In [1]:
```
import os
import zipfile
import random
import shutil
import numpy as np
from shutil import copyfile

%matplotlib inline
import matplotlib.image as mpimg
import matplotlib.pyplot as plt

import tensorflow as tf
import keras.utils as image
from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.applications import InceptionV3
from tensorflow.keras import layers
from tensorflow.keras import Model
import pandas as pd

import keras
from keras.layers import Dense,Dropout, Conv2D,MaxPooling2D , Activation, Flatten, BatchNormalization, SeparableConv2D
from keras.models import Sequential


os.environ['KMP_DUPLICATE_LIB_OK']='True'
```

In [2]:
```
print(tf.__version__)
```

```
2.9.1
```

In [3]:
```
base_dir = os.path.join("C:/Users/Abdoulla Yousef/Downloads/OriginalImage/Original Image/")
print('Base directory --> ', os.listdir(base_dir))
```

```
Base directory -->  ['FreshApple', 'FreshBanana', 'FreshGrape', 'FreshGuava', 'FreshJujube', 'FreshOrange', 'FreshPomegranate', 'FreshStrawberry', 'fruit-
dataset', 'RottenApple', 'RottenBanana', 'RottenGrape', 'RottenGuava', 'RottenJujube', 'RottenOrange', 'RottenPomegranate', 'RottenStrawberry']
```

In [4]:
```
def make_dir(PATH):
  if not os.path.exists(PATH):
      os.mkdir(PATH)
      return PATH
  else:
    shutil.rmtree(PATH)
    os.mkdir(PATH)
    return PATH
```

In [5]:
```
try:
    fruit_dir = make_dir(os.path.join(base_dir, 'fruit-dataset'))
    train_dir = make_dir(os.path.join(fruit_dir, 'train'))
    validation_dir = make_dir(os.path.join(fruit_dir, 'val'))
    test_dir = make_dir(os.path.join(fruit_dir, 'test'))
    preview_dir = make_dir(os.path.join(fruit_dir, 'preview'))

    train_fresh_apples_dir = make_dir(os.path.join(train_dir, 'Fresh Apples'))
    train_fresh_bananas_dir = make_dir(os.path.join(train_dir, 'Fresh Bananas'))
    train_fresh_oranges_dir = make_dir(os.path.join(train_dir, 'Fresh Oranges'))
    train_rotten_apples_dir = make_dir(os.path.join(train_dir, 'Rotten Apples'))
    train_rotten_bananas_dir = make_dir(os.path.join(train_dir, 'Rotten Bananas'))
    train_rotten_oranges_dir = make_dir(os.path.join(train_dir, 'Rotten Oranges'))

    validation_fresh_apples_dir = make_dir(os.path.join(validation_dir, 'Fresh Apples'))
    validation_fresh_bananas_dir = make_dir(os.path.join(validation_dir, 'Fresh Bananas'))
    validation_fresh_oranges_dir = make_dir(os.path.join(validation_dir, 'Fresh Oranges'))
    validation_rotten_apples_dir = make_dir(os.path.join(validation_dir, 'Rotten Apples'))
    validation_rotten_bananas_dir = make_dir(os.path.join(validation_dir, 'Rotten Bananas'))
    validation_rotten_oranges_dir = make_dir(os.path.join(validation_dir, 'Rotten Oranges'))

    test_fresh_apples_dir = make_dir(os.path.join(test_dir, 'Fresh Apples'))
    test_fresh_bananas_dir = make_dir(os.path.join(test_dir, 'Fresh Bananas'))
    test_fresh_oranges_dir = make_dir(os.path.join(test_dir, 'Fresh Oranges'))
    test_rotten_apples_dir = make_dir(os.path.join(test_dir, 'Rotten Apples'))
    test_rotten_bananas_dir = make_dir(os.path.join(test_dir, 'Rotten Bananas'))
    test_rotten_oranges_dir = make_dir(os.path.join(test_dir, 'Rotten Oranges'))

except OSError:
    pass
```

In [6]:
```
from PIL import Image

def split_data(SOURCE='', TRAINING='', VALIDATION='', SPLIT_SIZE=0):
  data = os.listdir(SOURCE)
  random_data = random.sample(data, len(data))

  train_size = len(data)*SPLIT_SIZE

  for i, filename in enumerate(random_data):
    filepath = os.path.join(SOURCE, filename)
    if os.path.getsize(filepath) > 0:
      if i < train_size:
        copyfile(filepath, os.path.join(TRAINING, filename))

      else:
        copyfile(filepath, os.path.join(VALIDATION, filename))
```

In [7]:
```
dataset_train_dir = base_dir
dataset_test_dir = base_dir

fapples_train_dir = os.path.join(dataset_train_dir, 'FreshApple')
fbananas_train_dir = os.path.join(dataset_train_dir, 'FreshBanana')
foranges_train_dir = os.path.join(dataset_train_dir, 'FreshOrange')
rapples_train_dir = os.path.join(dataset_train_dir, 'RottenApple')
rbananas_train_dir = os.path.join(dataset_train_dir, 'RottenBanana')
roranges_train_dir = os.path.join(dataset_train_dir, 'RottenOrange')

fapples_test_dir = os.path.join(dataset_test_dir, 'FreshApple')
fbananas_test_dir = os.path.join(dataset_test_dir, 'FreshBanana')
foranges_test_dir = os.path.join(dataset_test_dir, 'FreshOrange')
rapples_test_dir = os.path.join(dataset_test_dir, 'RottenApple')
rbananas_test_dir = os.path.join(dataset_test_dir, 'RottenBanana')
roranges_test_dir = os.path.join(dataset_test_dir, 'RottenOrange')

print('fapples_train images = ', len(os.listdir(fapples_train_dir)))
print('fbananas_train images = ', len(os.listdir(fbananas_train_dir)))
print('foranges_train images = ', len(os.listdir(foranges_train_dir)))
print('rapples_train images = ', len(os.listdir(rapples_train_dir)))
print('rbananas_train images = ', len(os.listdir(rbananas_train_dir)))
print('roranges_train images = ', len(os.listdir(roranges_train_dir)))
print()
print('fapples_test images = ', len(os.listdir(fapples_test_dir)))
print('fbananas_test images = ', len(os.listdir(fbananas_test_dir)))
print('foranges_test images = ', len(os.listdir(foranges_test_dir)))
print('rapples_test images = ', len(os.listdir(rapples_test_dir)))
print('rbananas_test images = ', len(os.listdir(rbananas_test_dir)))
print('roranges_test images = ', len(os.listdir(roranges_test_dir)))
```

```
fapples_train images =  200
fbananas_train images =  200
foranges_train images =  200
rapples_train images =  200
rbananas_train images =  200
roranges_train images =  200

fapples_test images =  200
fbananas_test images =  200
foranges_test images =  200
rapples_test images =  200
rbananas_test images =  200
roranges_test images =  200
```

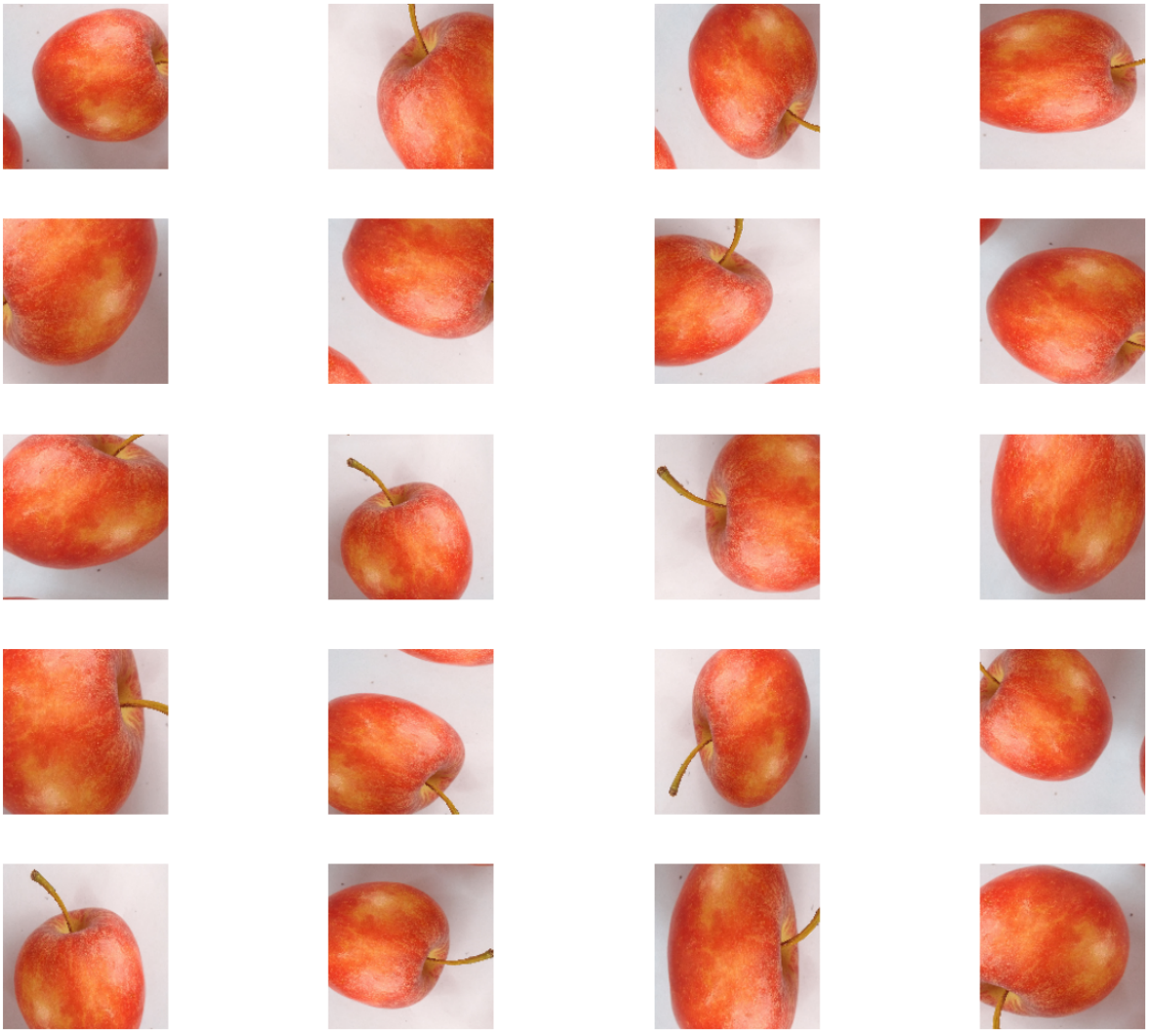
In [8]:
```
SPLIT_SIZE = 0.8
split_data(fapples_train_dir, train_fresh_apples_dir, validation_fresh_apples_dir, SPLIT_SIZE)
split_data(fbananas_train_dir, train_fresh_bananas_dir, validation_fresh_bananas_dir, SPLIT_SIZE)
split_data(foranges_train_dir, train_fresh_oranges_dir, validation_fresh_oranges_dir, SPLIT_SIZE)
split_data(rapples_train_dir, train_rotten_apples_dir, validation_rotten_apples_dir, SPLIT_SIZE)
split_data(rbananas_train_dir, train_rotten_bananas_dir, validation_rotten_bananas_dir, SPLIT_SIZE)
split_data(roranges_train_dir, train_rotten_oranges_dir, validation_rotten_oranges_dir, SPLIT_SIZE)

SPLIT_SIZE = 0.2
split_data(fapples_test_dir, test_fresh_apples_dir, validation_fresh_apples_dir, SPLIT_SIZE)
split_data(fbananas_test_dir, test_fresh_bananas_dir, validation_fresh_bananas_dir, SPLIT_SIZE)
split_data(foranges_test_dir, test_fresh_oranges_dir, validation_fresh_oranges_dir, SPLIT_SIZE)
split_data(rapples_test_dir, test_rotten_apples_dir, validation_rotten_apples_dir, SPLIT_SIZE)
split_data(rbananas_test_dir, test_rotten_bananas_dir, validation_rotten_bananas_dir, SPLIT_SIZE)
split_data(roranges_test_dir, test_rotten_oranges_dir, validation_rotten_oranges_dir, SPLIT_SIZE)
```

In [9]:
```
print(len(os.listdir('C:/Users/Abdoulla Yousef/Downloads/OriginalImage/Original Image/fruit-dataset/train/Fresh Apples/')))
print(len(os.listdir('C:/Users/Abdoulla Yousef/Downloads/OriginalImage/Original Image/fruit-dataset/train/Fresh Bananas/')))
print(len(os.listdir('C:/Users/Abdoulla Yousef/Downloads/OriginalImage/Original Image/fruit-dataset/train/Fresh Oranges/')))
print(len(os.listdir('C:/Users/Abdoulla Yousef/Downloads/OriginalImage/Original Image/fruit-dataset/train/Rotten Apples/')))
print(len(os.listdir('C:/Users/Abdoulla Yousef/Downloads/OriginalImage/Original Image/fruit-dataset/train/Rotten Bananas/')))
print(len(os.listdir('C:/Users/Abdoulla Yousef/Downloads/OriginalImage/Original Image/fruit-dataset/train/Rotten Oranges/')))
print()
print(len(os.listdir('C:/Users/Abdoulla Yousef/Downloads/OriginalImage/Original Image/fruit-dataset/val/Fresh Apples/')))
print(len(os.listdir('C:/Users/Abdoulla Yousef/Downloads/OriginalImage/Original Image/fruit-dataset/val/Fresh Bananas/')))
print(len(os.listdir('C:/Users/Abdoulla Yousef/Downloads/OriginalImage/Original Image/fruit-dataset/val/Fresh Oranges/')))
print(len(os.listdir('C:/Users/Abdoulla Yousef/Downloads/OriginalImage/Original Image/fruit-dataset/val/Rotten Apples/')))
print(len(os.listdir('C:/Users/Abdoulla Yousef/Downloads/OriginalImage/Original Image/fruit-dataset/val/Rotten Bananas/')))
print(len(os.listdir('C:/Users/Abdoulla Yousef/Downloads/OriginalImage/Original Image/fruit-dataset/val/Rotten Oranges/')))
print()
print(len(os.listdir('C:/Users/Abdoulla Yousef/Downloads/OriginalImage/Original Image/fruit-dataset/test/Fresh Apples/')))
print(len(os.listdir('C:/Users/Abdoulla Yousef/Downloads/OriginalImage/Original Image/fruit-dataset/test/Fresh Bananas/')))
print(len(os.listdir('C:/Users/Abdoulla Yousef/Downloads/OriginalImage/Original Image/fruit-dataset/test/Fresh Oranges/')))
print(len(os.listdir('C:/Users/Abdoulla Yousef/Downloads/OriginalImage/Original Image/fruit-dataset/test/Rotten Apples/')))
print(len(os.listdir('C:/Users/Abdoulla Yousef/Downloads/OriginalImage/Original Image/fruit-dataset/test/Rotten Bananas/')))
print(len(os.listdir('C:/Users/Abdoulla Yousef/Downloads/OriginalImage/Original Image/fruit-dataset/test/Rotten Oranges/')))
```

```
160
160
160
160
160
160

168
171
170
169
167
169

40
40
40
40
40
40
```

In [10]:
```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=[0.5, 1.0],
    rotation_range=90,
    horizontal_flip=True,
    vertical_flip=True,
    fill_mode='reflect'
)

validation_datagen = ImageDataGenerator(
    rescale=1./255
)
```

In [11]:
```
path_aug = os.path.join(train_fresh_apples_dir, os.listdir(train_fresh_apples_dir)[-1])
img_augmentation = image.load_img(path_aug)
x_aug = image.img_to_array(img_augmentation)
x_aug = x_aug.reshape((1,) + x_aug.shape)

i = 0
for batch in train_datagen.flow(x_aug, batch_size=1, save_to_dir=preview_dir, save_prefix='fruit', save_format='jpeg'):
    i += 1
    if i >= 20:
        break

preview_img = os.listdir(preview_dir)

plt.figure(figsize=(15, 15))
for n in range(len(preview_img)):
  plt.subplot(int(len(preview_img)/4)+1, 4, n+1)
  plt.subplots_adjust(hspace = 0.3)
  plt.imshow(image.load_img(os.path.join(preview_dir, preview_img[n]),
                            color_mode="rgb",
                            target_size=(150, 150),
                            interpolation="nearest"))
  plt.axis('off')
plt.show()

for fn in preview_img:
  os.system(f'rm {os.path.join(preview_dir, fn)}')
```



In [12]:
```
train_generator = train_datagen.flow_from_directory(train_dir,
                                                    batch_size=32,
                                                    color_mode="rgb",
                                                    # shuffle = False,
                                                    target_size=(150,150), #?
                                                    class_mode='categorical')

validation_generator = train_datagen.flow_from_directory(validation_dir,
                                                         batch_size=32,
                                                         color_mode="rgb",
                                                         # shuffle = False,
                                                         target_size=(150,150), #?
                                                         class_mode='categorical')
```

```
Found 960 images belonging to 6 classes.
Found 1014 images belonging to 6 classes.
```

In [13]:
```
class myCallback(tf.keras.callbacks.Callback):
  def on_epoch_end(self, epoch, logs={}):
    if(logs.get('accuracy') > 0.98):
      print("\nReached 98% accuracy. Stop Training")
      self.model.stop_training = True

callbacks = myCallback()
```

In [14]:
```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2,2),
    # tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    # tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(6, activation='softmax')
])

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 32)      896

 max_pooling2d (MaxPooling2D  (None, 74, 74, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 72, 72, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 36, 36, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 34, 34, 128)       73856

 max_pooling2d_2 (MaxPooling  (None, 17, 17, 128)      0
 2D)

 flatten (Flatten)           (None, 36992)             0

 dense (Dense)               (None, 512)               18940416

 dense_1 (Dense)             (None, 6)                 3078

=================================================================
Total params: 19,036,742
Trainable params: 19,036,742
Non-trainable params: 0
_________________________________________________________________
```

In [15]:
```
train_len = 0
for foldername in os.listdir('C:/Users/Abdoulla Yousef/Downloads/OriginalImage/Original Image/fruit-dataset/train'):
    train_len = train_len + len(os.listdir(os.path.join('C:/Users/Abdoulla Yousef/Downloads/OriginalImage/Original Image/fruit-dataset/train',foldername)))

val_len = 0
for foldername in os.listdir('C:/Users/Abdoulla Yousef/Downloads/OriginalImage/Original Image/fruit-dataset/val'):
    val_len = val_len + len(os.listdir(os.path.join('C:/Users/Abdoulla Yousef/Downloads/OriginalImage/Original Image/fruit-dataset/val',foldername)))

print(train_len)
print(val_len)
```

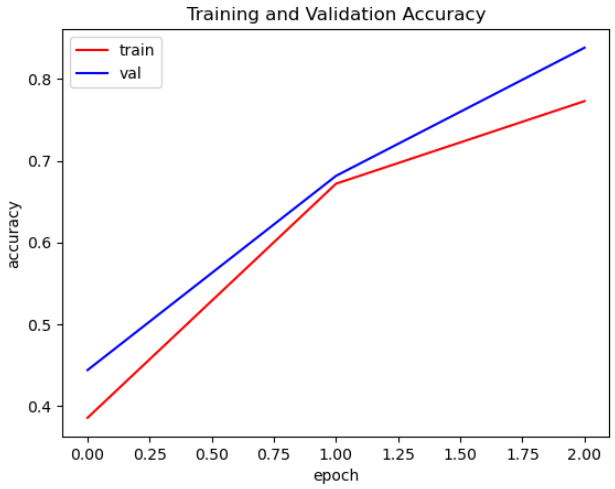
```
960
1014
```

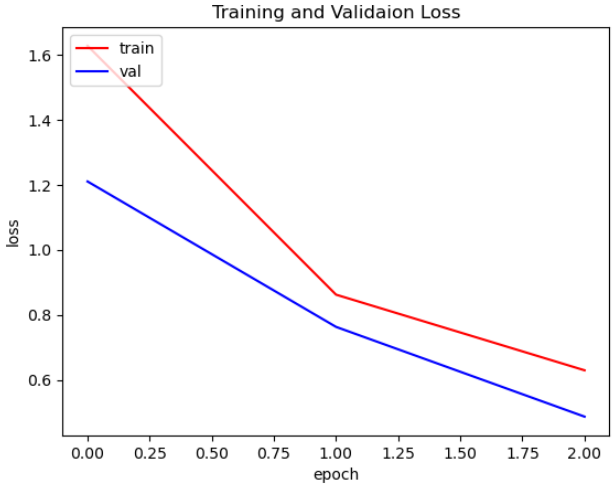
In [16]:
```
history = model.fit(
    train_generator,
    steps_per_epoch=(train_len/32),
    epochs=3,
    verbose=1,
    callbacks=[callbacks],
    validation_data=validation_generator,
    validation_steps=(val_len/32)
)
```

```
Epoch 1/3
30/30 [==============================] - 212s 7s/step - loss: 1.6272 - accuracy: 0.3854 - val_loss: 1.2101 - val_accuracy: 0.4438
Epoch 2/3
30/30 [==============================] - 203s 7s/step - loss: 0.8621 - accuracy: 0.6719 - val_loss: 0.7631 - val_accuracy: 0.6815
Epoch 3/3
30/30 [==============================] - 198s 7s/step - loss: 0.6297 - accuracy: 0.7729 - val_loss: 0.4872 - val_accuracy: 0.8383
```

In [17]:
```
%matplotlib inline
import matplotlib.image as mpimg
import matplotlib.pyplot as plt

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'r')
plt.plot(epochs, val_acc, 'b')
plt.title('Training and Validation Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.figure()

plt.plot(epochs, loss, 'r')
plt.plot(epochs, val_loss, 'b')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.title('Training and Validaion Loss')
plt.figure()
```

Out[17]:
```
<Figure size 640x480 with 0 Axes>
```




```
<Figure size 640x480 with 0 Axes>
```

In [18]:
```
test_datagen = ImageDataGenerator(rescale=1./255)

test_generator = test_datagen.flow_from_directory(test_dir,
                                                  batch_size=1,
                                                  target_size=(150, 150),
                                                  shuffle = False,
                                                  class_mode='categorical')

filenames = test_generator.filenames
nb_samples = len(filenames)

loss, acc = model.evaluate(test_generator,steps = (nb_samples), verbose=1)
print('accuracy test: ',acc)
```

```
Found 240 images belonging to 6 classes.
240/240 [==============================] - 48s 201ms/step - loss: 0.7429 - accuracy: 0.7375
accuracy test:  0.737500011920929
```

In [19]:
```
model.save('mymodel.h5')
```

In [ ]:
```

```