In [1]:

```
import os
import tensorflow as tf
from tensorflow.keras import Model
from tensorflow.keras import layers
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.preprocessing.image import ImageDataGenerator
#from tensorflow.keras.applications.inception_v3 import InceptionV3
```

model = tf.keras.models.Sequential([tf.keras.layers.Conv2D(16, (3,3), activation='relu', input_shape=(200,200,3)), tf.keras.layers.MaxPooling2D(2,2), tf.keras.layers.Conv2D(32,(3,3), activation='relu'), tf.keras.layers.MaxPooling2D(2,2), tf.keras.layers.Conv2D(32,(3,3), activation='relu'), tf.keras.layers.MaxPooling2D(2,2), tf.keras.layers.Dropout(0.5), tf.keras.layers.Flatten(), tf.keras.layers.Dense(1024, activation='relu'), tf.keras.layers.Dense(2, activation='softmax')])

model.compile(optimizer = 'Adam', loss = 'sparse_categorical_crossentropy', metrics = ['accuracy'])

In [2]:

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16, (3,3), activation='relu', input_shape=(254,254,3)),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(32,(3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(32,(3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(1024, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Dense(1, activation='sigmoid')])

model.compile(optimizer = SGD(lr=0.01, nesterov=True),
              loss = 'binary_crossentropy',
              metrics = ['accuracy'])

#model.summary()
```

In [3]:

```
base_dir = 'data/chest_xray'
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'test')

train_datagen = ImageDataGenerator(rescale = 1./255.,
                                   rotation_range = 40,
                                   width_shift_range = 0.2,
                                   height_shift_range = 0.2,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(train_dir,
                                                    target_size=(254,254),
                                                    batch_size=64,
                                                    class_mode='binary')
validation_generator = test_datagen.flow_from_directory(validation_dir,
                                                        target_size=(254,254),
                                                        batch_size=64,
                                                        class_mode='binary')
```

```
Found 5216 images belonging to 2 classes.
Found 624 images belonging to 2 classes.
```

In [4]:

```
history = model.fit(
            train_generator,
            validation_data = validation_generator,
            steps_per_epoch = 80,
            epochs = 10,
            validation_steps = 9,
            verbose = 1)
```

```
Epoch 1/10
80/80 [==============================] - 190s 2s/step - loss: 0.5653 - accur
acy: 0.7386 - val_loss: 0.7318 - val_accuracy: 0.6233
Epoch 2/10
80/80 [==============================] - 213s 3s/step - loss: 0.5454 - accur
acy: 0.7453 - val_loss: 0.6732 - val_accuracy: 0.6215
Epoch 3/10
80/80 [==============================] - 205s 3s/step - loss: 0.5393 - accur
acy: 0.7461 - val_loss: 0.6557 - val_accuracy: 0.6267
Epoch 4/10
80/80 [==============================] - 200s 3s/step - loss: 0.5242 - accur
acy: 0.7522 - val_loss: 0.7009 - val_accuracy: 0.6267
Epoch 5/10
80/80 [==============================] - 194s 2s/step - loss: 0.5218 - accur
acy: 0.7488 - val_loss: 0.7752 - val_accuracy: 0.6181
Epoch 6/10
80/80 [==============================] - 192s 2s/step - loss: 0.5157 - accur
acy: 0.7551 - val_loss: 0.8134 - val_accuracy: 0.6163
Epoch 7/10
80/80 [==============================] - 200s 2s/step - loss: 0.4903 - accur
acy: 0.7754 - val_loss: 0.6604 - val_accuracy: 0.6441
Epoch 8/10
80/80 [==============================] - 205s 3s/step - loss: 0.4902 - accur
acy: 0.7710 - val_loss: 0.5802 - val_accuracy: 0.6840
Epoch 9/10
80/80 [==============================] - 211s 3s/step - loss: 0.4852 - accur
acy: 0.7726 - val_loss: 0.7223 - val_accuracy: 0.6285
Epoch 10/10
80/80 [==============================] - 190s 2s/step - loss: 0.4689 - accur
acy: 0.7946 - val_loss: 0.6468 - val_accuracy: 0.6406
```

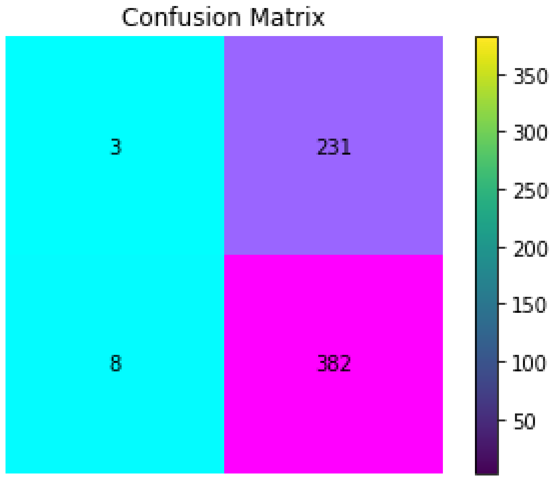
Model Accuracy

In [5]:

```
import numpy as np
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
```

In [6]:

```
pred = model.predict(validation_generator)
pred = np.round(pred)
matrix = confusion_matrix(validation_generator.classes, pred)
fig, ax = plt.subplots()
plt.title("Confusion Matrix")
plt.colorbar(plt.imshow(matrix))
ax.imshow(matrix, cmap = "cool")
ax.axis('off')
for i in range(2):
    for j in range(2):
        text = ax.text(j, i, matrix[i, j], ha="center", va="center", color="black")
```



In [7]:

```
TN = matrix[0][0]
TP = matrix[1][1]
FN = matrix[1][0]
FP = matrix[0][1]
print("True Negative: {}\t True Positive: {}".format(TN,TP))
print("False Negative: {}\t False Positive: {}\n".format(FN,FP))
print("False Positive Rate: {:.4f}".format(FP/(FP+TN)))
print("False Negative Rate: {:.4f}\n".format(FN/(FN+TP)))
print("Test Accuracy: {:.5f}".format((TP+TN)/(TP+TN+FN+FP)))
```

```
True Negative: 3         True Positive: 382
False Negative: 8        False Positive: 231

False Positive Rate: 0.9872
False Negative Rate: 0.0205

Test Accuracy: 0.61699
```

Let's get interactive

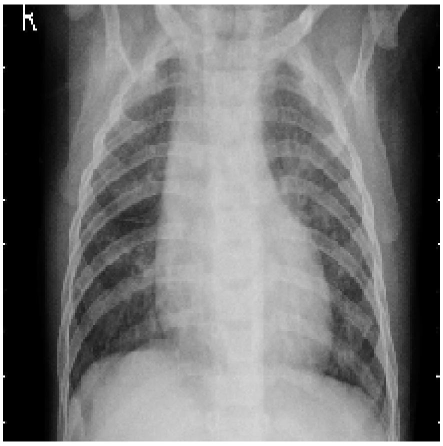
In [8]:

```
prediction_generator = test_datagen.flow_from_directory('data',                     # Conne
                                                        target_size=(254,254),      # Sets
                                                        batch_size=1,               # Sets
                                                        classes=['prediction'],     # Impor
                                                        class_mode='binary')        # Sets

plt.imshow(prediction_generator[0][0].squeeze())                                    # Plot
plt.axis("off")                                                                     # Disab
pneumonia_prob = model.predict(prediction_generator).squeeze()                      # Predi
normal_prob = 1-pneumonia_prob                                                      # Calcu

print("Probability of Pneumonia: {:.3f}% \nProbability of normal: {:.3f}%".format(pneumonia
```

```
Found 1 images belonging to 1 classes.
Probability of Pneumonia: 72.444%
Probability of normal: 27.556%
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: