In [1]:

```
import cv2
import matplotlib.pyplot as plt
import numpy
from matplotlib import pyplot
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Dense
from keras.layers import Flatten
from keras.optimizers import SGD
from keras.preprocessing.image import ImageDataGenerator
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
model = Sequential()
```

In [3]:

```
#add first convolution layer with the input shape
model.add(Conv2D(32, (3, 3),
                 activation='relu',
                 kernel_initializer='he_uniform',
padding='same', input_shape=(150, 150, 3)))
#add a Maxpooling layer
model.add(MaxPooling2D((2, 2)))
#add another convolution layer with more hidden units
model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform',
padding='same'))
model.add(MaxPooling2D((2, 2)))
#add another convolution layer with more hidden units
model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform',
padding='same'))
model.add(MaxPooling2D((2, 2)))
```

In [4]:

```
model.add(Flatten())
model.add(Dense(128,activation='relu',kernel_initializer='he_uniform'))
model.add(Dense(1, activation='sigmoid'))
```

In [5]:

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

In [6]:

```
# create data generators
train_datagen=ImageDataGenerator(rescale=1.0/255.0,
width_shift_range=0.1,height_shift_range=0.1, horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1.0/255.0)
# prepare iterators
train_it=train_datagen.flow_from_directory('training_set',class_mode='binary', batch_size=6
test_it=test_datagen.flow_from_directory('test_set',class_mode='binary', batch_size=64, tar
```

```
Found 160 images belonging to 2 classes.
Found 33 images belonging to 2 classes.
```

In [7]:

```
history=model.fit_generator(train_it,steps_per_epoch=len(train_it),validation_data=test_it,
```

```
Epoch 1/20
3/3 [==============================] - 3s 805ms/step - loss: 1.8597 - accura
cy: 0.5188 - val_loss: 0.7426 - val_accuracy: 0.4848
Epoch 2/20
3/3 [==============================] - 2s 642ms/step - loss: 0.7058 - accura
cy: 0.5625 - val_loss: 0.6942 - val_accuracy: 0.4848
Epoch 3/20
3/3 [==============================] - 2s 921ms/step - loss: 0.6862 - accura
cy: 0.5625 - val_loss: 0.6965 - val_accuracy: 0.4848
Epoch 4/20
3/3 [==============================] - 2s 903ms/step - loss: 0.6854 - accura
cy: 0.6250 - val_loss: 0.6875 - val_accuracy: 0.4848
Epoch 5/20
3/3 [==============================] - 2s 919ms/step - loss: 0.6724 - accura
cy: 0.5625 - val_loss: 0.6758 - val_accuracy: 0.5455
Epoch 6/20
3/3 [==============================] - 2s 655ms/step - loss: 0.6659 - accura
cy: 0.6313 - val_loss: 0.7124 - val_accuracy: 0.4848
Epoch 7/20
3/3 [==============================] - 2s 894ms/step - loss: 0.6579 - accura
cy: 0.6687 - val_loss: 0.6696 - val_accuracy: 0.5455
Epoch 8/20
3/3 [==============================] - 3s 992ms/step - loss: 0.6453 - accura
cy: 0.6500 - val_loss: 0.6547 - val_accuracy: 0.6061
Epoch 9/20
3/3 [==============================] - 2s 936ms/step - loss: 0.6770 - accura
cy: 0.6938 - val_loss: 0.6573 - val_accuracy: 0.5455
Epoch 10/20
3/3 [==============================] - 2s 681ms/step - loss: 0.6335 - accura
cy: 0.6938 - val_loss: 0.6500 - val_accuracy: 0.5758
Epoch 11/20
3/3 [==============================] - 2s 657ms/step - loss: 0.6236 - accura
cy: 0.7312 - val_loss: 0.6416 - val_accuracy: 0.6061
Epoch 12/20
3/3 [==============================] - 2s 678ms/step - loss: 0.6337 - accura
cy: 0.6812 - val_loss: 0.6733 - val_accuracy: 0.5152
Epoch 13/20
3/3 [==============================] - 2s 899ms/step - loss: 0.6477 - accura
cy: 0.6562 - val_loss: 0.6459 - val_accuracy: 0.6061
Epoch 14/20
3/3 [==============================] - 2s 735ms/step - loss: 0.6375 - accura
cy: 0.6562 - val_loss: 0.6125 - val_accuracy: 0.7879
Epoch 15/20
3/3 [==============================] - 2s 955ms/step - loss: 0.6104 - accura
cy: 0.6938 - val_loss: 0.6423 - val_accuracy: 0.5455
Epoch 16/20
3/3 [==============================] - 2s 941ms/step - loss: 0.6238 - accura
cy: 0.6500 - val_loss: 0.6347 - val_accuracy: 0.5455
Epoch 17/20
3/3 [==============================] - 2s 917ms/step - loss: 0.6002 - accura
cy: 0.7375 - val_loss: 0.7454 - val_accuracy: 0.4848
Epoch 18/20
3/3 [==============================] - 2s 666ms/step - loss: 0.6028 - accura
cy: 0.6625 - val_loss: 0.5894 - val_accuracy: 0.6667
Epoch 19/20
3/3 [==============================] - 2s 699ms/step - loss: 0.6421 - accura
```

```
cy: 0.6687 - val_loss: 0.6183 - val_accuracy: 0.6061
Epoch 20/20
3/3 [==============================] - 2s 976ms/step - loss: 0.5956 - accura
cy: 0.7188 - val_loss: 0.6998 - val_accuracy: 0.5152
```

In [8]:

```
model.save('monkeypox-vs-chickenpox.h5')
```

In [9]:

```
_,acc=model.evaluate_generator(test_it, steps=len(test_it), verbose=0)
print('> %.3f' % (acc * 100.0))
```

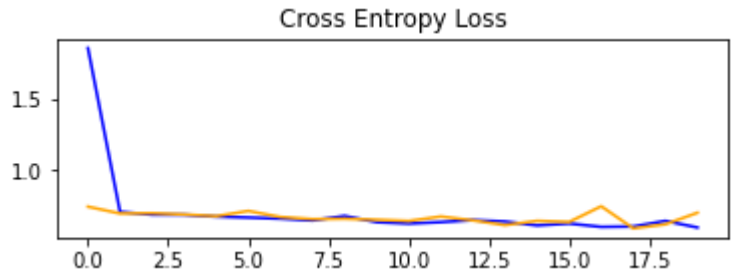
```
> 51.515
```

In [10]:

```
from matplotlib import pyplot
pyplot.subplot(211)
pyplot.title('Cross Entropy Loss')
pyplot.plot(history.history['loss'], color='blue', label='train')
pyplot.plot(history.history['val_loss'], color='orange', label='test')
```

Out[10]:

```
[<matplotlib.lines.Line2D at 0x1ed2f2ae560>]
```

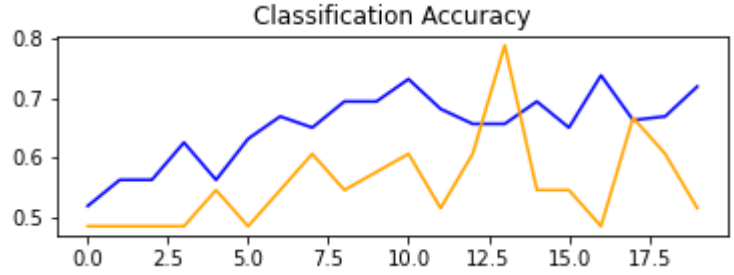


In [11]:

```
pyplot.subplot(212)
pyplot.title('Classification Accuracy')
pyplot.plot(history.history['accuracy'], color='blue', label='train')
pyplot.plot(history.history['val_accuracy'], color='orange', label='test')
```

Out[11]:

```
[<matplotlib.lines.Line2D at 0x1ed2f36cfa0>]
```



In [12]:

```
import sys
filename = sys.argv[0].split('/')[-1]
pyplot.savefig(filename + '_plot.png')
pyplot.close()
```

In [13]:

```
import keras
import tensorflow as tf
from tensorflow.keras.utils import load_img, img_to_array
```

In [14]:

```
import numpy as np
from keras.preprocessing import image
test_image = keras.utils.load_img('image.png', target_size = (150, 150))
test_image = tf.keras.preprocessing.image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)
result = model.predict(test_image)
train_it.class_indices
if result[0][0] == 1:
    prediction = 'Monkeypox'
    print(prediction)
else:
    prediction = 'Chickenpox'
    print(prediction)
```

```
1/1 [==============================] - 0s 65ms/step
Monkeypox
```