4 Final Project Submission

- name: Leticia D Drasler
- pace: Part time
- Scheduled project review data/time: January 28th, 2022, 1:00PM (Mountain Time)
- Course Instructor: Abhineet
- Blog post URL: [https://github.com/lddrasler/Pneumonia-Detection-Tensor-Flow-and-Keras (https://github.com/lddrasler/Pneumonia-Detection-Tensor-Flow-and-Keras)](https://github.com/lddrasler/Pneumonia-Detection-Tensor-Flow-and-Keras)
- GitHub repository: [https://wordpress.com/post/callableleticia.blog/101 (https://wordpress.com/post/callableleticia.blog/101)](https://wordpress.com/post/callableleticia.blog/101)

Importing Packages

In [1]:

```
import tensorflow as tf
import numpy as np
from tensorflow.keras.preprocessing import image_dataset_from_directory
import matplotlib.pyplot as plt
from tensorflow import keras
from tensorflow.keras import layers
```

Open the directories

In [2]:

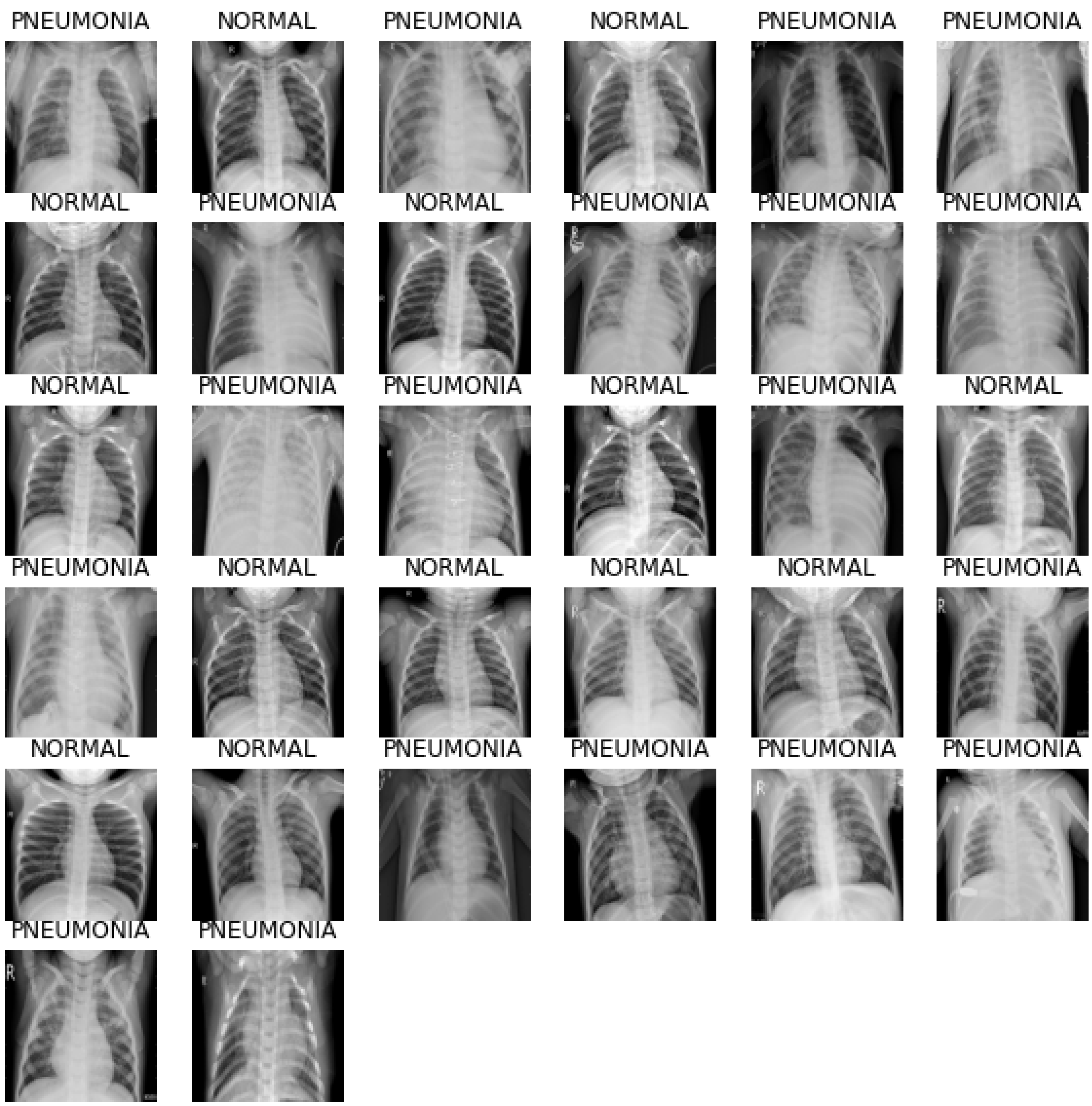
```
train_directory='chest_xray\\train'
val_directory='chest_xray\\val'
test_directory='chest_xray\\test'

# using image_dataset_from_directory
# to find out how many files and classes the dataset has

train_data=image_dataset_from_directory(train_directory,color_mode="grayscale")
val_data=image_dataset_from_directory(val_directory,color_mode="grayscale")
test_data=image_dataset_from_directory(test_directory,color_mode="grayscale")
```

```
Found 5216 files belonging to 2 classes.
Found 16 files belonging to 2 classes.
Found 624 files belonging to 2 classes.
```

Plotting random images from our train data file

In [3]:

```
plt.figure(figsize=(10, 10))
class_names = train_data.class_names
for images, labels in train_data.take(1):
    for i in range(len(images)):
        ax = plt.subplot(6, 6, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"),cmap='gray')
        plt.title(class_names[labels[i]])
        plt.axis("off")
```



In [ ]:

Baseline Fully Connected Neural Network

In [4]:

```
inputs = keras.Input(shape=(256, 256, 1))
x = layers.Rescaling(1./255)(inputs)
x = layers.Flatten()(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model_baseline = keras.Model(inputs=inputs, outputs=outputs)

model_baseline.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])
```

In [5]:

```python
callbacks_baseline = [
 keras.callbacks.ModelCheckpoint(
 filepath="baseline_model.keras",
 save_best_only=True,
 monitor="val_loss")
]

history_baseline = model_baseline.fit(
 train_data,
 epochs=15,
 validation_data=val_data,
 callbacks=callbacks_baseline)
```

```
Epoch 1/15
163/163 [==============================] - 13s 75ms/step - loss: 3.7488 - accuracy:
0.7427 - val_loss: 2.0293 - val_accuracy: 0.8125
Epoch 2/15
163/163 [==============================] - 13s 75ms/step - loss: 1.9834 - accuracy:
0.8317 - val_loss: 11.4065 - val_accuracy: 0.5000
Epoch 3/15
163/163 [==============================] - 15s 86ms/step - loss: 1.5900 - accuracy:
0.8637 - val_loss: 1.6170 - val_accuracy: 0.7500
Epoch 4/15
163/163 [==============================] - 18s 104ms/step - loss: 1.5290 - accuracy:
0.8750 - val_loss: 4.5848 - val_accuracy: 0.6875
Epoch 5/15
163/163 [==============================] - 17s 101ms/step - loss: 1.2871 - accuracy:
0.8905 - val_loss: 0.3674 - val_accuracy: 0.8125
Epoch 6/15
163/163 [==============================] - 18s 105ms/step - loss: 1.2331 - accuracy:
0.9018 - val_loss: 0.1379 - val_accuracy: 0.8750
Epoch 7/15
163/163 [==============================] - 17s 101ms/step - loss: 1.1809 - accuracy:
0.9041 - val_loss: 1.0325 - val_accuracy: 0.9375
Epoch 8/15
163/163 [==============================] - 18s 103ms/step - loss: 1.1693 - accuracy:
0.9095 - val_loss: 9.1917 - val_accuracy: 0.5625
Epoch 9/15
163/163 [==============================] - 17s 102ms/step - loss: 1.1143 - accuracy:
0.9137 - val_loss: 9.8717 - val_accuracy: 0.5625
Epoch 10/15
163/163 [==============================] - 17s 102ms/step - loss: 1.0918 - accuracy:
0.9160 - val_loss: 1.2066 - val_accuracy: 0.9375
Epoch 11/15
163/163 [==============================] - 18s 104ms/step - loss: 1.0175 - accuracy:
0.9208 - val_loss: 0.8982 - val_accuracy: 0.9375
Epoch 12/15
163/163 [==============================] - 17s 101ms/step - loss: 0.9975 - accuracy:
0.9158 - val_loss: 0.8909 - val_accuracy: 0.8125
Epoch 13/15
163/163 [==============================] - 18s 106ms/step - loss: 1.0235 - accuracy:
0.9204 - val_loss: 0.9798 - val_accuracy: 0.9375
Epoch 14/15
163/163 [==============================] - 16s 97ms/step - loss: 1.0374 - accuracy:
0.9241 - val_loss: 1.1385 - val_accuracy: 0.9375
Epoch 15/15
163/163 [==============================] - 17s 99ms/step - loss: 0.9668 - accuracy:
0.9314 - val_loss: 12.6372 - val_accuracy: 0.5625
```

In [7]:
```
test_loss, test_acc = model_baseline.evaluate(test_data)
print(f"Test accuracy: {test_acc:.3f}")
```

```
20/20 [==============================] - 1s 41ms/step - loss: 14.1111 - accuracy: 0.6
426
Test accuracy: 0.643
```

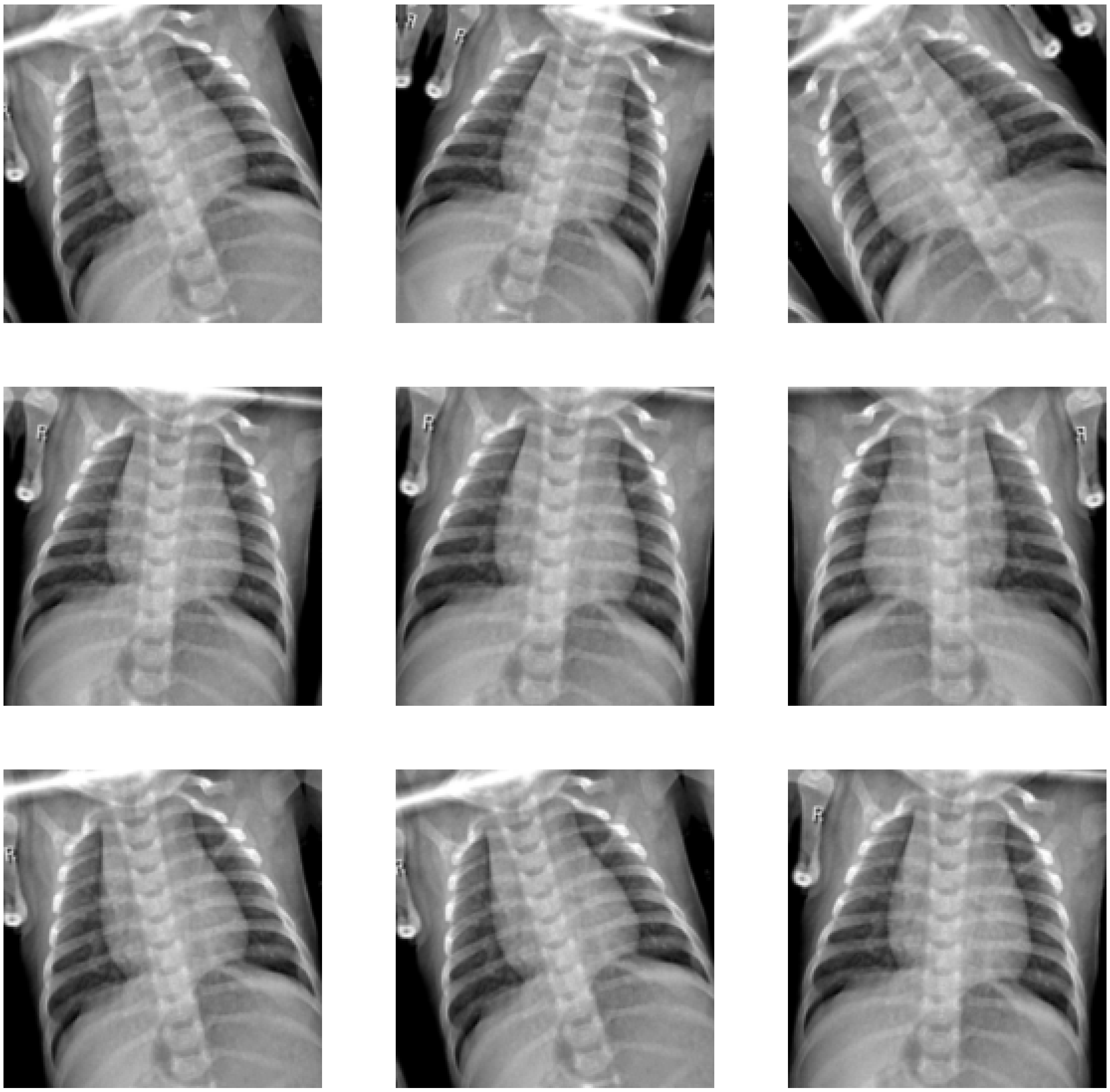
The simple Neural Network does not have a great performance because

Convolutional Neural Network

In [8]:
```
# adding data augmentation to help the model not to overfit

data_augmentation = keras.Sequential(
 [
     layers.RandomFlip("horizontal"),
     layers.RandomRotation(0.1),
     layers.RandomZoom(0.1),
 ]
)
```

In [9]:

```
plt.figure(figsize=(10, 10))
for images, _ in train_data.take(1):
    for i in range(9):
        augmented_images = data_augmentation(images)
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(augmented_images[0].numpy().astype("uint8"),cmap='gray')
        plt.axis("off")
```



In [10]:

```
inputs = keras.Input(shape=(256, 256, 1))

# adding data_augmentation to prevent overfitting
x = data_augmentation(inputs)

# first convolutional layer
x = layers.Rescaling(1./255)(x)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)

# second convolutional layer
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)

# third convolutional layer
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)

# fourth convolutional layer
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)

# fifth convolutional layer
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)

# flatten
x = layers.Flatten()(x)

# dropout to prevent overfitting

x = layers.Dropout(0.5)(x)
x = layers.Dense(512, activation="relu")(x)

outputs = layers.Dense(1, activation='sigmoid')(x)

model_3 = keras.Model(inputs=inputs, outputs=outputs)
model_3.compile(loss="binary_crossentropy",
                optimizer="rmsprop",
                metrics=["accuracy"])
```

In [11]:

```
callbacks_3 = [
 keras.callbacks.ModelCheckpoint(
     filepath='x_ray_covn_model_NN.{epoch:02d}.hdf5',
     save_best_only=False,
     monitor="val_loss")
]
```

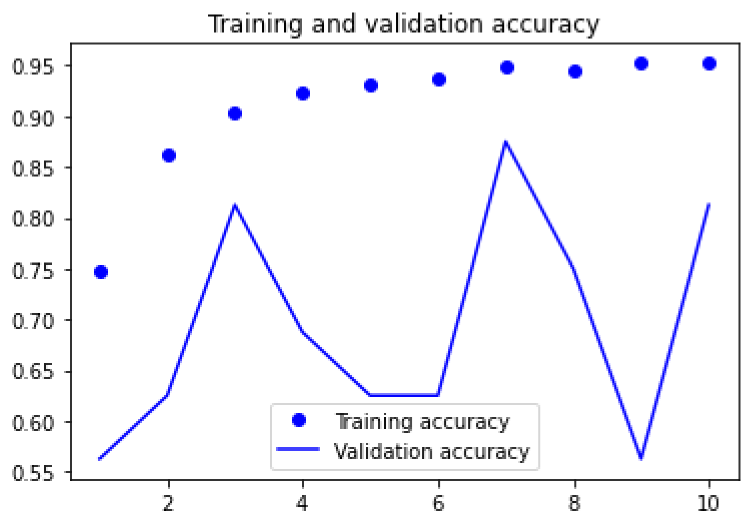
In [12]:

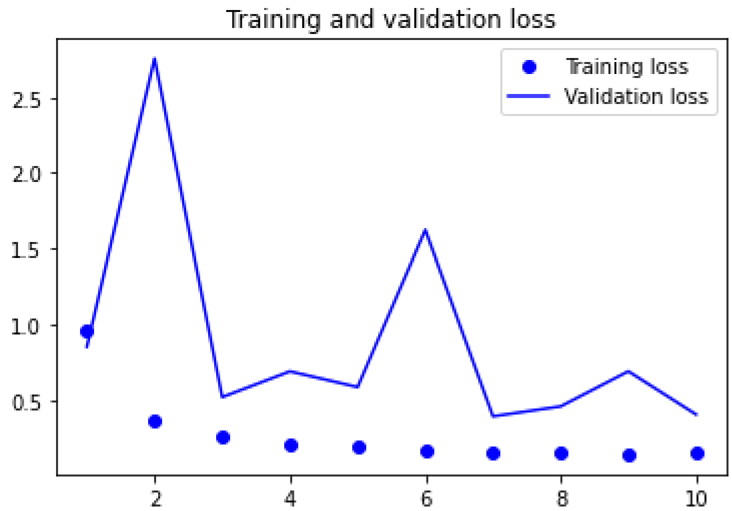
```
history_3 = model_3.fit(
 train_data,
 epochs=10,
 validation_data=val_data,
 callbacks=callbacks_3)
```

```
Epoch 1/10
163/163 [==============================] - 462s 3s/step - loss: 0.9677 - accuracy: 0.
7479 - val_loss: 0.8539 - val_accuracy: 0.5625
Epoch 2/10
163/163 [==============================] - 434s 3s/step - loss: 0.3636 - accuracy: 0.
8629 - val_loss: 2.7568 - val_accuracy: 0.6250
Epoch 3/10
163/163 [==============================] - 435s 3s/step - loss: 0.2650 - accuracy: 0.
9036 - val_loss: 0.5207 - val_accuracy: 0.8125
Epoch 4/10
163/163 [==============================] - 431s 3s/step - loss: 0.2111 - accuracy: 0.
9237 - val_loss: 0.6908 - val_accuracy: 0.6875
Epoch 5/10
163/163 [==============================] - 423s 3s/step - loss: 0.1896 - accuracy: 0.
9316 - val_loss: 0.5871 - val_accuracy: 0.6250
Epoch 6/10
163/163 [==============================] - 419s 3s/step - loss: 0.1688 - accuracy: 0.
9377 - val_loss: 1.6239 - val_accuracy: 0.6250
Epoch 7/10
163/163 [==============================] - 426s 3s/step - loss: 0.1478 - accuracy: 0.
9494 - val_loss: 0.3934 - val_accuracy: 0.8750
Epoch 8/10
163/163 [==============================] - 448s 3s/step - loss: 0.1542 - accuracy: 0.
9448 - val_loss: 0.4593 - val_accuracy: 0.7500
Epoch 9/10
163/163 [==============================] - 441s 3s/step - loss: 0.1421 - accuracy: 0.
9525 - val_loss: 0.6913 - val_accuracy: 0.5625
Epoch 10/10
163/163 [==============================] - 439s 3s/step - loss: 0.1499 - accuracy: 0.
9523 - val_loss: 0.4059 - val_accuracy: 0.8125
```

In [13]:

```
accuracy = history_3.history["accuracy"]
val_accuracy = history_3.history["val_accuracy"]
loss = history_3.history["loss"]
val_loss = history_3.history["val_loss"]
epochs = range(1, len(accuracy) + 1)

plt.plot(epochs, accuracy, "bo", label="Training accuracy")
plt.plot(epochs, val_accuracy, "b", label="Validation accuracy")
plt.title("Training and validation accuracy")
# plt.xlim([1, 25])
plt.legend()
plt.figure()
plt.plot(epochs, loss, "bo", label="Training loss")
plt.plot(epochs, val_loss, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.legend()
# plt.xlim([1, 25])

plt.show()
```





In [17]:

```
test_model_03 = keras.models.load_model("x_ray_covn_model_NN.24.hdf5")
test_loss, test_acc = test_model_03.evaluate(test_data)
print(f"Test accuracy: {test_acc:.3f}")
```

```
20/20 [==============================] - 21s 943ms/step - loss: 0.2494 - accuracy: 0.
9071
Test accuracy: 0.907
```

In [18]:

```
test_model_03.summary()
```

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_2 (InputLayer)        [(None, 256, 256, 1)]     0         
                                                                 
 sequential (Sequential)     (None, 256, 256, 1)       0         
                                                                 
 rescaling_1 (Rescaling)     (None, 256, 256, 1)       0         
                                                                 
 conv2d (Conv2D)             (None, 254, 254, 32)      320       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 127, 127, 32)     0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 125, 125, 64)      18496     
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 62, 62, 64)       0         
 2D)                                                             
                                                                 
 conv2d_2 (Conv2D)           (None, 60, 60, 128)       73856     
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 30, 30, 128)      0         
 2D)                                                             
                                                                 
 conv2d_3 (Conv2D)           (None, 28, 28, 256)       295168    
                                                                 
 max_pooling2d_3 (MaxPooling  (None, 14, 14, 256)      0         
 2D)                                                             
                                                                 
 conv2d_4 (Conv2D)           (None, 12, 12, 256)       590080    
                                                                 
 flatten_1 (Flatten)         (None, 36864)             0         
                                                                 
 dropout (Dropout)           (None, 36864)             0         
                                                                 
 dense_1 (Dense)             (None, 512)               18874880  
                                                                 
 dense_2 (Dense)             (None, 1)                 513       
                                                                 
=================================================================
Total params: 19,853,313
Trainable params: 19,853,313
Non-trainable params: 0
_________________________________________________________________
```

LIME PACKAGE

In [19]:

```
from lime import lime_image

explainer = lime_image.LimeImageExplainer()
```

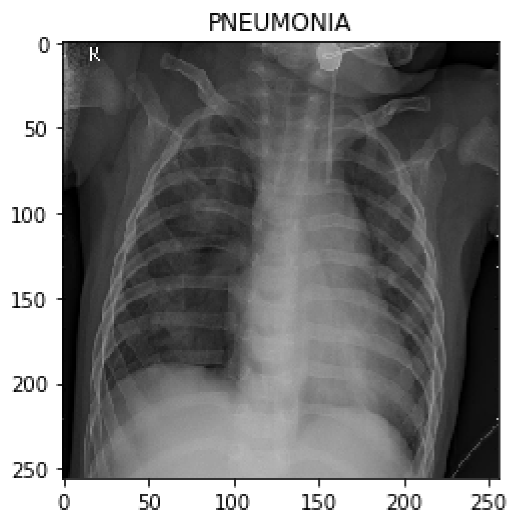
In [20]:

```python
data=image_dataset_from_directory(train_directory)
```

```
Found 5216 files belonging to 2 classes.
```

In [21]:

```python
# from skimage.color import gray2rgb
data =list(train_data.take(1))
image = data[0][0][4].numpy().astype(np.uint8)
label=data[0][1][4].numpy()
class_names = train_data.class_names

from matplotlib import pyplot as plt
plt.imshow(image.reshape(256,256), interpolation='nearest',cmap='gray')
plt.title(class_names[label])
# plt.imshow(image_color, interpolation='nearest')
plt.show()
```



In [22]:

```python
def reshape_img(image):
    return image.reshape(256,256)

def new_predict_fn(images):
    images=images[:,:,:,:1]
    return test_model_03.predict(images)

explanation = explainer.explain_instance(image.reshape(256,256),
                                         new_predict_fn,
                                         top_labels=2, hide_color=0,
                                         num_samples=1000)
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

In [23]:

```
from skimage.segmentation import mark_boundaries

temp_1, mask_1 = explanation.get_image_and_mask(explanation.top_labels[0], positive_onl
                                                num_features=5, hide_rest=True)
temp_2, mask_2 = explanation.get_image_and_mask(explanation.top_labels[0], positive_onl
                                                num_features=10, hide_rest=False)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15,15))
ax1.imshow(mark_boundaries(temp_1.astype('uint8'), mask_1))
ax2.imshow(mark_boundaries(temp_2.astype('uint8'), mask_2))
ax1.axis('off')
ax2.axis('off')
```

Out[23]: (-0.5, 255.5, 255.5, -0.5)