In [1]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
from tensorflow.keras.optimizers import SGD, RMSprop
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model
import tensorflow as  tf
import numpy as np
import cv2
import os
from keras.utils import np_utils
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceL
ROnPlateau
from keras.models import Sequential
from keras.layers import Dense,Flatten, Dropout
from tensorflow.keras.utils import load_img
from tensorflow.keras.utils import img_to_array
from keras.layers import Conv2D, MaxPooling2D
```

In [2]:
```
train = ImageDataGenerator(rescale = 1./255,
                                    shear_range = 0.2,
                                    zoom_range = 0.2,
                                    horizontal_flip = True)
validation = ImageDataGenerator(rescale = 1./255)
```

In [3]:
```
train_set=train.flow_from_directory('/content/drive/MyDrive/money/Train',targe
t_size = (64,64),batch_size = 12,class_mode = 'categorical')
test_set=validation.flow_from_directory('/content/drive/MyDrive/money/Validati
on',target_size = (64,64),batch_size = 12,class_mode = 'categorical')
```

```
Found 110 images belonging to 11 classes.
Found 55 images belonging to 11 classes.
```

In [11]:
```
train_set.class_indices
```

Out[11]:
```
{'1000': 0,
 '10000': 1,
 '100000': 2,
 '200': 3,
 '2000': 4,
 '20000': 5,
 '200000': 6,
 '500': 7,
 '5000': 8,
 '50000': 9,
 '500000': 10}
```

In [5]:
```
model = tf.keras.models.Sequential(
    [ tf.keras.layers.Conv2D(32,(3,3),activation = 'relu',kernel_initializer=
'he_uniform',padding='same',input_shape =(64,64,3)),
      tf.keras.layers.Conv2D(32,(3,3),activation = 'relu',kernel_initializer=
'he_uniform',padding='same'),
      tf.keras.layers.MaxPool2D(2,2),

      tf.keras.layers.Conv2D(32,(3,3),activation = 'relu',kernel_initializer=
'he_uniform',padding='same'),
      tf.keras.layers.Conv2D(32,(3,3),activation = 'relu',kernel_initializer=
'he_uniform',padding='same'),
      tf.keras.layers.MaxPool2D(2,2),

      tf.keras.layers.Conv2D(64,(3,3),activation = 'relu',kernel_initializer=
'he_uniform',padding='same'),
      tf.keras.layers.Conv2D(64,(3,3),activation = 'relu',kernel_initializer=
'he_uniform',padding='same'),
      tf.keras.layers.MaxPool2D(2,2),

      tf.keras.layers.Conv2D(128,(3,3),activation = 'relu',kernel_initializer=
'he_uniform',padding='same'),
      tf.keras.layers.Conv2D(128,(3,3),activation = 'relu',kernel_initializer=
'he_uniform',padding='same'),
      tf.keras.layers.MaxPool2D(2,2),

      tf.keras.layers.Flatten(),
      tf.keras.layers.Dense(256,activation = 'relu',kernel_initializer='he_uni
form'),
      tf.keras.layers.Dense(11,activation='softmax')])
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_8 (Conv2D)           (None, 64, 64, 32)        896

 conv2d_9 (Conv2D)           (None, 64, 64, 32)        9248

 max_pooling2d_4 (MaxPooling  (None, 32, 32, 32)       0
 2D)

 conv2d_10 (Conv2D)          (None, 32, 32, 32)        9248

 conv2d_11 (Conv2D)          (None, 32, 32, 32)        9248

 max_pooling2d_5 (MaxPooling  (None, 16, 16, 32)       0
 2D)

 conv2d_12 (Conv2D)          (None, 16, 16, 64)        18496

 conv2d_13 (Conv2D)          (None, 16, 16, 64)        36928

 max_pooling2d_6 (MaxPooling  (None, 8, 8, 64)         0
 2D)

 conv2d_14 (Conv2D)          (None, 8, 8, 128)         73856

 conv2d_15 (Conv2D)          (None, 8, 8, 128)         147584

 max_pooling2d_7 (MaxPooling  (None, 4, 4, 128)        0
 2D)

 flatten_1 (Flatten)         (None, 2048)              0

 dense_2 (Dense)             (None, 256)               524544

 dense_3 (Dense)             (None, 11)                2827

=================================================================
Total params: 832,875
Trainable params: 832,875
Non-trainable params: 0
_________________________________________________________________
```

In [6]:
```
opt = SGD(lr=0.001, momentum=0.9)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics = ['accu
racy'])
```

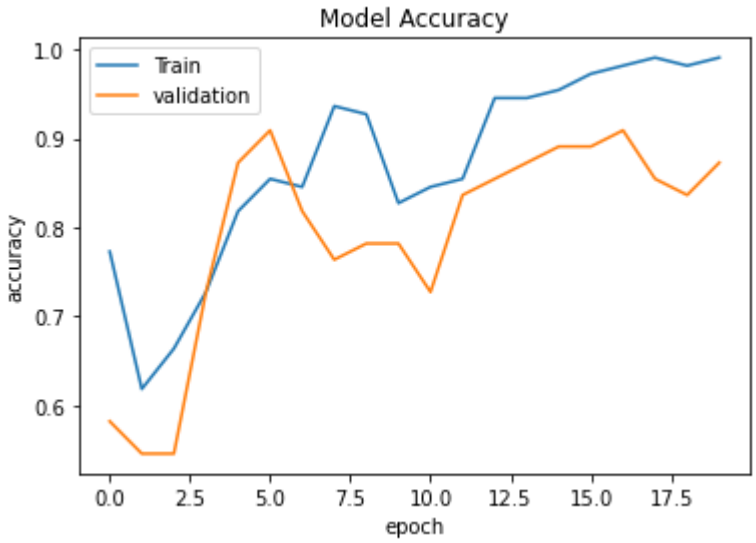
```
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descent.p
y:102: UserWarning: The `lr` argument is deprecated, use `learning_rate` inst
ead.
  super(SGD, self).__init__(name, **kwargs)
```

In [17]:
```
history = model.fit(train_set,batch_size=32,epochs=20,verbose=1,validation_dat
a=test_set)
```

```
Epoch 1/20
10/10 [==============================] - 3s 327ms/step - loss: 0.7874 - accur
acy: 0.7727 - val_loss: 1.4444 - val_accuracy: 0.5818
Epoch 2/20
10/10 [==============================] - 3s 315ms/step - loss: 1.4787 - accur
acy: 0.6182 - val_loss: 1.5787 - val_accuracy: 0.5455
Epoch 3/20
10/10 [==============================] - 3s 322ms/step - loss: 1.2217 - accur
acy: 0.6636 - val_loss: 1.3852 - val_accuracy: 0.5455
Epoch 4/20
10/10 [==============================] - 3s 322ms/step - loss: 0.8879 - accur
acy: 0.7273 - val_loss: 0.8400 - val_accuracy: 0.7273
Epoch 5/20
10/10 [==============================] - 3s 320ms/step - loss: 0.5542 - accur
acy: 0.8182 - val_loss: 0.5748 - val_accuracy: 0.8727
Epoch 6/20
10/10 [==============================] - 3s 306ms/step - loss: 0.4717 - accur
acy: 0.8545 - val_loss: 0.4423 - val_accuracy: 0.9091
Epoch 7/20
10/10 [==============================] - 3s 343ms/step - loss: 0.4527 - accur
acy: 0.8455 - val_loss: 0.5457 - val_accuracy: 0.8182
Epoch 8/20
10/10 [==============================] - 3s 310ms/step - loss: 0.2781 - accur
acy: 0.9364 - val_loss: 0.6302 - val_accuracy: 0.7636
Epoch 9/20
10/10 [==============================] - 3s 312ms/step - loss: 0.2029 - accur
acy: 0.9273 - val_loss: 0.7220 - val_accuracy: 0.7818
Epoch 10/20
10/10 [==============================] - 3s 316ms/step - loss: 0.5051 - accur
acy: 0.8273 - val_loss: 0.7406 - val_accuracy: 0.7818
Epoch 11/20
10/10 [==============================] - 3s 327ms/step - loss: 0.5089 - accur
acy: 0.8455 - val_loss: 0.7939 - val_accuracy: 0.7273
Epoch 12/20
10/10 [==============================] - 3s 319ms/step - loss: 0.4613 - accur
acy: 0.8545 - val_loss: 0.5503 - val_accuracy: 0.8364
Epoch 13/20
10/10 [==============================] - 3s 325ms/step - loss: 0.2228 - accur
acy: 0.9455 - val_loss: 0.4781 - val_accuracy: 0.8545
Epoch 14/20
10/10 [==============================] - 3s 319ms/step - loss: 0.1697 - accur
acy: 0.9455 - val_loss: 0.4388 - val_accuracy: 0.8727
Epoch 15/20
10/10 [==============================] - 3s 324ms/step - loss: 0.1128 - accur
acy: 0.9545 - val_loss: 0.3683 - val_accuracy: 0.8909
Epoch 16/20
10/10 [==============================] - 3s 322ms/step - loss: 0.0796 - accur
acy: 0.9727 - val_loss: 0.4021 - val_accuracy: 0.8909
Epoch 17/20
10/10 [==============================] - 3s 324ms/step - loss: 0.0680 - accur
acy: 0.9818 - val_loss: 0.4689 - val_accuracy: 0.9091
Epoch 18/20
10/10 [==============================] - 4s 342ms/step - loss: 0.0443 - accur
acy: 0.9909 - val_loss: 0.6551 - val_accuracy: 0.8545
Epoch 19/20
10/10 [==============================] - 3s 332ms/step - loss: 0.0766 - accur
acy: 0.9818 - val_loss: 0.6323 - val_accuracy: 0.8364
```

```
Epoch 20/20
10/10 [==============================] - 3s 342ms/step - loss: 0.0370 - accur
acy: 0.9909 - val_loss: 0.4119 - val_accuracy: 0.8727
```

In [18]:
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['Train','validation'],loc='upper left')
plt.show()
```



In [19]:
```
score=model.evaluate(test_set,verbose=1)
print('Sai số: ',score[0])
print('Độ chính xác: ',score[1])
```

```
5/5 [==============================] - 1s 114ms/step - loss: 0.4119 - accurac
y: 0.8727
Sai số:  0.4118559658527374
Độ chính xác:  0.8727272748947144
```

In [20]:
```
model.save('Money.h5')
```

In [21]:
```
from tensorflow.keras.models import load_model
model1=load_model('Money.h5')
```

In [24]:
```
test_img=load_img('/content/drive/MyDrive/money/Validation/10000/image (1).png',target_size=(64,64))
plt.imshow(test_img)
test_img= img_to_array(test_img)
test_img=test_img/255
test_img=np.expand_dims(test_img,axis=0)
result=model.predict(test_img)
if round(result[0][0])==1:
  prediction="1000"
elif round(result[0][1])==1:
  prediction="10000"
elif round(result[0][2])==1:
  prediction="100000"
elif round(result[0][3])==1:
  prediction="200"
elif round(result[0][4])==1:
  prediction="2000"
elif round(result[0][5])==1:
  prediction="20000"
elif round(result[0][6])==1:
  prediction="200000"
elif round(result[0][7])==1:
  prediction="500"
elif round(result[0][8])==1:
  prediction="5000"
elif round(result[0][9])==1:
  prediction="50000"
elif round(result[0][10])==1:
  prediction="500000"
print('dự đoán:', prediction)
```

```
dự đoán: 10000
```