```
"nbformat": 4,
 "nbformat_minor": 0,
 "metadata": {
  "colab": {
   "provenance": [],
   "collapsed_sections": []
  },
  "kernelspec": {
   "name": "python3",
   "display_name": "Python 3"
  },
  "language_info": {
   "name": "python"
  }
 },
 "cells": [
  {
   "cell_type": "code",
   "source": [
    "#Importing Neccessary Libraries\n",
    "\n",
    "import numpy as np\n",
    "#used for numerical analysis\n",
    "import tensorflow #open source used for both ML and DL for computation\n",
    "from tensorflow.keras.models import Sequential #it is a plain stack of layers\n",
    "from tensorflow.keras import layers # a layer consists of a tensor-in tensor-out computation function\n",
    "#Dense layer is the regular deeply connected neural network layer\n",
    "from tensorflow.keras.layers import Dense, Flatten \n",
    "#Flatten-used fot flattering the input or change the dimension\n",
    "from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dropout #convolutional layer\n",
    "#MaxPooling2D-for downsampling the image\n",
    "from keras.preprocessing.image import ImageDataGenerator"
   ],
   "metadata": {
    "id": "XdON9-MKqt-K"
   },
   "execution_count": 1,
   "outputs": []
  },
  {
   "cell_type": "code",
   "source": [
    "from google.colab import drive\n",
    "drive.mount('/content/drive')"
   ],
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "Yac7zN7nb907",
    "outputId": "1e2e3a47-587f-4485-8aab-1f5aa1945444"
   },
   "execution_count": 18,
   "outputs": [
```

```
    {
     "output_type": "stream",
     "name": "stdout",
     "text": [
      "Mounted at /content/drive\n"
     ]
    }
   ]
  },
  {
   "cell_type": "code",
   "source": [
    "import tensorflow as tf\n",
    "\n",
    "from tensorflow.keras import datasets, layers, models\n",
    "import matplotlib.pyplot as plt"
   ],
   "metadata": {
    "id": "NbjcLABwTD-f"
   },
   "execution_count": 5,
   "outputs": []
  },
  {
   "cell_type": "code",
   "source": [
    "(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_data()\n",
    "\n",
    "# Normalize pixel values to be between 0 and 1\n",
    "train_images, test_images = train_images / 255.0, test_images / 255.0"
   ],
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "fOxgYCO6THhC",
    "outputId": "8b9319aa-ca4c-465a-f05c-2df9cbd49f5a"
   },
   "execution_count": 6,
   "outputs": [
    {
     "output_type": "stream",
     "name": "stdout",
     "text": [
      "Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz\n",
      "170498071/170498071 [==============================] - 4s 0us/step\n"
     ]
    }
   ]
  },
  {
   "cell_type": "code",
   "source": [
    "#Creating the model\n",
    "model = models.Sequential()\n",
```

```
   "model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))\n",
   "model.add(layers.MaxPooling2D((2, 2)))\n",
   "model.add(layers.Conv2D(64, (3, 3), activation='relu'))\n",
   "model.add(layers.MaxPooling2D((2, 2)))\n",
   "model.add(layers.Conv2D(64, (3, 3), activation='relu'))\n",
   "model.add(layers.Flatten())\n",
   "model.add(layers.Dense(64, activation='relu'))\n",
   "model.add(layers.Dense(10))\n"
  ],
  "metadata": {
   "id": "169yIjH3lrlT"
  },
  "execution_count": 7,
  "outputs": []
 },
 {
  "cell_type": "code",
  "source": [
   "model.summary()"
  ],
  "metadata": {
   "colab": {
    "base_uri": "https://localhost:8080/"
   },
   "id": "q5fdM_m2lvrx",
   "outputId": "69314f45-2b78-460b-cb5e-893840945ff1"
  },
  "execution_count": 8,
  "outputs": [
   {
    "output_type": "stream",
    "name": "stdout",
    "text": [
     "Model: \"sequential\"\n",
     "_________________________________________________________________\n",
     " Layer (type)                Output Shape              Param #   \n",
     "=================================================================\n",
     " conv2d (Conv2D)             (None, 30, 30, 32)        896       \n",
     "                                                                 \n",
     " max_pooling2d (MaxPooling2D  (None, 15, 15, 32)       0         \n",
     " )                                                               \n",
     "                                                                 \n",
     " conv2d_1 (Conv2D)           (None, 13, 13, 64)        18496     \n",
     "                                                                 \n",
     " max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0         \n",
     " 2D)                                                             \n",
     "                                                                 \n",
     " conv2d_2 (Conv2D)           (None, 4, 4, 64)          36928     \n",
     "                                                                 \n",
     " flatten (Flatten)           (None, 1024)              0         \n",
     "                                                                 \n",
     " dense (Dense)               (None, 64)                65600     \n",
     "                                                                 \n",
     " dense_1 (Dense)             (None, 10)                650       \n",
     "                                                                 \n",
```

```
        "=================================================================\n",
        "Total params: 122,570\n",
        "Trainable params: 122,570\n",
        "Non-trainable params: 0\n",
        "_________________________________________________________________\n"
       ]
      }
     ]
    },
    {
     "cell_type": "code",
     "source": [
      "#Compiling the model\n",
      "model.compile(optimizer='adam',\n",
      "              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),\n",
      "              metrics=['accuracy'])\n",
      "#Fitting the model\n",
      "history = model.fit(train_images, train_labels, epochs=10, \n",
      "                    validation_data=(test_images, test_labels))"
     ],
     "metadata": {
      "colab": {
       "base_uri": "https://localhost:8080/"
      },
      "id": "7TWsgR2kl1K5",
      "outputId": "cd1fd463-08eb-43b2-a3eb-12ca48b1d855"
     },
     "execution_count": 9,
     "outputs": [
      {
       "output_type": "stream",
       "name": "stdout",
       "text": [
        "Epoch 1/10\n",
        "1563/1563 [==============================] - 85s 54ms/step - loss: 1.5241 - accuracy: 0.4449 - val_loss: 1.2775 - val_accuracy: 0.5395\n",
        "Epoch 2/10\n",
        "1563/1563 [==============================] - 78s 50ms/step - loss: 1.1802 - accuracy: 0.5837 - val_loss: 1.2141 - val_accuracy: 0.5757\n",
        "Epoch 3/10\n",
        "1563/1563 [==============================] - 78s 50ms/step - loss: 1.0330 - accuracy: 0.6392 - val_loss: 0.9934 - val_accuracy: 0.6493\n",
        "Epoch 4/10\n",
        "1563/1563 [==============================] - 78s 50ms/step - loss: 0.9367 - accuracy: 0.6738 - val_loss: 0.9645 - val_accuracy: 0.6568\n",
        "Epoch 5/10\n",
        "1563/1563 [==============================] - 78s 50ms/step - loss: 0.8598 - accuracy: 0.6982 - val_loss: 0.8927 - val_accuracy: 0.6906\n",
        "Epoch 6/10\n",
        "1563/1563 [==============================] - 77s 49ms/step - loss: 0.8069 - accuracy: 0.7185 - val_loss: 0.8897 - val_accuracy: 0.6925\n",
        "Epoch 7/10\n",
        "1563/1563 [==============================] - 77s 49ms/step - loss: 0.7532 - accuracy: 0.7351 - val_loss: 0.9193 - val_accuracy: 0.6885\n",
        "Epoch 8/10\n",
```

```
          "1563/1563 [==============================] - 76s 49ms/step - loss: 0.7127 - accuracy: 0.7508 - val_loss: 0.8996 - val_accuracy: 0.6962\n",
          "Epoch 9/10\n",
          "1563/1563 [==============================] - 76s 48ms/step - loss: 0.6726 - accuracy: 0.7641 - val_loss: 0.8864 - val_accuracy: 0.6996\n",
          "Epoch 10/10\n",
          "1563/1563 [==============================] - 76s 49ms/step - loss: 0.6358 - accuracy: 0.7755 - val_loss: 0.9306 - val_accuracy: 0.6936\n"
         ]
        }
       ]
      },
      {
       "cell_type": "code",
       "source": [
        "#Saving our model\n",
        "model.save('nutrition.h5')"
       ],
       "metadata": {
        "id": "WLvInNYXpQTz"
       },
       "execution_count": 11,
       "outputs": []
      },
      {
       "cell_type": "code",
       "source": [
        "#Prediciting our results\n",
        "from tensorflow.keras.models import load_model\n",
        "from tensorflow.keras.preprocessing import image\n",
        "model=load_model('nutrition.h5')"
       ],
       "metadata": {
        "id": "SW74x3bapQiW"
       },
       "execution_count": 16,
       "outputs": []
      },
      {
       "cell_type": "code",
       "source": [
        "img=image.load_img('/content/drive/MyDrive/Nutrition Image Analysis using CNN and Rapid API-20221106T044103Z-001/Nutrition Image Analysis using CNN and Rapid API/Dataset/TRAIN_SET/APPLES/n07740461_10065.jpg',target_size=(70,70))\n",
        "img"
       ],
       "metadata": {
        "colab": {
         "base_uri": "https://localhost:8080/",
         "height": 87
        },
        "id": "_OrLiTErq2xo",
        "outputId": "af059ad7-d898-4517-8f0f-5c16c437e8a0"
       },
       "execution_count": 19,
```

```
  "outputs": [
   {
    "output_type": "execute_result",
    "data": {
     "text/plain": [
      ""
     ],
     "image/png":          },
    "metadata": {},
    "execution_count": 19
   }
  ]
 },
 {
  "cell_type": "code",
  "source": [
   "x= image.img_to_array(img)\n"
  ],
  "metadata": {
   "id": "LN8xqzTar0Lb"
  },
  "execution_count": 20,
  "outputs": []
 },
 {
  "cell_type": "code",
  "source": [
   "x = np.expand_dims(x, axis=0)"
  ],
  "metadata": {
   "id": "p8lCho1Rr-E9"
  },
  "execution_count": 21,
  "outputs": []
 },
 {
  "cell_type": "code",
  "source": [
   "'\n",
   "index=['APPLES', 'BANANA', 'ORANGE', 'PINEAPPLE', 'WATERMELON']\n",
   "result=str(index[0])\n",
   "result"
  ],
  "metadata": {
   "colab": {
    "base_uri": "https://localhost:8080/",
    "height": 36
   },
   "id": "2fI9nwY7sSvT",
   "outputId": "acade409-66c4-4d6f-c974-7e3edb8f5570"
  },
  "execution_count": 27,
  "outputs": [
   {
    "output_type": "execute_result",
```

```
        "data": {
          "text/plain": [
            "'APPLES'"
          ],
          "application/vnd.google.colaboratory.intrinsic+json": {
            "type": "string"
          }
        },
        "metadata": {},
        "execution_count": 27
      }
    ]
  }
 ]
}
```