Diagnosing Pneumonia

•••

With Convolutional Neural Networks

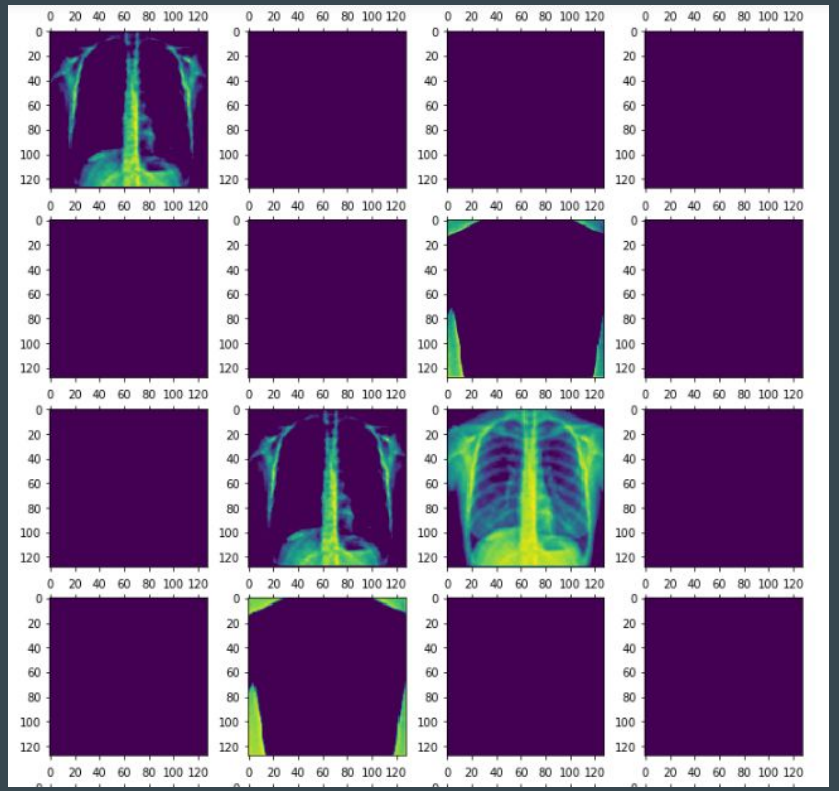
Data Collected From:

Dataset:

- Kaggle.com

Domain Information:

- Mayo Clinic
- iMedicalSociety.org
- MedlinePlus.gov



What is Pneumonia?

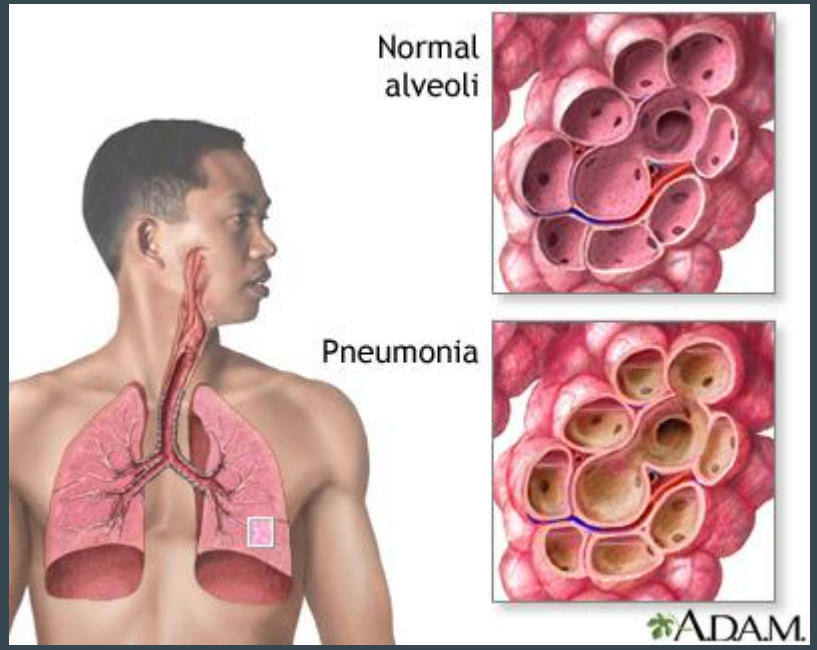


Image from MedlinePlus.gov

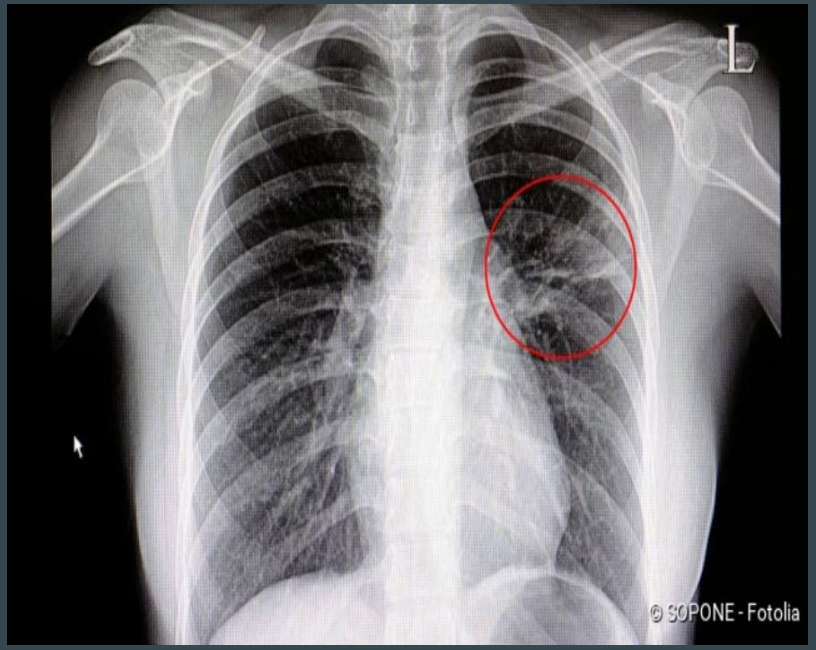
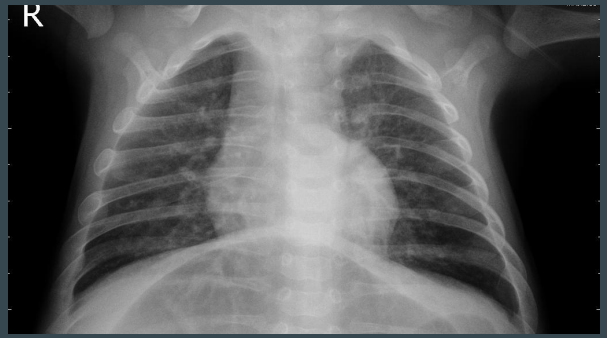
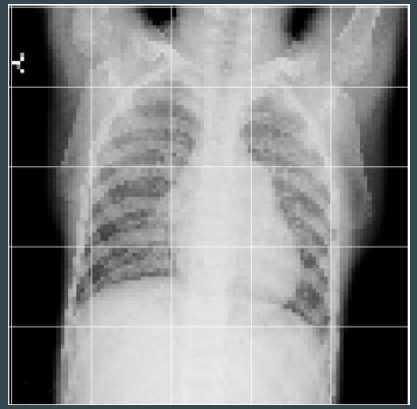
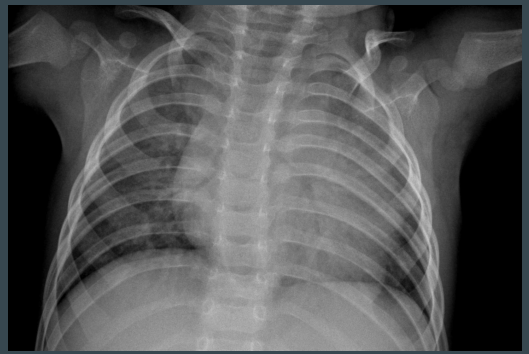


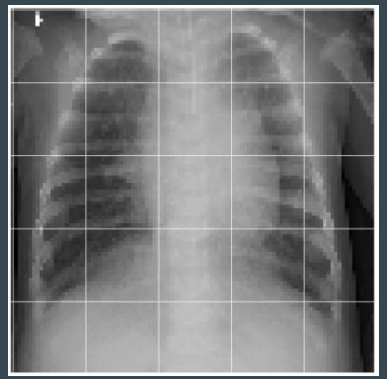
Image from iMedicalSociety.org

Obstacles for Machine Learning







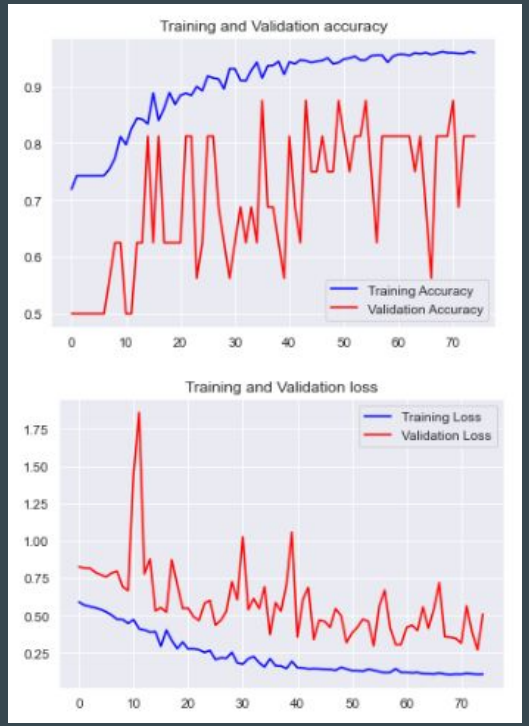


CNNs only view numbers that represent the pixel intensity for each channel

Noise in x-rays can be created by:

- Jittery patients (e.g., infants)
- Low quality x-rays that pick-up more body silhouettes
- Comparing x-rays of different resolutions after preprocessing

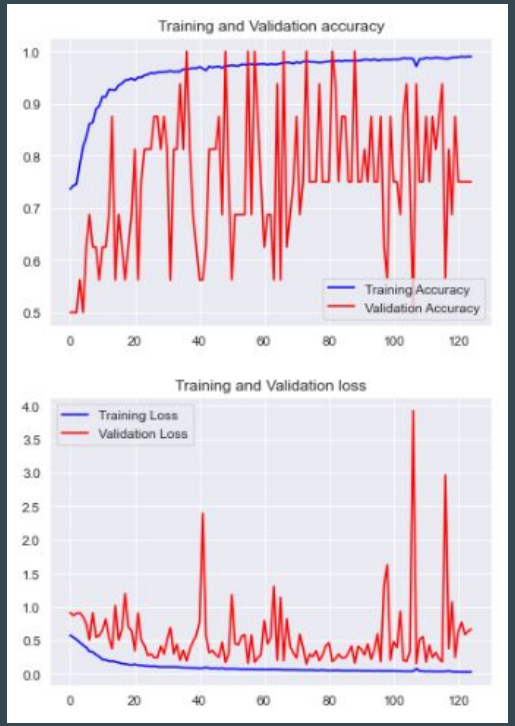
Baseline Model



Model Architecture:

- 4 Hidden Layers (HL)
- HL 1 & 2 contained **50 filters** each
- HL 3 contained **75 filters**
- HL 4 : Dense layer with **75 neurons**
- Output Layer: Dense layer with 1 neuron
- Optimizer: Stochastic Gradient Descent (SGD)
- Learning Rate: 0.01
- Ran for 75 epochs

Baseline Run Through More Epochs



Reran the model for 125 epochs

Results:

- Greater variance in the validation metrics
- Validation loss appears to diverge at the tail end
- Model Overfit
- Test Accuracy stayed at 72% - 73%

Gradient Tuning

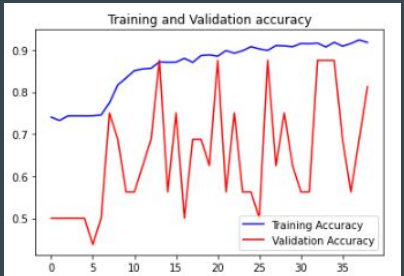
```
aug_datagen = ImageDataGenerator(rescale=1./255,
                                 rotation_range=30,
                                 width_shift_range=0.15,
                                 height_shift_range=0.15,
                                 shear_range=0.1,
                                 zoom_range=0.1,
                                 vertical_flip=True,
                                 fill_mode='nearest')
```

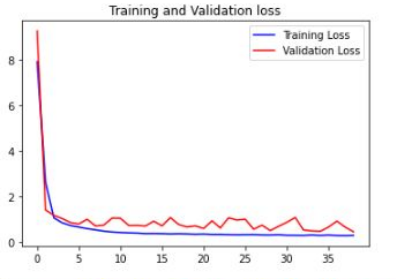
```
np.random.seed(42)

model_augreg3 = models.Sequential()

model_augreg3.add(layers.Conv2D(64, (3,3), padding='same', activation='relu',
                        kernel_initializer='he_normal',
                        kernel_regularizer='l2',
                        input_shape=(128, 128, 3)))
# model.add(layers.BatchNormalization())
model_augreg3.add(layers.MaxPooling2D((2,2)))


model_augreg3.add(layers.Conv2D(128, (3,3), padding='same',
                        kernel_initializer='he_normal',
                        kernel_regularizer='l2',
                        activation='relu'))
# model.add(layers.BatchNormalization())
model_augreg3.add(layers.MaxPooling2D((2,2)))

model_augreg3.add(layers.Conv2D(256, (3,3), padding='same',
                        kernel_initializer='he_normal',
                        kernel_regularizer='l2',
                        activation='relu'))
# model.add(layers.BatchNormalization())
model_augreg3.add(layers.MaxPooling2D((2,2)))

model_augreg3.add(layers.Flatten())
model_augreg3.add(layers.Dense(64, kernel_initializer='he_normal',
                        kernel_regularizer='l2',
                       activation='relu'))
# model.add(layers.Dropout(0.2))
model_augreg3.add(layers.Dense(1, activation='sigmoid'))

model_augreg3.compile(loss='binary_crossentropy',
              optimizer='Adam',
              metrics=['acc'])
model_augreg3.summary()
```

Short Version:

- Changed to Adam optimizer (learning rate: 0.001)
- Used He_normal to initialize starting weights.
- Changed input size to 128 x 128
- Utilized both L2 Regularization and Data Augmentation
  - This caused underfitting
- Increased model complexity
- Reduced epochs

Results!





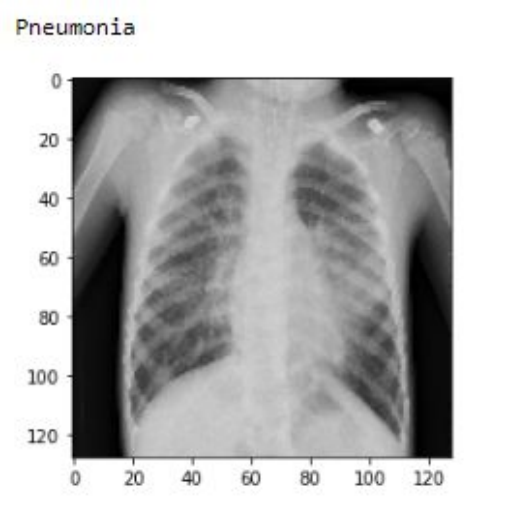

```
augreg_result_test2 = model_augreg4.evaluate(test_img, test_labels)
print(f"Model's Test Accuracy = {round((augreg_result_test2[1]*100), 3)}%")
print(f"Model's Test Loss = {augreg_result_test2[0]}")
```

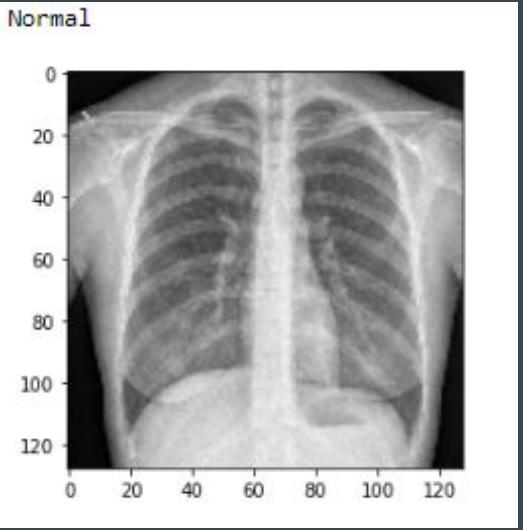
executed in 9.38s, finished 15:03:54 2020-09-17

```
624/624 [==============================] - 9s 15ms/step
Model's Test Accuracy = 85.417%
Model's Test Loss = 0.3848557946009514
```

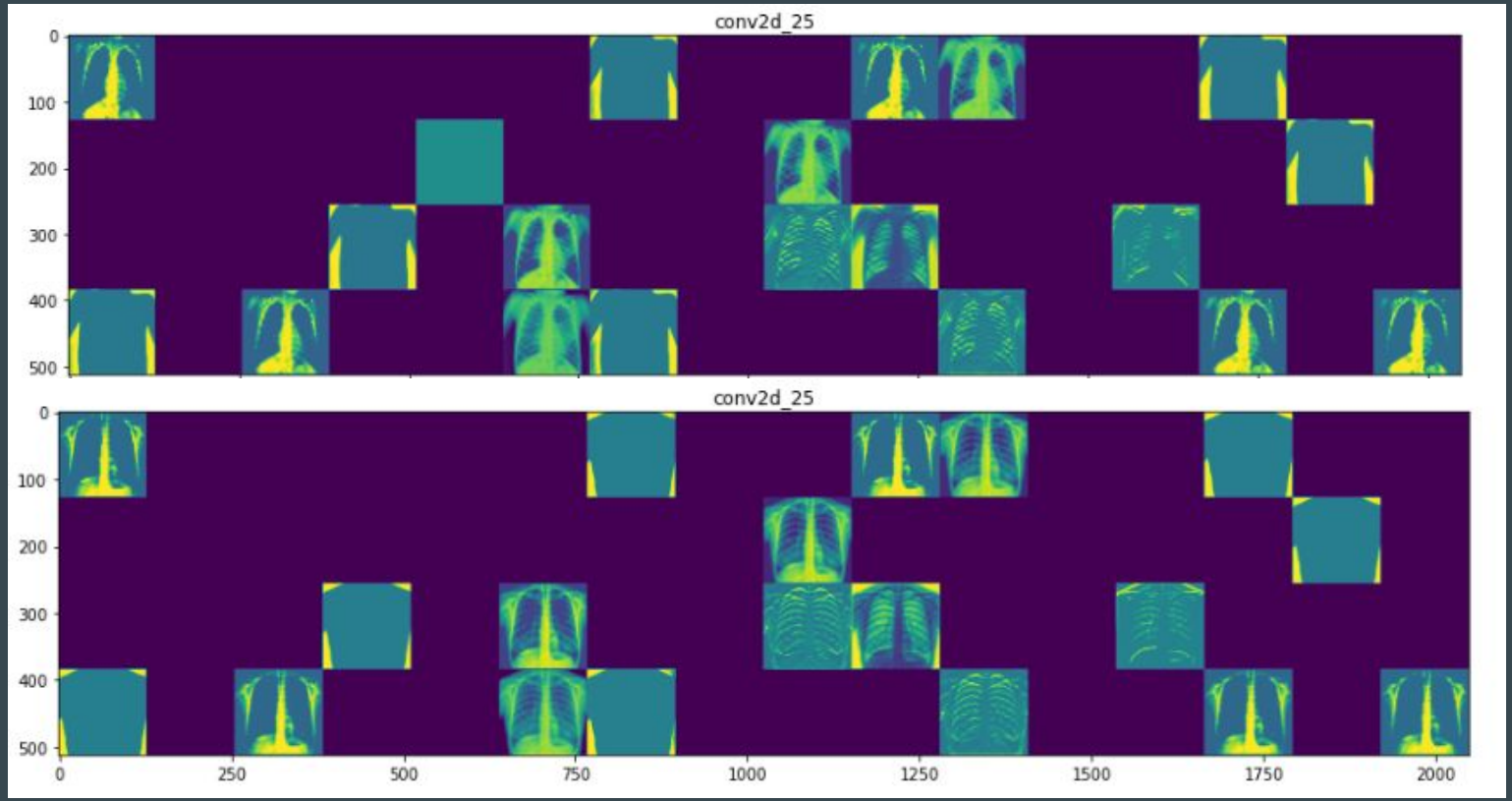
- Still high variance in the Validation accuracy
- Validation loss improved drastically
- Test accuracy jumped up to 85% and loss at an all time low.
- Further attempts to tune the model lead to either overfitting the model or underfitting

Under the Hood

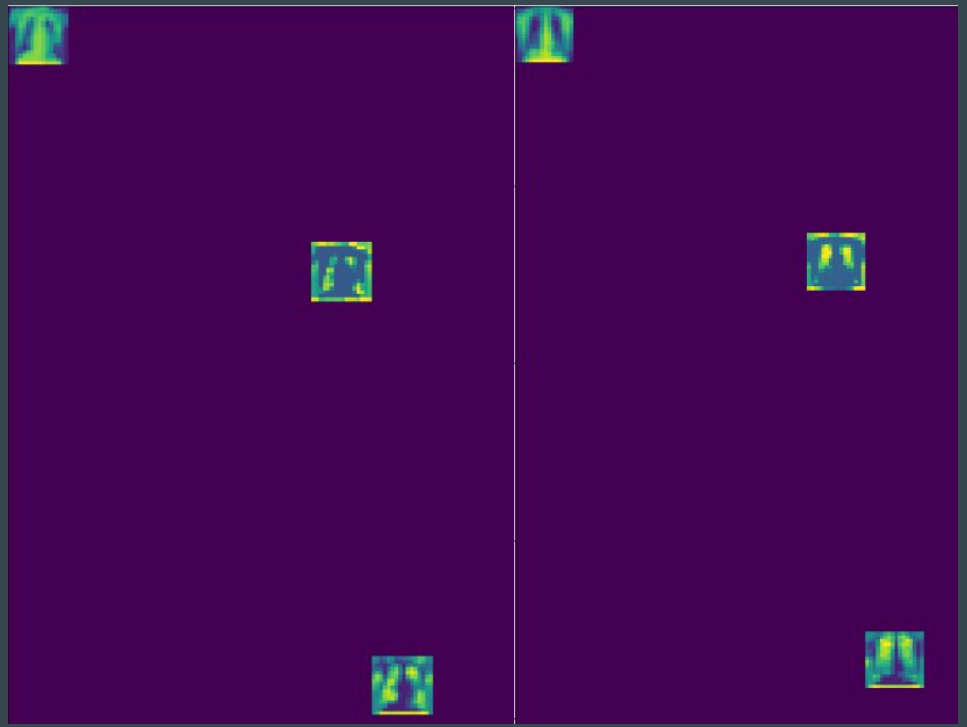




Under the Hood



Under the Hood



Conclusion:

- Model sifts through the various intensity values to filter out as much noise as possible
- It creates “negatives” of the images as a form of “check”
- Increasing the “contrast” allows it to check for “clarity”

Future Work

- Experiment more with adding strides to pooling layers
- Set seed parameters through Keras
- Increase efficiency through usage of Depthwise Convolutions

Special Thanks to these Medium Authors

Chi-Feng Wang: A Basic Introduction to Separable Convolutions

Atul Pandey: Depth-wise Convolution and Depth-wise Separable Convolution

Gabriel Pierobon: Visualizing intermediate activation in Convolutional Neural Networks with Keras

Shashank Ramesh: A guide to an efficient way to build neural network architectures- Part II: Hyper-parameter selection and tuning for Convolutional Neural Networks

Extra Special Thanks to You!

...

Ben Mauss
ben.mauss@gmail.com