FruitsClassification

May 23, 2022

[1]:
```
#importing important libraries
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
from tensorflow.keras.optimizers import SGD, RMSprop,Adam
from tensorflow.keras.utils import to_categorical, load_img, img_to_array
from tensorflow.keras.models import load_model
from keras.utils import np_utils
from keras.models import Sequential
from keras.layers import Dense,Flatten, Dropout, Conv2D, MaxPooling2D
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint,␣
↪ReduceLROnPlateau
```

[2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[3]:
```
train = ImageDataGenerator(rescale = 1./255, shear_range = 0.2, zoom_range = 0.
↪2, horizontal_flip = True)
train_data = '/content/drive/MyDrive/FruitsClassification/Train_DATA'
validation = ImageDataGenerator(rescale=1./255)
valid_data = '/content/drive/MyDrive/FruitsClassification/Validation_DATA'
```

[4]:
```
train_dataset = train.flow_from_directory(train_data, target_size = (150,150),␣
↪batch_size = 10, class_mode = 'categorical')
validation_dataset = validation.flow_from_directory(valid_data, target_size =␣
↪(150,150), batch_size = 10, class_mode = 'categorical')
```

```
Found 88 images belonging to 10 classes.
Found 88 images belonging to 10 classes.
```

[5]:
```
train_dataset.class_indices
```

[5]:
```
{'chomchom': 0,
 'mangcut': 1,
 'mit': 2,
```

```
 'nhan': 3,
 'oi': 4,
 'saurieng': 5,
 'tao': 6,
 'thanhlong': 7,
 'vai': 8,
 'xoai': 9}
```

[6]:
```
model = Sequential()
model.add(Conv2D(32,(3,3),activation =␣
 ↪'relu',kernel_initializer='he_uniform',padding='same',input_shape=(150,150,3)))
model.add(Conv2D(32,(3,3),activation =␣
 ↪'relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(32,(3,3),activation =␣
 ↪'relu',kernel_initializer='he_uniform',padding='same'))
model.add(Conv2D(32,(3,3),activation =␣
 ↪'relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(64,(3,3),activation =␣
 ↪'relu',kernel_initializer='he_uniform',padding='same'))
model.add(Conv2D(64,(3,3),activation =␣
 ↪'relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(128,(3,3),activation =␣
 ↪'relu',kernel_initializer='he_uniform',padding='same'))
model.add(Conv2D(128,(3,3),activation =␣
 ↪'relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(Flatten())
model.add(Dense(256,activation='relu',kernel_initializer='he_uniform',))
model.add(Dense(10,activation='softmax'))
```

[7]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 150, 150, 32)      896

 conv2d_1 (Conv2D)           (None, 150, 150, 32)      9248

 max_pooling2d (MaxPooling2D  (None, 75, 75, 32)       0
 )

 conv2d_2 (Conv2D)           (None, 75, 75, 32)        9248
```

```
conv2d_3 (Conv2D)           (None, 75, 75, 32)        9248

max_pooling2d_1 (MaxPooling  (None, 37, 37, 32)       0
2D)

conv2d_4 (Conv2D)           (None, 37, 37, 64)        18496

conv2d_5 (Conv2D)           (None, 37, 37, 64)        36928

max_pooling2d_2 (MaxPooling  (None, 18, 18, 64)       0
2D)

conv2d_6 (Conv2D)           (None, 18, 18, 128)       73856

conv2d_7 (Conv2D)           (None, 18, 18, 128)       147584

max_pooling2d_3 (MaxPooling  (None, 9, 9, 128)        0
2D)

flatten (Flatten)           (None, 10368)             0

dense (Dense)               (None, 256)               2654464

dense_1 (Dense)             (None, 10)                2570

=================================================================
Total params: 2,962,538
Trainable params: 2,962,538
Non-trainable params: 0
_________________________________________________________________
```

[8]:
```
opt = SGD(lr=0.001, momentum=0.9)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics =␣
↪['accuracy'])
```

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

[9]:
```
history = model.
↪fit(train_dataset,batch_size=32,epochs=50,verbose=1,validation_data=validation_dataset)
```

```
Epoch 1/50
9/9 [==============================] - 50s 4s/step - loss: 2.4528 - accuracy:
0.0682 - val_loss: 2.3100 - val_accuracy: 0.1705
Epoch 2/50
```

```
9/9 [==============================] - 3s 334ms/step - loss: 2.3609 - accuracy:
0.0795 - val_loss: 2.2914 - val_accuracy: 0.1364
Epoch 3/50
9/9 [==============================] - 3s 334ms/step - loss: 2.3067 - accuracy:
0.0682 - val_loss: 2.2739 - val_accuracy: 0.1023
Epoch 4/50
9/9 [==============================] - 3s 333ms/step - loss: 2.2640 - accuracy:
0.1591 - val_loss: 2.2179 - val_accuracy: 0.2841
Epoch 5/50
9/9 [==============================] - 3s 329ms/step - loss: 2.2502 - accuracy:
0.1818 - val_loss: 2.1681 - val_accuracy: 0.1818
Epoch 6/50
9/9 [==============================] - 3s 336ms/step - loss: 2.1590 - accuracy:
0.2273 - val_loss: 2.0849 - val_accuracy: 0.2727
Epoch 7/50
9/9 [==============================] - 4s 434ms/step - loss: 2.1229 - accuracy:
0.1477 - val_loss: 1.9887 - val_accuracy: 0.2614
Epoch 8/50
9/9 [==============================] - 3s 330ms/step - loss: 2.1132 - accuracy:
0.2500 - val_loss: 2.0472 - val_accuracy: 0.3068
Epoch 9/50
9/9 [==============================] - 3s 343ms/step - loss: 2.0641 - accuracy:
0.3068 - val_loss: 1.9314 - val_accuracy: 0.3523
Epoch 10/50
9/9 [==============================] - 3s 345ms/step - loss: 1.9124 - accuracy:
0.3636 - val_loss: 1.6767 - val_accuracy: 0.5000
Epoch 11/50
9/9 [==============================] - 4s 474ms/step - loss: 1.7213 - accuracy:
0.4205 - val_loss: 1.5486 - val_accuracy: 0.4432
Epoch 12/50
9/9 [==============================] - 3s 333ms/step - loss: 1.7333 - accuracy:
0.4091 - val_loss: 1.2829 - val_accuracy: 0.6136
Epoch 13/50
9/9 [==============================] - 3s 329ms/step - loss: 1.4759 - accuracy:
0.4773 - val_loss: 1.2462 - val_accuracy: 0.5227
Epoch 14/50
9/9 [==============================] - 3s 346ms/step - loss: 1.5486 - accuracy:
0.4886 - val_loss: 2.0111 - val_accuracy: 0.3409
Epoch 15/50
9/9 [==============================] - 3s 401ms/step - loss: 1.7204 - accuracy:
0.4432 - val_loss: 1.4731 - val_accuracy: 0.4432
Epoch 16/50
9/9 [==============================] - 5s 560ms/step - loss: 1.4263 - accuracy:
0.4432 - val_loss: 1.1408 - val_accuracy: 0.5795
Epoch 17/50
9/9 [==============================] - 3s 395ms/step - loss: 1.0619 - accuracy:
0.6250 - val_loss: 0.7329 - val_accuracy: 0.7614
Epoch 18/50
```

```
9/9 [==============================] - 4s 437ms/step - loss: 1.3916 - accuracy:
0.4659 - val_loss: 2.0146 - val_accuracy: 0.4318
Epoch 19/50
9/9 [==============================] - 3s 382ms/step - loss: 1.7868 - accuracy:
0.4659 - val_loss: 1.2832 - val_accuracy: 0.5682
Epoch 20/50
9/9 [==============================] - 3s 355ms/step - loss: 1.2791 - accuracy:
0.5909 - val_loss: 0.9930 - val_accuracy: 0.7273
Epoch 21/50
9/9 [==============================] - 3s 342ms/step - loss: 1.0867 - accuracy:
0.6364 - val_loss: 0.8603 - val_accuracy: 0.7273
Epoch 22/50
9/9 [==============================] - 3s 337ms/step - loss: 1.1065 - accuracy:
0.6023 - val_loss: 0.6831 - val_accuracy: 0.7614
Epoch 23/50
9/9 [==============================] - 3s 363ms/step - loss: 0.9184 - accuracy:
0.7045 - val_loss: 0.8166 - val_accuracy: 0.6932
Epoch 24/50
9/9 [==============================] - 3s 335ms/step - loss: 0.8169 - accuracy:
0.7386 - val_loss: 0.5325 - val_accuracy: 0.8523
Epoch 25/50
9/9 [==============================] - 3s 337ms/step - loss: 0.5695 - accuracy:
0.7841 - val_loss: 0.3763 - val_accuracy: 0.8636
Epoch 26/50
9/9 [==============================] - 3s 334ms/step - loss: 0.6530 - accuracy:
0.7386 - val_loss: 0.3213 - val_accuracy: 0.8750
Epoch 27/50
9/9 [==============================] - 3s 359ms/step - loss: 1.1059 - accuracy:
0.6023 - val_loss: 0.9287 - val_accuracy: 0.7614
Epoch 28/50
9/9 [==============================] - 3s 332ms/step - loss: 1.1487 - accuracy:
0.6250 - val_loss: 0.8348 - val_accuracy: 0.7386
Epoch 29/50
9/9 [==============================] - 3s 385ms/step - loss: 0.8773 - accuracy:
0.6818 - val_loss: 0.5416 - val_accuracy: 0.8636
Epoch 30/50
9/9 [==============================] - 3s 394ms/step - loss: 0.5933 - accuracy:
0.7727 - val_loss: 0.2845 - val_accuracy: 0.9318
Epoch 31/50
9/9 [==============================] - 3s 390ms/step - loss: 0.4351 - accuracy:
0.8636 - val_loss: 0.2226 - val_accuracy: 0.9318
Epoch 32/50
9/9 [==============================] - 3s 365ms/step - loss: 0.3202 - accuracy:
0.8977 - val_loss: 0.2073 - val_accuracy: 0.9432
Epoch 33/50
9/9 [==============================] - 3s 337ms/step - loss: 0.2721 - accuracy:
0.9091 - val_loss: 0.1385 - val_accuracy: 0.9886
Epoch 34/50
```

```
9/9 [==============================] - 3s 380ms/step - loss: 0.2893 - accuracy:
0.8523 - val_loss: 0.3011 - val_accuracy: 0.8864
Epoch 35/50
9/9 [==============================] - 3s 387ms/step - loss: 0.4429 - accuracy:
0.8636 - val_loss: 0.4720 - val_accuracy: 0.8523
Epoch 36/50
9/9 [==============================] - 3s 337ms/step - loss: 0.3917 - accuracy:
0.8864 - val_loss: 0.1131 - val_accuracy: 0.9773
Epoch 37/50
9/9 [==============================] - 3s 384ms/step - loss: 0.2488 - accuracy:
0.9091 - val_loss: 0.0631 - val_accuracy: 0.9886
Epoch 38/50
9/9 [==============================] - 3s 367ms/step - loss: 0.1570 - accuracy:
0.9432 - val_loss: 0.0394 - val_accuracy: 1.0000
Epoch 39/50
9/9 [==============================] - 3s 369ms/step - loss: 0.2179 - accuracy:
0.9318 - val_loss: 0.0284 - val_accuracy: 1.0000
Epoch 40/50
9/9 [==============================] - 3s 331ms/step - loss: 0.2124 - accuracy:
0.9432 - val_loss: 0.1351 - val_accuracy: 0.9432
Epoch 41/50
9/9 [==============================] - 4s 473ms/step - loss: 0.4405 - accuracy:
0.8864 - val_loss: 0.1784 - val_accuracy: 0.9318
Epoch 42/50
9/9 [==============================] - 4s 417ms/step - loss: 0.4478 - accuracy:
0.8636 - val_loss: 0.0942 - val_accuracy: 0.9773
Epoch 43/50
9/9 [==============================] - 4s 506ms/step - loss: 0.2942 - accuracy:
0.8750 - val_loss: 0.1208 - val_accuracy: 0.9659
Epoch 44/50
9/9 [==============================] - 4s 488ms/step - loss: 0.2050 - accuracy:
0.9432 - val_loss: 0.0920 - val_accuracy: 0.9659
Epoch 45/50
9/9 [==============================] - 4s 493ms/step - loss: 0.0907 - accuracy:
1.0000 - val_loss: 0.0483 - val_accuracy: 1.0000
Epoch 46/50
9/9 [==============================] - 3s 370ms/step - loss: 0.1131 - accuracy:
0.9659 - val_loss: 0.0874 - val_accuracy: 0.9773
Epoch 47/50
9/9 [==============================] - 3s 371ms/step - loss: 0.2067 - accuracy:
0.9205 - val_loss: 0.0588 - val_accuracy: 0.9886
Epoch 48/50
9/9 [==============================] - 3s 362ms/step - loss: 0.1311 - accuracy:
0.9659 - val_loss: 0.1944 - val_accuracy: 0.9659
Epoch 49/50
9/9 [==============================] - 3s 344ms/step - loss: 0.0799 - accuracy:
0.9659 - val_loss: 0.0275 - val_accuracy: 1.0000
Epoch 50/50
```

```
9/9 [==============================] - 3s 361ms/step - loss: 0.0620 - accuracy:
0.9773 - val_loss: 0.1136 - val_accuracy: 0.9773
```

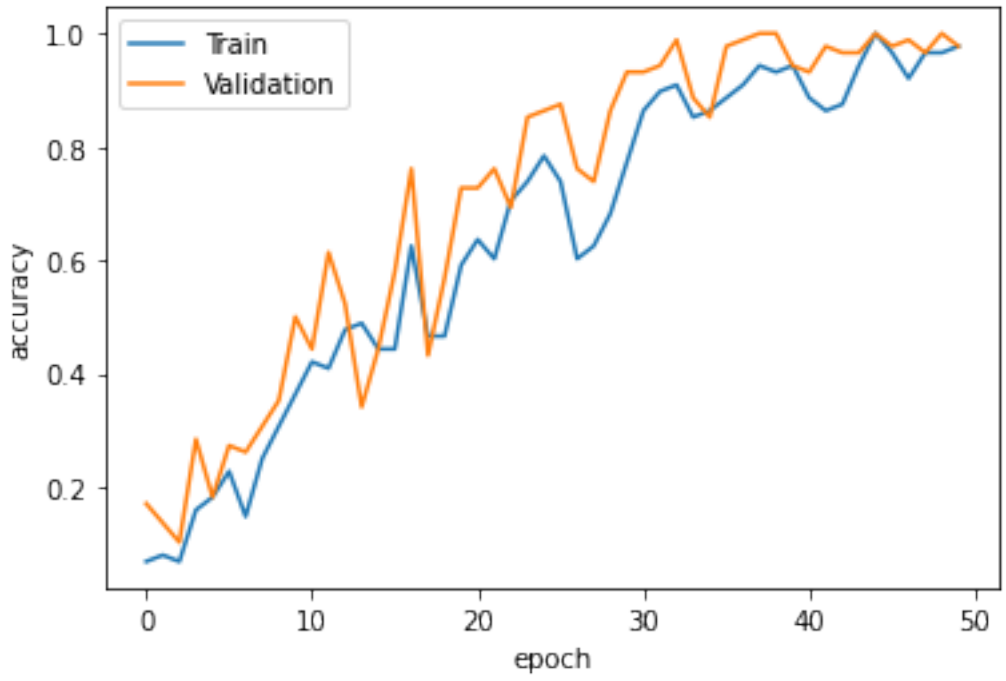
[12]:
```
model.save('/content/drive/MyDrive/FruitsClassification/FruitsClassification.
↪h5')
```

[17]:
```
classificationFruits_model = load_model('/content/drive/MyDrive/
↪FruitsClassification/FruitsClassification.h5')
```

[18]:
```
score=classificationFruits_model.evaluate(validation_dataset,verbose=1)
print('Test loss = ',score[0])
print('Test accuracy = ',score[1])
```

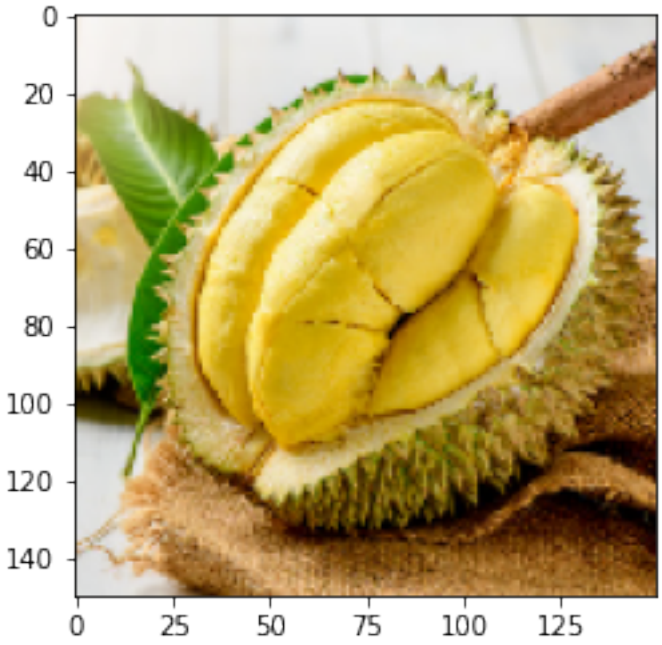
```
9/9 [==============================] - 2s 147ms/step - loss: 0.1136 - accuracy:
0.9773
Test loss =  0.11359824985265732
Test accuracy =  0.9772727489471436
```

[19]:
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['Train','Validation'],loc='upper left')
plt.show()
```



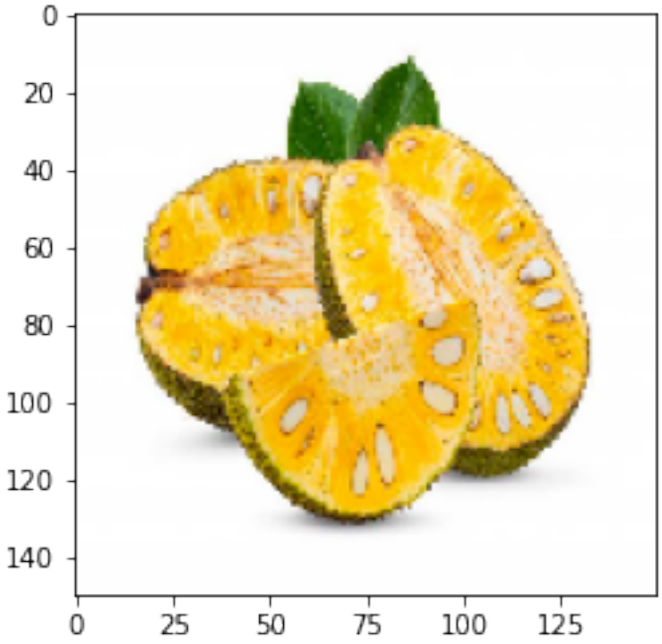
[20]:
```
img = load_img('/content/drive/MyDrive/FruitsClassification/Predict_Object/1.
 ↪jpg',target_size=(150,150))
plt.imshow(img)
img = img_to_array(img)
img = img.reshape(1,150,150,3)
img = img.astype('float32')
img = img/255
Label =␣
 ↪['chomchom','mangcut','mit','nhan','oi','saurieng','tao','thanhlong','vai','xoai']
print('Object: ',Label[int(np.argmax(classificationFood_model.
 ↪predict(img),axis=-1))])
```

```
Object:  saurieng
```



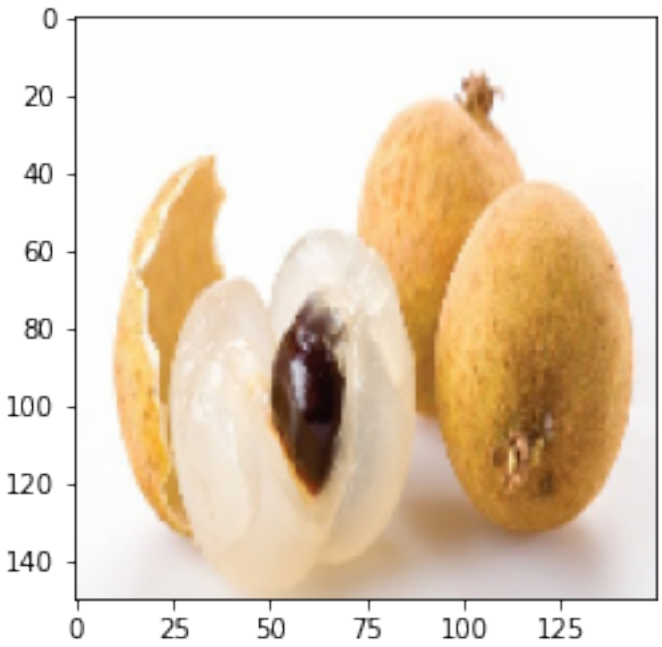
[22]:
```
img = load_img('/content/drive/MyDrive/FruitsClassification/Predict_Object/2.
 ↪jpg',target_size=(150,150))
plt.imshow(img)
img = img_to_array(img)
img = img.reshape(1,150,150,3)
img = img.astype('float32')
img = img/255
Label =␣
 ↪['chomchom','mangcut','mit','nhan','oi','saurieng','tao','thanhlong','vai','xoai']
print('Object: ',Label[int(np.argmax(classificationFruits_model.
 ↪predict(img),axis=-1))])
```

```
Object:  mit
```



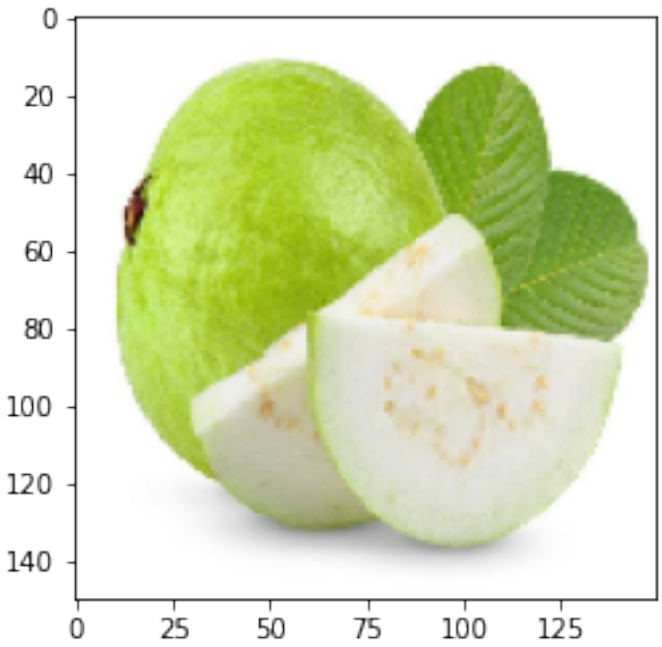
[23]:
```
img = load_img('/content/drive/MyDrive/FruitsClassification/Predict_Object/3.
 ↪jpg',target_size=(150,150))
plt.imshow(img)
img = img_to_array(img)
img = img.reshape(1,150,150,3)
img = img.astype('float32')
img = img/255
Label =␣
 ↪['chomchom','mangcut','mit','nhan','oi','saurieng','tao','thanhlong','vai','xoai']
print('Object: ',Label[int(np.argmax(classificationFruits_model.
 ↪predict(img),axis=-1))])
```

```
Object:  nhan
```



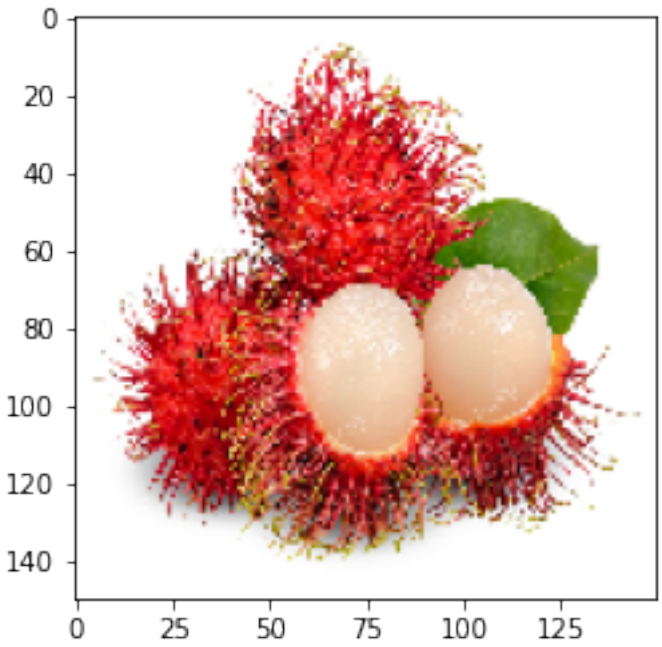
[25]:
```
img = load_img('/content/drive/MyDrive/FruitsClassification/Predict_Object/4.
 ↪png',target_size=(150,150))
plt.imshow(img)
img = img_to_array(img)
img = img.reshape(1,150,150,3)
img = img.astype('float32')
img = img/255
Label =␣
 ↪['chomchom','mangcut','mit','nhan','oi','saurieng','tao','thanhlong','vai','xoai']
print('Object: ',Label[int(np.argmax(classificationFruits_model.
 ↪predict(img),axis=-1))])
```

```
Object:  oi
```



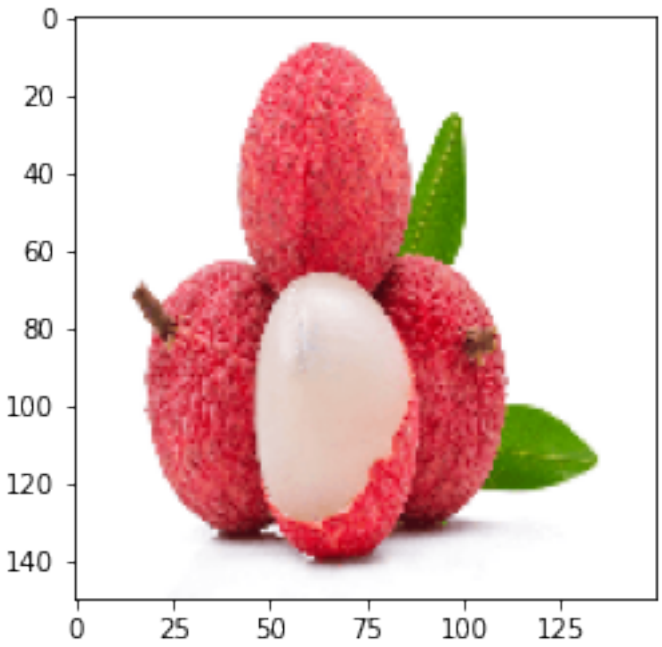
[26]:
```
img = load_img('/content/drive/MyDrive/FruitsClassification/Predict_Object/5.
 ↪jpg',target_size=(150,150))
plt.imshow(img)
img = img_to_array(img)
img = img.reshape(1,150,150,3)
img = img.astype('float32')
img = img/255
Label =␣
 ↪['chomchom','mangcut','mit','nhan','oi','saurieng','tao','thanhlong','vai','xoai']
print('Object: ',Label[int(np.argmax(classificationFruits_model.
 ↪predict(img),axis=-1))])
```

```
Object:  chomchom
```



[27]:
```
img = load_img('/content/drive/MyDrive/FruitsClassification/Predict_Object/6.
 ↪jpg',target_size=(150,150))
plt.imshow(img)
img = img_to_array(img)
img = img.reshape(1,150,150,3)
img = img.astype('float32')
img = img/255
Label =␣
 ↪['chomchom','mangcut','mit','nhan','oi','saurieng','tao','thanhlong','vai','xoai']
print('Object: ',Label[int(np.argmax(classificationFruits_model.
 ↪predict(img),axis=-1))])
```

```
Object:  vai
```



[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf('FruitsClassification.ipynb')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
--2022-05-23 06:15:27--  https://raw.githubusercontent.com/brpy/colab-
pdf/master/colab_pdf.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)…
185.199.108.133, 185.199.109.133, 185.199.110.133, …
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.108.133|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 1864 (1.8K) [text/plain]
Saving to: ‘colab_pdf.py’

colab_pdf.py        100%[===================>]   1.82K  --.-KB/s    in 0s

2022-05-23 06:15:27 (38.0 MB/s) - ‘colab_pdf.py’ saved [1864/1864]


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.
```

```
WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Extracting templates from packages: 100%
```

[ ]: