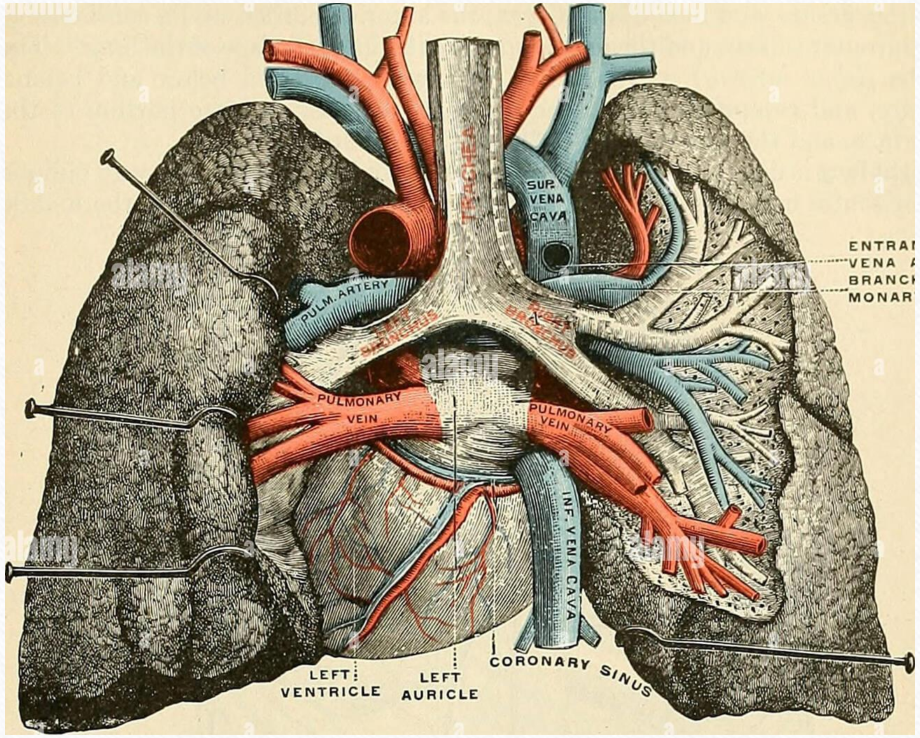
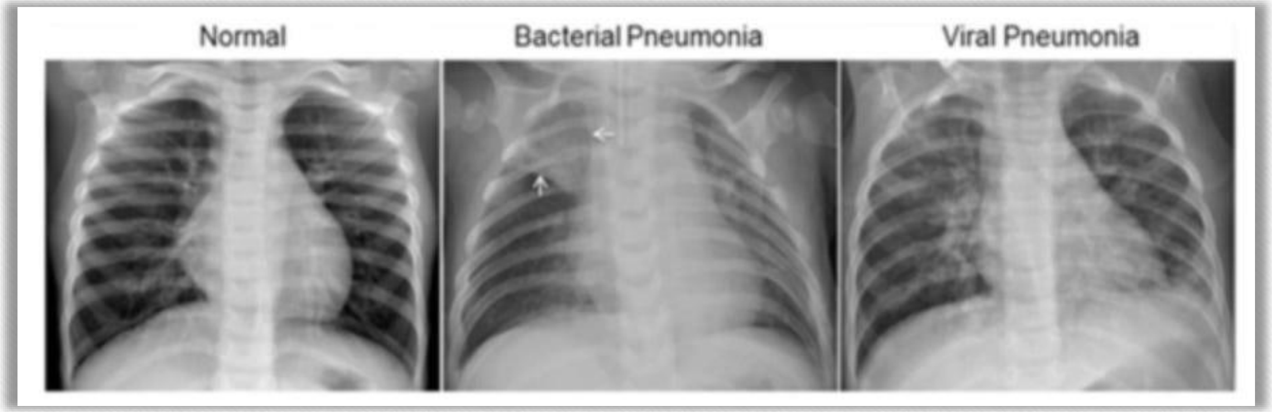
Pixels to Prognosis ||

A Visual Odyssey of Lung Diagnosis





Machine Learning & AI

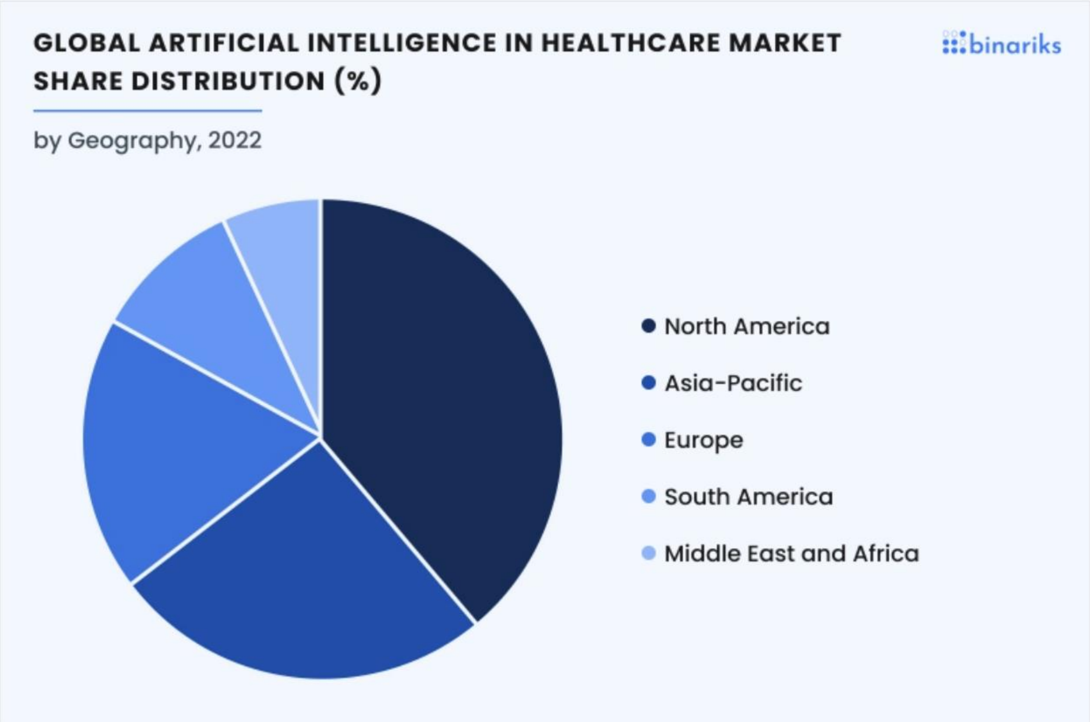
// In Medicine

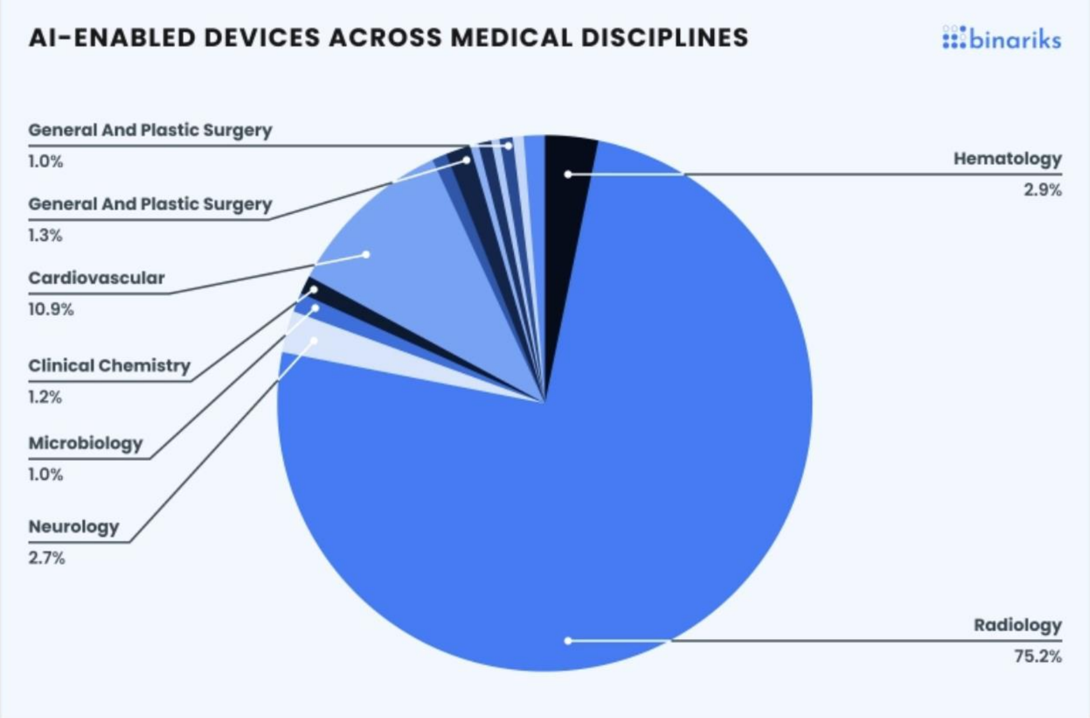
- CT Scans
- Mental Health
- Assessment of Disease
- Care

- Profile
- Blood work
- Genetic History
- Drug Discovery

Growing Field //

- 16.3 Billion (2022) -> 17.55 Billion (2029)
- Data 10 trillion gigabytes by 2025
- 86 % of the industry





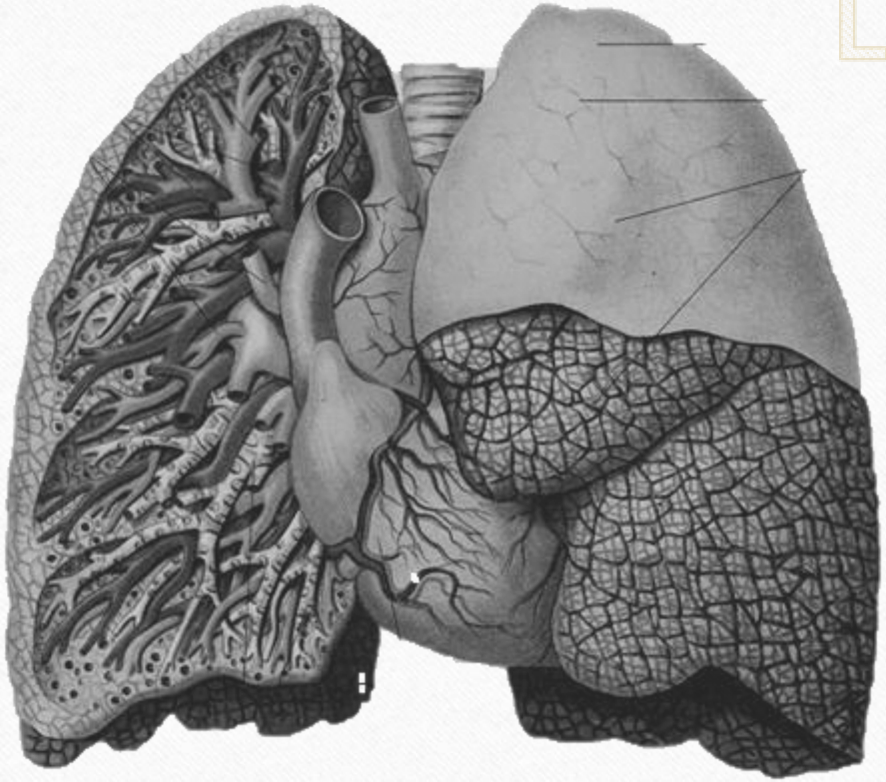
// Not Just Medicine

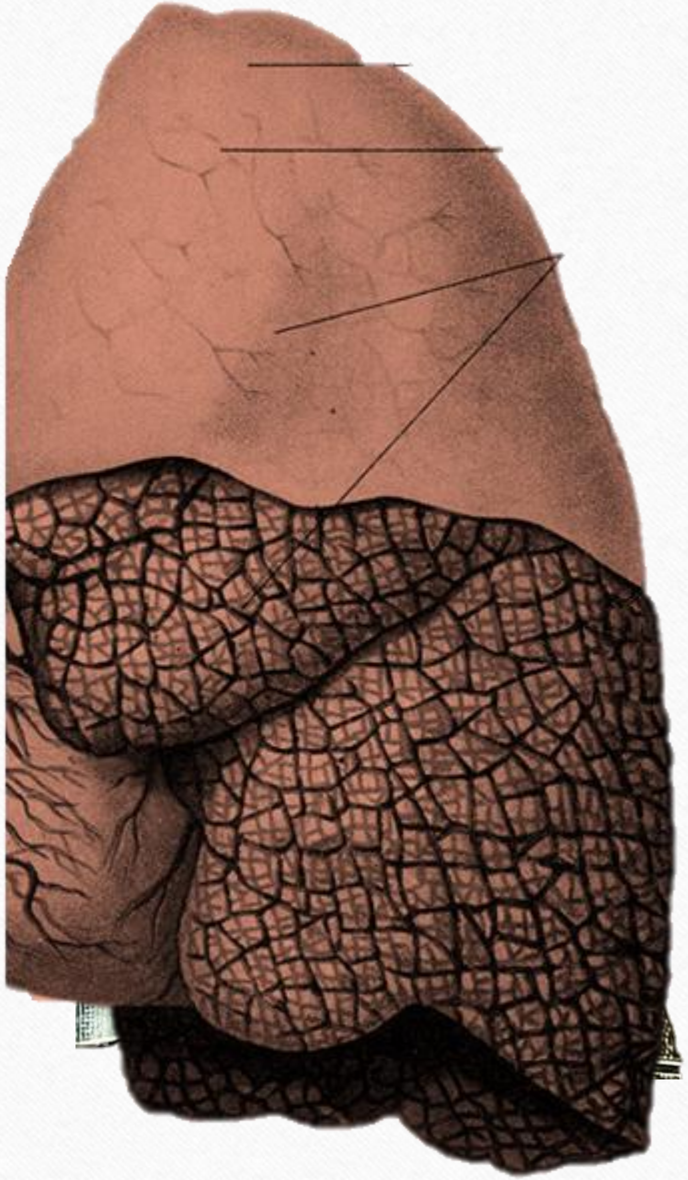
- Cyber-Security
- 24/7 Service
- Technology (watches)
- Payroll
- Prescriptions

Closer look at Layers

Layers

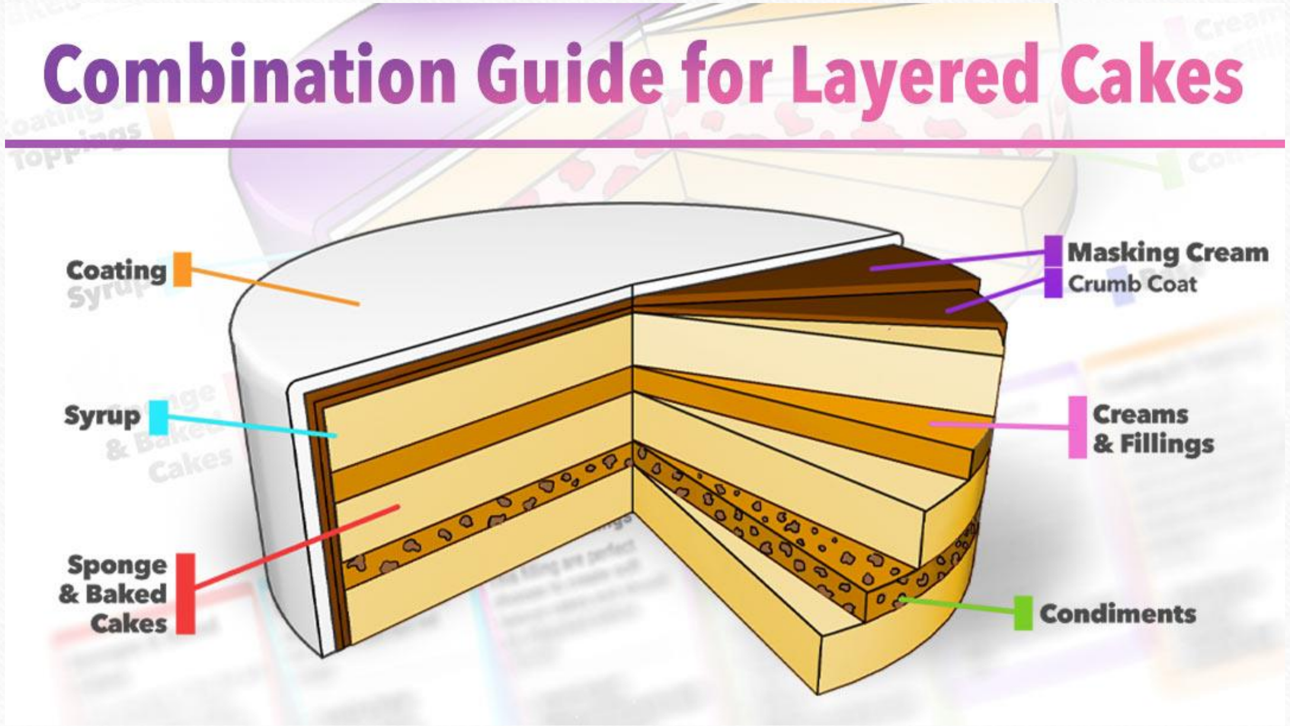
Layers





CONV2D

→ Base layer with processed and mix ingredients (pixels)

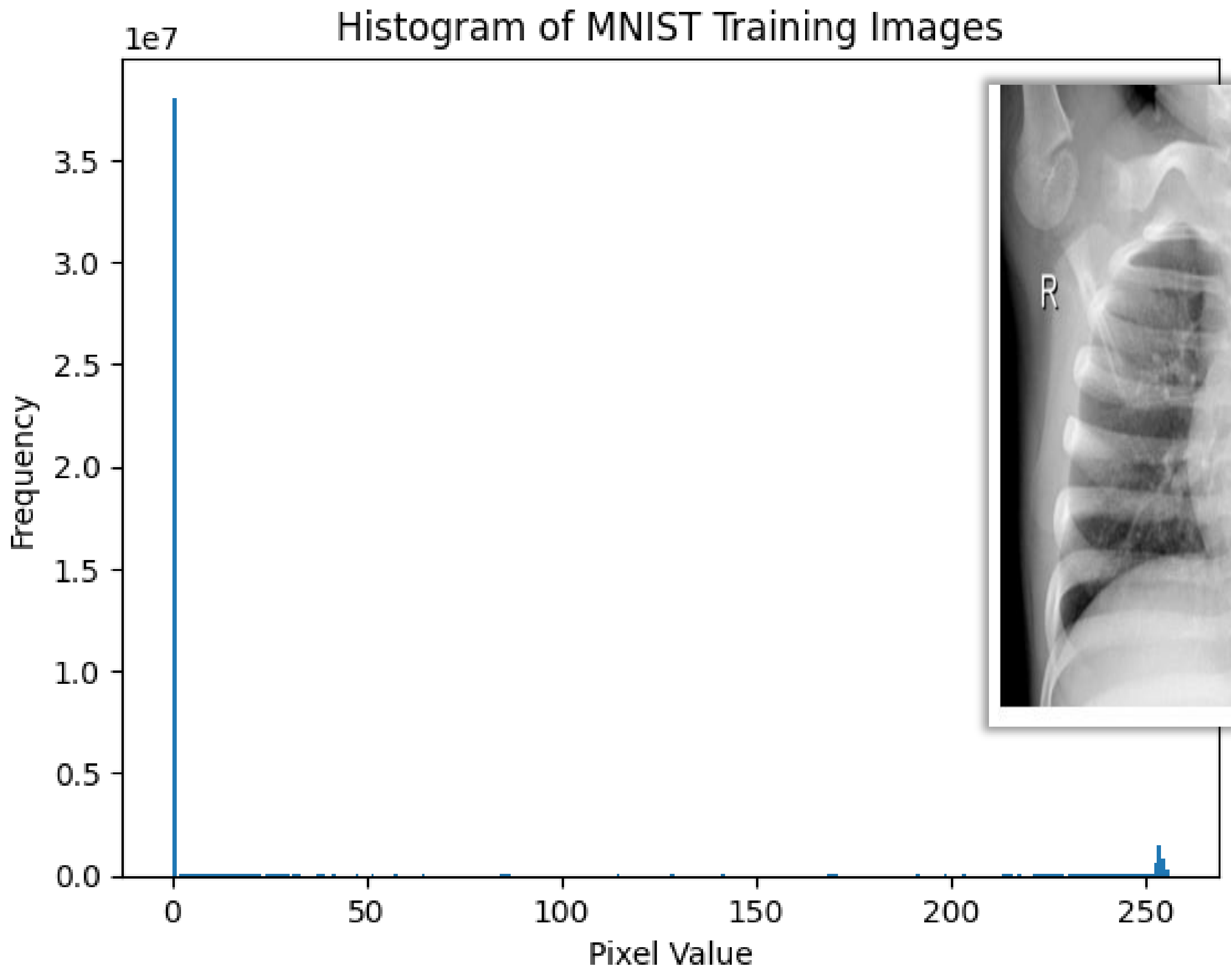


FLATTEN

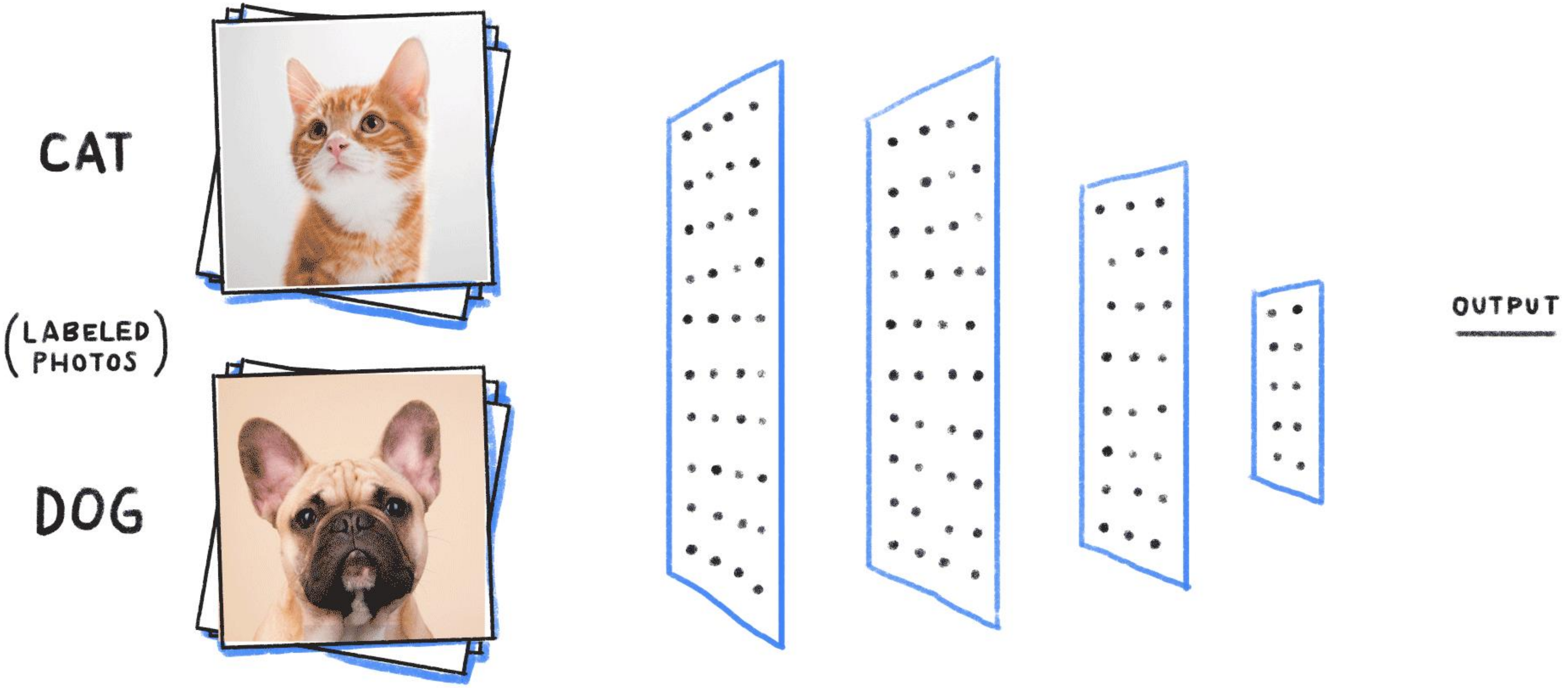
→ Each stack makes one cohesive item

DENSE

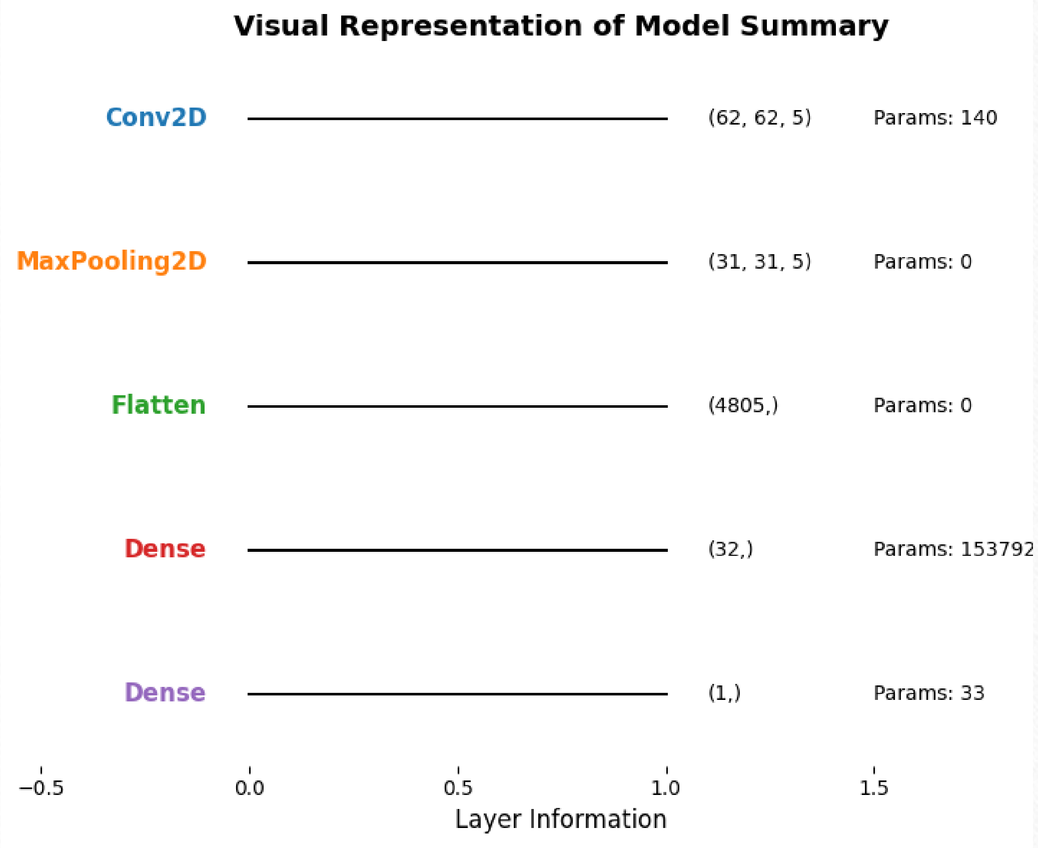
→ The filling that contributes to the overall taste



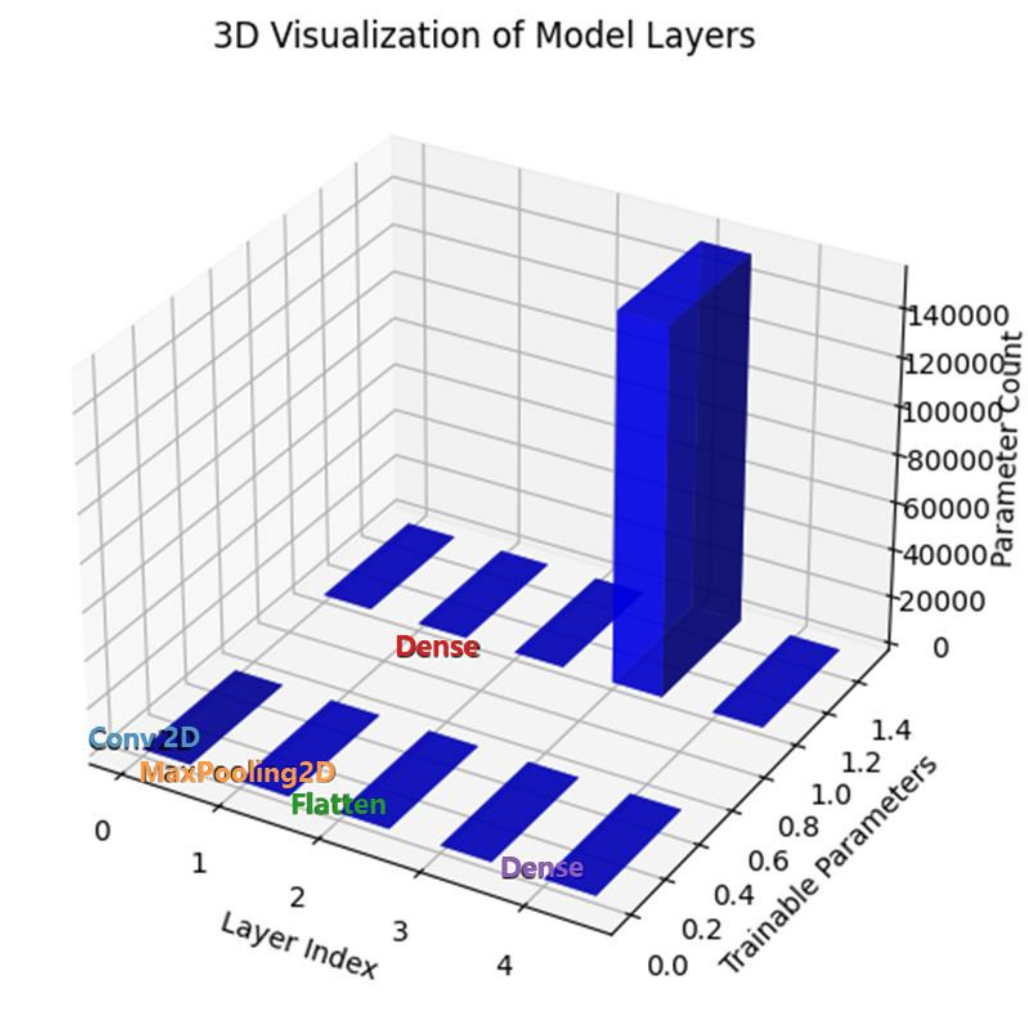
- A **limitation** of training and analyzing **black & white** images
- **No color** differential means **fewer ways** to identify difference


CAT
(LABELED PHOTOS)
DOG
OUTPUT

Layer // Data

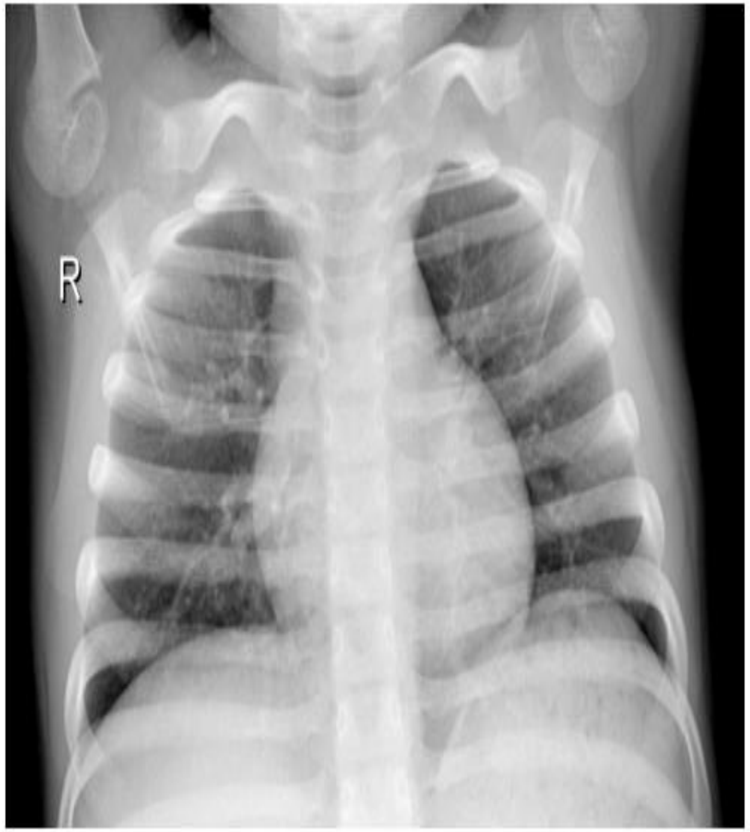


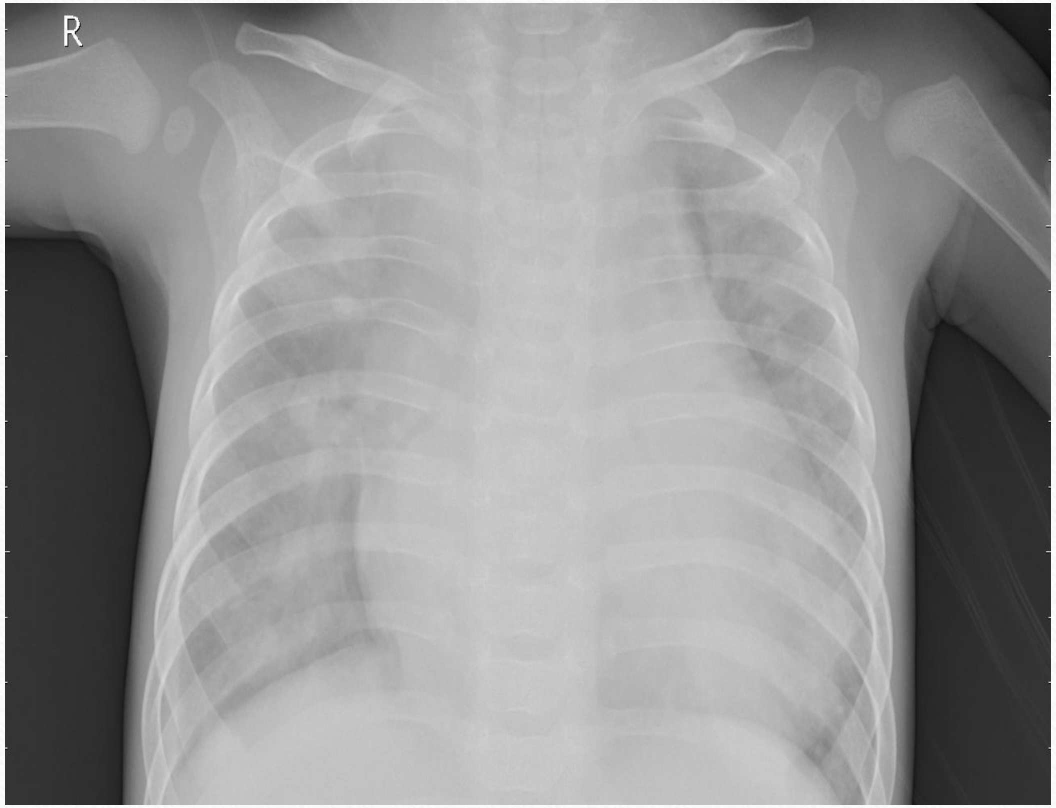
Visual Representation of Model Summary
Conv2D
(62, 62, 5)
Params: 140
MaxPooling2D
(31, 31, 5)
Params: 0
Flatten
(4805,)
Params: 0
Dense
(32,)
Params: 153792
Dense
(1,)
Params: 33
−0.5
0.0
0.5
1.0
1.5
Layer Information



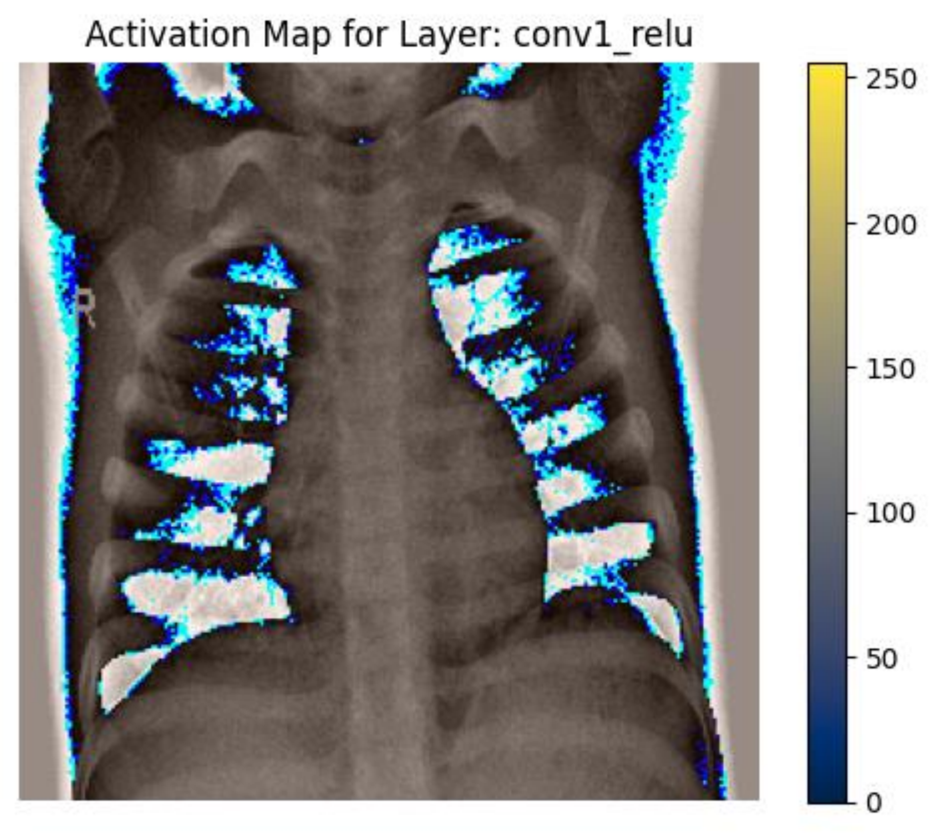
3D Visualization of Model Layers
Parameter Count
140000
120000
100000
80000
60000
40000
20000
0
Dense
Conv2D
MaxPooling2D
Flatten
Dense
Layer Index
0
1
2
3
4
Trainable Parameters
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

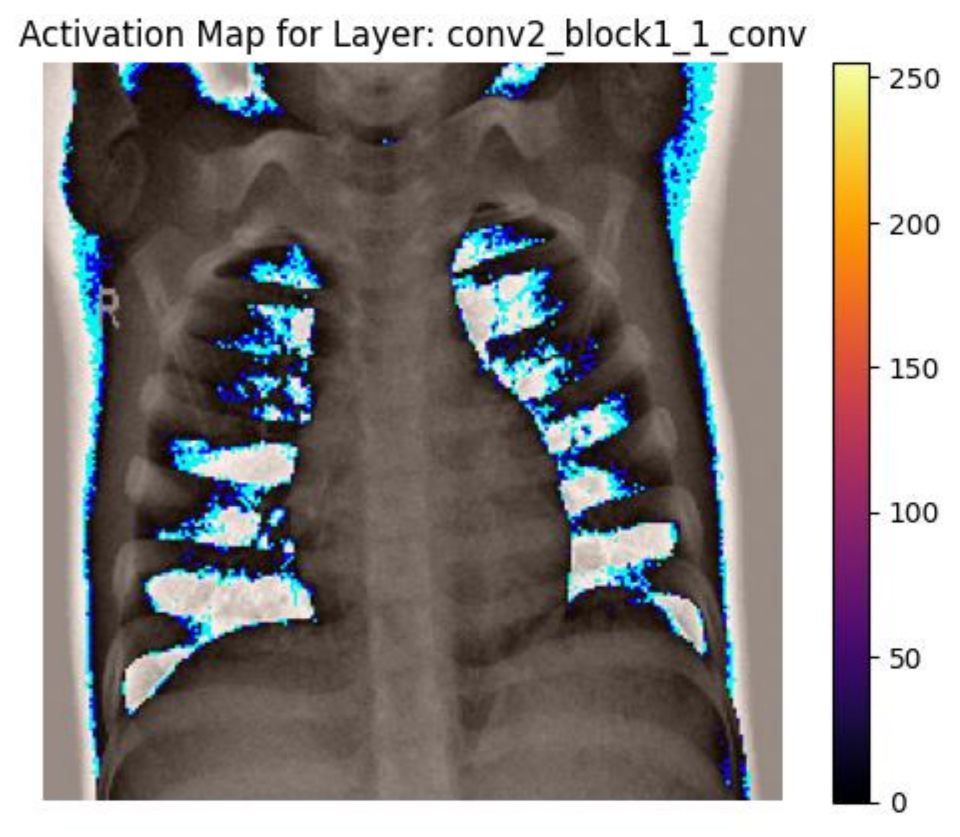
Normal Lungs // Bacteria





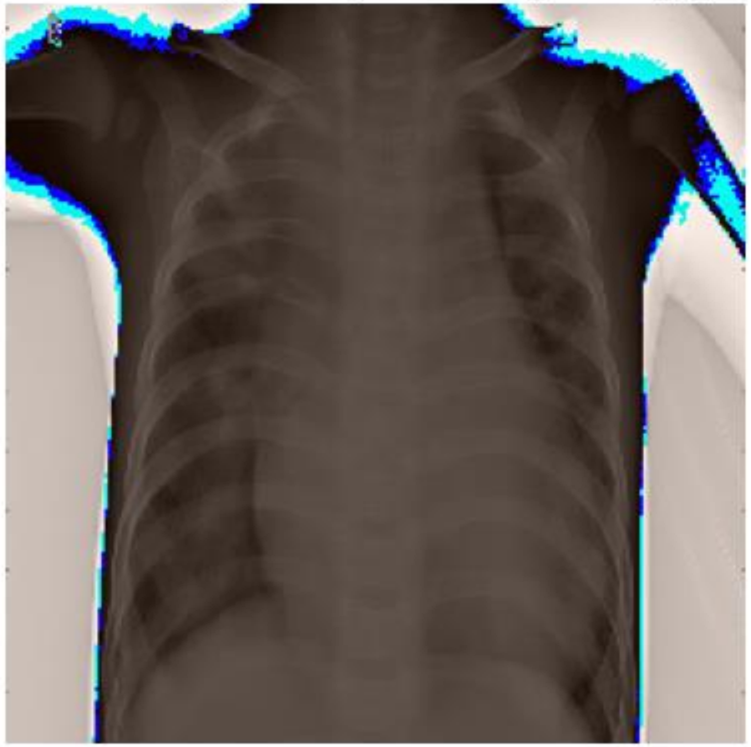
Normal Lungs // Layering



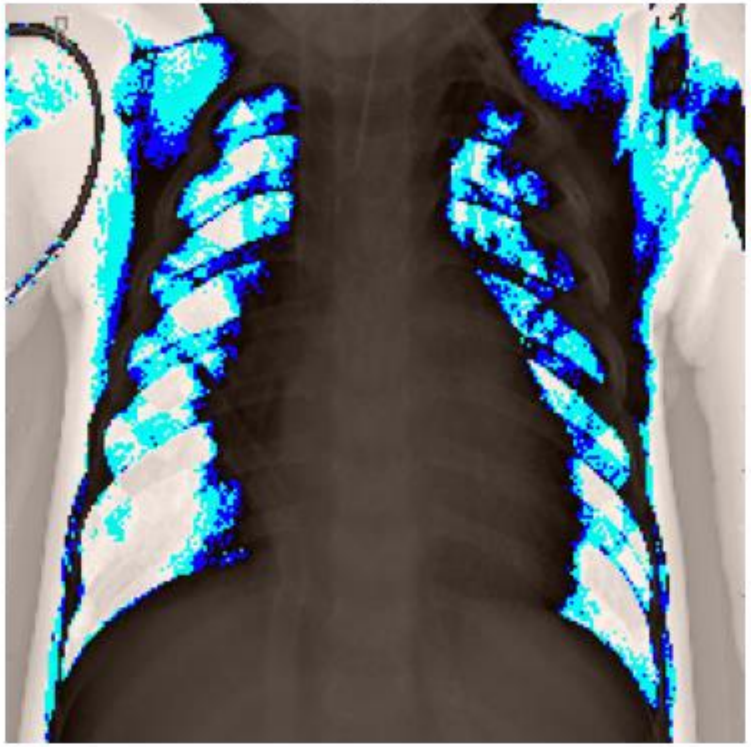


CNN Layer Visual

Activation Map for Layer: conv2_block1_2_bn

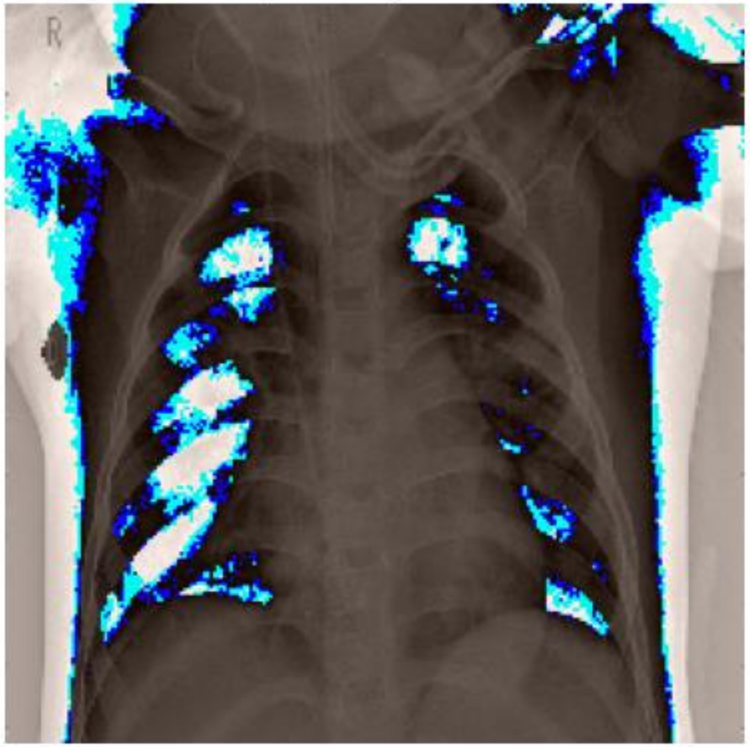


Activation Map for Layer: conv2_block1_out

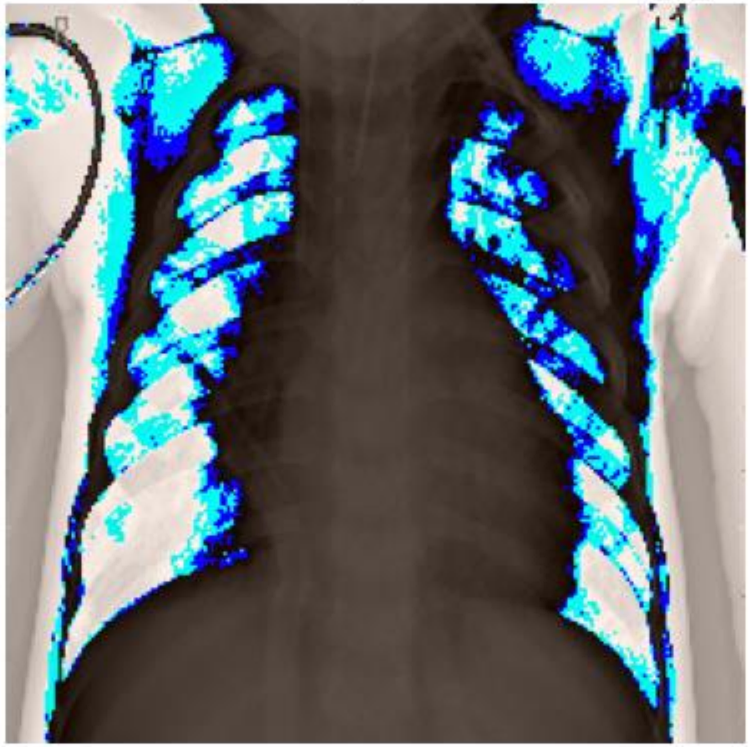


CNN Layer Visual

Activation Map for Layer 80 - Channel 40



Activation Map for Layer: conv2_block1_3_bn



Code // Layers

```
# Model definition
model = tf.keras.models.Sequential()
# Add initial convolutional layer
model.add(tf.keras.layers.Conv2D(filt
ers=5, kernel_size=3,
activation='relu', input_shape=[64,
64, 3]))
# Add maximum pooling layer
model.add(tf.keras.layers.MaxPool2D(p
ool_size=2, strides=2))
# Add flattening layer
model.add(tf.keras.layers.Flatten())
# Add neural network
model.add(tf.keras.layers.Dense(units
=32, activation='relu'))
# Add final layer output
model.add(tf.keras.layers.Dense(units
=1, activation='relu'))

model.summary()
```

Layer // Output

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_1 (Conv2D)           (None, 64, 64, 5)         140

 max_pooling2d_1 (MaxPoolin  (None, 32, 32, 5)         0
 g2D)

 flatten_1 (Flatten)         (None, 5120)              0

 dense_3 (Dense)             (None, 8)                 40968

 dense_4 (Dense)             (None, 4)                 36

 dense_5 (Dense)             (None, 1)                 5

=================================================================
Total params: 41149 (160.74 KB)
Trainable params: 41149 (160.74 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```python
# Compile and run model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(x=training_set, validation_data=test_set, epochs=10)
```

```
Epoch 1/10
163/163 [==============================] - 585s 3s/step - loss: 0.6758 - accuracy: 0.6835 - val_loss: 0.6510 - val_accuracy: 0.6138
Epoch 2/10
163/163 [==============================] - 60s 367ms/step - loss: 0.4823 - accuracy: 0.7747 - val_loss: 0.5353 - val_accuracy: 0.8093
Epoch 3/10
163/163 [==============================] - 58s 357ms/step - loss: 0.4554 - accuracy: 0.8250 - val_loss: 0.6877 - val_accuracy: 0.7997
Epoch 4/10
163/163 [==============================] - 58s 356ms/step - loss: 0.4219 - accuracy: 0.8355 - val_loss: 0.4773 - val_accuracy: 0.8013
Epoch 5/10
163/163 [==============================] - 59s 360ms/step - loss: 0.6128 - accuracy: 0.6904 - val_loss: 0.4504 - val_accuracy: 0.8333
Epoch 6/10
163/163 [==============================] - 60s 368ms/step - loss: 0.4242 - accuracy: 0.8294 - val_loss: 0.5663 - val_accuracy: 0.8109
Epoch 7/10
163/163 [==============================] - 58s 357ms/step - loss: 0.4023 - accuracy: 0.8434 - val_loss: 0.5056 - val_accuracy: 0.8205
Epoch 8/10
163/163 [==============================] - 59s 361ms/step - loss: 0.3710 - accuracy: 0.8530 - val_loss: 0.4693 - val_accuracy: 0.8381
Epoch 9/10
163/163 [==============================] - 59s 359ms/step - loss: 0.3562 - accuracy: 0.8627 - val_loss: 0.4584 - val_accuracy: 0.8349
Epoch 10/10
163/163 [==============================] - 58s 359ms/step - loss: 0.4487 - accuracy: 0.8248 - val_loss: 0.4937 - val_accuracy: 0.7997
<keras.src.callbacks.History at 0x7a1b90477e20>
```

```python
# Evaluate the model's performance
model_loss, model_accuracy = model.evaluate(test_set, verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
20/20 - 5s - loss: 0.4937 - accuracy: 0.7997 - 5s/epoch - 252ms/step
Loss: 0.49368688464164734, Accuracy: 0.7996794581413269
```

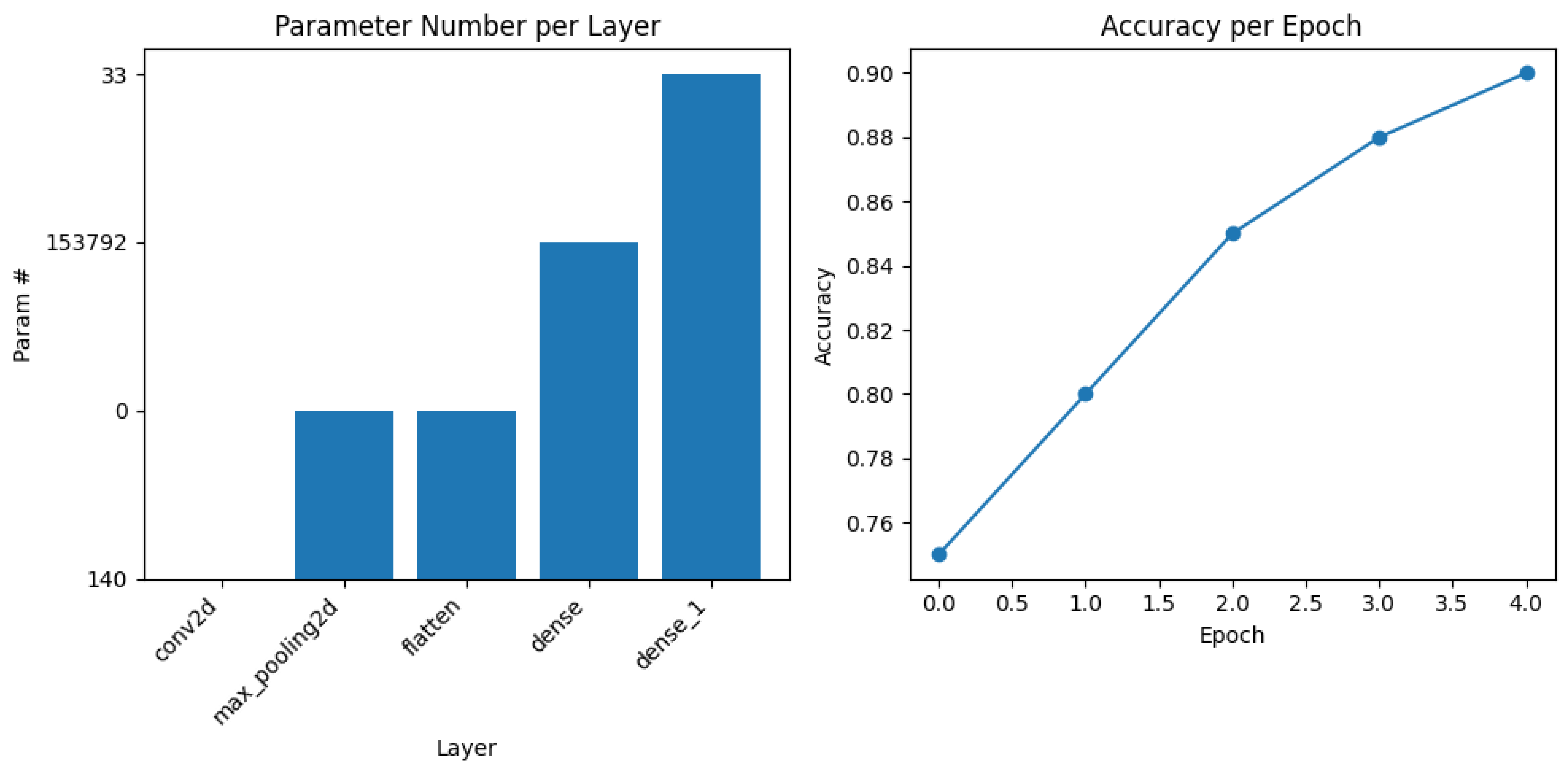
Final // Output

```
163/163 [==============================] - 92s 568ms/step - loss: 0.0497 - accuracy: 0.9843 - val_loss: 2.7677 - val_ac
Epoch 38/50
163/163 [==============================] - 92s 562ms/step - loss: 0.0475 - accuracy: 0.9854 - val_loss: 2.8572 - val_accuracy: 0.7500
Epoch 39/50
163/163 [==============================] - 92s 562ms/step - loss: 0.0807 - accuracy: 0.9745 - val_loss: 1.7909 - val_accuracy: 0.7837
Epoch 40/50
163/163 [==============================] - 92s 566ms/step - loss: 0.0541 - accuracy: 0.9854 - val_loss: 3.0313 - val_accuracy: 0.7436
Epoch 41/50
163/163 [==============================] - 91s 556ms/step - loss: 0.0435 - accuracy: 0.9891 - val_loss: 2.7641 - val_accuracy: 0.7564
Epoch 42/50
163/163 [==============================] - 94s 577ms/step - loss: 0.0380 - accuracy: 0.9910 - val_loss: 3.2735 - val_accuracy: 0.7179
Epoch 43/50
163/163 [==============================] - 92s 562ms/step - loss: 0.0369 - accuracy: 0.9904 - val_loss: 3.3030 - val_accuracy: 0.7212
Epoch 44/50
163/163 [==============================] - 92s 563ms/step - loss: 0.0330 - accuracy: 0.9937 - val_loss: 2.8620 - val_accuracy: 0.7532
Epoch 45/50
163/163 [==============================] - 93s 572ms/step - loss: 0.0313 - accuracy: 0.9944 - val_loss: 3.2988 - val_accuracy: 0.7276
Epoch 46/50
163/163 [==============================] - 93s 571ms/step - loss: 0.0304 - accuracy: 0.9948 - val_loss: 3.2151 - val_accuracy: 0.7436
Epoch 47/50
163/163 [==============================] - 93s 568ms/step - loss: 0.0286 - accuracy: 0.9946 - val_loss: 3.4359 - val_accuracy: 0.7244
Epoch 48/50
163/163 [==============================] - 90s 554ms/step - loss: 0.0273 - accuracy: 0.9964 - val_loss: 3.3581 - val_accuracy: 0.7276
Epoch 49/50
163/163 [==============================] - 94s 576ms/step - loss: 0.0250 - accuracy: 0.9975 - val_loss: 3.1801 - val_accuracy: 0.7468
Epoch 50/50
163/163 [==============================] - 92s 567ms/step - loss: 0.0238 - accuracy: 0.9975 - val_loss: 2.8972 - val_accuracy: 0.7708
<keras.src.callbacks.History at 0x79e0dfd19d80>
```

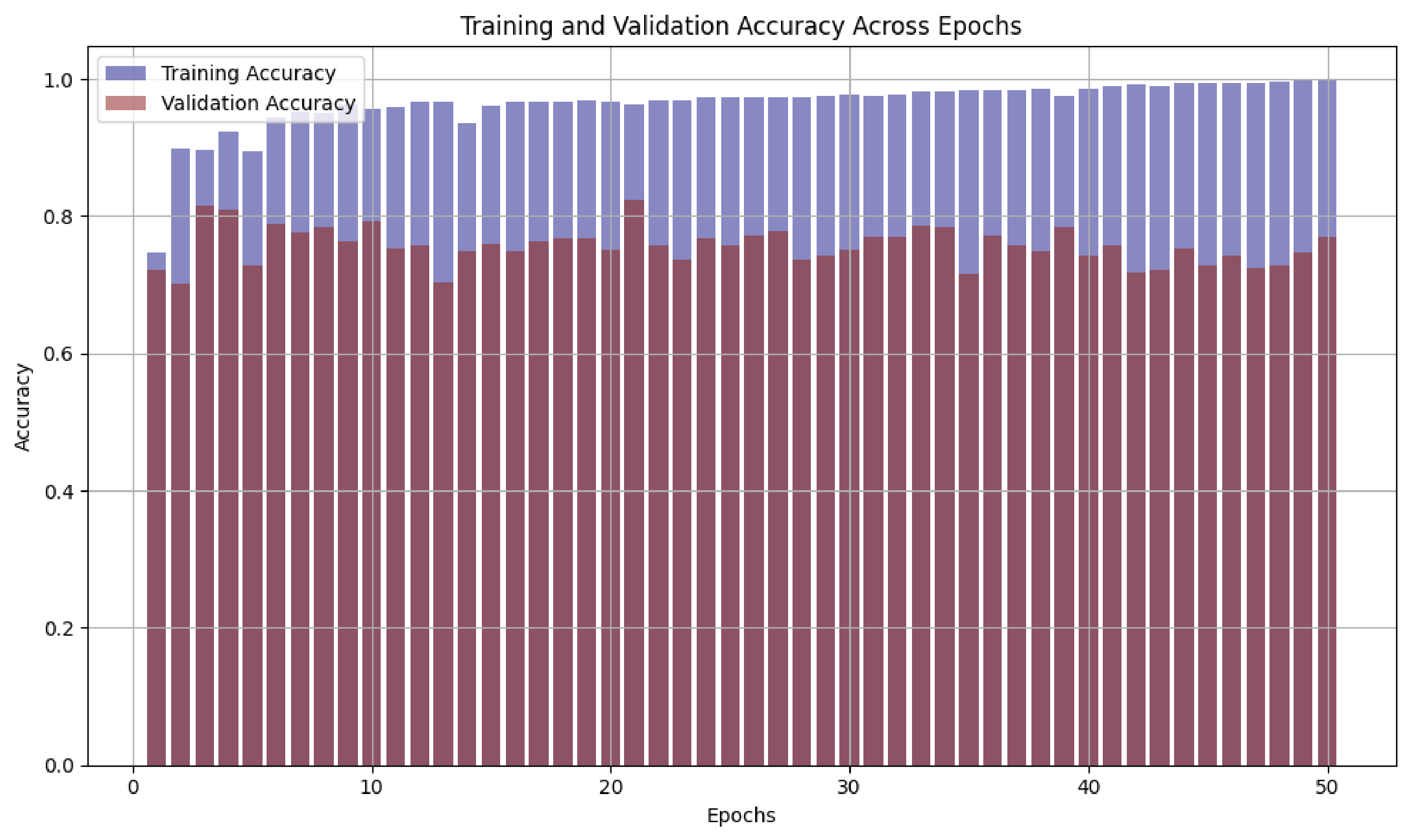
```
[10] # Evaluate the model's performance
     val_datagen = ImageDataGenerator(rescale=1./255, zoom_range = 0, horizontal_flip = False)
     val_set = val_datagen.flow_from_directory('/content/drive/MyDrive/chest_xray/val', target_size=(64, 64), batch_size=32, class_mode='binary')
     model_loss, model_accuracy = model.evaluate(val_set, verbose=2)
     print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
Found 154 images belonging to 2 classes.
5/5 - 29s - loss: 0.3136 - accuracy: 0.9740 - 29s/epoch - 6s/step
Loss: 0.31359970569610596, Accuracy: 0.9740259647369385
```

Test and Train // Accuracy 2



Kernel // Training & Validation



Pixels to Prognosis // Conclusion

Sources / /

- https://binariks.com/blog/artificial-intelligence-ai-healthcare-market/

- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7325854/