```
import numpy as np
import os

from keras.models import Sequential
from keras.layers import Dense, Conv2D
from keras.layers import Dropout
from keras.layers import Flatten
from keras.constraints import maxnorm
from tensorflow.keras.optimizers import Adam

from keras.layers.convolutional import Convolution2D
from keras.layers.convolutional import MaxPooling2D
from keras.callbacks import ModelCheckpoint, LearningRateScheduler
from keras.callbacks import ReduceLROnPlateau
from keras.callbacks import EarlyStopping


from keras.utils import np_utils
import matplotlib.pyplot as plt
from keras.preprocessing.image import ImageDataGenerator
```

+ Code + Text

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/con
```

```
x_train= '/content/drive/MyDrive/money_recognition/train'
x_test= '/content/drive/MyDrive/money_recognition/test'
```

```
x_train = ImageDataGenerator(rescale=1/255)
x_test = ImageDataGenerator(rescale=1/255)
```

```
x_train_data = x_train.flow_from_directory(
    directory=r"/content/drive/MyDrive/money_recognition/train",
    target_size=(224, 224),
    batch_size=3,
    class_mode='categorical',
    )
x_test_data = x_test.flow_from_directory(
    directory=r"/content/drive/MyDrive/money_recognition/test",
    target_size=(224, 224),
    batch_size=3,
    class_mode= "categorical",
    )
```

```
Found 40 images belonging to 4 classes.
Found 21 images belonging to 4 classes.
```

```
x_train_data.class_indices
```

```
{'100k': 0, '10k': 1, '200k': 2, '50k': 3}
```

```
model = Sequential()

model.add(Conv2D(32,(3,3),input_shape=(224,224,3),padding='same',activation='relu'))
model.add(Dropout(0.2))

model.add(Conv2D(32,(3,3),activation='relu',padding='same'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(64,(3,3),activation='relu',padding='same'))
model.add(Dropout(0.2))

model.add(Conv2D(64,(3,3),activation='relu',padding='same'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(128,(3,3),activation='relu',padding='same'))
model.add(Dropout(0.2))

model.add(Conv2D(128,(3,3),activation='relu',padding='same'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())
model.add(Dropout(0.2))

model.add(Dense(1024,activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(512,activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(4,activation='softmax'))
model.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_42 (Conv2D)          (None, 224, 224, 32)      896

 dropout_42 (Dropout)        (None, 224, 224, 32)      0

 conv2d_43 (Conv2D)          (None, 224, 224, 32)      9248

 max_pooling2d_21 (MaxPoolin  (None, 112, 112, 32)     0
 g2D)

 conv2d_44 (Conv2D)          (None, 112, 112, 64)      18496

 dropout_43 (Dropout)        (None, 112, 112, 64)      0

 conv2d_45 (Conv2D)          (None, 112, 112, 64)      36928

 max_pooling2d_22 (MaxPoolin  (None, 56, 56, 64)       0
 g2D)

 conv2d_46 (Conv2D)          (None, 56, 56, 128)       73856

 dropout_44 (Dropout)        (None, 56, 56, 128)       0

 conv2d_47 (Conv2D)          (None, 56, 56, 128)       147584

 max_pooling2d_23 (MaxPoolin  (None, 28, 28, 128)      0
 g2D)
```

```
 flatten_7 (Flatten)         (None, 100352)            0

 dropout_45 (Dropout)        (None, 100352)            0

 dense_21 (Dense)            (None, 1024)              102761472

 dropout_46 (Dropout)        (None, 1024)              0

 dense_22 (Dense)            (None, 512)               524800

 dropout_47 (Dropout)        (None, 512)               0

 dense_23 (Dense)            (None, 4)                 2052

=================================================================
Total params: 103,575,332
Trainable params: 103,575,332
Non-trainable params: 0
_________________________________________________________________
```

```
from tensorflow.keras.optimizers import SGD
#opt = SGD(lr = 0.01, momentum= 0.9)
model.compile(optimizer=Adam(learning_rate=0.0005), loss='categorical_crossentropy', metrics=['accura
history=model.fit(x_train_data,epochs=2,batch_size=64,verbose=1,validation_data= x_test_data)
```

```
Epoch 1/2
14/14 [==============================] - 5s 278ms/step - loss: 0.4729 - accuracy: 0.9000 - val_
Epoch 2/2
14/14 [==============================] - 4s 259ms/step - loss: 0.1557 - accuracy: 0.9250 - val_
```

```
model.save('money_recognition.h5')
```

```
from keras.models import load_model
recognition = load_model('money_recognition.h5')
```

```
labels = {0: '100k', 1: '10k', 2: '200k',3: '50k'}
```

```
from keras.preprocessing.image import load_img, img_to_array
img = load_img('50k.jpg', target_size =(224,224))
plt.imshow(img)
img = img_to_array(img)
img = img.reshape(1,224,224,3)
img = img.astype('float32')
img = img/255
img.shape

val = recognition.predict(img)
np.argmax(val,axis=1)
print('Cost: ', labels[np.argmax(val)])
```

```
Cost:  50k
```