```
import tensorflow as tf

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

```
# Download and preprocess the CIFAR-10 dataset
# This is a dataset of 50,000 32x32 color training images and 10,000 test images, labeled over 10 categories.
# We normalize training data, since we can make sure that the various features have similar value ranges so that gradient

# Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_data()
```

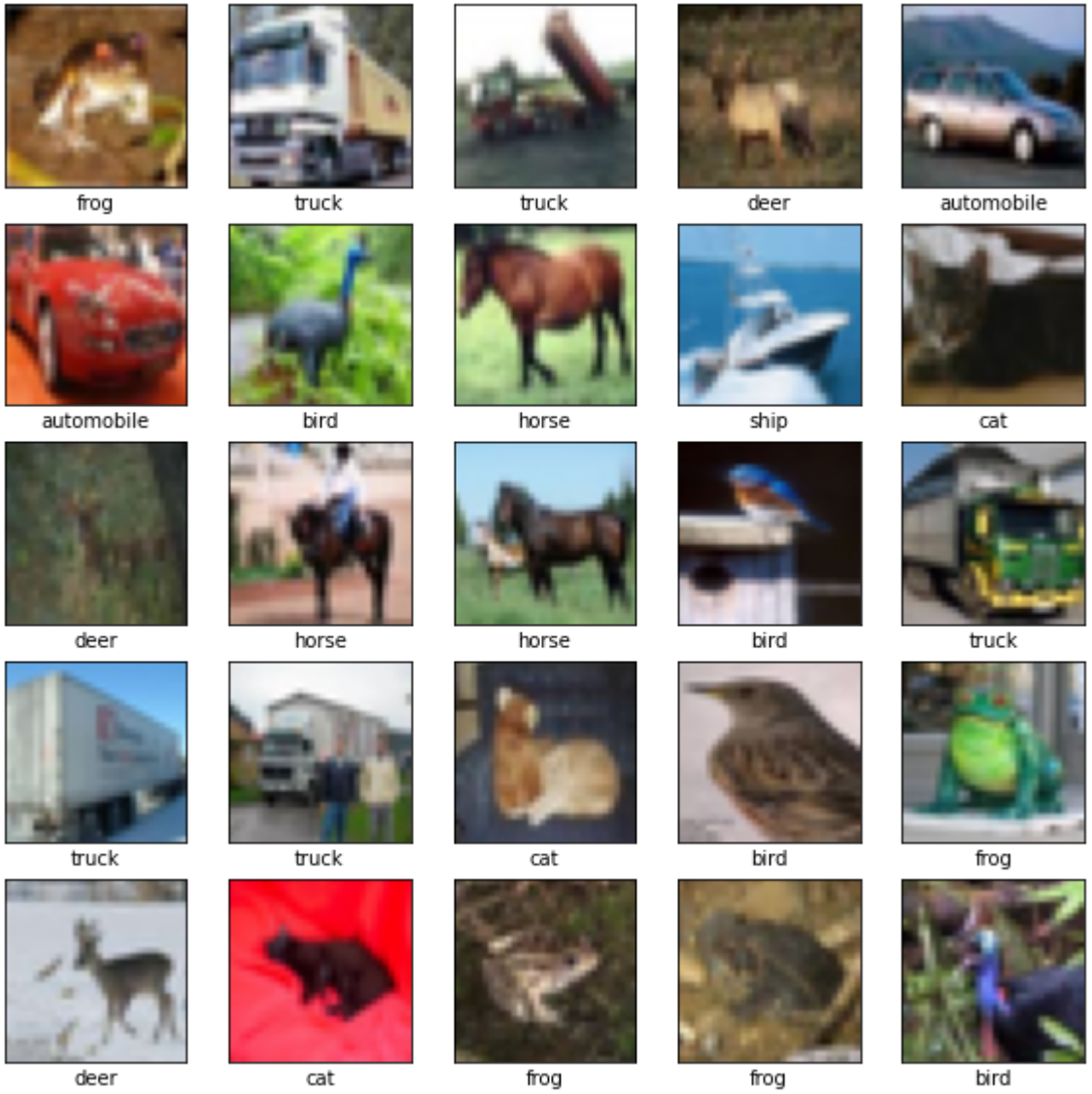
```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [==============================] - 11s 0us/step
170508288/170498071 [==============================] - 11s 0us/step
```

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

```
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']
```

```
asdf=[]
plt.figure(figsize=(10,10))

for i in range(25): # mod25, {0,1,2...24}
    plt.subplot(5,5,i+1) # 5 row, 5 columns, and 'something'
    plt.xticks([]) # no 'ticks'
    plt.yticks([])
    plt.grid(False) # don't show background 'grid lines'

    plt.imshow(train_images_norm[i])

    # The CIFAR labels happen to be arrays,
    # which is why you need the extra index
```

```
    # asdf.append(train_labels[i][0]) # >> [6, 9, 9, 4, 1, 1, 2, 7, 8, 3, 4, 7 ...]
    name1 = train_labels[i][0]
    plt.xlabel(class_names[name1])

plt.show()
```



```
# build your own VGG16 with tensorflow.keras
model1 = models.Sequential()

# 'filtering' stuff
model1.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model1.add(layers.Conv2D(32, (3, 3), activation='relu'))
model1.add(layers.MaxPooling2D((2, 2)))
model1.add(layers.Conv2D(64, (3, 3), activation='relu'))
model1.add(layers.Conv2D(64, (3, 3), activation='relu'))
# model1.add(layers.MaxPooling2D((2, 2)))
# model1.add(layers.Convolution2D(32, 3, 3, activation='relu'))
model1.add(layers.MaxPooling2D((2,2)))

# 'classifying' stuff
model1.add(layers.Flatten())
model1.add(layers.Dense(64, activation='relu'))
model1.add(layers.Dense(10)) # the last layer have to 'match' the 'class list size' (10)
```

```
model1.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_9 (Conv2D)           (None, 30, 30, 32)        896

 conv2d_10 (Conv2D)          (None, 28, 28, 32)        9248

 max_pooling2d_9 (MaxPooling  (None, 14, 14, 32)       0
 2D)

 conv2d_11 (Conv2D)          (None, 12, 12, 64)        18496

 conv2d_12 (Conv2D)          (None, 10, 10, 64)        36928

 max_pooling2d_10 (MaxPoolin  (None, 5, 5, 64)         0
 g2D)

 flatten_3 (Flatten)         (None, 1600)              0
```

```
 dense_6 (Dense)             (None, 64)                102464

 dense_7 (Dense)             (None, 10)                650

=================================================================
Total params: 168,682
Trainable params: 168,682
Non-trainable params: 0
_________________________________________________________________
```

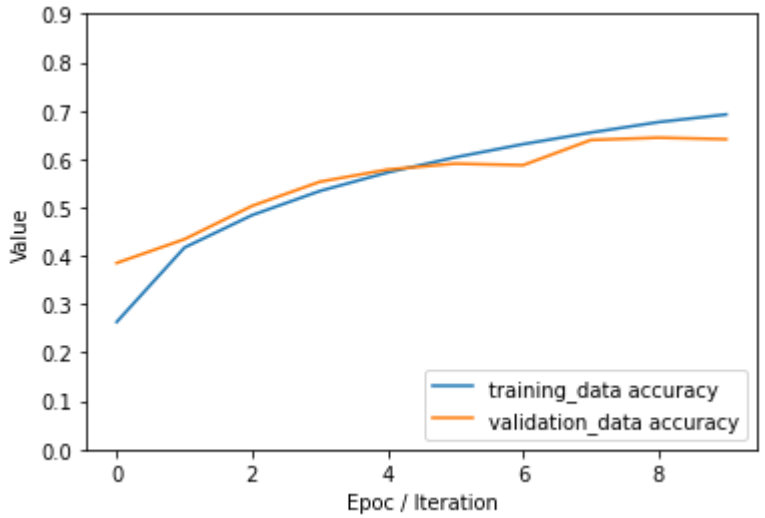
```
# 'Train' (CREATE) the CNN 'model' (the equation)
# Here, we use SGD optimizer and cross entropy loss function.

model1.compile(optimizer='SGD',
               loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
               metrics=['accuracy'])

history1 = model1.fit(train_images_norm, train_labels,
           epochs=10,
           validation_data=(test_images_norm, test_labels) )
```

```
Epoch 1/10
1563/1563 [==============================] - 187s 119ms/step - loss: 2.0168 - accuracy: 0.2634 - val_loss: 1.7348 - va
Epoch 2/10
1563/1563 [==============================] - 182s 117ms/step - loss: 1.6267 - accuracy: 0.4170 - val_loss: 1.5764 - va
Epoch 3/10
1563/1563 [==============================] - 183s 117ms/step - loss: 1.4359 - accuracy: 0.4841 - val_loss: 1.3797 - va
Epoch 4/10
1563/1563 [==============================] - 183s 117ms/step - loss: 1.3102 - accuracy: 0.5339 - val_loss: 1.2654 - va
Epoch 5/10
1563/1563 [==============================] - 182s 117ms/step - loss: 1.2075 - accuracy: 0.5722 - val_loss: 1.1821 - va
Epoch 6/10
1563/1563 [==============================] - 181s 116ms/step - loss: 1.1266 - accuracy: 0.6035 - val_loss: 1.1446 - va
Epoch 7/10
1563/1563 [==============================] - 183s 117ms/step - loss: 1.0499 - accuracy: 0.6307 - val_loss: 1.1758 - va
Epoch 8/10
1563/1563 [==============================] - 182s 116ms/step - loss: 0.9881 - accuracy: 0.6546 - val_loss: 1.0314 - va
Epoch 9/10
1563/1563 [==============================] - 181s 116ms/step - loss: 0.9281 - accuracy: 0.6762 - val_loss: 1.0167 - va
```

```
Epoch 10/10
1563/1563 [==============================] - 182s 116ms/step - loss: 0.8800 - accuracy: 0.6920 - val_loss: 1.0298 - va
```

```
what1 = history1.history['loss'][0]
print(what1)


x_data1 = history1.history['accuracy']
x_data2 = history1.history['val_accuracy']
x_data3 = history1.history['loss']

plt.plot( x_data1, label='training_data accuracy')
plt.plot( x_data2, label='validation_data accuracy')

plt.xlabel('Epoc / Iteration')
plt.ylabel('Value')
plt.legend(loc='lower right')
plt.ylim(0.0, 0.9)
```

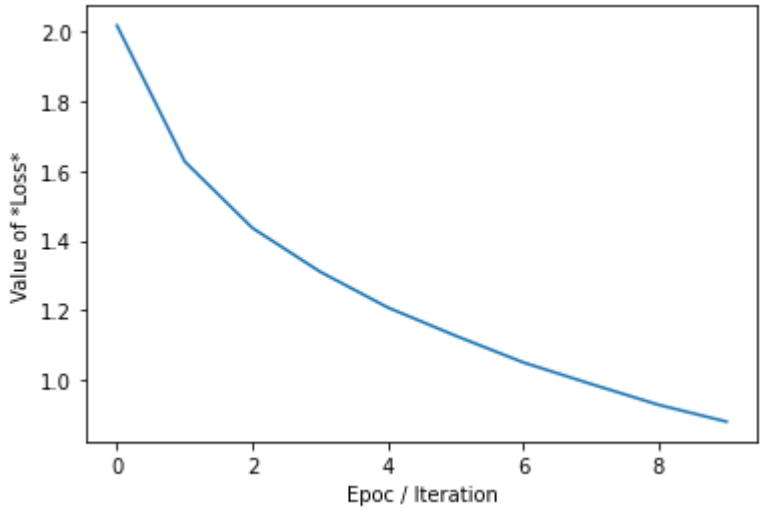
```
2.0167994499206543
(0.0, 0.9)
```




```
plt.plot( x_data3, label='Loss* value (training data)')
```

```
plt.xlabel('Epoc / Iteration')
plt.ylabel('Value of *Loss*')
```

```
Text(0, 0.5, 'Value of *Loss*')
```




```
# 'looking' at what is INSIDE 'history1'
# history1.
```

```
# 'Evaluate' the 'model' with the data in 'group test'
test_loss_value, test_accuracy_value = model1.evaluate(test_images_norm, test_labels)
print(f"Accuracy value for 'Test Data': {test_accuracy_value}\n'Loss value' for 'Test Data': {test_loss_value}")
```

```
313/313 [==============================] - 10s 32ms/step - loss: 1.0298 - accuracy: 0.6408
Accuracy value for 'Test Data': 0.6407999992370605
'Loss value' for 'Test Data': 1.02983558177948
```

0s completed at 3:38 AM