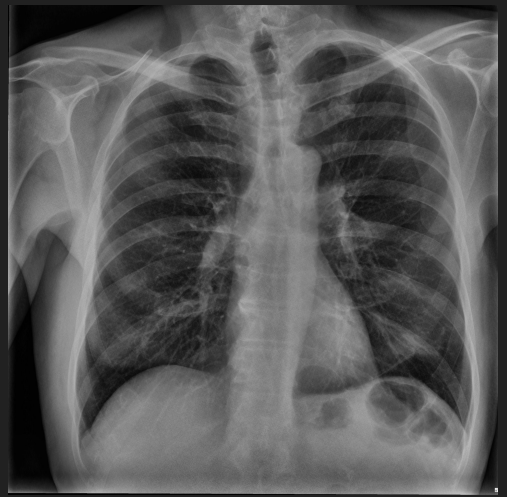
Final Research Report

Matthew Miers, SooJung Lee, DaeHo Kim

Research Goal

Can we build a system that can provide meaningful assistance to medical professionals to identify pneumonia from x-ray images?



Schedule

Selecting Data	Select a medical condition and dataset - one week
Research	Research into what ML frameworks exist, what might be the best fit for our data, and how the by hand classification is done - two weeks
Cleaning and Prep	Cleaning and preparing the dataset - two weeks
Machine Learning	Feeding the data into selected model(s) - two weeks
Analysis and Presentation	Evaluating how well the ML models did, and building the final presentation - four weeks

Image Conversion

First we need to do some processing to convert images into grayscale. Original images are in RGBA, but the additional color information doesn't hold any extra value

```python
from PIL import Image
import os
import shutil

imageDir = "images"
greyDir = "imagesGrey"
if os.path.isdir(greyDir):
    shutil.rmtree(greyDir)
os.mkdir(greyDir)

images = os.listdir(imageDir)

for x in range(0, len(images)):
    filename = images[x]
    print("{}/{} - {}".format(x, len(images), filename))
    if('.png' not in filename):
        continue

    filePath = os.path.join(imageDir, filename)
    filePathDest = os.path.join(greyDir, filename)

    img = Image.open(filePath)
    imgGray = img.convert('L')
    imgGray.save(filePathDest)

```

Removing Low Instance Data

There were some additional pneumonia types that did not have very many cases. Dropping fungal, none, undefined and Pneumonia. Potentially ‘None’ might mean No Pneumonia, but it is also python’s null designation, so it might be best to drop

```
matthew@Matthews-MacBook-Pro gatheringAndPrepSave % python removeBadPneumonia.py
Fungal Pneumonia - 17
None - 76
Undefined Pneumonia - 5
Pneumonia - 5
Viral Pneumonia - 1832
Bacterial Pneumonia - 2801
No Pneumonia (healthy) - 1601
```

Splitting Training and Testing Sets

Since keras requires a specific dir structure, we need to pre-split testing and training sets. Splitting using a 90-10 train test split

```
import os
import random
import shutil

greyDir = "imagesGrey"
feedDir = "feedData"
trainingDir = os.path.join(feedDir, 'train', 'images')
testingDir = os.path.join(feedDir, 'validation', 'images')

if os.path.isdir(feedDir):
    shutil.rmtree(feedDir)
os.mkdir(feedDir)
os.mkdir(os.path.dirname(trainingDir))
os.mkdir(trainingDir)
os.mkdir(os.path.dirname(testingDir))
os.mkdir(testingDir)

images = os.listdir(greyDir)

numImages = len(images)
testingRatio = 0.1
numTestingImg = int(numImages * testingRatio)
testingIndexes = random.sample(range(numImages), numTestingImg)

for x in range(0, len(images)):
    if x in testingIndexes:
        print("{}/{} TESTING - {}".format(x, len(images), images[x]))
        shutil.copyfile(os.path.join(greyDir, images[x]),
            os.path.join(testingDir, images[x]))
    else:
        print("{}/{} TRAINING - {}".format(x, len(images), images[x]))
        shutil.copyfile(os.path.join(greyDir, images[x]),
            os.path.join(trainingDir, images[x]))

print("Total: {}".format(numImages))
print("Training: {}".format(numImages - numTestingImg))
print("Testing: {}".format(numTestingImg))

```

Labeling Data

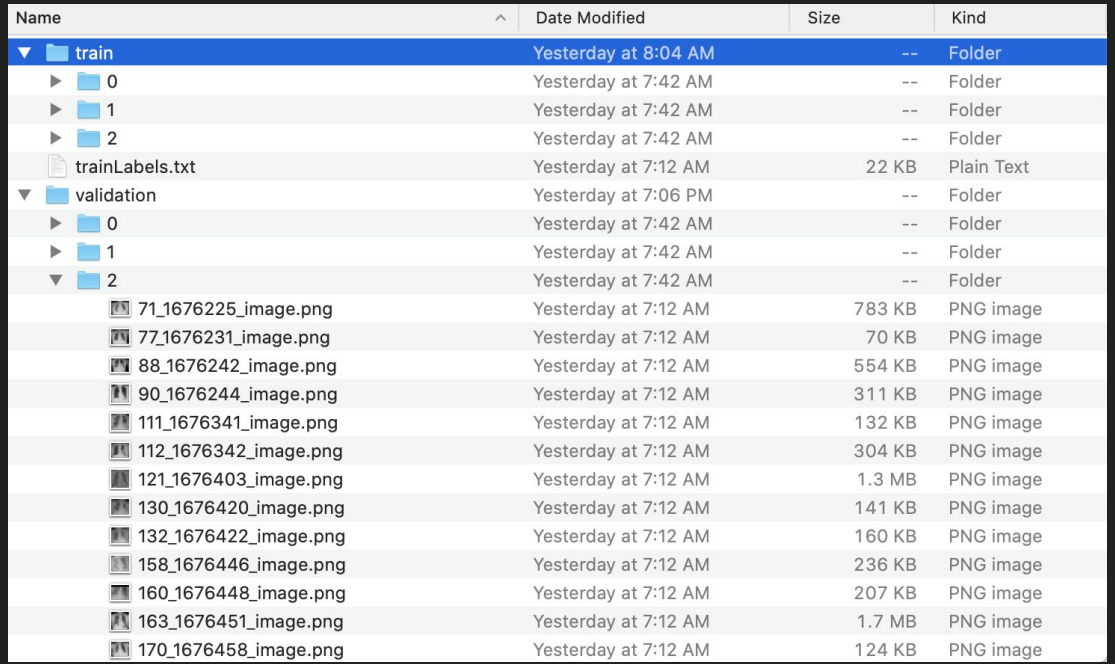
```
PneumoniaKey = {"No Pneumonia (healthy)":0,
    "Bacterial Pneumonia":1,
    "Viral Pneumonia":2,
}
```

Need to do an additional sorting to label the data into labeled sub directories

```
labelFile = "feedData/trainLabels.txt"
with open(labelFile, newline='') as f:
    lines = f.readlines()

lines = [x.split(',')[0] for x in lines]
print(lines)

x = 0
os.mkdir(os.path.join(feedDir, 'train', '0'))
os.mkdir(os.path.join(feedDir, 'train', '1'))
os.mkdir(os.path.join(feedDir, 'train', '2'))
for root, dirs, files in os.walk(trainingDir):
    for f in files:
        os.rename(os.path.join(trainingDir, f), os.path.join(feedDir, 'train', lines[x], f))
        x = x + 1
```

Final Dir Structure

One dir split between testing and training data, and for both, another split based on classes



Name	Date Modified	Size	Kind
train	Yesterday at 8:04 AM	--	Folder
0	Yesterday at 7:42 AM	--	Folder
1	Yesterday at 7:42 AM	--	Folder
2	Yesterday at 7:42 AM	--	Folder
trainLabels.txt	Yesterday at 7:12 AM	22 KB	Plain Text
validation	Yesterday at 7:06 PM	--	Folder
0	Yesterday at 7:42 AM	--	Folder
1	Yesterday at 7:42 AM	--	Folder
2	Yesterday at 7:42 AM	--	Folder
71_1676225_image.png	Yesterday at 7:12 AM	783 KB	PNG image
77_1676231_image.png	Yesterday at 7:12 AM	70 KB	PNG image
88_1676242_image.png	Yesterday at 7:12 AM	554 KB	PNG image
90_1676244_image.png	Yesterday at 7:12 AM	311 KB	PNG image
111_1676341_image.png	Yesterday at 7:12 AM	132 KB	PNG image
112_1676342_image.png	Yesterday at 7:12 AM	304 KB	PNG image
121_1676403_image.png	Yesterday at 7:12 AM	1.3 MB	PNG image
130_1676420_image.png	Yesterday at 7:12 AM	141 KB	PNG image
132_1676422_image.png	Yesterday at 7:12 AM	160 KB	PNG image
158_1676446_image.png	Yesterday at 7:12 AM	236 KB	PNG image
160_1676448_image.png	Yesterday at 7:12 AM	207 KB	PNG image
163_1676451_image.png	Yesterday at 7:12 AM	1.7 MB	PNG image
170_1676458_image.png	Yesterday at 7:12 AM	124 KB	PNG image

Testing/Training Data

Greyscale images:

- Represented as an array of numbers between 0 and 255

```
In [11]: img1 = Image.open('feedData/train/0/390_1676586_image.png')
         display(img1)
```

```
In [12]: np.array(img1) / 255.0

Out[12]: array([[[0.00392157, 0.01568627, 0.01176471, ..., 0.16470588, 0.16470588,
                  0.16862745],
                 [0.01176471, 0.01960784, 0.00784314, ..., 0.14901961, 0.15294118,
                  0.15686275],
                 [0.01568627, 0.01568627, 0.00392157, ..., 0.13333333, 0.1372549 ,
                  0.14901961],
                 ...,
                 [0.01176471, 0.01176471, 0.01176471, ..., 0.45882353, 0.4745098 ,
                  0.4745098 ],
                 [0.01176471, 0.01176471, 0.01176471, ..., 0.4627451 , 0.48235294,
                  0.49019608],
                 [0.01176471, 0.01176471, 0.01176471, ..., 0.16862745, 0.18039216,
                  0.19215686]])
```

Loading Data with Keras

- Transforming pixel info to be in range (0,1)
- Resizing images to be 256x256

In [14]:
```
train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1/255.0)
valid_datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1/255.0)
```

In [15]:
```
valid_data = valid_datagen.flow_from_directory(
    directory='feedData/validation/',
    target_size=(256, 256),
    class_mode='categorical',
    batch_size=64,
    seed=42
)
train_data = train_datagen.flow_from_directory(
    directory='feedData/train/',
    target_size=(256, 256),
    class_mode='categorical',
    batch_size=64,
    seed=42
)
```

```
Found 623 images belonging to 3 classes.
Found 5611 images belonging to 3 classes.
```

Testing Some Models

```
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(filters=32, kernel_size=(3, 3), input_shape=(256, 256, 3), activation='relu'),
    tf.keras.layers.MaxPool2D(pool_size=(2, 2), padding='same'),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax')
])
```

```
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(filters=32, kernel_size=(4, 4), input_shape=(256, 256, 3), activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPool2D(pool_size=(2, 2), padding='same'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax')
])
```

```
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(filters=32, kernel_size=(3, 3), input_shape=(256, 256, 3), activation='relu'),
    tf.keras.layers.MaxPool2D(pool_size=(2, 2), padding='same'),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax')
])
```

Best and Cleanest Results

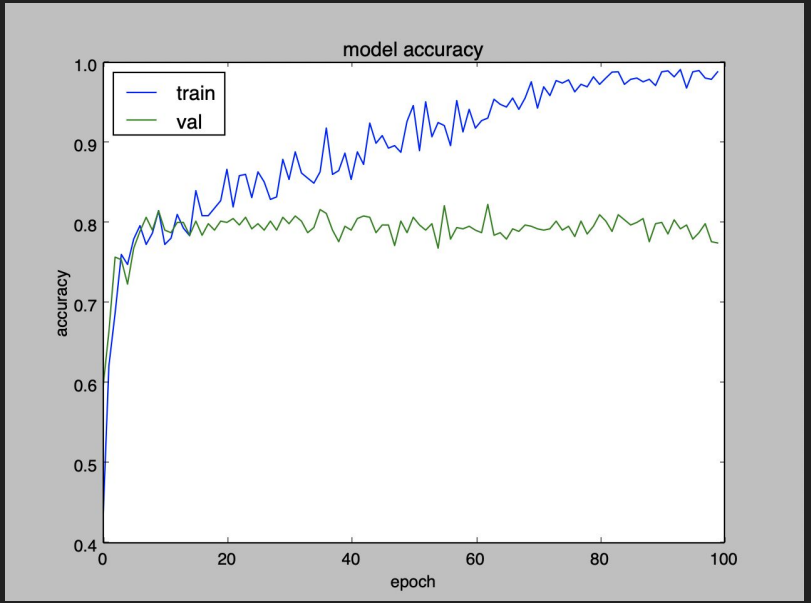
All models had roughly the same accuracy, and eventually started overfitting. Setting the steps per epoch to be low appeared to help, but the model eventually overfit

```
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(filters=32, kernel_size=(3, 3), input_shape=(256, 256, 3), activation='relu'),
    tf.keras.layers.MaxPool2D(pool_size=(2, 2), padding='same'),
    tf.keras.layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu'),
    tf.keras.layers.MaxPool2D(pool_size=(2, 2), padding='same'),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax')
])

model.compile(
    loss=tf.keras.losses.categorical_crossentropy,
    optimizer=tf.keras.optimizers.Adam(),
    metrics=[tf.keras.metrics.CategoricalAccuracy(name='accuracy')]
)

history = model.fit(
    train_data,
    steps_per_epoch=10,
    validation_data=valid_data,
    epochs=100
)
```

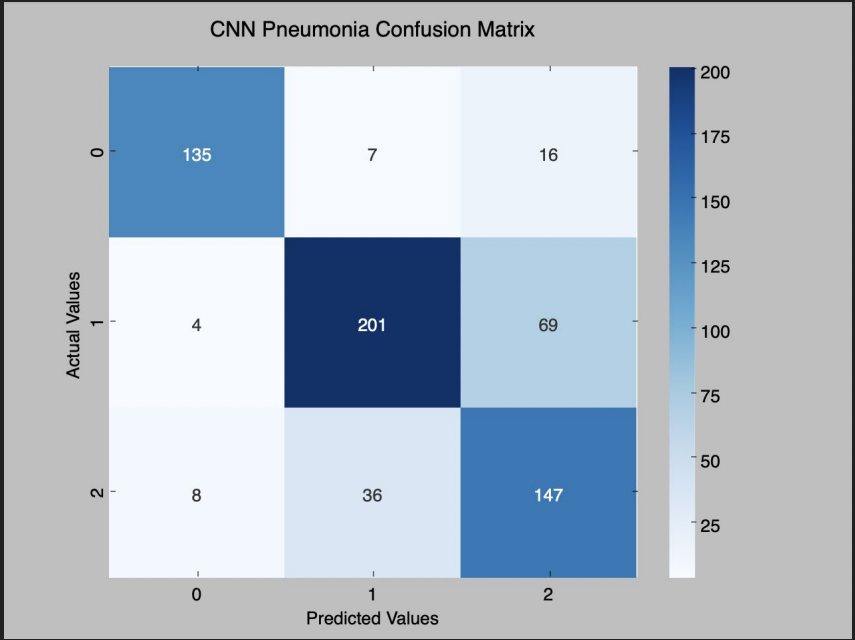
Results - Accuracy

All had roughly the same final accuracy

- Overfitting after around 20 epochs
- Overall validation accuracy ended around 0.79 (max at 0.8234)



Results - Confusion Matrix




```
PneumoniaKey = {"No Pneumonia (healthy)":0,
    "Bacterial Pneumonia":1,
    "Viral Pneumonia":2,
}
```

	precision	recall	f1-score	support
0	0.918	0.854	0.885	158
1	0.824	0.734	0.776	274
2	0.634	0.770	0.695	191
micro avg	0.775	0.775	0.775	623
macro avg	0.792	0.786	0.785	623
weighted avg	0.789	0.775	0.779	623

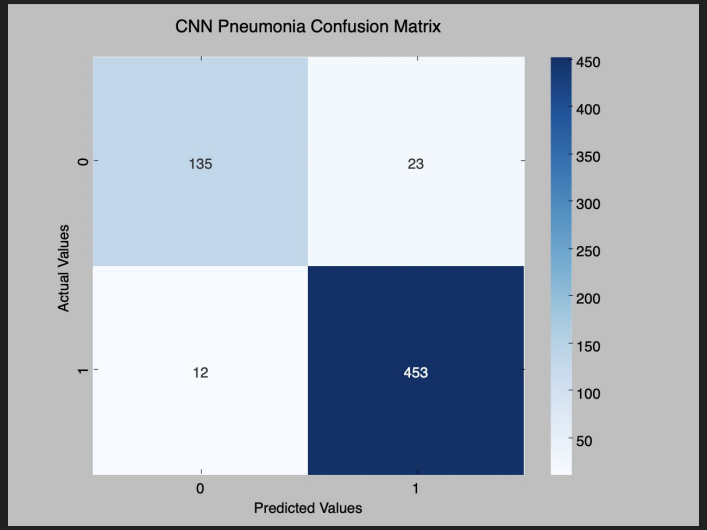
Results - Confusion Matrix Continued

Much better accuracy for pneumonia/non-pneumonia - **94.38%**

(TP+TN)/total = (135+453)/623 = 0.9438

```
print(classification_report(labels_mod, predictions_mod, digits=3))
```

	precision	recall	f1-score	support
0	0.918	0.854	0.885	158
1	0.952	0.974	0.963	465
micro avg	0.944	0.944	0.944	623
macro avg	0.935	0.914	0.924	623
weighted avg	0.943	0.944	0.943	623



Potential Improvements

More training data

Additional medical information

- Temperature
- Lung capacity
- Age

More tuning of the CNN

Manufacturing additional training data from the training set

Oversampling and undersampling the data

Potentially the training data might be mis-classified

Conclusion - Did we achieve the original goal?

Not exactly. The original goal was to be able to detect COVID cases from xray images, and that attempt was abandoned to focus only on pneumonia. Additionally the pneumonia classification accuracy was not quite high enough to be able to provide assistance, although the pneumonia/no pneumonia classification seems like it could be useful.