Very Deep Convolutional Networks for Large-Scale Image Recognition(VGG-19)

Assignment 16

Referral id: SIRSS1088
NAME: ONASVEE BANARSE
EMAIL : 2obanarse@gmail.com
COLLEGE: AISSMS IOIT
GitHub : https://github.com/ORION-22/RegexSoftware_ASSIGNMENT.git
Kaggle:https://www.kaggle.com/onasveebanarse

In [14]:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf

from keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Dropout,BatchNormalization,GlobalAverag
```

In [15]:

```
cifar100=tf.keras.datasets.cifar100
(X_train,Y_train),(X_test,Y_test)=cifar100.load_data()
```

In [16]:

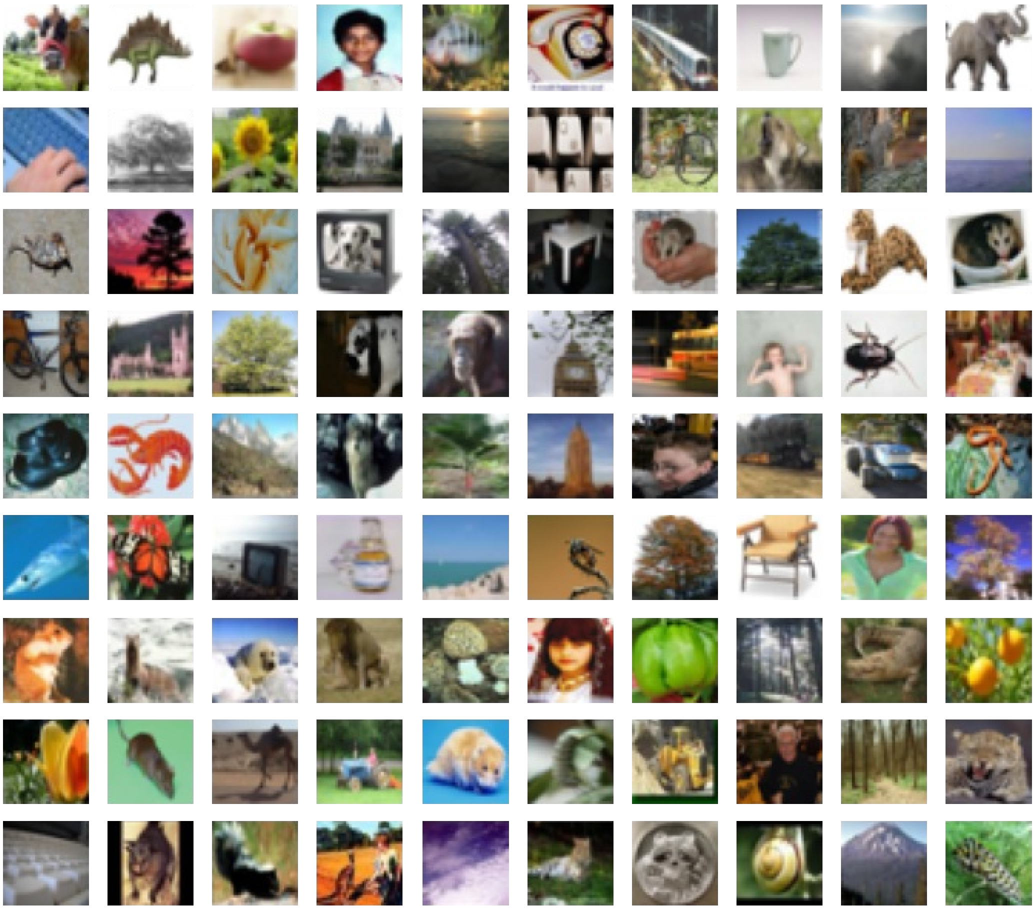
```
print(X_train.shape)
print(Y_train.shape)

print(X_test.shape)
print(Y_test.shape)
```

```
(50000, 32, 32, 3)
(50000, 1)
(10000, 32, 32, 3)
(10000, 1)
```

In [17]:

```
plt.figure(figsize = (12,12))
for i in range(100):
  plt.subplot(10,10,1+i)
  plt.axis('off')
  plt.imshow(X_train[i], cmap = 'gray')
```



In [18]:

```
from sklearn.model_selection import train_test_split
x_train,x_val,y_train,y_val=train_test_split(X_train, Y_train,test_size=0.2)
```

In [19]:

```
from keras.utils.np_utils import to_categorical

y_train=to_categorical(y_train,num_classes=100)
y_val=to_categorical(y_val,num_classes=100)
Y_test=to_categorical(Y_test,num_classes=100)
```

In [20]:

```
x_train = x_train*1.0/255
x_val = x_val*1.0/255
X_test = X_test*1.0/255
```

In [21]:

```
print(X_train.shape)
print(Y_train.shape)

print(x_train.shape)
print(y_train.shape)

print(x_val.shape)
print(y_val.shape)
```

```
(50000, 32, 32, 3)
(50000, 1)
(40000, 32, 32, 3)
(40000, 100)
(10000, 32, 32, 3)
(10000, 100)
```

In [22]:

```
train_datagen = ImageDataGenerator(
        rotation_range=10,
        zoom_range = 0.1,
        width_shift_range=0.1,
        height_shift_range=0.1,
        shear_range = 0.1,
        horizontal_flip=True,
        vertical_flip=False
        )
train_datagen.fit(x_train)
```

In [23]:

```
from keras.callbacks import ReduceLROnPlateau
learning_rate_reduction = ReduceLROnPlateau(
    monitor='val_accuracy',
    patience=3,
    verbose=1,
    factor=0.6,
    min_lr=1e-6)
```

In [24]:

```
vgg_model_1=tf.keras.applications.VGG19(
    include_top=False,
    weights='imagenet',
    input_shape=(32,32,3),
    classes=100
)
```

In [25]:

```
vgg_model_1.summary()
```

```
Model: "vgg19"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 32, 32, 3)]       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 32, 32, 64)        1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 32, 32, 64)        36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 16, 16, 64)        0
```

```
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 16, 16, 128)       73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 16, 16, 128)       147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 8, 8, 128)         0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 8, 8, 256)         295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 8, 8, 256)         590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 8, 8, 256)         590080    
_________________________________________________________________
block3_conv4 (Conv2D)        (None, 8, 8, 256)         590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 4, 4, 256)         0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 4, 4, 512)         1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 4, 4, 512)         2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 4, 4, 512)         2359808   
_________________________________________________________________
block4_conv4 (Conv2D)        (None, 4, 4, 512)         2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 2, 2, 512)         0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 2, 2, 512)         2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 2, 2, 512)         2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 2, 2, 512)         2359808   
_________________________________________________________________
block5_conv4 (Conv2D)        (None, 2, 2, 512)         2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 1, 1, 512)         0         
=================================================================
Total params: 20,024,384
Trainable params: 20,024,384
Non-trainable params: 0
_________________________________________________________________
```

In [26]:

```
model=tf.keras.models.Sequential()

model.add(vgg_model_1)
# model.add(Flatten())
# model.add(Dense(1024,activation='relu'))
# model.add(Dropout(0.25))
# model.add(Dense(512, activation='relu'))
# model.add(BatchNormalization())
# model.add(Dense(100, activation='softmax'))
model.add(GlobalAveragePooling2D())
model.add(Dropout(.25))
model.add(Dense(256, activation='relu'))
model.add(BatchNormalization())
model.add(Dense(100, activation='softmax'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
vgg19 (Functional)           (None, 1, 1, 512)         20024384  
_________________________________________________________________
global_average_pooling2d (Gl (None, 512)               0         
_________________________________________________________________
dropout (Dropout)            (None, 512)               0         
_________________________________________________________________
dense (Dense)                (None, 256)               131328    
_________________________________________________________________
batch_normalization (BatchNo (None, 256)               1024      
_________________________________________________________________
dense_1 (Dense)              (None, 100)               25700     
=================================================================
Total params: 20,182,436
Trainable params: 20,181,924
Non-trainable params: 512
_________________________________________________________________
```

In [27]:

In [27]:

```
optimizer = tf.keras.optimizers.SGD(learning_rate=1e-3, momentum=0.9)
```

In [28]:

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

In [29]:

```
history = model.fit(
    train_datagen.flow(x_train, y_train, batch_size = 128),
    validation_data = (x_val, y_val),
    epochs = 30,
    verbose = 1,
    callbacks =[learning_rate_reduction]
)
```

```
Epoch 1/30
313/313 [==============================] - 45s 116ms/step - loss: 4.4699 - accuracy: 0.0640 - val_loss: 3.0518 -
val_accuracy: 0.2579
Epoch 2/30
313/313 [==============================] - 36s 114ms/step - loss: 3.0704 - accuracy: 0.2500 - val_loss: 2.5471 -
val_accuracy: 0.3576
Epoch 3/30
313/313 [==============================] - 35s 113ms/step - loss: 2.5744 - accuracy: 0.3438 - val_loss: 2.1979 -
val_accuracy: 0.4260cy: 0.331 - ETA: 22s - loss: 2.6276 - accur - ETA: 21s - loss: 2.625 - ETA: 1s - loss: 2.5
Epoch 4/30
313/313 [==============================] - 35s 112ms/step - loss: 2.2867 - accuracy: 0.4064 - val_loss: 2.1469 -
val_accuracy: 0.4325
Epoch 5/30
313/313 [==============================] - 35s 113ms/step - loss: 2.1011 - accuracy: 0.4451 - val_loss: 2.1229 -
val_accuracy: 0.4437 - loss: 2.1923 - accuracy: 0 - ETA: 28s - loss: 2.1807 - accuracy: 0.41 - - ETA: 25 - ETA: 2
2s - loss: 2.1308 - accuracy: - ETA: 22s - loss: 2.1280 - accuracy - ETA: 21s - loss: 2.1259 - accur - ETA: 19s -
loss: 2.1237 - accu - ETA: 18s - loss: 2.1214 - accura - ETA: 12s - loss: 2.1147  - ETA: 10s - loss: - ETA: 6s -
loss: 2.1082 - accu - ETA: 3s - l - ETA: 2s - - ETA: 0s - loss: 2.1017 - accu
Epoch 6/30
313/313 [==============================] - 35s 113ms/step - loss: 1.9413 - accuracy: 0.4832 - val_loss: 2.0706 -
val_accuracy: 0.4586
Epoch 7/30
313/313 [==============================] - 35s 113ms/step - loss: 1.8397 - accuracy: 0.5038 - val_loss: 1.7739 -
val_accuracy: 0.5209- loss: 1.8397 - accu - E
Epoch 8/30
313/313 [==============================] - 35s 113ms/step - loss: 1.7553 - accuracy: 0.5218 - val_loss: 1.8426 -
val_accuracy: 0.5002
Epoch 9/30
313/313 [==============================] - 35s 113ms/step - loss: 1.6611 - accuracy: 0.5453 - val_loss: 1.7113 -
val_accuracy: 0.5338
Epoch 10/30
313/313 [==============================] - 35s 112ms/step - loss: 1.5981 - accuracy: 0.5587 - val_loss: 1.6089 -
val_accuracy: 0.5505
Epoch 11/30
313/313 [==============================] - 35s 111ms/step - loss: 1.5100 - accuracy: 0.5787 - val_loss: 1.7316 -
val_accuracy: 0.5305
Epoch 12/30
313/313 [==============================] - 35s 113ms/step - loss: 1.4789 - accuracy: 0.5876 - val_loss: 1.7425 -
val_accuracy: 0.5308
Epoch 13/30
313/313 [==============================] - 35s 111ms/step - loss: 1.4104 - accuracy: 0.6031 - val_loss: 1.5704 -
val_accuracy: 0.5697
Epoch 14/30
313/313 [==============================] - 35s 113ms/step - loss: 1.3593 - accuracy: 0.6144 - val_loss: 1.5528 -
val_accuracy: 0.5743
Epoch 15/30
313/313 [==============================] - 35s 111ms/step - loss: 1.2842 - accuracy: 0.6367 - val_loss: 1.5941 -
val_accuracy: 0.5663
Epoch 16/30
313/313 [==============================] - 36s 113ms/step - loss: 1.2491 - accuracy: 0.6449 - val_loss: 1.6448 -
val_accuracy: 0.5623loss: 1.2372 - accuracy: 0. - ETA: 17s - loss: 1.2374 - accuracy: 0. - - ETA: 8s - loss: 1.24
23 -  - ETA: 3s - los - ETA: 1s
Epoch 17/30
313/313 [==============================] - 35s 111ms/step - loss: 1.2041 - accuracy: 0.6528 - val_loss: 1.6076 -
val_accuracy: 0.5626

Epoch 00017: ReduceLROnPlateau reducing learning rate to 0.0006000000284984708.
Epoch 18/30
313/313 [==============================] - 35s 113ms/step - loss: 1.1016 - accuracy: 0.6842 - val_loss: 1.4829 -
val_accuracy: 0.5926s - loss: 1.1044 - accuracy - ETA: 4s - loss: 1.1042 - accu - ETA: 4s - loss: 1.1037 - accura
- ETA: 3s - l
Epoch 19/30
313/313 [==============================] - 35s 112ms/step - loss: 1.0411 - accuracy: 0.6984 - val_loss: 1.4965 -
val_accuracy: 0.59579s - loss: 1.0605 -  - ETA: 17s - loss: 1.0417 - ETA: 15s - loss: 1.0415 - accuracy - ETA: 14
s - loss: 1.0414 - accurac - ETA: 6s - loss: 1.0411 - accuracy: 0.69 - ETA: 6s - loss: 1.0411  - ETA: 5s - ETA: 1
s - loss: 1.0408 - accuracy - ETA: 1s -
```

```
Epoch 20/30
313/313 [==============================] - 35s 113ms/step - loss: 0.9879 - accuracy: 0.7142 - val_loss: 1.5217 -
val_accuracy: 0.5908- ETA: 2 - ETA: 19s - loss: 0.9780 - accuracy: 0.717 - ETA: 19s - loss - ETA: 16s - loss: 0.9
800 - accuracy: - ETA: 15s - loss: 0.9803 - accur - ETA: 14s - loss:  - ET
Epoch 21/30
313/313 [==============================] - 35s 112ms/step - loss: 0.9593 - accuracy: 0.7218 - val_loss: 1.5036 -
val_accuracy: 0.5953A: 32s  - ET
Epoch 22/30
313/313 [==============================] - 35s 113ms/step - loss: 0.9289 - accuracy: 0.7301 - val_loss: 1.4647 -
val_accuracy: 0.6034
Epoch 23/30
313/313 [==============================] - 35s 113ms/step - loss: 0.8971 - accuracy: 0.7361 - val_loss: 1.4981 -
val_accuracy: 0.5994
Epoch 24/30
313/313 [==============================] - 35s 113ms/step - loss: 0.8795 - accuracy: 0.7433 - val_loss: 1.4929 -
val_accuracy: 0.6018 0.8783 - accuracy: 0 - ETA: 19s - loss: 0 - ETA: 17s - ETA: 9s - loss: 0.8776  - ETA: 8s - l
oss: 0.8779 -  - ETA - ETA: 5s - loss: - ETA: 2s - ETA: 0s - loss: 0.8794 - accu
Epoch 25/30
313/313 [==============================] - 35s 112ms/step - loss: 0.8439 - accuracy: 0.7512 - val_loss: 1.5364 -
val_accuracy: 0.5912

Epoch 00025: ReduceLROnPlateau reducing learning rate to 0.0003600000170990825.
Epoch 26/30
313/313 [==============================] - 36s 114ms/step - loss: 0.7703 - accuracy: 0.7754 - val_loss: 1.4764 -
val_accuracy: 0.6083
Epoch 27/30
313/313 [==============================] - 35s 113ms/step - loss: 0.7468 - accuracy: 0.7813 - val_loss: 1.4496 -
val_accuracy: 0.61680.7478 - accuracy: 0. - ETA: 18s - loss: 0.7480 -  - ETA: 17s  - ETA: 4s - loss: 0.747 - ETA:
3s - loss: 0.7470 - accuracy:  - ETA: 3s - loss: 0.747
Epoch 28/30
313/313 [==============================] - 35s 112ms/step - loss: 0.7216 - accuracy: 0.7906 - val_loss: 1.5099 -
val_accuracy: 0.6083
Epoch 29/30
313/313 [==============================] - 35s 113ms/step - loss: 0.6903 - accuracy: 0.8022 - val_loss: 1.5445 -
val_accuracy: 0.6044
Epoch 30/30
313/313 [==============================] - 35s 112ms/step - loss: 0.6802 - accuracy: 0.8007 - val_loss: 1.5440 -
val_accuracy: 0.6034ETA: 0s - loss: 0.6802 - accuracy

Epoch 00030: ReduceLROnPlateau reducing learning rate to 0.00021600000327453016.
```

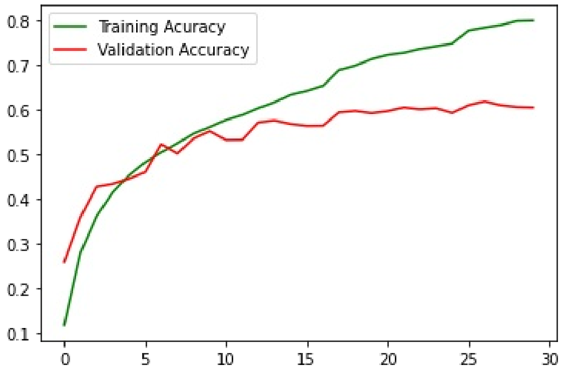
In [30]:

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

plt.figure()
plt.plot(acc,color = 'green',label = 'Training Acuracy')
plt.plot(val_acc,color = 'red',label = 'Validation Accuracy')
plt.legend()
```

Out[30]:

```
<matplotlib.legend.Legend at 0x7f83dc7b1490>
```



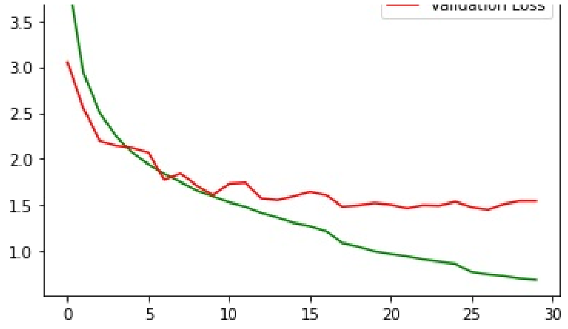
In [31]:

```
loss = history.history['loss']
val_loss = history.history['val_loss']

plt.figure()
plt.plot(loss,color = 'green',label = 'Training Loss')
plt.plot(val_loss,color = 'red',label = 'Validation Loss')
plt.legend()
```

Out[31]:

```
<matplotlib.legend.Legend at 0x7f83dc764c10>
```



In [32]:

```
y_pred = model.predict_classes(X_test)
y_true = np.argmax(Y_test, axis = 1)
print(y_pred.shape)
print(y_true.shape)
```

```
/opt/conda/lib/python3.7/site-packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning: `model.pred
ict_classes()` is deprecated and will be removed after 2021-01-01. Please use instead:* `np.argmax(model.predict(
x), axis=-1)`,   if your model does multi-class classification   (e.g. if it uses a `softmax` last-layer activati
on).* `(model.predict(x) > 0.5).astype("int32")`,   if your model does binary classification   (e.g. if it uses a
`sigmoid` last-layer activation).
  warnings.warn('`model.predict_classes()` is deprecated and '
(10000,)
(10000,)
```

In [33]:

```
from sklearn.metrics import confusion_matrix, accuracy_score
print("Testing Accuracy: ", accuracy_score(y_true,y_pred))
```

```
Testing Accuracy:  0.6059
```

In [34]:

```
from sklearn.metrics import confusion_matrix, accuracy_score
print("Testing Accuracy: ", accuracy_score(y_true,y_pred))
cm = confusion_matrix(y_true,y_pred)
cm
```

```
Testing Accuracy:  0.6059
```

Out[34]:

```
array([[79,  0,  0, ...,  0,  0,  0],
       [ 0, 63,  2, ...,  0,  0,  0],
       [ 1,  0, 52, ...,  0,  6,  0],
       ...,
       [ 0,  0,  0, ..., 48,  0,  1],
       [ 0,  0,  9, ...,  0, 30,  0],
       [ 0,  0,  0, ...,  0,  0, 72]])
```

In [ ]:

In [47]:

```
from sklearn.model_selection import train_test_split
x_train,x_val,y_train,y_val=train_test_split(X_train, Y_train,test_size=0.2)

from keras.utils.np_utils import to_categorical

y_train=to_categorical(y_train,num_classes=100)
y_val=to_categorical(y_val,num_classes=100)
Y_test=to_categorical(Y_test,num_classes=100)

x_train = x_train*1.0/255
x_val = x_val*1.0/255
X_test = X_test*1.0/255


vgg_model_2=tf.keras.applications.VGG19(
    include_top=False,
    weights='imagenet',
    input_shape=(32,32,3),
    classes=100
)

# for layer in vgg_model_2.layers:
#     layer.trainable = False
```

In [48]:

```
model_1=tf.keras.models.Sequential()

model_1.add(vgg_model_2)
model_1.add(Flatten())
model_1.add(Dense(1024,activation='relu'))
model_1.add(Dropout(0.25))
model_1.add(Dense(512, activation='relu'))
model_1.add(BatchNormalization())
model_1.add(Dense(100, activation='softmax'))

model_1.compile(
    optimizer = optimizer,
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
model_1.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
vgg19 (Functional)           (None, 1, 1, 512)         20024384  
_________________________________________________________________
flatten_5 (Flatten)          (None, 512)               0         
_________________________________________________________________
dense_17 (Dense)             (None, 1024)              525312    
_________________________________________________________________
dropout_6 (Dropout)          (None, 1024)              0         
_________________________________________________________________
dense_18 (Dense)             (None, 512)               524800    
_________________________________________________________________
batch_normalization_6 (Batch (None, 512)               2048      
_________________________________________________________________
dense_19 (Dense)             (None, 100)               51300     
=================================================================
Total params: 21,127,844
Trainable params: 21,126,820
Non-trainable params: 1,024
_________________________________________________________________
```

In [50]:

```
history_1= model_1.fit(
    train_datagen.flow(x_train, y_train, batch_size = 128),
    validation_data = (x_val, y_val),
    epochs = 30,
    verbose = 1,
    callbacks =[learning_rate_reduction]
)
```

```
Epoch 1/30
313/313 [==============================] - 34s 110ms/step - loss: 4.0660 - accuracy: 0.1067 - val_loss: 3.4763 -
val_accuracy: 0.2023
Epoch 2/30
313/313 [==============================] - 35s 111ms/step - loss: 3.3740 - accuracy: 0.2029 - val_loss: 2.8887 -
val_accuracy: 0.2926
Epoch 3/30
313/313 [==============================] - 34s 110ms/step - loss: 2.9810 - accuracy: 0.2702 - val_loss: 2.6089 -
val_accuracy: 0.3429
Epoch 4/30
313/313 [==============================] - 35s 112ms/step - loss: 2.7219 - accuracy: 0.3159 - val_loss: 2.4671 -
val_accuracy: 0.3676
Epoch 5/30
313/313 [==============================] - 34s 109ms/step - loss: 2.5399 - accuracy: 0.3529 - val_loss: 2.4101 -
val_accuracy: 0.3795
Epoch 6/30
313/313 [==============================] - 35s 112ms/step - loss: 2.3956 - accuracy: 0.3809 - val_loss: 2.2810 -
val_accuracy: 0.3997
Epoch 7/30
313/313 [==============================] - 35s 113ms/step - loss: 2.2781 - accuracy: 0.4022 - val_loss: 2.1573 -
val_accuracy: 0.4305
Epoch 8/30
313/313 [==============================] - 34s 110ms/step - loss: 2.1775 - accuracy: 0.4264 - val_loss: 2.0392 -
val_accuracy: 0.4577
Epoch 9/30
313/313 [==============================] - 35s 112ms/step - loss: 2.0973 - accuracy: 0.4478 - val_loss: 2.0746 -
val_accuracy: 0.4505
Epoch 10/30
```

```
313/313 [==============================] - 35s 111ms/step - loss: 2.0273 - accuracy: 0.4602 - val_loss: 1.9870 -
val_accuracy: 0.4651
Epoch 11/30
313/313 [==============================] - 35s 111ms/step - loss: 1.9672 - accuracy: 0.4717 - val_loss: 1.8890 -
val_accuracy: 0.4848- loss: 1.9726 - accuracy:  - ETA: 28s -
Epoch 12/30
313/313 [==============================] - 35s 111ms/step - loss: 1.9100 - accuracy: 0.4862 - val_loss: 1.8903 -
val_accuracy: 0.4848
Epoch 13/30
313/313 [==============================] - 35s 111ms/step - loss: 1.8547 - accuracy: 0.4986 - val_loss: 1.8446 -
val_accuracy: 0.5000
Epoch 14/30
313/313 [==============================] - 35s 112ms/step - loss: 1.7971 - accuracy: 0.5152 - val_loss: 1.8412 -
val_accuracy: 0.4993
Epoch 15/30
313/313 [==============================] - 35s 111ms/step - loss: 1.7594 - accuracy: 0.5211 - val_loss: 1.8355 -
val_accuracy: 0.4992
Epoch 16/30
313/313 [==============================] - 35s 112ms/step - loss: 1.7149 - accuracy: 0.5289 - val_loss: 1.9143 -
val_accuracy: 0.4888

Epoch 00016: ReduceLROnPlateau reducing learning rate to 0.00012960000021848827.
Epoch 17/30
313/313 [==============================] - 35s 113ms/step - loss: 1.6413 - accuracy: 0.5506 - val_loss: 1.7438 -
val_accuracy: 0.5215
Epoch 18/30
313/313 [==============================] - 35s 111ms/step - loss: 1.6160 - accuracy: 0.5547 - val_loss: 1.7485 -
val_accuracy: 0.5243
Epoch 19/30
313/313 [==============================] - 36s 114ms/step - loss: 1.5924 - accuracy: 0.5611 - val_loss: 1.7951 -
val_accuracy: 0.5151
Epoch 20/30
313/313 [==============================] - 35s 111ms/step - loss: 1.5667 - accuracy: 0.5678 - val_loss: 1.6754 -
val_accuracy: 0.5390
Epoch 21/30
313/313 [==============================] - 35s 112ms/step - loss: 1.5532 - accuracy: 0.5700 - val_loss: 1.7633 -
val_accuracy: 0.5201
Epoch 22/30
313/313 [==============================] - 34s 108ms/step - loss: 1.5268 - accuracy: 0.5768 - val_loss: 1.7691 -
val_accuracy: 0.5210
Epoch 23/30
313/313 [==============================] - 35s 113ms/step - loss: 1.4988 - accuracy: 0.5829 - val_loss: 1.7047 -
val_accuracy: 0.5341

Epoch 00023: ReduceLROnPlateau reducing learning rate to 7.775999838486313e-05.
Epoch 24/30
313/313 [==============================] - 35s 110ms/step - loss: 1.4572 - accuracy: 0.5942 - val_loss: 1.6426 -
val_accuracy: 0.5473
Epoch 25/30
313/313 [==============================] - 35s 112ms/step - loss: 1.4435 - accuracy: 0.5981 - val_loss: 1.6584 -
val_accuracy: 0.5407
Epoch 26/30
313/313 [==============================] - 34s 108ms/step - loss: 1.4277 - accuracy: 0.6029 - val_loss: 1.6607 -
val_accuracy: 0.5433
Epoch 27/30
313/313 [==============================] - 35s 111ms/step - loss: 1.4148 - accuracy: 0.6060 - val_loss: 1.7710 -
val_accuracy: 0.5248

Epoch 00027: ReduceLROnPlateau reducing learning rate to 4.6655999904032795e-05.
Epoch 28/30
313/313 [==============================] - 35s 112ms/step - loss: 1.3903 - accuracy: 0.6086 - val_loss: 1.6150 -
val_accuracy: 0.5530
Epoch 29/30
313/313 [==============================] - 35s 112ms/step - loss: 1.3790 - accuracy: 0.6146 - val_loss: 1.6400 -
val_accuracy: 0.545917 -   - ETA: 1s -
Epoch 30/30
313/313 [==============================] - 35s 112ms/step - loss: 1.3703 - accuracy: 0.6193 - val_loss: 1.6344 -
val_accuracy: 0.5519
```

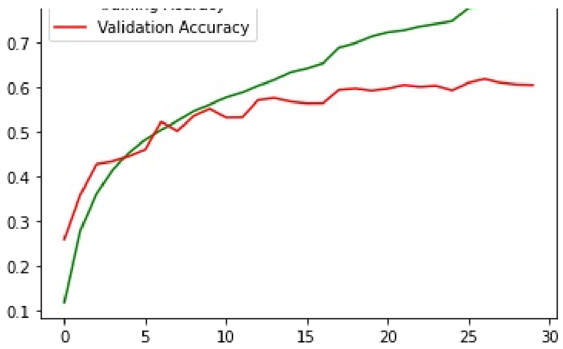
In [51]:

```python
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

plt.figure()
plt.plot(acc,color = 'green',label = 'Training Acuracy')
plt.plot(val_acc,color = 'red',label = 'Validation Accuracy')
plt.legend()
```

Out[51]:

```
<matplotlib.legend.Legend at 0x7f83dc646890>
```



```
In [ ]:
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js