```
%tensorflow_version 2.x  # this line is not required unless you are in a notebook
import tensorflow as tf
from tensorflow.keras import layers, models, datasets, utils
import matplotlib.pyplot as plt
```

```
Colab only includes TensorFlow 2.x; %tensorflow_version has no effect.
```

```
(train_images , train_labels) , (test_images , test_labels) = datasets.cifar10.load_data()
train_images , test_images = train_images / 255.0 , test_images / 255.0
```

```
train_images.shape
train_labels.shape
train_images
```

```
          [0.60392157, 0.62745098, 0.66666667],
          ...,
          [0.16470588, 0.13333333, 0.14117647],
          [0.23921569, 0.20784314, 0.22352941],
          [0.36470588, 0.3254902 , 0.35686275]],

         [[0.64705882, 0.60392157, 0.50196078],
          [0.61176471, 0.59607843, 0.50980392],
          [0.62352941, 0.63137255, 0.55686275],
          ...,
          [0.40392157, 0.36470588, 0.37647059],
          [0.48235294, 0.44705882, 0.47058824],
          [0.51372549, 0.4745098 , 0.51372549]],

         [[0.63921569, 0.58039216, 0.47058824],
          [0.61960784, 0.58039216, 0.47843137],
          [0.63921569, 0.61176471, 0.52156863],
          ...,
          [0.56078431, 0.52156863, 0.54509804],
          [0.56078431, 0.5254902 , 0.55686275],
          [0.56078431, 0.52156863, 0.56470588]]],


        [[[1.        , 1.        , 1.        ],
          [0.99215686, 0.99215686, 0.99215686],
          [0.99215686, 0.99215686, 0.99215686],
          ...,
          [0.99215686, 0.99215686, 0.99215686],
          [0.99215686, 0.99215686, 0.99215686],
          [0.99215686, 0.99215686, 0.99215686]],

         [[1.        , 1.        , 1.        ],
          [1.        , 1.        , 1.        ],
          [1.        , 1.        , 1.        ],
          ...,
          [1.        , 1.        , 1.        ],
          [1.        , 1.        , 1.        ],
          [1.        , 1.        , 1.        ]],

         [[1.        , 1.        , 1.        ],
          [0.99607843, 0.99607843, 0.99607843],
          [0.99607843, 0.99607843, 0.99607843],
          ...,
          [0.99607843, 0.99607843, 0.99607843],
          [0.99607843, 0.99607843, 0.99607843],
          [0.99607843, 0.99607843, 0.99607843]],

         ...,

         [[0.44313725, 0.47058824, 0.43921569],
          [0.43529412, 0.4627451 , 0.43529412],
          [0.41176471, 0.43921569, 0.41568627],
          ...,
          [0.28235294, 0.31764706, 0.31372549],
          [0.28235294, 0.31372549, 0.30980392],
          [0.28235294, 0.31372549, 0.30980392]],

         [[0.43529412, 0.4627451 , 0.43137255],
          [0.40784314, 0.43529412, 0.40784314],
```

```
IMG_INDEX = 49999  # change this to look at other images
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']
# plt.figure()
# plt.imshow(train_images[IMG_INDEX] ,cmap=plt.cm.binary)
# plt.xlabel(class_names[train_labels[IMG_INDEX][0]])
# plt.show()
```

```
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))
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_sha
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
model.compile(optimizer='Adamax',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

```
history = model.fit(train_images, train_labels, epochs=8,
                    validation_data=(test_images, test_labels))
```

```
Epoch 1/8
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 80s 50ms/step - accuracy: 0.2960 - loss: 1.9035 - val_accuracy: 0.4797 - val_loss: 1.4424
Epoch 2/8
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 81s 49ms/step - accuracy: 0.4959 - loss: 1.4019 - val_accuracy: 0.5503 - val_loss: 1.2601
Epoch 3/8
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 81s 49ms/step - accuracy: 0.5630 - loss: 1.2375 - val_accuracy: 0.5844 - val_loss: 1.1680
Epoch 4/8
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 81s 48ms/step - accuracy: 0.6024 - loss: 1.1251 - val_accuracy: 0.6275 - val_loss: 1.0573
Epoch 5/8
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 77s 49ms/step - accuracy: 0.6402 - loss: 1.0282 - val_accuracy: 0.6449 - val_loss: 1.0034
Epoch 6/8
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 75s 48ms/step - accuracy: 0.6645 - loss: 0.9615 - val_accuracy: 0.6597 - val_loss: 0.9715
Epoch 7/8
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 83s 53ms/step - accuracy: 0.6834 - loss: 0.9120 - val_accuracy: 0.6621 - val_loss: 0.9543
Epoch 8/8
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 77s 49ms/step - accuracy: 0.7029 - loss: 0.8637 - val_accuracy: 0.6945 - val_loss: 0.8860
```

```
model.evaluate(test_images,  test_labels, verbose=0.01)
```

```
[0.8859587907791138, 0.6945000290870667]
```

```
train_labels
```

```
array([[6],
       [9],
       [9],
       ...,
       [9],
       [1],
       [1]], dtype=uint8)
```

```
from tensorflow.keras.preprocessing import image
import numpy as np


img_path = '/test_image.jpg'  # Provide the path to your image here
img = image.load_img(img_path, target_size=(32, 32))

# Step 2: Convert the image to a numpy array
img_array = image.img_to_array(img)

# Step 3: Normalize the image
img_array = img_array / 255.0  # Scaling the image to be between 0 and 1

# Step 4: Expand dimensions to match the model's input shape
img_array = np.expand_dims(img_array, axis=0)  # (1, 32, 32, 3)

# The image is now ready for model evaluation
# Step 5: Evaluate the model on this single image
predictions = model.predict(img_array)

# Optional: Print the predictions
print(predictions)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 96ms/step
[[ 5.626914   -0.32364175  0.75023305 -2.6693144  -6.802434   -6.196869
  -2.8286593  -7.92513     4.421582   -1.6097699 ]]
```

```
predicted_class_index = np.argmax(predictions[0])

# Access the class name using the predicted index
print(class_names[predicted_class_index])
```

```
airplane
```

```
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator # Correct the import statement

datagen = ImageDataGenerator( # Correct the typo
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode = 'nearest'
)
# test_img = train_images[20]
# img = image.img_to_array(test_img)  # convert image to numpy arry
# img = img.reshape((1,) + img.shape)  # reshape image
# plt.figure()
# img.shape
augmented_images = []
augmented_labels = []

for i in range(len(train_images)):
  img = image.img_to_array(train_images[i])
  img = img.reshape((1,) + img.shape)
  if i< len(train_labels):
    label = class_names[train_labels[i][0]]

  # Generate augmented images for each image in the training set
  for batch in datagen.flow(img, batch_size=1, save_prefix='test', save_format='jpeg'): # Reduced batch size to 1 to augment each i
      augmented_images.append(batch[0])
      augmented_labels.append(label)
      break # Generate only one augmented image per original image
```

```
import numpy as np
# Convert the lists to NumPy arrays
augmented_images = np.array(augmented_images)
```

```
augmented_labels_numeric = np.array([class_names.index(label) for label in augmented_labels]).reshape(-1, 1)

# Append the augmented data to the training set
train_images = np.concatenate((train_images, augmented_images))
train_labels = np.concatenate((train_labels, augmented_labels_numeric))
```

```
train_images.shape
```

```
(100000, 32, 32, 3)
```

```
model2=models.Sequential()
model2.add(layers.Conv2D(32,(3,3),activation='relu',input_shape=(32,32,3)))
model2.add(layers.MaxPooling2D((2,2)))
model2.add(layers.Conv2D(64,(3,3),activation='relu'))
model2.add(layers.MaxPooling2D((2,2)))
model2.add(layers.Conv2D(64,(3,3),activation='relu'))
model2.add(layers.Flatten())
model2.add(layers.Dense(64,activation='relu'))
model2.add(layers.Dense(10))
model2.compile(
    optimizer='Adamax',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy']
)
history2 = model2.fit(train_images, train_labels, epochs=8,
                    validation_data=(test_images, test_labels))
model2.evaluate(test_images,  test_labels, verbose = 0.01)
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_sha
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/8
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 149s 47ms/step - accuracy: 0.3026 - loss: 1.8815 - val_accuracy: 0.4842 - val_loss: 1.4177
Epoch 2/8
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 149s 48ms/step - accuracy: 0.4778 - loss: 1.4607 - val_accuracy: 0.5959 - val_loss: 1.1744
Epoch 3/8
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 148s 47ms/step - accuracy: 0.5366 - loss: 1.3080 - val_accuracy: 0.6370 - val_loss: 1.0579
Epoch 4/8
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 203s 48ms/step - accuracy: 0.5664 - loss: 1.2228 - val_accuracy: 0.6196 - val_loss: 1.0871
Epoch 5/8
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 203s 48ms/step - accuracy: 0.5969 - loss: 1.1460 - val_accuracy: 0.6790 - val_loss: 0.9267
Epoch 6/8
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 201s 48ms/step - accuracy: 0.6166 - loss: 1.0852 - val_accuracy: 0.6821 - val_loss: 0.9079
Epoch 7/8
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 202s 48ms/step - accuracy: 0.6367 - loss: 1.0311 - val_accuracy: 0.6960 - val_loss: 0.8807
Epoch 8/8
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 201s 47ms/step - accuracy: 0.6554 - loss: 0.9828 - val_accuracy: 0.7121 - val_loss: 0.8399
[0.8398774266242981, 0.7121000289916992]
```