In [1]:
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import Adam, SGD
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.regularizers import l1_l2
from tensorflow.keras import models, layers
import os
from keras.models import load_model
```

In [2]:
```
import tensorflow_datasets as tfds
(ds_train, ds_test),ds_info=tfds.load('cats_vs_dogs',
                                      split=['train[:10%]', 'train[98%:]'],
                                      shuffle_files=True,
                                      as_supervised=True,
                                      with_info=True)
```

In [3]:
```
image_size = (256, 256)
```

In [4]:
```
#ds_train=tf.keras.utils.image_dataset_from_directory("ProjectDir_Pet/train",batch_
```

In [5]:
```
#ds_test=tf.keras.utils.image_dataset_from_directory("ProjectDir_Pet/test",batch_s
```

In [6]:
```
def normalize_img(image, label):
    # Resize the image to the desired dimensions
    image = tf.image.resize(image, image_size)
    # Normalize images: uint8 -> float32
    image = tf.cast(image, tf.float32) / 255.0
    return image, label

ds_train = ds_train.map(normalize_img, num_parallel_calls=tf.data.AUTOTUNE)
ds_train = ds_train.cache()
ds_train = ds_train.shuffle(40)
ds_train = ds_train.batch(128)
ds_train = ds_train.prefetch(tf.data.AUTOTUNE)
```

In [7]:
```
ds_test = ds_test.map(normalize_img, num_parallel_calls=tf.data.AUTOTUNE)
ds_test = ds_test.batch(128)
ds_test = ds_test.cache()
ds_test = ds_test.prefetch(tf.data.AUTOTUNE)
```

In [8]:
```
data_augmentation = tf.keras.Sequential([
  layers.RandomFlip("horizontal_and_vertical"),
  layers.RandomRotation(0.2),
])
```

In [9]:
```
model = models.Sequential()
model.add(layers.Input(shape=(256, 256, 3)))
model.add(data_augmentation)
model.add( layers.Conv2D( 32, (5, 5 ), activation = 'tanh',kernel_initializer='glo
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D(2,2))
model.add(layers.Conv2D(64, (5, 5 ),kernel_initializer='glorot_uniform', activatio
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Dropout(0.5))
model.add(layers.Flatten())
model.add(layers.Dense(32,kernel_initializer='he_normal',activation = 'relu'))
model.add(layers.Dropout(0.5))
```

```
model.add(layers.BatchNormalization())
model.add(layers.Dense(1,kernel_initializer='he_normal', activation = 'sigmoid'))
```

In [10]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 sequential (Sequential)     (None, 256, 256, 3)       0         
                                                                 
 conv2d (Conv2D)             (None, 252, 252, 32)      2432      
                                                                 
 batch_normalization (Batch  (None, 252, 252, 32)      128       
 Normalization)                                                  
                                                                 
 max_pooling2d (MaxPooling2  (None, 126, 126, 32)      0         
 D)                                                              
                                                                 
 conv2d_1 (Conv2D)           (None, 122, 122, 64)      51264     
                                                                 
 batch_normalization_1 (Bat  (None, 122, 122, 64)      256       
 chNormalization)                                                
                                                                 
 max_pooling2d_1 (MaxPoolin  (None, 61, 61, 64)        0         
 g2D)                                                            
                                                                 
 dropout (Dropout)           (None, 61, 61, 64)        0         
                                                                 
 flatten (Flatten)           (None, 238144)            0         
                                                                 
 dense (Dense)               (None, 32)                7620640   
                                                                 
 dropout_1 (Dropout)         (None, 32)                0         
                                                                 
 batch_normalization_2 (Bat  (None, 32)                128       
 chNormalization)                                                
                                                                 
 dense_1 (Dense)             (None, 1)                 33        
                                                                 
=================================================================
Total params: 7674881 (29.28 MB)
Trainable params: 7674625 (29.28 MB)
Non-trainable params: 256 (1.00 KB)
_________________________________________________________________
```

In [11]:
```
history = model.compile(
    optimizer = tf.keras.optimizers.Adam(0.01),
    #loss='binary_crossentropy',
    #metrics=['accuracy']
    loss = tf.keras.losses.BinaryCrossentropy(from_logits=False),
    metrics=[tf.keras.metrics.BinaryAccuracy()]
)
```

In [12]:
```
history = model.fit(
    ds_train,
    epochs=50,
    validation_data=ds_test,
)
print('Number of total epochs ran:')
len(history.history['val_binary_accuracy'])
```

```
Epoch 1/50
19/19 [==============================] - 250s 13s/step - loss: 0.8349 - binary_acc
uracy: 0.5284 - val_loss: 1.0589 - val_binary_accuracy: 0.4903
Epoch 2/50
19/19 [==============================] - 245s 12s/step - loss: 0.7188 - binary_acc
uracy: 0.5353 - val_loss: 1.0581 - val_binary_accuracy: 0.4817
Epoch 3/50
19/19 [==============================] - 253s 13s/step - loss: 0.6956 - binary_acc
uracy: 0.5331 - val_loss: 0.7093 - val_binary_accuracy: 0.5097
Epoch 4/50
19/19 [==============================] - 248s 13s/step - loss: 0.6762 - binary_acc
uracy: 0.5731 - val_loss: 0.6974 - val_binary_accuracy: 0.5226
Epoch 5/50
19/19 [==============================] - 245s 13s/step - loss: 0.6685 - binary_acc
uracy: 0.5860 - val_loss: 0.7032 - val_binary_accuracy: 0.5075
Epoch 6/50
19/19 [==============================] - 240s 13s/step - loss: 0.6713 - binary_acc
uracy: 0.5709 - val_loss: 0.7270 - val_binary_accuracy: 0.5075
Epoch 7/50
19/19 [==============================] - 242s 13s/step - loss: 0.6728 - binary_acc
uracy: 0.5752 - val_loss: 0.6963 - val_binary_accuracy: 0.5226
Epoch 8/50
19/19 [==============================] - 246s 13s/step - loss: 0.6664 - binary_acc
uracy: 0.5847 - val_loss: 0.9562 - val_binary_accuracy: 0.5118
Epoch 9/50
19/19 [==============================] - 239s 12s/step - loss: 0.6673 - binary_acc
uracy: 0.5929 - val_loss: 1.4730 - val_binary_accuracy: 0.5161
Epoch 10/50
19/19 [==============================] - 239s 12s/step - loss: 0.6726 - binary_acc
uracy: 0.5761 - val_loss: 0.6976 - val_binary_accuracy: 0.5312
Epoch 11/50
19/19 [==============================] - 235s 12s/step - loss: 0.6577 - binary_acc
uracy: 0.6028 - val_loss: 0.6651 - val_binary_accuracy: 0.5677
Epoch 12/50
19/19 [==============================] - 236s 12s/step - loss: 0.6583 - binary_acc
uracy: 0.6062 - val_loss: 0.6636 - val_binary_accuracy: 0.5742
Epoch 13/50
19/19 [==============================] - 228s 12s/step - loss: 0.6483 - binary_acc
uracy: 0.6148 - val_loss: 0.7301 - val_binary_accuracy: 0.5183
Epoch 14/50
19/19 [==============================] - 229s 12s/step - loss: 0.6423 - binary_acc
uracy: 0.6281 - val_loss: 0.7350 - val_binary_accuracy: 0.5247
Epoch 15/50
19/19 [==============================] - 232s 12s/step - loss: 0.6383 - binary_acc
uracy: 0.6303 - val_loss: 0.6696 - val_binary_accuracy: 0.5806
Epoch 16/50
19/19 [==============================] - 228s 12s/step - loss: 0.6316 - binary_acc
uracy: 0.6337 - val_loss: 0.7402 - val_binary_accuracy: 0.5247
Epoch 17/50
19/19 [==============================] - 226s 12s/step - loss: 0.6360 - binary_acc
uracy: 0.6307 - val_loss: 0.8598 - val_binary_accuracy: 0.5204
Epoch 18/50
19/19 [==============================] - 245s 12s/step - loss: 0.6267 - binary_acc
uracy: 0.6419 - val_loss: 0.9740 - val_binary_accuracy: 0.5183
Epoch 19/50
19/19 [==============================] - 235s 12s/step - loss: 0.6232 - binary_acc
uracy: 0.6470 - val_loss: 1.3085 - val_binary_accuracy: 0.5118
Epoch 20/50
19/19 [==============================] - 228s 12s/step - loss: 0.6329 - binary_acc
uracy: 0.6316 - val_loss: 0.8805 - val_binary_accuracy: 0.5441
Epoch 21/50
19/19 [==============================] - 225s 12s/step - loss: 0.6194 - binary_acc
uracy: 0.6599 - val_loss: 0.7905 - val_binary_accuracy: 0.5419
Epoch 22/50
```

```
19/19 [==============================] - 227s 12s/step - loss: 0.5935 - binary_acc
uracy: 0.6758 - val_loss: 0.7130 - val_binary_accuracy: 0.5720
Epoch 23/50
19/19 [==============================] - 226s 12s/step - loss: 0.6057 - binary_acc
uracy: 0.6780 - val_loss: 0.6037 - val_binary_accuracy: 0.6839
Epoch 24/50
19/19 [==============================] - 230s 12s/step - loss: 0.5932 - binary_acc
uracy: 0.6879 - val_loss: 0.6067 - val_binary_accuracy: 0.6172
Epoch 25/50
19/19 [==============================] - 228s 12s/step - loss: 0.5880 - binary_acc
uracy: 0.6862 - val_loss: 0.5989 - val_binary_accuracy: 0.6473
Epoch 26/50
19/19 [==============================] - 229s 12s/step - loss: 0.5863 - binary_acc
uracy: 0.6801 - val_loss: 1.1014 - val_binary_accuracy: 0.5505
Epoch 27/50
19/19 [==============================] - 229s 12s/step - loss: 0.5773 - binary_acc
uracy: 0.6883 - val_loss: 0.8039 - val_binary_accuracy: 0.5806
Epoch 28/50
19/19 [==============================] - 225s 12s/step - loss: 0.5739 - binary_acc
uracy: 0.6952 - val_loss: 0.5916 - val_binary_accuracy: 0.6667
Epoch 29/50
19/19 [==============================] - 228s 12s/step - loss: 0.5701 - binary_acc
uracy: 0.7034 - val_loss: 0.6450 - val_binary_accuracy: 0.6172
Epoch 30/50
19/19 [==============================] - 227s 12s/step - loss: 0.5710 - binary_acc
uracy: 0.7068 - val_loss: 0.6807 - val_binary_accuracy: 0.6108
Epoch 31/50
19/19 [==============================] - 228s 12s/step - loss: 0.6431 - binary_acc
uracy: 0.6260 - val_loss: 0.8155 - val_binary_accuracy: 0.5183
Epoch 32/50
19/19 [==============================] - 222s 12s/step - loss: 0.6251 - binary_acc
uracy: 0.6457 - val_loss: 0.6681 - val_binary_accuracy: 0.6215
Epoch 33/50
19/19 [==============================] - 225s 12s/step - loss: 0.6124 - binary_acc
uracy: 0.6711 - val_loss: 0.6843 - val_binary_accuracy: 0.5892
Epoch 34/50
19/19 [==============================] - 237s 12s/step - loss: 0.5991 - binary_acc
uracy: 0.6780 - val_loss: 0.6479 - val_binary_accuracy: 0.6151
Epoch 35/50
19/19 [==============================] - 227s 12s/step - loss: 0.5895 - binary_acc
uracy: 0.6827 - val_loss: 0.6256 - val_binary_accuracy: 0.6516
Epoch 36/50
19/19 [==============================] - 228s 12s/step - loss: 0.5861 - binary_acc
uracy: 0.6887 - val_loss: 0.6168 - val_binary_accuracy: 0.6452
Epoch 37/50
19/19 [==============================] - 222s 12s/step - loss: 0.5699 - binary_acc
uracy: 0.6986 - val_loss: 0.6980 - val_binary_accuracy: 0.5828
Epoch 38/50
19/19 [==============================] - 227s 12s/step - loss: 0.5706 - binary_acc
uracy: 0.7068 - val_loss: 0.6146 - val_binary_accuracy: 0.6602
Epoch 39/50
19/19 [==============================] - 226s 12s/step - loss: 0.5702 - binary_acc
uracy: 0.6930 - val_loss: 0.5929 - val_binary_accuracy: 0.6753
Epoch 40/50
19/19 [==============================] - 225s 12s/step - loss: 0.5603 - binary_acc
uracy: 0.7034 - val_loss: 0.6137 - val_binary_accuracy: 0.6473
Epoch 41/50
19/19 [==============================] - 226s 12s/step - loss: 0.5573 - binary_acc
uracy: 0.7085 - val_loss: 0.5483 - val_binary_accuracy: 0.6946
Epoch 42/50
19/19 [==============================] - 226s 12s/step - loss: 0.5533 - binary_acc
uracy: 0.7137 - val_loss: 0.6239 - val_binary_accuracy: 0.6559
Epoch 43/50
19/19 [==============================] - 222s 12s/step - loss: 0.5488 - binary_acc
```

```
uracy: 0.7120 - val_loss: 0.5386 - val_binary_accuracy: 0.6882
Epoch 44/50
19/19 [==============================] - 231s 12s/step - loss: 0.5659 - binary_acc
uracy: 0.6999 - val_loss: 0.5490 - val_binary_accuracy: 0.7118
Epoch 45/50
19/19 [==============================] - 224s 12s/step - loss: 0.5603 - binary_acc
uracy: 0.7115 - val_loss: 0.6304 - val_binary_accuracy: 0.6366
Epoch 46/50
19/19 [==============================] - 224s 12s/step - loss: 0.5599 - binary_acc
uracy: 0.7120 - val_loss: 0.5710 - val_binary_accuracy: 0.6946
Epoch 47/50
19/19 [==============================] - 227s 12s/step - loss: 0.5494 - binary_acc
uracy: 0.7201 - val_loss: 0.5689 - val_binary_accuracy: 0.7075
Epoch 48/50
19/19 [==============================] - 224s 12s/step - loss: 0.5371 - binary_acc
uracy: 0.7184 - val_loss: 0.5359 - val_binary_accuracy: 0.7226
Epoch 49/50
19/19 [==============================] - 227s 12s/step - loss: 0.5420 - binary_acc
uracy: 0.7248 - val_loss: 0.5394 - val_binary_accuracy: 0.7269
Epoch 50/50
19/19 [==============================] - 236s 12s/step - loss: 0.5469 - binary_acc
uracy: 0.7167 - val_loss: 0.5774 - val_binary_accuracy: 0.6796
Number of total epochs ran:
```

Out[12]:

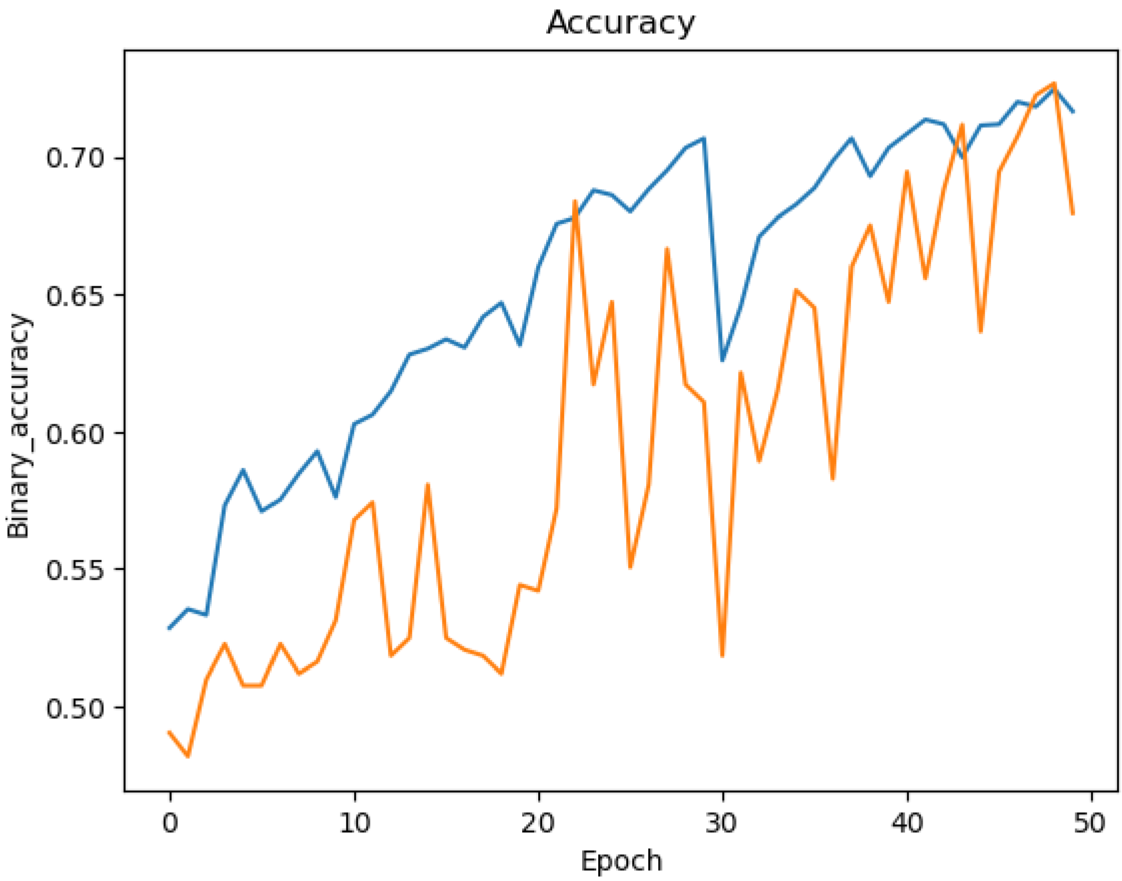
```
50
```

In [13]:

```
#model.load_weights('best_model.h5')
```

In [14]:

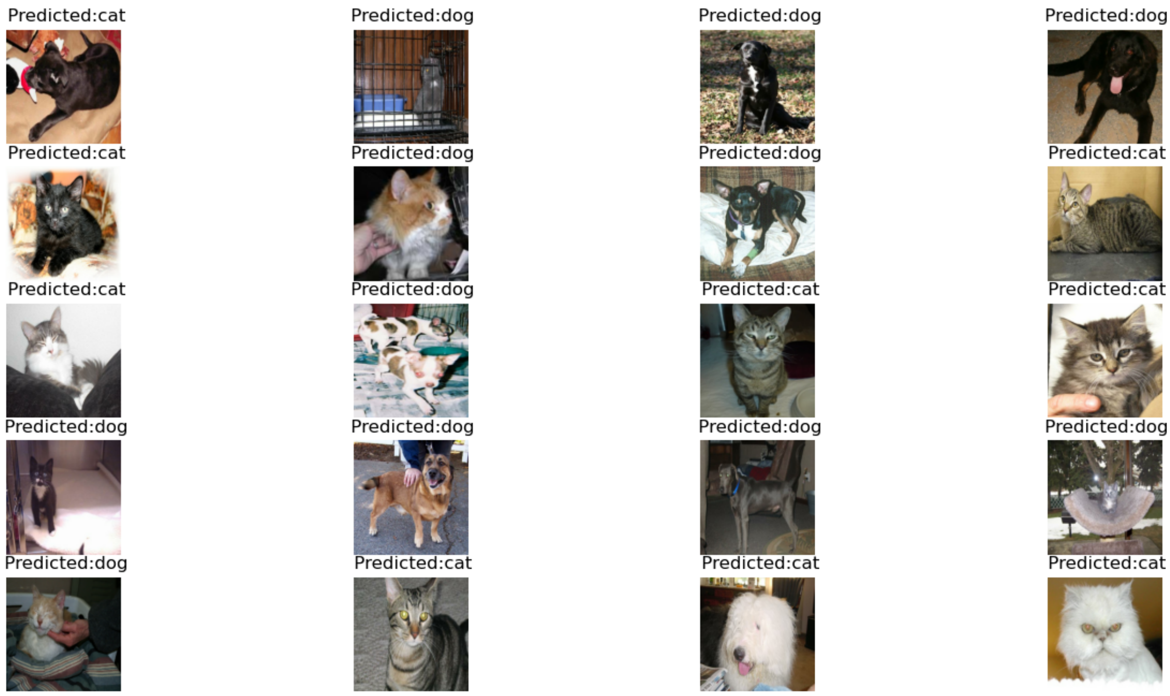
```
import matplotlib.pyplot as plt
epochs= range(1, 50+1)
plt.plot(history.history['binary_accuracy'])
plt.plot(history.history['val_binary_accuracy'])
plt.title('Accuracy')
plt.ylabel('Binary_accuracy')
plt.xlabel('Epoch')
plt.show()
```



In [15]:
```
import matplotlib.pyplot as plt

class_name=['cat', 'dog']
```

In [16]:
```
for images, labels in ds_test.take(20):
    predictions = model.predict(images)

def image_print(i, prediction_arr, img):
    prediction_label = int(prediction_arr[i] >0.5) #1 if greater than 0.5, 0 if le
    plt.imshow(img[i])
    plt.title(f'\n Predicted:{class_name[prediction_label]}')
    plt.axis('off')

fig, axes = plt.subplots(4,5, figsize=(16,8))
for i in range(20):#since there are 20 images
    plt.subplot(5,4, i+1)
    image_print(i, predictions, images)
plt.show()
```

```
4/4 [==============================] - 3s 601ms/step
4/4 [==============================] - 2s 597ms/step
4/4 [==============================] - 3s 642ms/step
3/3 [==============================] - 2s 458ms/step
C:\Users\selpa\AppData\Local\Temp\ipykernel_24768\2471148957.py:12: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 and will be removed two minor releases later; explicitly call ax.remove() as needed.
  plt.subplot(5,4, i+1)
```



In [17]:
```
history = model.compile(
    optimizer = tf.keras.optimizers.Adam(0.0001),
    #loss='binary_crossentropy',
    #metrics=['accuracy']
    loss = tf.keras.losses.BinaryCrossentropy(from_logits=False),
    metrics=[tf.keras.metrics.BinaryAccuracy()]
)
```

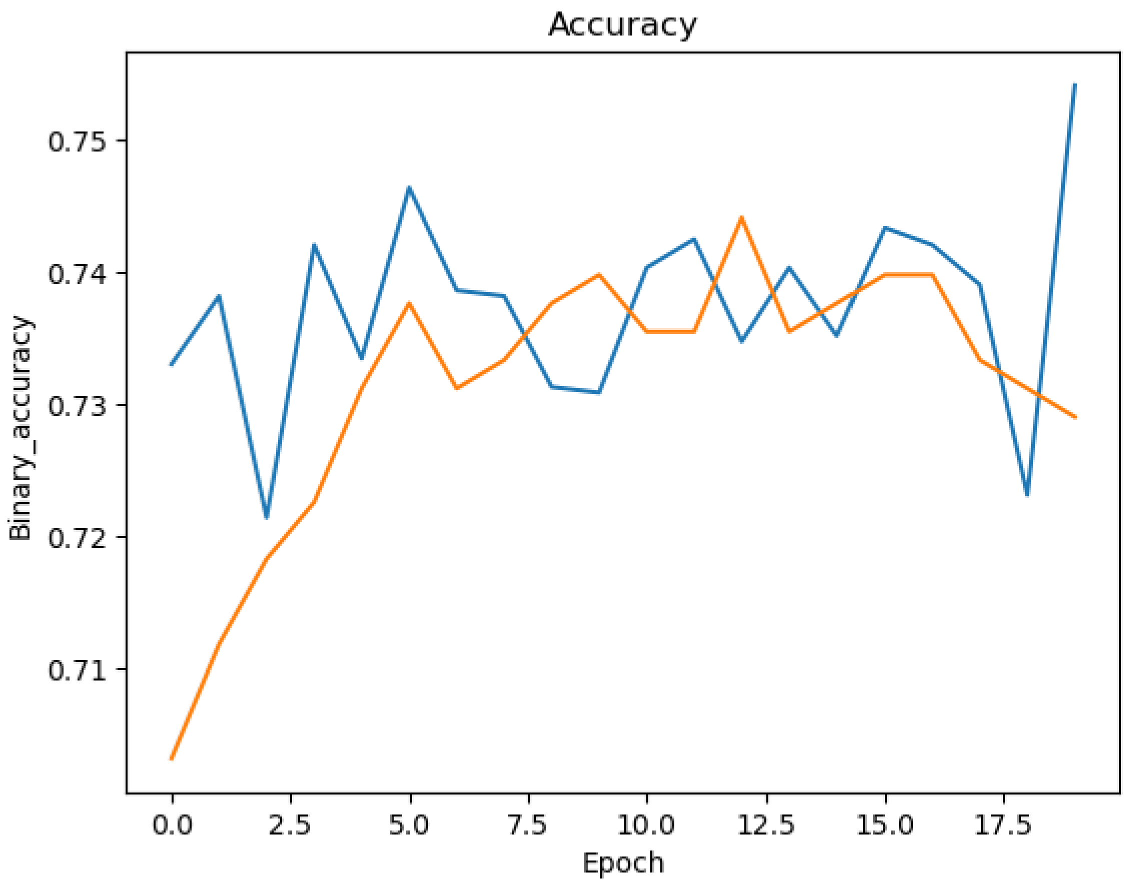
In [18]:
```
history = model.fit(
    ds_train,
    epochs=20,
    validation_data=ds_test,
)
print('Number of total epochs ran:')
len(history.history['val_binary_accuracy'])
```

```
Epoch 1/20
19/19 [==============================] - 236s 12s/step - loss: 0.5265 - binary_acc
uracy: 0.7330 - val_loss: 0.5507 - val_binary_accuracy: 0.7032
Epoch 2/20
19/19 [==============================] - 226s 12s/step - loss: 0.5302 - binary_acc
uracy: 0.7382 - val_loss: 0.5366 - val_binary_accuracy: 0.7118
Epoch 3/20
19/19 [==============================] - 226s 12s/step - loss: 0.5277 - binary_acc
uracy: 0.7214 - val_loss: 0.5259 - val_binary_accuracy: 0.7183
Epoch 4/20
19/19 [==============================] - 222s 12s/step - loss: 0.5221 - binary_acc
uracy: 0.7420 - val_loss: 0.5189 - val_binary_accuracy: 0.7226
Epoch 5/20
19/19 [==============================] - 238s 12s/step - loss: 0.5229 - binary_acc
uracy: 0.7334 - val_loss: 0.5140 - val_binary_accuracy: 0.7312
Epoch 6/20
19/19 [==============================] - 245s 13s/step - loss: 0.5186 - binary_acc
uracy: 0.7463 - val_loss: 0.5115 - val_binary_accuracy: 0.7376
Epoch 7/20
19/19 [==============================] - 247s 13s/step - loss: 0.5211 - binary_acc
uracy: 0.7386 - val_loss: 0.5098 - val_binary_accuracy: 0.7312
Epoch 8/20
19/19 [==============================] - 245s 13s/step - loss: 0.5179 - binary_acc
uracy: 0.7382 - val_loss: 0.5083 - val_binary_accuracy: 0.7333
Epoch 9/20
19/19 [==============================] - 262s 14s/step - loss: 0.5249 - binary_acc
uracy: 0.7313 - val_loss: 0.5076 - val_binary_accuracy: 0.7376
Epoch 10/20
19/19 [==============================] - 248s 13s/step - loss: 0.5230 - binary_acc
uracy: 0.7309 - val_loss: 0.5073 - val_binary_accuracy: 0.7398
Epoch 11/20
19/19 [==============================] - 241s 12s/step - loss: 0.5119 - binary_acc
uracy: 0.7403 - val_loss: 0.5070 - val_binary_accuracy: 0.7355
Epoch 12/20
19/19 [==============================] - 290s 15s/step - loss: 0.5137 - binary_acc
uracy: 0.7425 - val_loss: 0.5072 - val_binary_accuracy: 0.7355
Epoch 13/20
19/19 [==============================] - 295s 15s/step - loss: 0.5195 - binary_acc
uracy: 0.7347 - val_loss: 0.5075 - val_binary_accuracy: 0.7441
Epoch 14/20
19/19 [==============================] - 241s 12s/step - loss: 0.5124 - binary_acc
uracy: 0.7403 - val_loss: 0.5070 - val_binary_accuracy: 0.7355
Epoch 15/20
19/19 [==============================] - 239s 13s/step - loss: 0.5146 - binary_acc
uracy: 0.7352 - val_loss: 0.5067 - val_binary_accuracy: 0.7376
Epoch 16/20
19/19 [==============================] - 230s 12s/step - loss: 0.5123 - binary_acc
uracy: 0.7433 - val_loss: 0.5070 - val_binary_accuracy: 0.7398
Epoch 17/20
19/19 [==============================] - 229s 12s/step - loss: 0.5135 - binary_acc
uracy: 0.7420 - val_loss: 0.5066 - val_binary_accuracy: 0.7398
Epoch 18/20
19/19 [==============================] - 228s 12s/step - loss: 0.5175 - binary_acc
uracy: 0.7390 - val_loss: 0.5069 - val_binary_accuracy: 0.7333
Epoch 19/20
19/19 [==============================] - 230s 12s/step - loss: 0.5195 - binary_acc
uracy: 0.7231 - val_loss: 0.5074 - val_binary_accuracy: 0.7312
Epoch 20/20
19/19 [==============================] - 223s 12s/step - loss: 0.5076 - binary_acc
uracy: 0.7541 - val_loss: 0.5070 - val_binary_accuracy: 0.7290
Number of total epochs ran:
```

Out[18]:

```
20
```

In [19]:
```
import matplotlib.pyplot as plt
epochs= range(1, 20+1)
plt.plot(history.history['binary_accuracy'])
plt.plot(history.history['val_binary_accuracy'])
plt.title('Accuracy')
plt.ylabel('Binary_accuracy')
plt.xlabel('Epoch')
plt.show()
```



In [20]:
```
history = model.compile(
    optimizer = tf.keras.optimizers.Adam(0.001),
    #loss='binary_crossentropy',
    #metrics=['accuracy']
    loss = tf.keras.losses.BinaryCrossentropy(from_logits=False),
    metrics=[tf.keras.metrics.BinaryAccuracy()]
)
```

In [21]:
```
history = model.fit(
    ds_train,
    epochs=30,
    validation_data=ds_test,
)
print('Number of total epochs ran:')
len(history.history['loss'])
```

```
Epoch 1/30
19/19 [==============================] - 226s 12s/step - loss: 0.5197 - binary_acc
uracy: 0.7395 - val_loss: 0.5079 - val_binary_accuracy: 0.7462
Epoch 2/30
19/19 [==============================] - 222s 12s/step - loss: 0.5153 - binary_acc
uracy: 0.7451 - val_loss: 0.5077 - val_binary_accuracy: 0.7398
Epoch 3/30
19/19 [==============================] - 222s 12s/step - loss: 0.5151 - binary_acc
uracy: 0.7425 - val_loss: 0.5064 - val_binary_accuracy: 0.7290
Epoch 4/30
19/19 [==============================] - 225s 12s/step - loss: 0.4958 - binary_acc
uracy: 0.7635 - val_loss: 0.5020 - val_binary_accuracy: 0.7398
Epoch 5/30
19/19 [==============================] - 218s 11s/step - loss: 0.5014 - binary_acc
uracy: 0.7541 - val_loss: 0.5047 - val_binary_accuracy: 0.7398
Epoch 6/30
19/19 [==============================] - 224s 12s/step - loss: 0.4994 - binary_acc
uracy: 0.7519 - val_loss: 0.5038 - val_binary_accuracy: 0.7226
Epoch 7/30
19/19 [==============================] - 237s 12s/step - loss: 0.5169 - binary_acc
uracy: 0.7390 - val_loss: 0.5051 - val_binary_accuracy: 0.7269
Epoch 8/30
19/19 [==============================] - 229s 12s/step - loss: 0.5057 - binary_acc
uracy: 0.7549 - val_loss: 0.5015 - val_binary_accuracy: 0.7398
Epoch 9/30
19/19 [==============================] - 232s 12s/step - loss: 0.4938 - binary_acc
uracy: 0.7537 - val_loss: 0.4993 - val_binary_accuracy: 0.7548
Epoch 10/30
19/19 [==============================] - 247s 13s/step - loss: 0.4886 - binary_acc
uracy: 0.7644 - val_loss: 0.5003 - val_binary_accuracy: 0.7462
Epoch 11/30
19/19 [==============================] - 255s 13s/step - loss: 0.5018 - binary_acc
uracy: 0.7489 - val_loss: 0.5008 - val_binary_accuracy: 0.7548
Epoch 12/30
19/19 [==============================] - 297s 16s/step - loss: 0.4974 - binary_acc
uracy: 0.7489 - val_loss: 0.5089 - val_binary_accuracy: 0.7333
Epoch 13/30
19/19 [==============================] - 258s 14s/step - loss: 0.4950 - binary_acc
uracy: 0.7494 - val_loss: 0.5075 - val_binary_accuracy: 0.7398
Epoch 14/30
19/19 [==============================] - 330s 17s/step - loss: 0.4949 - binary_acc
uracy: 0.7657 - val_loss: 0.4990 - val_binary_accuracy: 0.7677
Epoch 15/30
19/19 [==============================] - 232s 12s/step - loss: 0.4856 - binary_acc
uracy: 0.7567 - val_loss: 0.5008 - val_binary_accuracy: 0.7527
Epoch 16/30
19/19 [==============================] - 229s 12s/step - loss: 0.4967 - binary_acc
uracy: 0.7545 - val_loss: 0.4965 - val_binary_accuracy: 0.7527
Epoch 17/30
19/19 [==============================] - 228s 12s/step - loss: 0.4848 - binary_acc
uracy: 0.7648 - val_loss: 0.4878 - val_binary_accuracy: 0.7699
Epoch 18/30
19/19 [==============================] - 237s 12s/step - loss: 0.4811 - binary_acc
uracy: 0.7627 - val_loss: 0.4920 - val_binary_accuracy: 0.7484
Epoch 19/30
19/19 [==============================] - 340s 18s/step - loss: 0.4737 - binary_acc
uracy: 0.7666 - val_loss: 0.4899 - val_binary_accuracy: 0.7570
Epoch 20/30
19/19 [==============================] - 314s 16s/step - loss: 0.4813 - binary_acc
uracy: 0.7567 - val_loss: 0.4945 - val_binary_accuracy: 0.7505
Epoch 21/30
19/19 [==============================] - 298s 15s/step - loss: 0.4803 - binary_acc
uracy: 0.7644 - val_loss: 0.4891 - val_binary_accuracy: 0.7441
Epoch 22/30
```

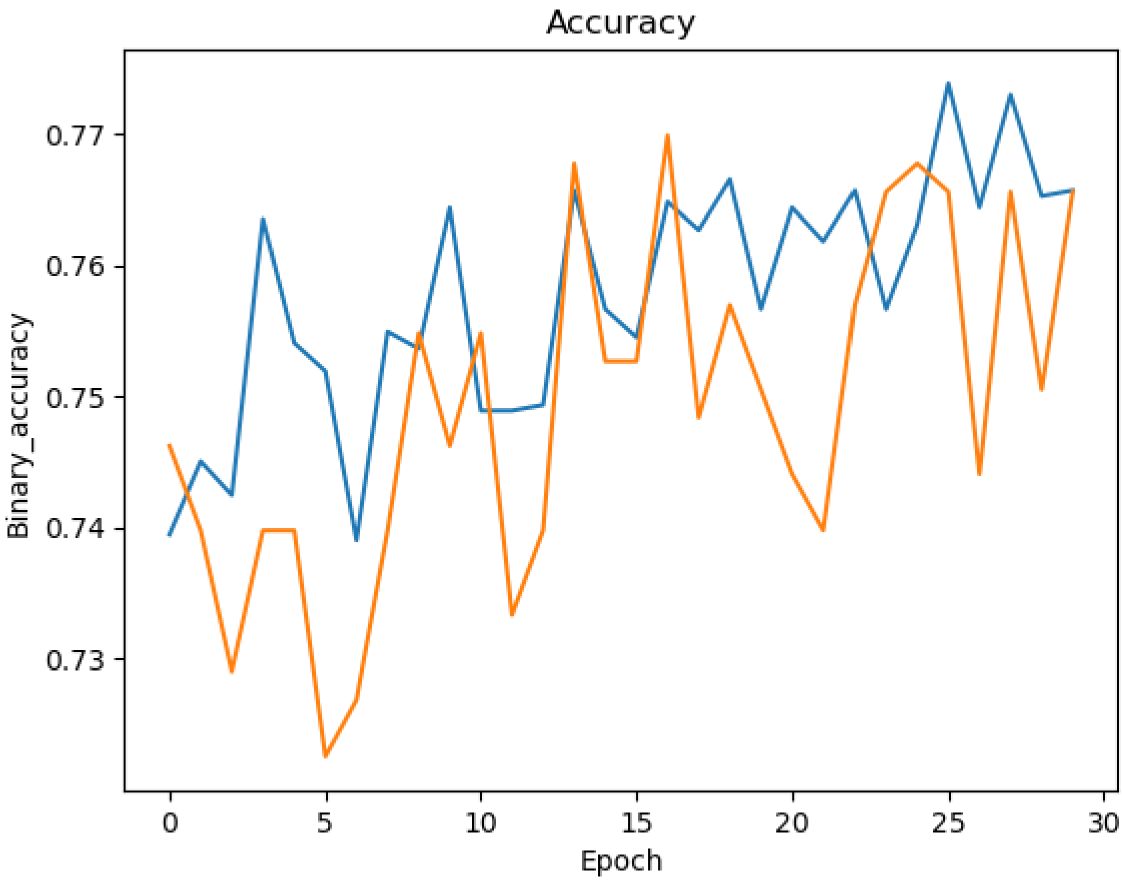
```
19/19 [==============================] - 320s 16s/step - loss: 0.4852 - binary_acc
uracy: 0.7618 - val_loss: 0.4950 - val_binary_accuracy: 0.7398
Epoch 23/30
19/19 [==============================] - 255s 13s/step - loss: 0.4779 - binary_acc
uracy: 0.7657 - val_loss: 0.4995 - val_binary_accuracy: 0.7570
Epoch 24/30
19/19 [==============================] - 274s 15s/step - loss: 0.4780 - binary_acc
uracy: 0.7567 - val_loss: 0.4960 - val_binary_accuracy: 0.7656
Epoch 25/30
19/19 [==============================] - 303s 16s/step - loss: 0.4764 - binary_acc
uracy: 0.7631 - val_loss: 0.4968 - val_binary_accuracy: 0.7677
Epoch 26/30
19/19 [==============================] - 278s 14s/step - loss: 0.4791 - binary_acc
uracy: 0.7739 - val_loss: 0.4893 - val_binary_accuracy: 0.7656
Epoch 27/30
19/19 [==============================] - 268s 14s/step - loss: 0.4741 - binary_acc
uracy: 0.7644 - val_loss: 0.4926 - val_binary_accuracy: 0.7441
Epoch 28/30
19/19 [==============================] - 336s 18s/step - loss: 0.4662 - binary_acc
uracy: 0.7730 - val_loss: 0.4895 - val_binary_accuracy: 0.7656
Epoch 29/30
19/19 [==============================] - 252s 13s/step - loss: 0.4765 - binary_acc
uracy: 0.7653 - val_loss: 0.4848 - val_binary_accuracy: 0.7505
Epoch 30/30
19/19 [==============================] - 235s 12s/step - loss: 0.4860 - binary_acc
uracy: 0.7657 - val_loss: 0.4851 - val_binary_accuracy: 0.7656
Number of total epochs ran:
```

Out[21]:

```
30
```

In [22]:

```
import matplotlib.pyplot as plt
epochs= range(1, 30+1)
plt.plot(history.history['binary_accuracy'])
plt.plot(history.history['val_binary_accuracy'])
plt.title('Accuracy')
plt.ylabel('Binary_accuracy')
plt.xlabel('Epoch')
plt.show()
```


Accuracy
Binary_accuracy
0.77
0.76
0.75
0.74
0.73
Epoch
0
5
10
15
20
25
30