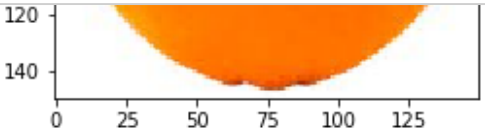
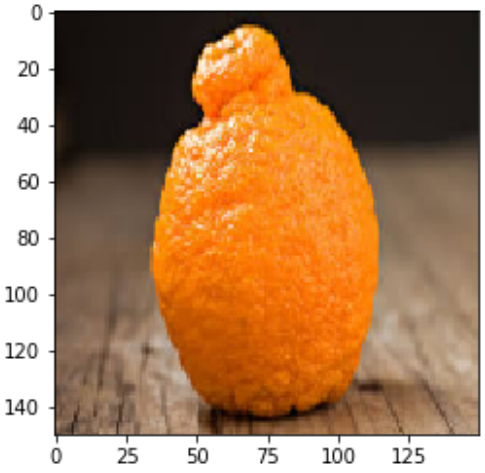
In [173]:
```
import cv2
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
%matplotlib inline

import os
import random
```

In [174]:
```
TRAINDIR = 'C:/Users/ADMIN/Desktop/Snehil devops/train'
CATEGORIES = ["apple","orange"]
```

In [175]:
```
nrows = 150
ncolumns = 150
channels = 3
```

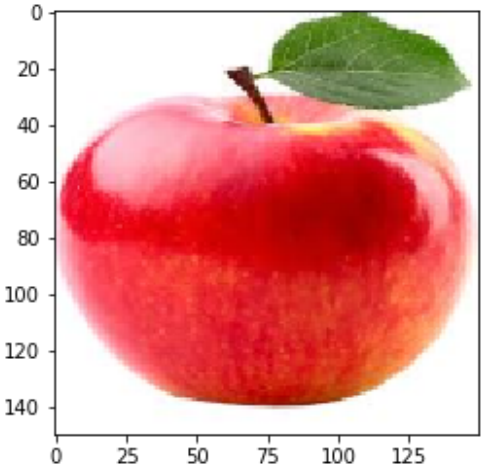
In [176]:
```
train_data=[]
def create_training_data():
    for category in CATEGORIES:
        path = os.path.join(TRAINDIR,category)      # path to apple or orange dir
        class_num = CATEGORIES.index(category)

        for image in os.listdir(path):              # Bunch of images in the path

            image_array1 = cv2.imread(os.path.join(path,image),cv2.IMREAD_COLOR)

            new_array1 = cv2.resize(image_array1,(nrows,ncolumns))       # Normal

            color_img = cv2.cvtColor(new_array1, cv2.COLOR_BGR2RGB)
            imgplot=plt.imshow(color_img)

            train_data.append([color_img,class_num])        # Collecting all trai
            plt.show()
create_training_data()
```



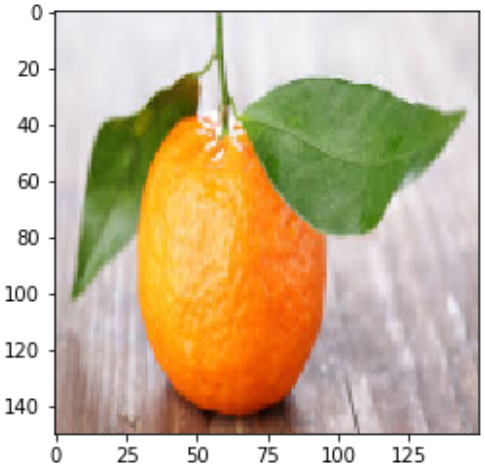


In [177]:
```
random.shuffle(train_data)
```

In [178]:
```
plt.imshow(train_data[1][0])
plt.show()
```



In [179]:
```
plt.imshow(train_data[2][0])
plt.show()
```



In [180]:
```
X = []
y = []

for feature,label in train_data:
    X.append(feature)
    y.append(label)

X = np.array(X).reshape(-1,nrows,ncolumns,channels)
y = np.array(y)
```

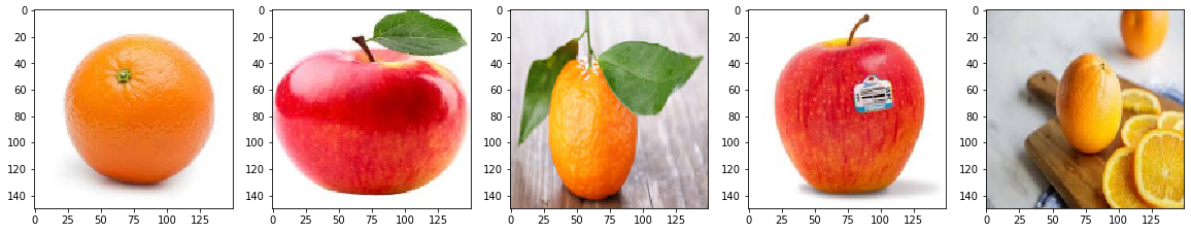
In [181]:
```
X[0]
```

Out[181]:
```
array([[[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       ...,

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]]], dtype=uint8)
```

In [182]:
```
y
```

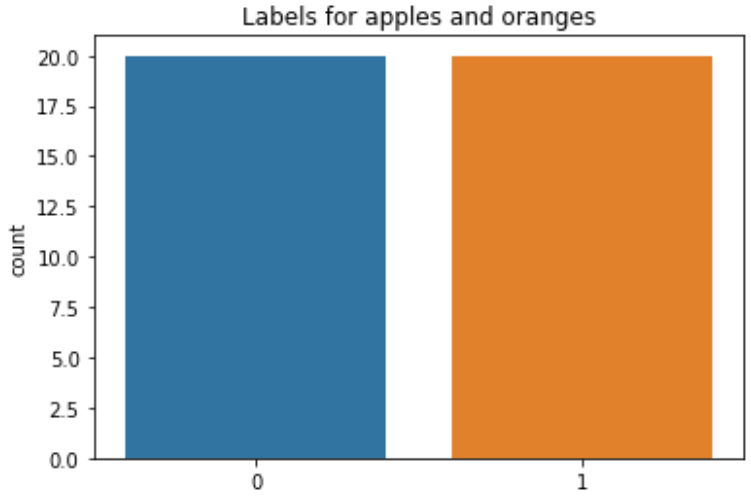
Out[182]:
```
array([1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1,
       0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0])
```

In [183]:
```
plt.figure(figsize=(20,20))
columns = 5
for i in range(columns):
    plt.subplot(5/columns + 1, columns, i + 1)
    plt.imshow(X[i])
```



In [184]:
```
import seaborn as sns
sns.countplot(y)
plt.title('Labels for apples and oranges')
```

Out[184]:
```
Text(0.5, 1.0, 'Labels for apples and oranges')
```



In [185]:
```
X.shape
```

Out[185]:
```
(40, 150, 150, 3)
```

In [186]:
```
y.shape
```

Out[186]:
```
(40,)
```

In [187]:
```
from sklearn.model_selection import train_test_split
X_train, X_val, y_train, y_val = train_test_split(X,y,test_size=0.2,random_state=
```

In [188]:
```
print(X_train.shape)
print(X_val.shape)
print(y_train.shape)
print(y_val.shape)
```

```
(32, 150, 150, 3)
(8, 150, 150, 3)
(32,)
(8,)
```

In [189]:
```
#get the length of the train and validation data
ntrain = len(X_train)
nval = len(X_val)

#We will use a batch size of 32. Note: batch size should be a factor of 2.***4,8,
batch_size = 32

print(ntrain)
print(nval)
```

```
32
8
```

In [190]:
```
from keras import layers
from keras import models
from keras import optimizers
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing.image import img_to_array, load_img

model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',input_shape=(150, 150, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dropout(0.5))  #Dropout for regularization
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))  #Sigmoid function at the end be
```

In [191]:
```
model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_21 (Conv2D)           (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d_21 (MaxPooling (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 72, 72, 64)        18496
_________________________________________________________________
max_pooling2d_22 (MaxPooling (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_23 (Conv2D)           (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_23 (MaxPooling (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_24 (Conv2D)           (None, 15, 15, 128)       147584
_________________________________________________________________
max_pooling2d_24 (MaxPooling (None, 7, 7, 128)         0
_________________________________________________________________
flatten_6 (Flatten)          (None, 6272)              0
_________________________________________________________________
dropout_6 (Dropout)          (None, 6272)              0
_________________________________________________________________
dense_11 (Dense)             (None, 512)               3211776
_________________________________________________________________
dense_12 (Dense)             (None, 1)                 513
=================================================================
Total params: 3,453,121
Trainable params: 3,453,121
Non-trainable params: 0
_________________________________________________________________
```

In [192]:
```
model.compile(loss='binary_crossentropy', optimizer=optimizers.RMSprop(lr=1e-4),
```

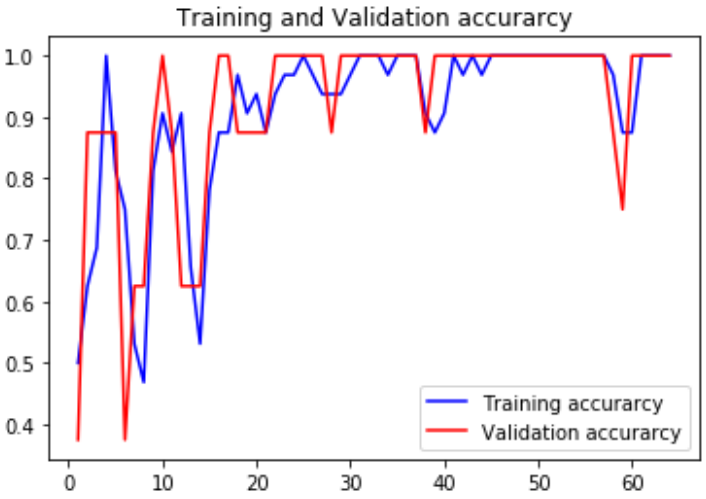
In [193]:
```
train_datagen = ImageDataGenerator(rescale=1/255,
                                   rotation_range=40,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True,)

val_datagen = ImageDataGenerator(rescale=1/255)
```

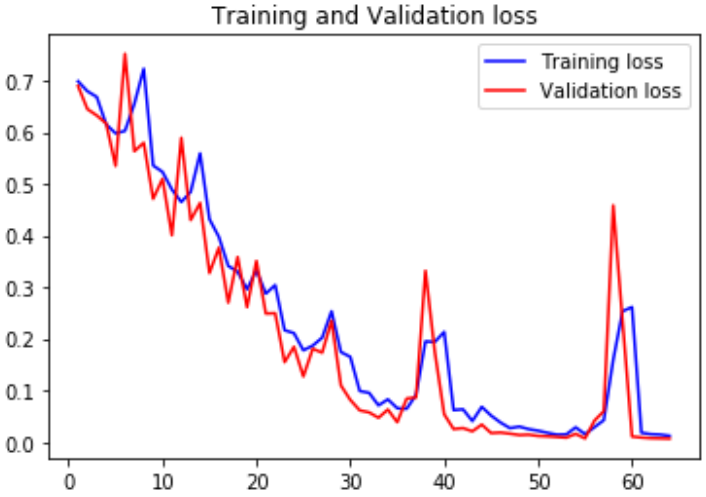
In [194]:
```
train_generator = train_datagen.flow(X_train, y_train, batch_size=batch_size)
val_generator = val_datagen.flow(X_val, y_val, batch_size=batch_size)
```

In [195]:
```
history = model.fit_generator(train_generator,
                              steps_per_epoch=ntrain // batch_size,
                              epochs=64,
                              validation_data=val_generator,
                              validation_steps=nval // batch_size)
```

```
1/1 [==============================] - 1s 1s/step - loss: 0.0644 - acc: 1.000
0 - val_loss: 0.0842 - val_acc: 1.0000
Epoch 37/64
1/1 [==============================] - 1s 1s/step - loss: 0.0914 - acc: 1.000
0 - val_loss: 0.0863 - val_acc: 1.0000
Epoch 38/64
1/1 [==============================] - 1s 1s/step - loss: 0.1949 - acc: 0.906
2 - val_loss: 0.3317 - val_acc: 0.8750
Epoch 39/64
1/1 [==============================] - 1s 1s/step - loss: 0.1939 - acc: 0.875
0 - val_loss: 0.1744 - val_acc: 1.0000
Epoch 40/64
1/1 [==============================] - 1s 1s/step - loss: 0.2132 - acc: 0.906
2 - val_loss: 0.0537 - val_acc: 1.0000
Epoch 41/64
1/1 [==============================] - 1s 1s/step - loss: 0.0620 - acc: 1.000
0 - val_loss: 0.0253 - val_acc: 1.0000
Epoch 42/64
1/1 [==============================] - 1s 1s/step - loss: 0.0634 - acc: 0.968
8 - val loss: 0.0270 - val acc: 1.0000
```

In [196]:
```
#lets plot the train and val curve
#get the details form the history object
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

#Train and validation accuracy
plt.plot(epochs, acc, 'b', label='Training accurarcy')
plt.plot(epochs, val_acc, 'r', label='Validation accurarcy')
plt.title('Training and Validation accurarcy')
plt.legend()

plt.figure()
#Train and validation loss
plt.plot(epochs, loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and Validation loss')
plt.legend()

plt.show()
```





In [214]:
```
TRAINDIR2 = 'C:/Users/ADMIN/Desktop/Snehil devops/train2'
CATEGORIES = ["apple","orange"]
```

In [215]:
```
nrows = 150
ncolumns = 150
channels = 3
```

In [216]:
```
train_data=[]
def create_training_data():
    for category in CATEGORIES:
        path = os.path.join(TRAINDIR2,category)      # path to apple or orange di
        class_num = CATEGORIES.index(category)

        for image in os.listdir(path):             # Bunch of images in the path

            image_array1 = cv2.imread(os.path.join(path,image))

            new_array1 = cv2.resize(image_array1,(nrows,ncolumns))       # Normal

            train_data.append([new_array1,class_num])        # Collecting all tra

create_training_data()
```

In [217]:
```
random.shuffle(train_data)
```

In [218]:
```
X = []
y = []

for feature,label in train_data:
    X.append(feature)
    y.append(label)

X = np.array(X).reshape(-1,nrows,ncolumns,channels)
y = np.array(y)
```

In [219]:
```
from sklearn.model_selection import train_test_split
X_train, X_val, y_train, y_val = train_test_split(X,y,test_size=0.2,random_state=
```

In [220]:
```
from keras import layers
from keras import models
from keras import optimizers
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing.image import img_to_array, load_img

model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',input_shape=(150, 150, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dropout(0.5))  #Dropout for regularization
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))  #Sigmoid function at the end be
```

In [221]:
```
model.compile(loss='binary_crossentropy', optimizer=optimizers.RMSprop(lr=1e-4),
```

In [222]:
```
train_datagen = ImageDataGenerator(rescale=1/255,
                                   rotation_range=40,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True,)

val_datagen = ImageDataGenerator(rescale=1/255)
```

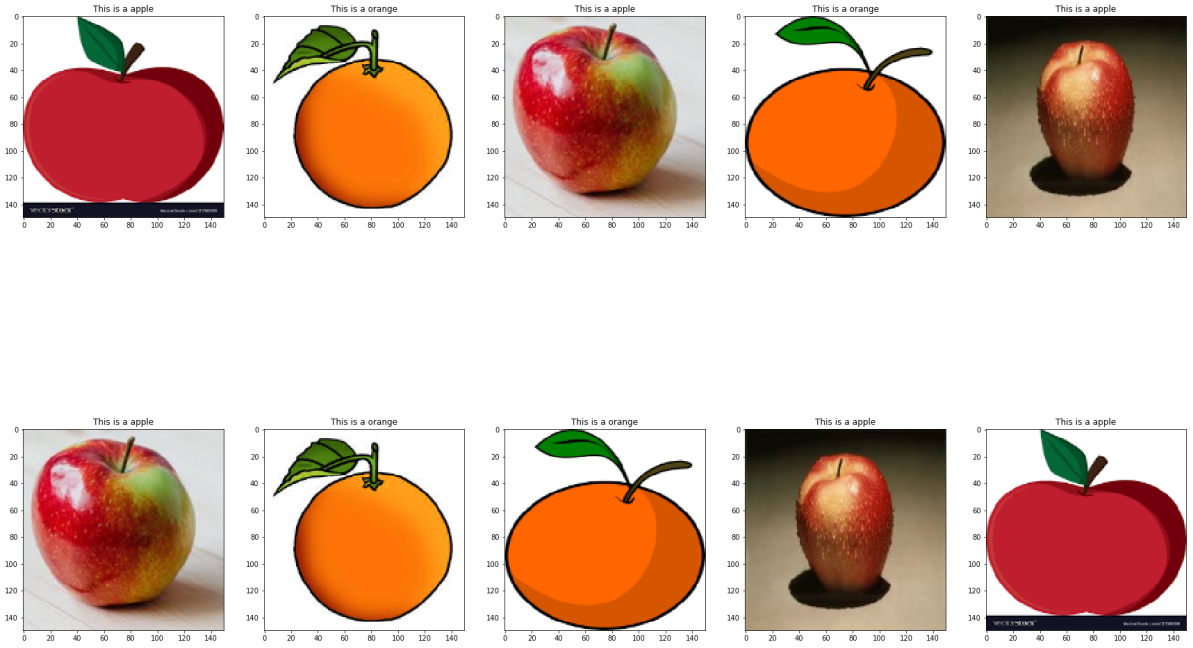
In [223]:
```
train_generator = train_datagen.flow(X_train, y_train, batch_size=batch_size)
val_generator = val_datagen.flow(X_val, y_val, batch_size=batch_size)
```

In [224]:
```
history = model.fit_generator(train_generator,
                              steps_per_epoch=ntrain // batch_size,
                              epochs=64,
                              validation_data=val_generator,
                              validation_steps=nval // batch_size)
```

```
0000 - val_loss: 0.3210 - val_acc: 1.0000
Epoch 37/64
1/1 [==============================] - 0s 228ms/step - loss: 0.3466 - acc: 1.
0000 - val_loss: 0.6908 - val_acc: 1.0000
Epoch 38/64
1/1 [==============================] - 0s 249ms/step - loss: 0.3102 - acc: 1.
0000 - val_loss: 0.4033 - val_acc: 1.0000
Epoch 39/64
1/1 [==============================] - 0s 237ms/step - loss: 0.2068 - acc: 1.
0000 - val_loss: 0.4103 - val_acc: 1.0000
Epoch 40/64
1/1 [==============================] - 0s 217ms/step - loss: 0.2023 - acc: 1.
0000 - val_loss: 0.3812 - val_acc: 1.0000
Epoch 41/64
1/1 [==============================] - 0s 237ms/step - loss: 0.1621 - acc: 1.
0000 - val_loss: 0.5058 - val_acc: 1.0000
Epoch 42/64
1/1 [==============================] - 0s 241ms/step - loss: 0.2003 - acc: 1.
0000 - val_loss: 0.2233 - val_acc: 1.0000
Epoch 43/64
```

In [225]:
```
test_datagen=ImageDataGenerator(rescale=1./255)
```

In [226]:
```
i = 0
text_labels = []
plt.figure(figsize=(30,20))
for batch in test_datagen.flow(X, batch_size=1):
    pred = model.predict(batch)
    if pred > 0.5:
        text_labels.append('orange')
    else:
        text_labels.append('apple')
    plt.subplot(5 / columns + 1, columns, i + 1)
    plt.title('This is a ' + text_labels[i])
    color_img = cv2.cvtColor(batch[0], cv2.COLOR_BGR2RGB)
    imgplot=plt.imshow(color_img)
    i += 1
    if i % 10 == 0:
        break
plt.show()
```



In [ ]: