```
import numpy as np
#used for numerical analysis
import  tensorflow  #open  source  used  for  both  ML  and  DL  for
computation from tensorflow.keras.models import Sequential #it is a
plain stack oflayers
from tensorflow.keras import layers # a layer consists of a tensor-
intensor-out computation function
#Dense layer is the regular deeply connected neural network layer
from tensorflow.keras.layers import Dense, Flatten
#Flatten-used fot flattering the input or change the dimension
from tensorflow.keras.layers import Conv2D, MaxPooling2D,
Dropout#convolutional layer
#MaxPooling2D-for downsampling the image
from keras.preprocessing.image import ImageDataGenerator

from google.colab import
drive
drive.mount('/content/drive')

import tensorflow as tf

from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt

(train_images, train_labels), (test_images, test_labels) =
datasets.cifar10.load_data()

# Normalize pixel values to be between 0 and 1
train_images, test_images = train_images / 255.0, test_images / 255.0

#Creating the model
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3),
activation='relu',
input_shape=(32, 32, 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(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))

model.summary()

#Compiling the model
model.compile(optimizer='adam',
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

#Fitting the model
history = model.fit(train_images, train_labels, epochs=10,
                    validation_data=(test_images, test_labels))

#Saving our model
model.save('nutrition.h5')

#Prediciting our results
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
model=load_model('nutrition.h5')

img=image.load_img('https://drive.google.com/drive/folders/1iHlwnX4bE
zukqsTUwAbvK-9tRrqp2Mpr?usp=share_link 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)
)img

x= image.img_to_array(img)

x = np.expand_dims(x, axis=0)

index=['APPLES', 'BANANA', 'ORANGE', 'PINEAPPLE', 'WATERMELON']
result=str(index[0]
)result
```

```
Mounted at /content/drive
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-
python.tar.gz
170498071/170498071 [==============================] - 2s 0us/step
Model: "sequential"

 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 32)        896

 max_pooling2d (MaxPooling2D  (None, 15, 15, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)
```

conv2d_2 (Conv2D)	(None,	4, 4,	64)	36928
flatten (Flatten)	(None,	1024)		0
dense (Dense)	(None,	64)		65600

```
 dense_1 (Dense)             (None, 10)                650

=================================================================
Total params: 122,570
Trainable params: 122,570
Non-trainable params: 0

Epoch 1/10
1563/1563 [==============================] - 71s 45ms/step - loss:
1.5489 - accuracy: 0.4334 - val_loss: 1.2797 - val_accuracy: 0.5380
Epoch 2/10
1563/1563 [==============================] - 69s 44ms/step - loss:
1.1728 - accuracy: 0.5837 - val_loss: 1.0645 - val_accuracy: 0.6289
Epoch 3/10
1563/1563 [==============================] - 68s 44ms/step - loss:
1.0175 - accuracy: 0.6424 - val_loss: 0.9828 - val_accuracy: 0.6563
Epoch 4/10
1563/1563 [==============================] - 70s 45ms/step - loss:
0.9222 - accuracy: 0.6790 - val_loss: 0.9710 - val_accuracy: 0.6605
Epoch 5/10
1563/1563 [==============================] - 68s 43ms/step - loss:
0.8525 - accuracy: 0.6999 - val_loss: 0.9227 - val_accuracy: 0.6818
Epoch 6/10
1563/1563 [==============================] - 68s 44ms/step - loss:
0.7994 - accuracy: 0.7194 - val_loss: 0.8692 - val_accuracy: 0.7029
Epoch 7/10
1563/1563 [==============================] - 69s 44ms/step - loss:
0.7552 - accuracy: 0.7335 - val_loss: 0.8711 - val_accuracy: 0.6974
Epoch 8/10
1563/1563 [==============================] - 70s 45ms/step - loss:
0.7136 - accuracy: 0.7503 - val_loss: 0.9073 - val_accuracy: 0.6943
Epoch 9/10
1563/1563 [==============================] - 69s 44ms/step - loss:
0.6729 - accuracy: 0.7617 - val_loss: 0.8794 - val_accuracy: 0.7047
Epoch 10/10
1563/1563 [==============================] - 69s 44ms/step - loss:
0.6421 - accuracy: 0.7751 - val_loss: 0.8812 - val_accuracy: 0.7042
```