, network building, parameter training.

Coding.

Finished the final edition of PPT.

D10-04 Yiyang Zhang:

Overall thinking, idea providing.

Digged into other deliverables.

Finished the final edition of PPT and source code editing.

Other team members:

D10-05 Zhiye He: made the draft of PPT

D10-02 Chao Sun: group discussion

D10-01 Jincheng Yi