

CNN Project: X-Ray Image Semantic Segmentation

Melchor Ronquillo

Jason Luce

Michelle Nesbit



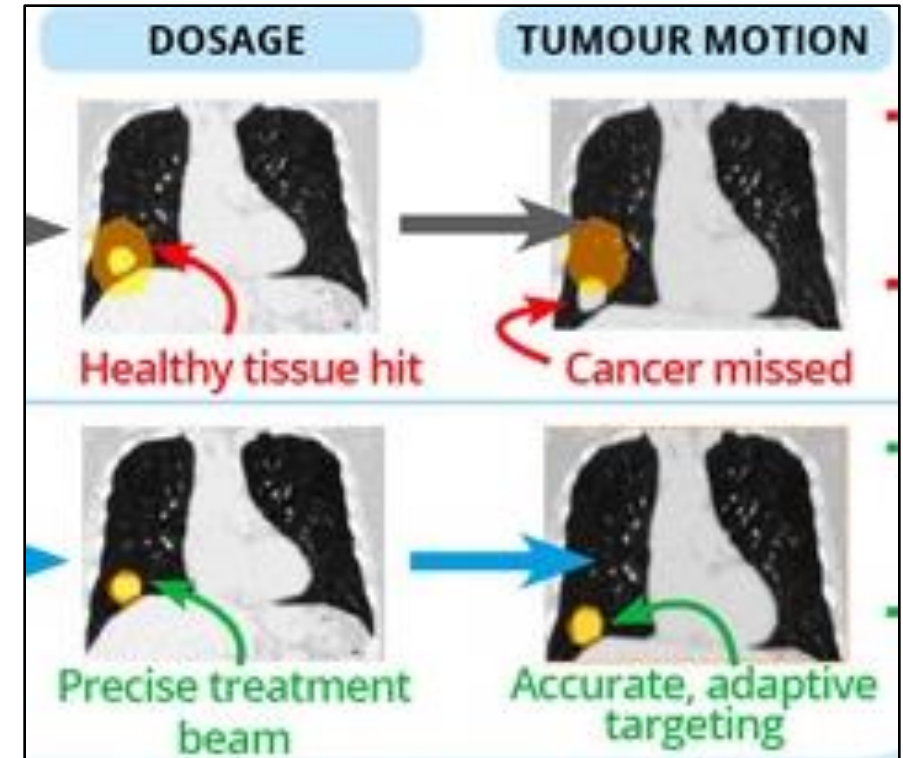
Outline

- Project Overview
- Image Preprocessing
- Model Selection
- Model Building
- Accuracy & Predictions
- Iterative Thoughts
- Questions & Answers

Project Overview

- In radiation therapy, i.e. the treatment of tumors with high-dose radiation, it would be helpful to be able to accurately track the location of a tumor during treatment
- This would allow a more accurate treatment beam, resulting in a greater percentage of the radiation dose to be applied to tumor vs. the surrounding healthy tissue
- The goal of this project is to train a neural network to automatically segment a medical image into 'tumor' and 'not-tumor' to help identify tumor location in medical images

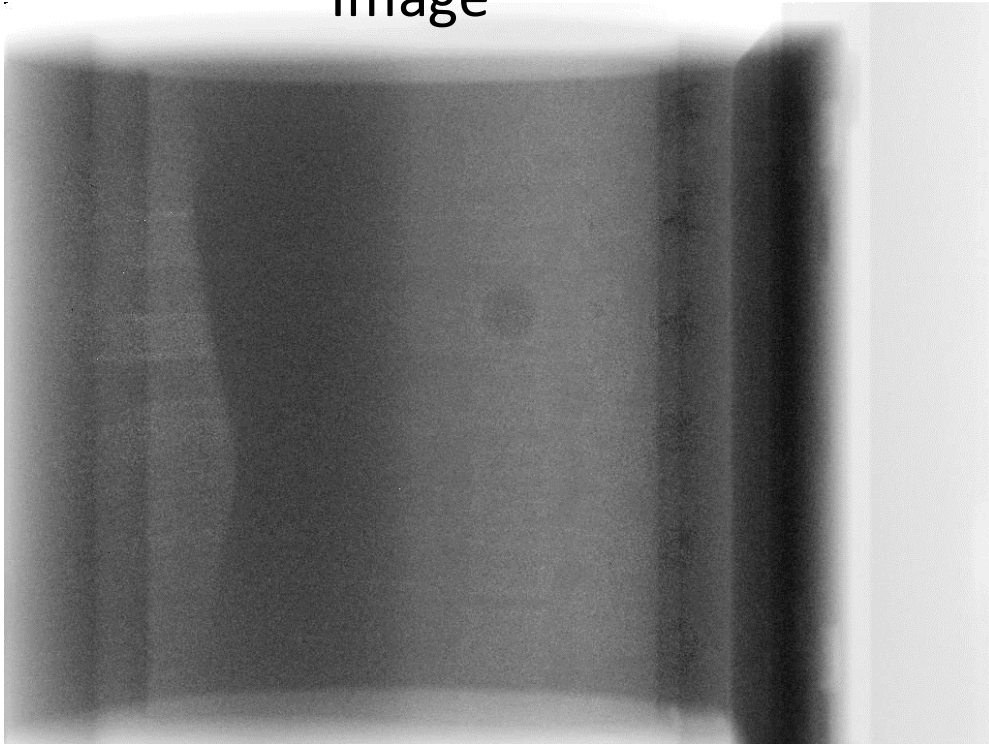
Radiation Therapy



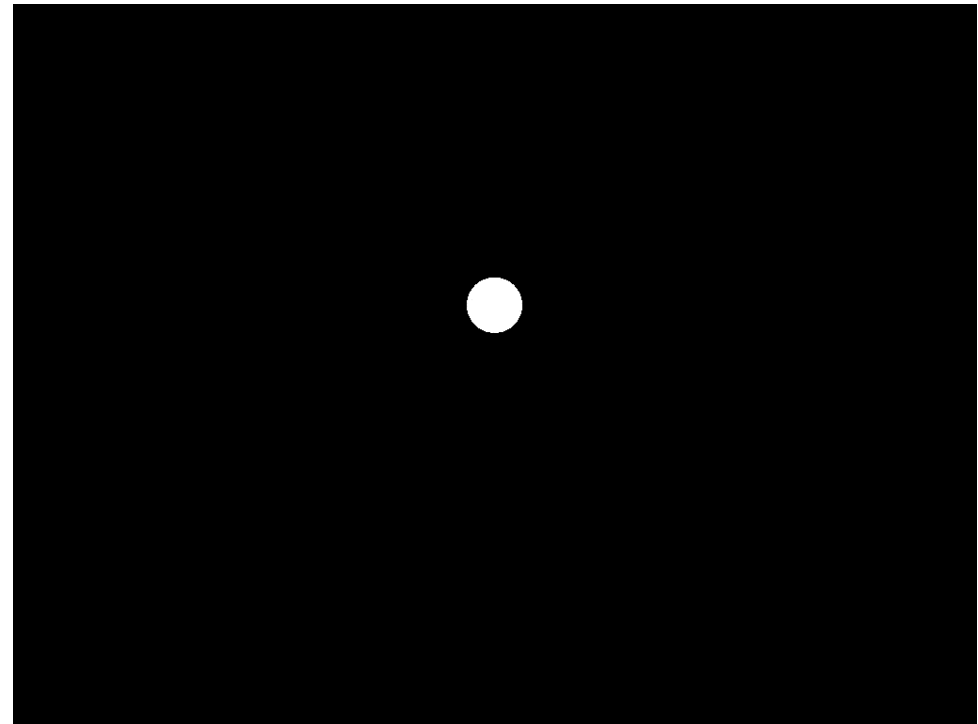
Project Overview

- The image on the left is an x-ray image of a medical phantom, i.e. a fake patient
 - The location of the 'tumor' is a known quantity
- The image on the right is a user generated label for the location of the tumor
 - Tumor location is labeled as '65535', all other locations are labeled '0'

Image

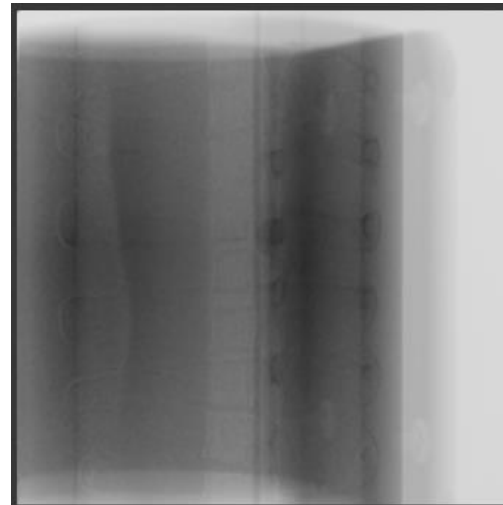


Label

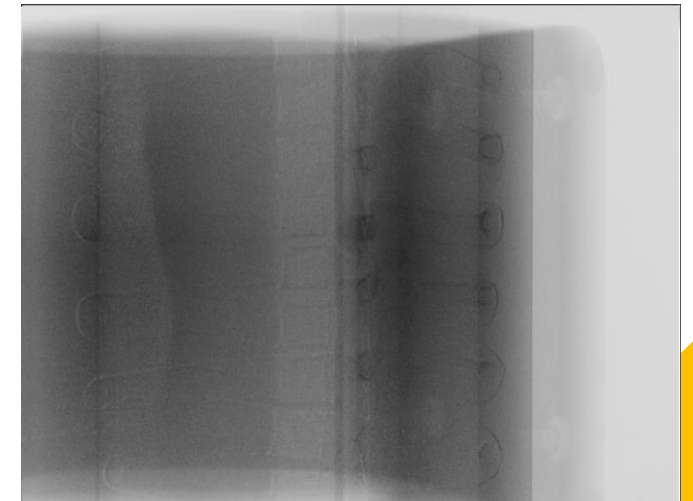


Preprocessing

- Image --> Array
 - Values ranged from 0 to 65,000
- Normalize values
 - Changed to range (0,1) where 1 is target value
- Reshape arrays
 - Maintain the original shape 768 x 1024
 - Initially reshaped to 96 x 96 for testing
 - Images were too distorted, wanted to use original size for real life application
 - Add channel for greyscale
 - 768 x 1024 x 1

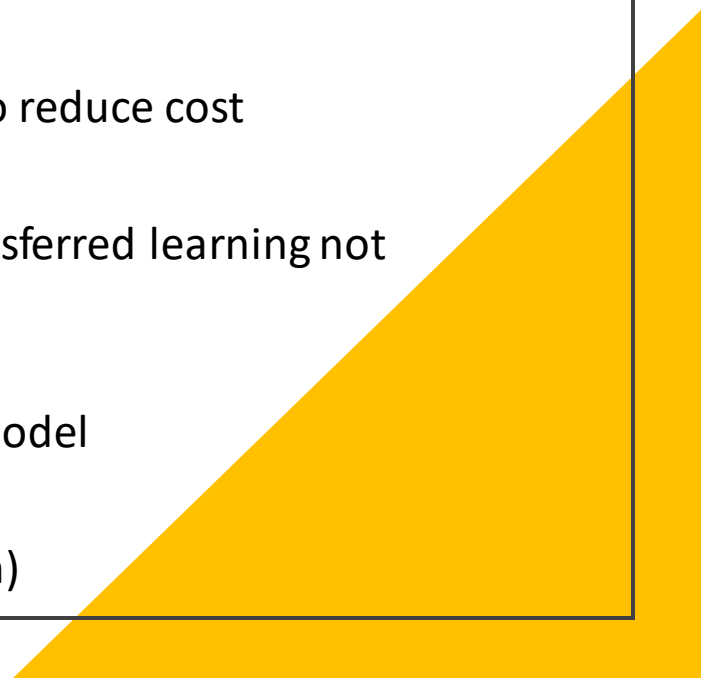


```
[39096 57415 57176 ... 56322 56257 57412]
[58469 57215 57090 ... 56146 56323 54047]
[57220 57076 57066 ... 56159 56254 54237]
...
[39279 43736 43337 ... 55189 54949 59038]
[40232 44800 44642 ... 55219 55019 59070]
[41689 45243 45485 ... 55252 55080 59105]]
```



```
X_train.shape, y_train.shape, X_test.shape, y_test.shape
((183, 768, 1024, 1),
 (183, 768, 1024, 1),
 (46, 768, 1024, 1),
 (46, 768, 1024, 1))
data[0].min(), data[0].max()
(0.0, 1.0)
```

Model Selection

- We want to identify or predict a single object
 - Need to label each pixel into two categories that are semantically interpretable aka real objects here tumor/non-tumor so our best tool is semantic segmentation model
 - Design a network of convolutional layers, but use down and up sampling to reduce cost
 - Example Semantic Segmentation was pretrained on colored images so transferred learning not useful
 - Built our own ResNet-18 as the backbone of the Semantic Segmentation model
 - Added learnable upsampling (aka convolution transpose or upconvolution)
- 
- A large yellow triangle is positioned in the bottom right corner of the slide, pointing towards the top right.

Model Overview

```
## going from 768 x 1024 x 1 image size, downsampling to values, upsampling to image
## blocks are downsampling

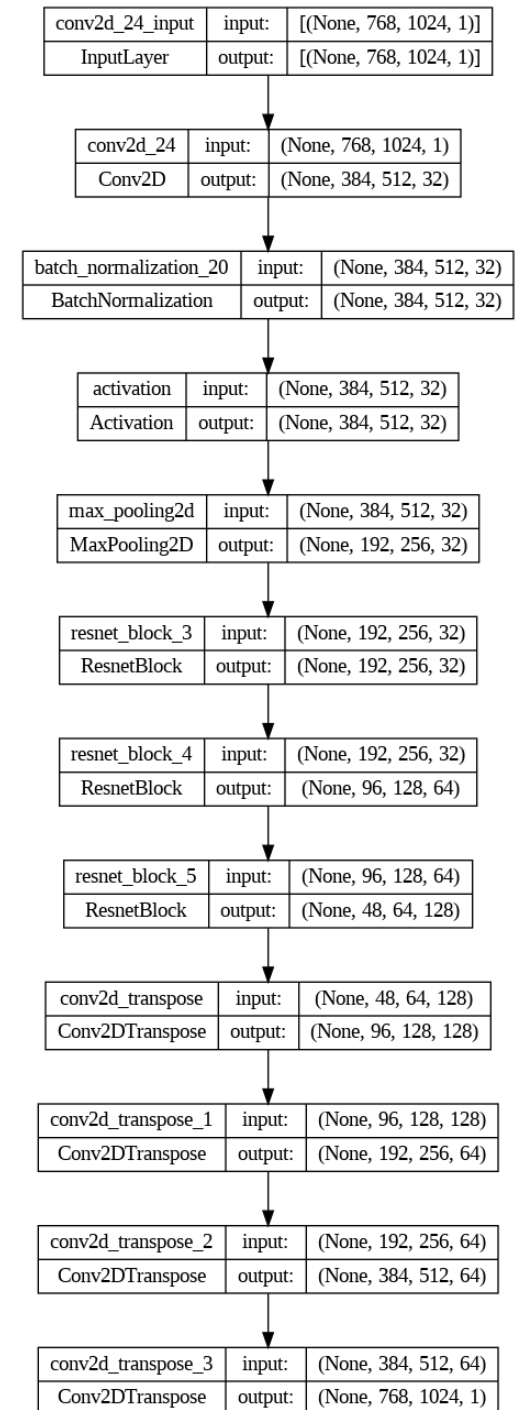
b2 = ResnetBlock(32, 2, first_block=True)
b3 = ResnetBlock(64, 2)
b4 = ResnetBlock(128, 1)

def net():
    model = tf.keras.Sequential([
        ## first 4 lines are block 1
        tf.keras.layers.Conv2D(32, kernel_size=7, strides=2, padding='same', input_shape=(768,1024,1)),
        tf.keras.layers.BatchNormalization(),
        tf.keras.layers.Activation('relu'),
        tf.keras.layers.MaxPool2D(pool_size=3, strides=2, padding='same'),
        b2,
        b3,
        b4, #downsampling
        # upsampling
        tf.keras.layers.Conv2DTranspose(128, kernel_size=3, strides=2, padding='same', activation='relu'),
        tf.keras.layers.Conv2DTranspose(64, kernel_size=3, strides=2, padding='same', activation='relu'),
        tf.keras.layers.Conv2DTranspose(64, kernel_size=3, strides=2, padding='same', activation='relu'),
        tf.keras.layers.Conv2DTranspose(1, kernel_size=3, strides=2, padding='same', activation='sigmoid'),
    ])

    optimizer=tf.optimizers.Adam(learning_rate=0.001)

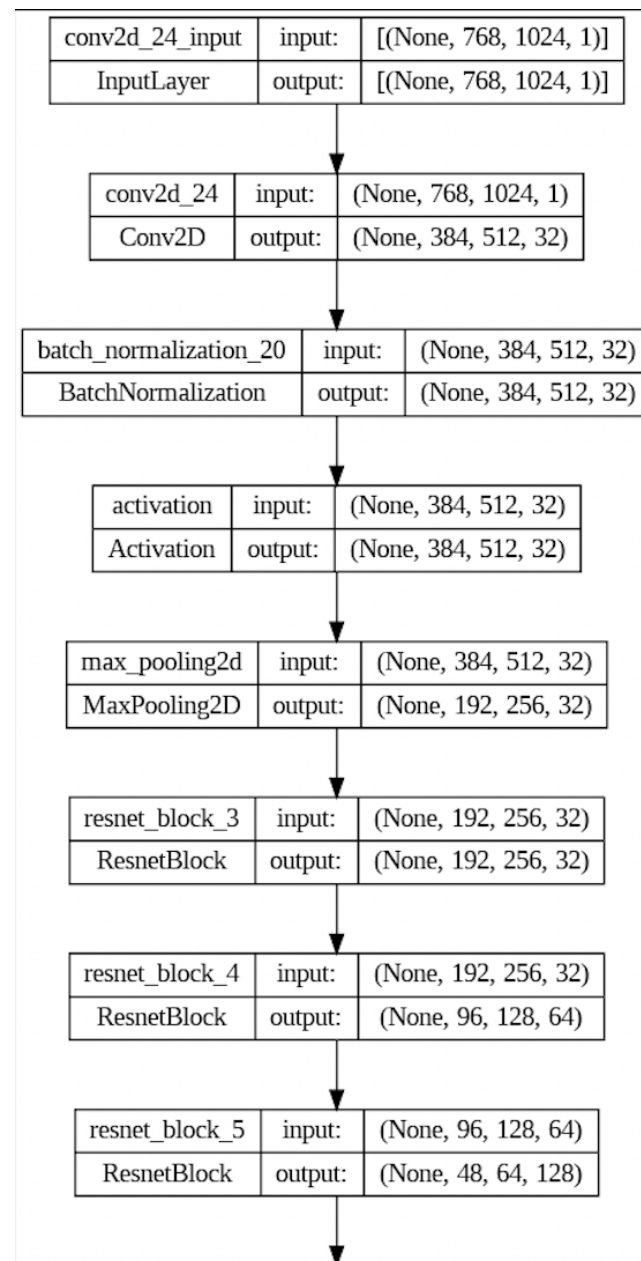
    model.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=tf.metrics.BinaryAccuracy(threshold=0.3))

    return model
```



Downsampling with ResNet backbone

- 1st "block"
 - Input shape
 - Convolutional2D
 - 7x7 kernel, 2 stride
 - Batch Normalization
 - Relu Activation
 - Max Pool
- 2, 3, 4 blocks
 - Residual Blocks
 - 2, 3 x 3 convolutional layers
 - Batch Normalization
 - Relu Activation




```
# upsampling
tf.keras.layers.Conv2DTranspose(128, kernel_size=3, strides=2, padding='same', activation='relu'),
tf.keras.layers.Conv2DTranspose(64, kernel_size=3, strides=2, padding='same', activation='relu'),
tf.keras.layers.Conv2DTranspose(64, kernel_size=3, strides=2, padding='same', activation='relu'),
tf.keras.layers.Conv2DTranspose(1, kernel_size=3, strides=2, padding='same', activation='sigmoid'),
])

optimizer=tf.optimizers.Adam(learning_rate=0.001)

model.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=tf.metrics.BinaryAccuracy(threshold=0.3))

return model
```

Upsampling Blocks & Design Choices

- Pick your hyperparameters: filter size, activation functions, optimizer, learning rate, loss, and threshold

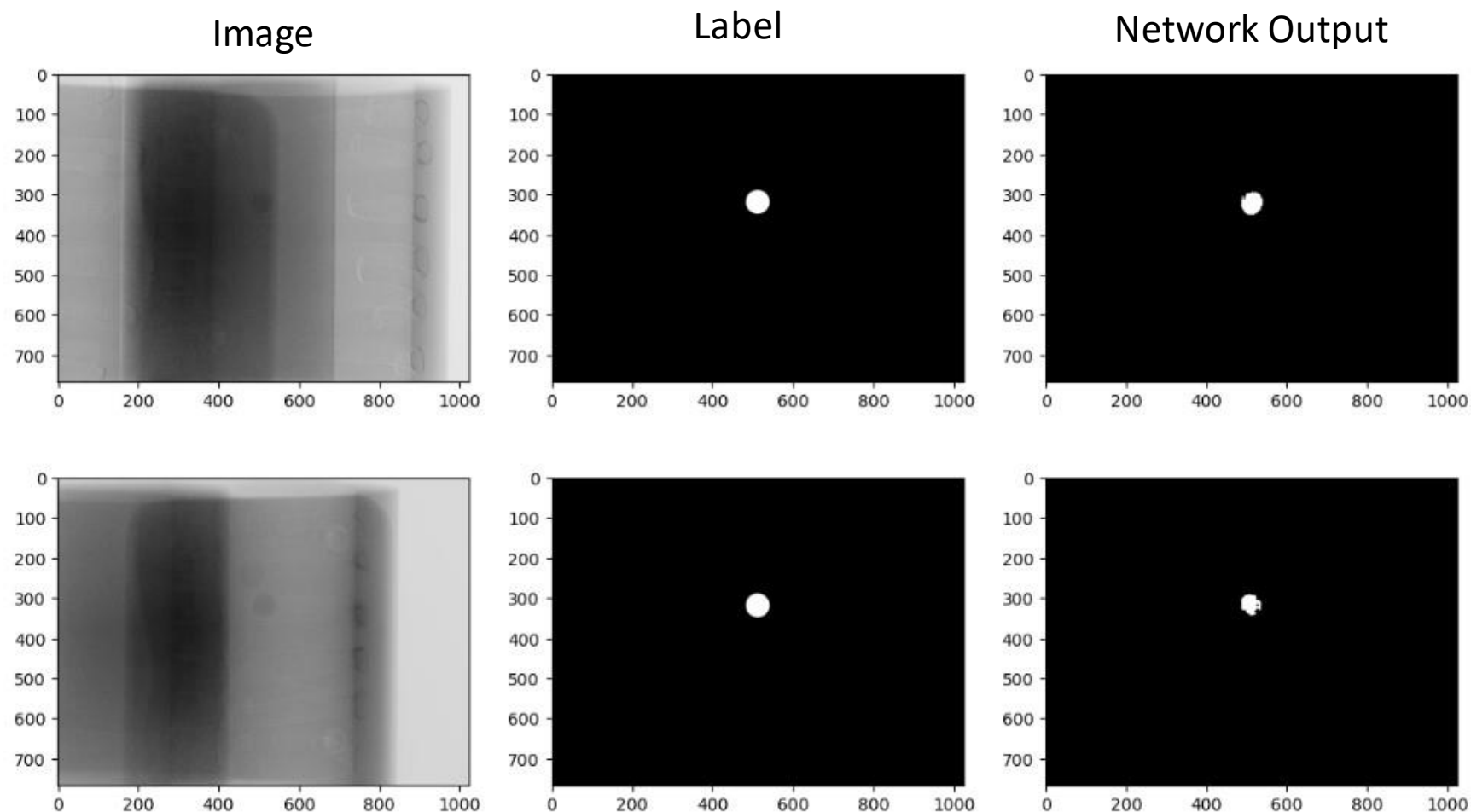
Checking the Model Shape

Model: Sequential

Layer (type)	Output Shape	Param #
conv2d_24 (Conv2D)	(None, 384, 512, 32)	1600
batch_normalization_20 (Batch Normalization)	(None, 384, 512, 32)	128
activation (Activation)	(None, 384, 512, 32)	0
max_pooling2d (MaxPooling2D)	(None, 192, 256, 32)	0
resnet_block_3 (ResnetBlock)	(None, 192, 256, 32)	37504
resnet_block_4 (ResnetBlock)	(None, 96, 128, 64)	132416
resnet_block_5 (ResnetBlock)	(None, 48, 64, 128)	230784
conv2d_transpose (Conv2DTranspose)	(None, 96, 128, 128)	147584
conv2d_transpose_1 (Conv2DTranspose)	(None, 192, 256, 64)	73792
conv2d_transpose_2 (Conv2DTranspose)	(None, 384, 512, 64)	36928
conv2d_transpose_3 (Conv2DTranspose)	(None, 768, 1024, 1)	577
Total params: 661,313		
Trainable params: 659,969		
Non-trainable params: 1,344		

Model Evaluation and Predictions

- Accuracy on training data: 0.9996
- Accuracy on test data: 0.9992
- Loss on test data: 0.0032



Epoch 50/50

22/22 [=====] - 2s 91ms/step - loss: 0.0027 - binary_accuracy: 0.9996 - val_loss: 0.0032 - val_binary_accuracy: 0.9992



Future Work

- Try adding noise with image augmentation to further prevent overfitting
 - Play with more design choices like learning rate, threshold
 - Test on real chest x-ray images and/or larger dataset
 - See if it predicts other images well like tumors on stomach x-rays
 - Would transfer learning from this model be useful for tumor detection on MRI scans if they are also grayscale?
-

Questions & Answers

Show us more of your
code please.

[Absolutely, let's click here!](#)