{  
 "nbformat": *4*,  
 "nbformat\_minor": *0*,  
 "metadata": {  
 "accelerator": "GPU",  
 "colab": {  
 "name": "VGG19Model\_(CIFAR100).ipynb",  
 "provenance": []  
 },  
 "kernelspec": {  
 "display\_name": "Python 3",  
 "language": "python",  
 "name": "python3"  
 },  
 "language\_info": {  
 "codemirror\_mode": {  
 "name": "ipython",  
 "version": *3* },  
 "file\_extension": ".py",  
 "mimetype": "text/x-python",  
 "name": "python",  
 "nbconvert\_exporter": "python",  
 "pygments\_lexer": "ipython3",  
 "version": "3.8.5"  
 }  
 },  
 "cells": [  
 {  
 "cell\_type": "markdown",  
 "metadata": {  
 "id": "qN\_Sprsqsvg9"  
 },  
 "source": [  
 "Import libraries"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "metadata": {  
 "id": "WJbVE-AVaGpA"  
 },  
 "source": [  
 "import numpy as np*\n*",  
 "import matplotlib.pyplot as plt*\n*",  
 "import seaborn as sns*\n*",  
 "import tensorflow as tf*\n*",  
 "*\n*",  
 "from tensorflow.keras.optimizers import RMSprop*\n*",  
 "from keras.preprocessing import image*\n*",  
 "from tensorflow.keras.preprocessing.image import ImageDataGenerator*\n*",  
 "from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Dropout, BatchNormalization*\n*",  
 "*\n*",  
 "%matplotlib inline"  
 ],  
 "execution\_count": *null*,  
 "outputs": []  
 },  
 {  
 "cell\_type": "markdown",  
 "metadata": {  
 "id": "nxgdjVg0suGK"  
 },  
 "source": [  
 "Extract data and train and test dataset"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "metadata": {  
 "id": "wRRU6txYaZ1m"  
 },  
 "source": [  
 "cifar100 = tf.keras.datasets.cifar100*\n*",  
 "(X\_train,Y\_train) , (X\_test,Y\_test) = cifar10.load\_data()"  
 ],  
 "execution\_count": *null*,  
 "outputs": []  
 },  
 {  
 "cell\_type": "code",  
 "metadata": {  
 "id": "OrVy37gUa4Jc"  
 },  
 "source": [  
 "classes = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']"  
 ],  
 "execution\_count": *null*,  
 "outputs": []  
 },  
 {  
 "cell\_type": "markdown",  
 "metadata": {  
 "id": "vI8qStc7uGvO"  
 },  
 "source": [  
 "Let's look into the dataset images"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "metadata": {  
 "colab": {  
 "base\_uri": "https://localhost:8080/",  
 "height": *901* },  
 "id": "Tgzh-Ot\_bCh0",  
 "outputId": "13e00d6a-1d11-4196-a583-8e0cf8d0d2ce"  
 },  
 "source": [  
 "plt.figure(figsize = (16,16))*\n*",  
 "for i in range(100):*\n*",  
 " plt.subplot(10,10,1+i)*\n*",  
 " plt.axis('off')*\n*",  
 " plt.imshow(X\_train[i], cmap = 'gray')"  
 ],  
 "execution\_count": *null*,  
 "outputs": [  
 {  
 "output\_type": "display\_data",  
 "data": {  
 "image/png": "\n",  
 "text/plain": [  
 "<Figure size *1152*x*1152* with *100* Axes>"  
 ]  
 },  
 "metadata": {  
 "tags": [],  
 "needs\_background": "light"  
 }  
 }  
 ]  
 },  
 {  
 "cell\_type": "markdown",  
 "metadata": {  
 "id": "RwaCXlELuXUL"  
 },  
 "source": [  
 "Training , Validating and Splitting trained and tested data"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "metadata": {  
 "id": "oWB7JmUKbWkQ"  
 },  
 "source": [  
 "from sklearn.model\_selection import train\_test\_split\n",  
 "x\_train, x\_val, y\_train, y\_val = train\_test\_split(X\_train,Y\_train,test\_size=0.2)"  
 ],  
 "execution\_count": null,  
 "outputs": []  
 },  
 {  
 "cell\_type": "code",  
 "metadata": {  
 "id": "WQorTIBHb6Z2"  
 },  
 "source": [  
 "from keras.utils.np\_utils import to\_categorical\n",  
 "y\_train = to\_categorical(y\_train, num\_classes = 10)\n",  
 "y\_val = to\_categorical(y\_val, num\_classes = 10)"  
 ],  
 "execution\_count": null,  
 "outputs": []  
 },  
 {  
 "cell\_type": "code",  
 "metadata": {  
 "colab": {  
 "base\_uri": "https://localhost:8080/"  
 },  
 "id": "rmR3CUmscaOR",  
 "outputId": "98f019ce-2f94-449b-f218-31a981e9cead"  
 },  
 "source": [  
 "print(x\_train.shape)*\n*",  
 "print(y\_train.shape)*\n*",  
 "print(x\_val.shape)*\n*",  
 "print(y\_val.shape)*\n*",  
 "print(X\_test.shape)*\n*",  
 "print(Y\_test.shape)"  
 ],  
 "execution\_count": *null*,  
 "outputs": [  
 {  
 "output\_type": "stream",  
 "text": [  
 "(40000, 32, 32, 3)*\n*",  
 "(40000, 10)*\n*",  
 "(10000, 32, 32, 3)*\n*",  
 "(10000, 10)*\n*",  
 "(10000, 32, 32, 3)*\n*",  
 "(10000, 1)*\n*"  
 ],  
 "name": "stdout"  
 }  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "metadata": {  
 "id": "JWuVD37zcdaH"  
 },  
 "source": [  
 "train\_datagen = ImageDataGenerator(*\n*",  
 " preprocessing\_function = tf.keras.applications.vgg19.preprocess\_input,*\n*",  
 " rotation\_range=10,*\n*",  
 " zoom\_range = 0.1,*\n*",  
 " width\_shift\_range = 0.1,*\n*",  
 " height\_shift\_range = 0.1,*\n*",  
 " shear\_range = 0.1,*\n*",  
 " horizontal\_flip = True*\n*",  
 ")*\n*",  
 "train\_datagen.fit(x\_train)*\n*",  
 "*\n*",  
 "val\_datagen = ImageDataGenerator(preprocessing\_function = tf.keras.applications.vgg19.preprocess\_input)*\n*",  
 "val\_datagen.fit(x\_val)"  
 ],  
 "execution\_count": *null*,  
 "outputs": []  
 },  
 {  
 "cell\_type": "code",  
 "metadata": {  
 "id": "Ym3AEEqneTeA"  
 },  
 "source": [  
 "from keras.callbacks import ReduceLROnPlateau*\n*",  
 "learning\_rate\_reduction = ReduceLROnPlateau(monitor='val\_accuracy', *\n*",  
 " patience=3, *\n*",  
 " verbose=1, *\n*",  
 " factor=0.5, *\n*",  
 " min\_lr=0.00001)"  
 ],  
 "execution\_count": *null*,  
 "outputs": []  
 },  
 {  
 "cell\_type": "markdown",  
 "metadata": {  
 "id": "Sfw8mMkbyCJl"  
 },  
 "source": [  
 "We have used only 16 layers out of 19 layers in the CNN"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "metadata": {  
 "colab": {  
 "base\_uri": "https://localhost:8080/"  
 },  
 "id": "imMzq9dmej6u",  
 "outputId": "72a9bf94-0e02-4803-dcf8-b2eaac689122"  
 },  
 "source": [  
 "vgg\_model = tf.keras.applications.VGG19(*\n*",  
 " include\_top=False,*\n*",  
 " weights=*\"*imagenet*\"*,*\n*",  
 " input\_shape=(32,32,3),*\n*",  
 ")*\n*",  
 "*\n*",  
 "vgg\_model.summary()"  
 ],  
 "execution\_count": *null*,  
 "outputs": [  
 {  
 "output\_type": "stream",  
 "text": [  
 "Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19\_weights\_tf\_dim\_ordering\_tf\_kernels\_notop.h5*\n*",  
 "80142336/80134624 [==============================] - 1s 0us/step*\n*",  
 "Model: *\"*vgg19*\"\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*",  
 "Layer (type) Output Shape Param # *\n*",  
 "=================================================================*\n*",  
 "input\_1 (InputLayer) [(None, 32, 32, 3)] 0 *\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*",  
 "block1\_conv1 (Conv2D) (None, 32, 32, 64) 1792 *\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*",  
 "block1\_conv2 (Conv2D) (None, 32, 32, 64) 36928 *\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*",  
 "block1\_pool (MaxPooling2D) (None, 16, 16, 64) 0 *\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*",  
 "block2\_conv1 (Conv2D) (None, 16, 16, 128) 73856 *\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*",  
 "block2\_conv2 (Conv2D) (None, 16, 16, 128) 147584 *\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*",  
 "block2\_pool (MaxPooling2D) (None, 8, 8, 128) 0 *\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*",  
 "block3\_conv1 (Conv2D) (None, 8, 8, 256) 295168 *\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*",  
 "block3\_conv2 (Conv2D) (None, 8, 8, 256) 590080 *\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*",  
 "block3\_conv3 (Conv2D) (None, 8, 8, 256) 590080 *\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*",  
 "block3\_conv4 (Conv2D) (None, 8, 8, 256) 590080 *\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*",  
 "block3\_pool (MaxPooling2D) (None, 4, 4, 256) 0 *\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*",  
 "block4\_conv1 (Conv2D) (None, 4, 4, 512) 1180160 *\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*",  
 "block4\_conv2 (Conv2D) (None, 4, 4, 512) 2359808 *\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*",  
 "block4\_conv3 (Conv2D) (None, 4, 4, 512) 2359808 *\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*",  
 "block4\_conv4 (Conv2D) (None, 4, 4, 512) 2359808 *\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*",  
 "block4\_pool (MaxPooling2D) (None, 2, 2, 512) 0 *\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*",  
 "block5\_conv1 (Conv2D) (None, 2, 2, 512) 2359808 *\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*",  
 "block5\_conv2 (Conv2D) (None, 2, 2, 512) 2359808 *\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*",  
 "block5\_conv3 (Conv2D) (None, 2, 2, 512) 2359808 *\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*",  
 "block5\_conv4 (Conv2D) (None, 2, 2, 512) 2359808 *\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*",  
 "block5\_pool (MaxPooling2D) (None, 1, 1, 512) 0 *\n*",  
 "=================================================================*\n*",  
 "Total params: 20,024,384*\n*",  
 "Trainable params: 20,024,384*\n*",  
 "Non-trainable params: 0*\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*"  
 ],  
 "name": "stdout"  
 }  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "metadata": {  
 "colab": {  
 "base\_uri": "https://localhost:8080/"  
 },  
 "id": "PdpYiRhFhZ3U",  
 "outputId": "486959e1-bf59-4e74-b3df-e40d1e0bc085"  
 },  
 "source": [  
 "model = tf.keras.Sequential()*\n*",  
 "model.add(vgg\_model)*\n*",  
 "model.add(Flatten())*\n*",  
 "model.add(Dense(1024, activation = 'relu'))*\n*",  
 "model.add(Dense(1024, activation = 'relu'))*\n*",  
 "model.add(Dense(256, activation = 'relu'))*\n*",  
 "model.add(Dense(10, activation = 'softmax'))*\n*",  
 "*\n*",  
 "model.summary()"  
 ],  
 "execution\_count": *null*,  
 "outputs": [  
 {  
 "output\_type": "stream",  
 "text": [  
 "Model: *\"*sequential*\"\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*",  
 "Layer (type) Output Shape Param # *\n*",  
 "=================================================================*\n*",  
 "vgg19 (Functional) (None, 1, 1, 512) 20024384 *\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*",  
 "flatten (Flatten) (None, 512) 0 *\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*",  
 "dense (Dense) (None, 1024) 525312 *\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*",  
 "dense\_1 (Dense) (None, 1024) 1049600 *\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*",  
 "dense\_2 (Dense) (None, 256) 262400 *\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*",  
 "dense\_3 (Dense) (None, 10) 2570 *\n*",  
 "=================================================================*\n*",  
 "Total params: 21,864,266*\n*",  
 "Trainable params: 21,864,266*\n*",  
 "Non-trainable params: 0*\n*",  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\n*"  
 ],  
 "name": "stdout"  
 }  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "metadata": {  
 "colab": {  
 "base\_uri": "https://localhost:8080/"  
 },  
 "id": "2r3HsiQVjLhE",  
 "outputId": "2b37a085-b924-498d-e2f6-32d30b11955f"  
 },  
 "source": [  
 "optimizer = tf.keras.optimizers.SGD(lr = 0.001, momentum = 0.9)*\n*",  
 "model.compile(optimizer= optimizer,*\n*",  
 " loss='categorical\_crossentropy',*\n*",  
 " metrics=['accuracy'])"  
 ],  
 "execution\_count": *null*,  
 "outputs": [  
 {  
 "output\_type": "stream",  
 "text": [  
 "/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/optimizer\_v2/optimizer\_v2.py:375: UserWarning: The `lr` argument is deprecated, use `learning\_rate` instead.*\n*",  
 " *\"*The `lr` argument is deprecated, use `learning\_rate` instead.*\"*)*\n*"  
 ],  
 "name": "stderr"  
 }  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "metadata": {  
 "colab": {  
 "base\_uri": "https://localhost:8080/"  
 },  
 "id": "IqKsNoKejwF9",  
 "outputId": "a25fa5b1-dc80-477a-cdfc-da166059c349"  
 },  
 "source": [  
 "history = model.fit(*\n*",  
 " train\_datagen.flow(x\_train, y\_train, batch\_size = 128),*\n*",  
 " validation\_data = val\_datagen.flow(x\_val,y\_val, batch\_size = 128),*\n*",  
 " epochs = 25,*\n*",  
 " verbose = 1,*\n*",  
 " callbacks = [learning\_rate\_reduction]*\n*",  
 ")"  
 ],  
 "execution\_count": *null*,  
 "outputs": [  
 {  
 "output\_type": "stream",  
 "text": [  
 "Epoch 1/25*\n*",  
 "313/313 [==============================] - 84s 165ms/step - loss: 1.6671 - accuracy: 0.3875 - val\_loss: 1.1241 - val\_accuracy: 0.6056*\n*",  
 "Epoch 2/25*\n*",  
 "313/313 [==============================] - 50s 159ms/step - loss: 0.9631 - accuracy: 0.6640 - val\_loss: 0.7466 - val\_accuracy: 0.7404*\n*",  
 "Epoch 3/25*\n*",  
 "313/313 [==============================] - 50s 160ms/step - loss: 0.7464 - accuracy: 0.7430 - val\_loss: 0.6792 - val\_accuracy: 0.7716*\n*",  
 "Epoch 4/25*\n*",  
 "313/313 [==============================] - 50s 160ms/step - loss: 0.6533 - accuracy: 0.7782 - val\_loss: 0.6814 - val\_accuracy: 0.7829*\n*",  
 "Epoch 5/25*\n*",  
 "313/313 [==============================] - 50s 159ms/step - loss: 0.5779 - accuracy: 0.8013 - val\_loss: 0.5932 - val\_accuracy: 0.8058*\n*",  
 "Epoch 6/25*\n*",  
 "313/313 [==============================] - 50s 160ms/step - loss: 0.5369 - accuracy: 0.8136 - val\_loss: 0.5455 - val\_accuracy: 0.8157*\n*",  
 "Epoch 7/25*\n*",  
 "313/313 [==============================] - 50s 160ms/step - loss: 0.4925 - accuracy: 0.8299 - val\_loss: 0.5119 - val\_accuracy: 0.8269*\n*",  
 "Epoch 8/25*\n*",  
 "313/313 [==============================] - 50s 160ms/step - loss: 0.4660 - accuracy: 0.8398 - val\_loss: 0.5036 - val\_accuracy: 0.8342*\n*",  
 "Epoch 9/25*\n*",  
 "313/313 [==============================] - 50s 160ms/step - loss: 0.4315 - accuracy: 0.8523 - val\_loss: 0.4470 - val\_accuracy: 0.8461*\n*",  
 "Epoch 10/25*\n*",  
 "313/313 [==============================] - 50s 160ms/step - loss: 0.4110 - accuracy: 0.8569 - val\_loss: 0.4712 - val\_accuracy: 0.8440*\n*",  
 "Epoch 11/25*\n*",  
 "313/313 [==============================] - 50s 159ms/step - loss: 0.3797 - accuracy: 0.8673 - val\_loss: 0.4721 - val\_accuracy: 0.8504*\n*",  
 "Epoch 12/25*\n*",  
 "313/313 [==============================] - 50s 160ms/step - loss: 0.3641 - accuracy: 0.8728 - val\_loss: 0.4315 - val\_accuracy: 0.8541*\n*",  
 "Epoch 13/25*\n*",  
 "313/313 [==============================] - 50s 159ms/step - loss: 0.3399 - accuracy: 0.8826 - val\_loss: 0.4618 - val\_accuracy: 0.8501*\n*",  
 "Epoch 14/25*\n*",  
 "313/313 [==============================] - 50s 159ms/step - loss: 0.3259 - accuracy: 0.8860 - val\_loss: 0.4929 - val\_accuracy: 0.8448*\n*",  
 "Epoch 15/25*\n*",  
 "313/313 [==============================] - 50s 159ms/step - loss: 0.3130 - accuracy: 0.8925 - val\_loss: 0.4325 - val\_accuracy: 0.8604*\n*",  
 "Epoch 16/25*\n*",  
 "313/313 [==============================] - 50s 159ms/step - loss: 0.2969 - accuracy: 0.8955 - val\_loss: 0.4943 - val\_accuracy: 0.8477*\n*",  
 "Epoch 17/25*\n*",  
 "313/313 [==============================] - 50s 159ms/step - loss: 0.2805 - accuracy: 0.9008 - val\_loss: 0.4315 - val\_accuracy: 0.8650*\n*",  
 "Epoch 18/25*\n*",  
 "313/313 [==============================] - 50s 159ms/step - loss: 0.2638 - accuracy: 0.9089 - val\_loss: 0.4128 - val\_accuracy: 0.8674*\n*",  
 "Epoch 19/25*\n*",  
 "313/313 [==============================] - 50s 159ms/step - loss: 0.2528 - accuracy: 0.9108 - val\_loss: 0.4517 - val\_accuracy: 0.8616*\n*",  
 "Epoch 20/25*\n*",  
 "313/313 [==============================] - 50s 159ms/step - loss: 0.2420 - accuracy: 0.9147 - val\_loss: 0.4273 - val\_accuracy: 0.8658*\n*",  
 "Epoch 21/25*\n*",  
 "313/313 [==============================] - 50s 159ms/step - loss: 0.2311 - accuracy: 0.9158 - val\_loss: 0.4323 - val\_accuracy: 0.8693*\n*",  
 "Epoch 22/25*\n*",  
 "313/313 [==============================] - 50s 159ms/step - loss: 0.2142 - accuracy: 0.9244 - val\_loss: 0.4716 - val\_accuracy: 0.8679*\n*",  
 "Epoch 23/25*\n*",  
 "313/313 [==============================] - 50s 159ms/step - loss: 0.2091 - accuracy: 0.9272 - val\_loss: 0.4171 - val\_accuracy: 0.8765*\n*",  
 "Epoch 24/25*\n*",  
 "313/313 [==============================] - 50s 159ms/step - loss: 0.1964 - accuracy: 0.9304 - val\_loss: 0.4461 - val\_accuracy: 0.8724*\n*",  
 "Epoch 25/25*\n*",  
 "313/313 [==============================] - 50s 159ms/step - loss: 0.1879 - accuracy: 0.9337 - val\_loss: 0.4827 - val\_accuracy: 0.8671*\n*"  
 ],  
 "name": "stdout"  
 }  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "metadata": {  
 "colab": {  
 "base\_uri": "https://localhost:8080/",  
 "height": *283* },  
 "id": "sca37Zc1nguf",  
 "outputId": "86f930b2-c0e7-4c3f-da2a-b139529c3f25"  
 },  
 "source": [  
 "acc = history.history['accuracy']*\n*",  
 "val\_acc = history.history['val\_accuracy']*\n*",  
 "*\n*",  
 "plt.figure()*\n*",  
 "plt.plot(acc,color = 'purple',label = 'Training Acuracy')*\n*",  
 "plt.plot(val\_acc,color = 'blue',label = 'Validation Accuracy')*\n*",  
 "plt.legend()"  
 ],  
 "execution\_count": *null*,  
 "outputs": [  
 {  
 "output\_type": "execute\_result",  
 "data": {  
 "text/plain": [  
 "<matplotlib.legend.Legend at 0x7fc5601fe610>"  
 ]  
 },  
 "metadata": {  
 "tags": []  
 },  
 "execution\_count": *15* },  
 {  
 "output\_type": "display\_data",  
 "data": {  
 "image/png": "*\n*",  
 "text/plain": [  
 "<Figure size 432x288 with 1 Axes>"  
 ]  
 },  
 "metadata": {  
 "tags": [],  
 "needs\_background": "light"  
 }  
 }  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "metadata": {  
 "colab": {  
 "base\_uri": "https://localhost:8080/",  
 "height": *283* },  
 "id": "Y5CCvx2Fnv6I",  
 "outputId": "274cb9d8-f8f1-4182-89dd-56110b67decc"  
 },  
 "source": [  
 "loss = history.history['loss']*\n*",  
 "val\_loss = history.history['val\_loss']*\n*",  
 "*\n*",  
 "plt.figure()*\n*",  
 "plt.plot(loss,color = 'green',label = 'Training Loss')*\n*",  
 "plt.plot(val\_loss,color = 'red',label = 'Validation Loss')*\n*",  
 "plt.legend()"  
 ],  
 "execution\_count": *null*,  
 "outputs": [  
 {  
 "output\_type": "execute\_result",  
 "data": {  
 "text/plain": [  
 "<matplotlib.legend.Legend at 0x7fc5bd878ad0>"  
 ]  
 },  
 "metadata": {  
 "tags": []  
 },  
 "execution\_count": *16* },  
 {  
 "output\_type": "display\_data",  
 "data": {  
 "image/png": "*\n*",  
 "text/plain": [  
 "<Figure size 432x288 with 1 Axes>"  
 ]  
 },  
 "metadata": {  
 "tags": [],  
 "needs\_background": "light"  
 }  
 }  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "metadata": {  
 "colab": {  
 "base\_uri": "https://localhost:8080/"  
 },  
 "id": "ins28-Enp6el",  
 "outputId": "9888802d-9398-4b41-a3fa-e14579e7027e"  
 },  
 "source": [  
 "x\_test = tf.keras.applications.vgg19.preprocess\_input(X\_test)*\n*",  
 "y\_pred = model.predict\_classes(x\_test)*\n*",  
 "y\_pred[:10]"  
 ],  
 "execution\_count": *null*,  
 "outputs": [  
 {  
 "output\_type": "stream",  
 "text": [  
 "/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/sequential.py:455: UserWarning: `model.predict\_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 activation).\* `(model.predict(x) > 0.5).astype(*\"*int32*\"*)`, if your model does binary classification (e.g. if it uses a `sigmoid` last-layer activation).*\n*",  
 " warnings.warn('`model.predict\_classes()` is deprecated and '*\n*"  
 ],  
 "name": "stderr"  
 },  
 {  
 "output\_type": "execute\_result",  
 "data": {  
 "text/plain": [  
 "array([3, 8, 8, 0, 6, 6, 1, 6, 3, 1])"  
 ]  
 },  
 "metadata": {  
 "tags": []  
 },  
 "execution\_count": *17* }  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "metadata": {  
 "colab": {  
 "base\_uri": "https://localhost:8080/"  
 },  
 "id": "mBdO\_JoGn2go",  
 "outputId": "d2e32d19-583e-4314-d9ca-f5556444c55e"  
 },  
 "source": [  
 "from sklearn.metrics import confusion\_matrix, accuracy\_score*\n*",  
 "print('Testing Accuarcy : ', accuracy\_score(Y\_test, y\_pred))"  
 ],  
 "execution\_count": *null*,  
 "outputs": [  
 {  
 "output\_type": "stream",  
 "text": [  
 "Testing Accuarcy : 0.8591*\n*"  
 ],  
 "name": "stdout"  
 }  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "metadata": {  
 "colab": {  
 "base\_uri": "https://localhost:8080/"  
 },  
 "id": "7J7Nf0vMq2iT",  
 "outputId": "68782688-b8fe-42d1-825e-41b1addea739"  
 },  
 "source": [  
 "cm = confusion\_matrix(Y\_test, y\_pred)*\n*",  
 "cm"  
 ],  
 "execution\_count": *null*,  
 "outputs": [  
 {  
 "output\_type": "execute\_result",  
 "data": {  
 "text/plain": [  
 "array([[882, 11, 20, 1, 12, 1, 5, 9, 43, 16],*\n*",  
 " [ 5, 944, 0, 2, 0, 1, 2, 0, 8, 38],*\n*",  
 " [ 22, 3, 800, 11, 54, 30, 56, 9, 10, 5],*\n*",  
 " [ 10, 7, 42, 536, 57, 228, 74, 26, 6, 14],*\n*",  
 " [ 6, 1, 22, 5, 884, 20, 35, 23, 2, 2],*\n*",  
 " [ 5, 4, 9, 54, 36, 845, 19, 26, 0, 2],*\n*",  
 " [ 5, 1, 11, 7, 16, 10, 941, 3, 3, 3],*\n*",  
 " [ 4, 1, 12, 7, 31, 28, 8, 902, 2, 5],*\n*",  
 " [ 25, 16, 1, 1, 4, 0, 4, 1, 929, 19],*\n*",  
 " [ 6, 41, 1, 2, 4, 0, 6, 2, 10, 928]])"  
 ]  
 },  
 "metadata": {  
 "tags": []  
 },  
 "execution\_count": *19* }  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "metadata": {  
 "id": "nl6TTxEarCW1"  
 },  
 "source": [  
 "import itertools*\n*",  
 "def plot\_confusion\_matrix(cm, classes,*\n*",  
 " normalize=False,*\n*",  
 " title='Confusion matrix',*\n*",  
 " cmap=plt.cm.Greens):*\n*",  
 " *\"\"\"\n*",  
 " This function prints and plots the confusion matrix.*\n*",  
 " Normalization can be applied by setting `normalize=True`.*\n*",  
 " *\"\"\"\n*",  
 " plt.imshow(cm, interpolation='nearest', cmap=cmap)*\n*",  
 " plt.title(title)*\n*",  
 " plt.colorbar()*\n*",  
 " tick\_marks = np.arange(len(classes))*\n*",  
 " plt.xticks(tick\_marks, classes, rotation=30)*\n*",  
 " plt.yticks(tick\_marks, classes)*\n*",  
 "*\n*",  
 " if normalize:*\n*",  
 " cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]*\n*",  
 " print(*\"*Normalized confusion matrix*\"*)*\n*",  
 " else:*\n*",  
 " print('Confusion matrix, without normalization')*\n*",  
 "*\n*",  
 " #print(cm)*\n*",  
 "*\n*",  
 " thresh = cm.max() / 2.*\n*",  
 " for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):*\n*",  
 " plt.text(j, i, cm[i, j],*\n*",  
 " horizontalalignment=*\"*center*\"*,*\n*",  
 " color=*\"*white*\"* if cm[i, j] > thresh else *\"*black*\"*)*\n*",  
 "*\n*",  
 " plt.tight\_layout()*\n*",  
 " plt.ylabel('True label')*\n*",  
 " plt.xlabel('Predicted label')"  
 ],  
 "execution\_count": *null*,  
 "outputs": []  
 },  
 {  
 "cell\_type": "code",  
 "metadata": {  
 "colab": {  
 "base\_uri": "https://localhost:8080/",  
 "height": *603* },  
 "id": "Bx49dDWbrWCI",  
 "outputId": "4f406745-2534-4fd2-b5a1-d1a4d070e07b"  
 },  
 "source": [  
 "plt.figure(figsize=(8,8))*\n*",  
 "plot\_confusion\_matrix(cm,classes)"  
 ],  
 "execution\_count": *null*,  
 "outputs": [  
 {  
 "output\_type": "stream",  
 "text": [  
 "Confusion matrix, without normalization*\n*"  
 ],  
 "name": "stdout"  
 },  
 {  
 "output\_type": "display\_data",  
 "data": {  
 "image/png": "\n",  
 "text/plain": [  
 "<Figure size *576*x*576* with *2* Axes>"  
 ]  
 },  
 "metadata": {  
 "tags": [],  
 "needs\_background": "light"  
 }  
 }  
 ]  
 }  
 ]  
}