 Convolutional Neural Network

In this notebook, we'll walk through the steps required to train your own convolutional neural network (CNN) on the CIFAR dataset

In [2]:
```
import sys
import os

project_root = os.path.abspath(os.path.join(os.getcwd(), '../../..'))
if project_root not in sys.path:
    sys.path.insert(0, project_root)

import numpy as np

from tensorflow.keras import layers, models, optimizers, utils, datasets
from notebooks.utils import display
```

0. Parameters

In [2]:
```
NUM_CLASSES = 10
```

1. Prepare the Data

In [3]:
```
(x_train, y_train), (x_test, y_test) = datasets.cifar10.load_data()
```

In [4]:
```
x_train = x_train.astype("float32") / 255.0
x_test = x_test.astype("float32") / 255.0

y_train = utils.to_categorical(y_train, NUM_CLASSES)
y_test = utils.to_categorical(y_test, NUM_CLASSES)
```

In [5]:
```
display(x_train[:10])
print(y_train[:10])
```

```
array([[[[0.23137255, 0.24313726, 0.24705882],
         [0.16862746, 0.18039216, 0.1764706 ],
         [0.19607843, 0.1882353 , 0.16862746],
         ...,
         [0.61960787, 0.5176471 , 0.42352942],
         [0.59607846, 0.49019608, 0.4       ],
         [0.5803922 , 0.4862745 , 0.40392157]],

        [[0.0627451 , 0.07843138, 0.07843138],
         [0.        , 0.        , 0.        ],
         [0.07058824, 0.03137255, 0.        ],
         ...,
         [0.48235294, 0.34509805, 0.21568628],
         [0.46666667, 0.3254902 , 0.19607843],
         [0.47843137, 0.34117648, 0.22352941]],

        [[0.09803922, 0.09411765, 0.08235294],
         [0.0627451 , 0.02745098, 0.        ],
         [0.19215687, 0.10588235, 0.03137255],
         ...,
         [0.4627451 , 0.32941177, 0.19607843],
         [0.47058824, 0.32941177, 0.19607843],
         [0.42745098, 0.28627452, 0.16470589]],

        ...,

        [[0.8156863 , 0.6666667 , 0.3764706 ],
         [0.7882353 , 0.6       , 0.13333334],
         [0.7764706 , 0.6313726 , 0.10196079],
         ...,
         [0.627451  , 0.52156866, 0.27450982],
         [0.21960784, 0.12156863, 0.02745098],
         [0.20784314, 0.13333334, 0.07843138]],

        [[0.7058824 , 0.54509807, 0.3764706 ],
         [0.6784314 , 0.48235294, 0.16470589],
         [0.7294118 , 0.5647059 , 0.11764706],
         ...,
         [0.72156864, 0.5803922 , 0.36862746],
         [0.38039216, 0.24313726, 0.13333334],
         [0.3254902 , 0.20784314, 0.13333334]],

        [[0.69411767, 0.5647059 , 0.45490196],
         [0.65882355, 0.5058824 , 0.36862746],
         [0.7019608 , 0.5568628 , 0.34117648],
         ...,
         [0.84705883, 0.72156864, 0.54901963],
         [0.5921569 , 0.4627451 , 0.32941177],
         [0.48235294, 0.36078432, 0.28235295]]],


       [[[0.6039216 , 0.69411767, 0.73333335],
         [0.49411765, 0.5372549 , 0.53333336],
         [0.4117647 , 0.40784314, 0.37254903],
         ...,
         [0.35686275, 0.37254903, 0.2784314 ],
```

```
  [0.34117648, 0.3529412 , 0.2784314 ],
  [0.30980393, 0.31764707, 0.27450982]],

 [[0.54901963, 0.627451  , 0.6627451 ],
  [0.5686275 , 0.6       , 0.6039216 ],
  [0.49019608, 0.49019608, 0.4627451 ],
  ...,
  [0.3764706 , 0.3882353 , 0.30588236],
  [0.3019608 , 0.3137255 , 0.24313726],
  [0.2784314 , 0.28627452, 0.23921569]],

 [[0.54901963, 0.60784316, 0.6431373 ],
  [0.54509807, 0.57254905, 0.58431375],
  [0.4509804 , 0.4509804 , 0.4392157 ],
  ...,
  [0.30980393, 0.32156864, 0.2509804 ],
  [0.26666668, 0.27450982, 0.21568628],
  [0.2627451 , 0.27058825, 0.21568628]],

 ...,

 [[0.6862745 , 0.654902  , 0.6509804 ],
  [0.6117647 , 0.6039216 , 0.627451  ],
  [0.6039216 , 0.627451  , 0.6666667 ],
  ...,
  [0.16470589, 0.13333334, 0.14117648],
  [0.23921569, 0.20784314, 0.22352941],
  [0.3647059 , 0.3254902 , 0.35686275]],

 [[0.64705884, 0.6039216 , 0.5019608 ],
  [0.6117647 , 0.59607846, 0.50980395],
  [0.62352943, 0.6313726 , 0.5568628 ],
  ...,
  [0.40392157, 0.3647059 , 0.3764706 ],
  [0.48235294, 0.44705883, 0.47058824],
  [0.5137255 , 0.4745098 , 0.5137255 ]],

 [[0.6392157 , 0.5803922 , 0.47058824],
  [0.61960787, 0.5803922 , 0.47843137],
  [0.6392157 , 0.6117647 , 0.52156866],
  ...,
  [0.56078434, 0.52156866, 0.54509807],
  [0.56078434, 0.5254902 , 0.5568628 ],
  [0.56078434, 0.52156866, 0.5647059 ]]],


[[[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]],

 ...,

 [[0.44313726, 0.47058824, 0.4392157 ],
  [0.43529412, 0.4627451 , 0.43529412],
  [0.4117647 , 0.4392157 , 0.41568628],
  ...,
  [0.28235295, 0.31764707, 0.3137255 ],
  [0.28235295, 0.3137255 , 0.30980393],
  [0.28235295, 0.3137255 , 0.30980393]],

 [[0.43529412, 0.4627451 , 0.43137255],
  [0.40784314, 0.43529412, 0.40784314],
  [0.3882353 , 0.41568628, 0.38431373],
  ...,
  [0.26666668, 0.29411766, 0.28627452],
  [0.27450982, 0.29803923, 0.29411766],
  [0.30588236, 0.32941177, 0.32156864]],

 [[0.41568628, 0.44313726, 0.4117647 ],
  [0.3882353 , 0.41568628, 0.38431373],
  [0.37254903, 0.4       , 0.36862746],
  ...,
  [0.30588236, 0.33333334, 0.3254902 ],
  [0.30980393, 0.33333334, 0.3254902 ],
  [0.3137255 , 0.3372549 , 0.32941177]]],


...,


[[[0.10980392, 0.13725491, 0.15294118],
  [0.11764706, 0.13333334, 0.17254902],
  [0.12941177, 0.17254902, 0.18431373],
  ...,
  [0.16862746, 0.21960784, 0.1764706 ],
  [0.20392157, 0.2509804 , 0.20784314],
  [0.18039216, 0.22745098, 0.18431373]],

 [[0.10588235, 0.11764706, 0.14901961],
  [0.10588235, 0.10980392, 0.16078432],
  [0.08235294, 0.12156863, 0.15294118],
  ...,
```

```
  [0.4392157 , 0.53333336, 0.38039216],
  [0.45882353, 0.54901963, 0.39607844],
  [0.4509804 , 0.5411765 , 0.39215687]],

 [[0.13333334, 0.14117648, 0.16470589],
  [0.12941177, 0.12941177, 0.16862746],
  [0.09411765, 0.11764706, 0.15686275],
  ...,
  [0.6862745 , 0.8156863 , 0.56078434],
  [0.69411767, 0.81960785, 0.5647059 ],
  [0.6901961 , 0.8156863 , 0.56078434]],

 ...,

 [[0.5568628 , 0.6901961 , 0.4627451 ],
  [0.5568628 , 0.6901961 , 0.4627451 ],
  [0.5882353 , 0.72156864, 0.49803922],
  ...,
  [0.5254902 , 0.6862745 , 0.46666667],
  [0.5019608 , 0.65882355, 0.4392157 ],
  [0.5254902 , 0.6862745 , 0.46666667]],

 [[0.54901963, 0.6901961 , 0.4862745 ],
  [0.5686275 , 0.7058824 , 0.5058824 ],
  [0.5882353 , 0.7294118 , 0.5254902 ],
  ...,
  [0.5137255 , 0.6666667 , 0.46666667],
  [0.50980395, 0.6666667 , 0.46666667],
  [0.47843137, 0.63529414, 0.43529412]],

 [[0.5254902 , 0.67058825, 0.48235294],
  [0.53333336, 0.67058825, 0.4862745 ],
  [0.53333336, 0.67058825, 0.4862745 ],
  ...,
  [0.41568628, 0.5647059 , 0.39215687],
  [0.40784314, 0.5568628 , 0.3882353 ],
  [0.39607844, 0.54901963, 0.3764706 ]]],


[[[0.5254902 , 0.7294118 , 0.8745098 ],
  [0.5137255 , 0.72156864, 0.8627451 ],
  [0.5019608 , 0.7137255 , 0.85490197],
  ...,
  [0.49803922, 0.70980394, 0.87058824],
  [0.49803922, 0.70980394, 0.87058824],
  [0.5019608 , 0.7137255 , 0.8745098 ]],

 [[0.52156866, 0.7411765 , 0.89411765],
  [0.5058824 , 0.7294118 , 0.8784314 ],
  [0.5019608 , 0.7294118 , 0.8784314 ],
  ...,
  [0.49803922, 0.7176471 , 0.8784314 ],
  [0.49803922, 0.7176471 , 0.8784314 ],
  [0.5019608 , 0.72156864, 0.88235295]],

 [[0.5019608 , 0.7254902 , 0.8862745 ],
```

```
       [0.49803922, 0.7137255 , 0.8745098 ],
       [0.5019608 , 0.7137255 , 0.8745098 ],
       ...,
       [0.49411765, 0.70980394, 0.87058824],
       [0.49411765, 0.70980394, 0.87058824],
       [0.49411765, 0.7058824 , 0.8666667 ]],

      ...,

      [[0.68235296, 0.8156863 , 0.92156863],
       [0.67058825, 0.80784315, 0.8980392 ],
       [0.60784316, 0.7411765 , 0.84705883],
       ...,
       [0.10588235, 0.36862746, 0.53333336],
       [0.11372549, 0.3764706 , 0.5372549 ],
       [0.10980392, 0.36862746, 0.53333336]],

      [[0.7607843 , 0.8666667 , 0.95686275],
       [0.7411765 , 0.84313726, 0.9372549 ],
       [0.62352943, 0.76862746, 0.88235295],
       ...,
       [0.11764706, 0.37254903, 0.5411765 ],
       [0.11764706, 0.3764706 , 0.54509807],
       [0.11764706, 0.37254903, 0.54901963]],

      [[0.75686276, 0.8509804 , 0.92941177],
       [0.70980394, 0.8156863 , 0.9019608 ],
       [0.65882355, 0.7882353 , 0.8901961 ],
       ...,
       [0.12156863, 0.36862746, 0.53333336],
       [0.1254902 , 0.36862746, 0.5372549 ],
       [0.1254902 , 0.36862746, 0.5411765 ]]],


     [[[0.49019608, 0.49019608, 0.45490196],
       [0.43137255, 0.39607844, 0.35686275],
       [0.4       , 0.3529412 , 0.3254902 ],
       ...,
       [0.7921569 , 0.8117647 , 0.8392157 ],
       [0.78431374, 0.8039216 , 0.83137256],
       [0.7921569 , 0.8156863 , 0.8392157 ]],

      [[0.5568628 , 0.57254905, 0.5568628 ],
       [0.57254905, 0.5647059 , 0.54509807],
       [0.6901961 , 0.6745098 , 0.6666667 ],
       ...,
       [0.7647059 , 0.7882353 , 0.8039216 ],
       [0.7764706 , 0.8039216 , 0.81960785],
       [0.8       , 0.827451  , 0.84313726]],

      [[0.7058824 , 0.7254902 , 0.7176471 ],
       [0.56078434, 0.57254905, 0.57254905],
       [0.6117647 , 0.6156863 , 0.6156863 ],
       ...,
       [0.47843137, 0.43529412, 0.44313726],
       [0.54509807, 0.5019608 , 0.5137255 ],
```

```
         [0.61960787, 0.5764706 , 0.5882353 ]],

        ...,

        [[0.40784314, 0.32156864, 0.16078432],
         [0.39607844, 0.3137255 , 0.15294118],
         [0.39607844, 0.31764707, 0.14901961],
         ...,
         [0.49411765, 0.40392157, 0.2627451 ],
         [0.49411765, 0.40392157, 0.27058825],
         [0.49019608, 0.39607844, 0.26666668]],

        [[0.40784314, 0.31764707, 0.15686275],
         [0.4117647 , 0.32941177, 0.16078432],
         [0.42745098, 0.34509805, 0.16862746],
         ...,
         [0.5411765 , 0.44313726, 0.30588236],
         [0.5372549 , 0.44313726, 0.3137255 ],
         [0.5372549 , 0.4392157 , 0.31764707]],

        [[0.4117647 , 0.3254902 , 0.16470589],
         [0.42352942, 0.34117648, 0.1764706 ],
         [0.4509804 , 0.36862746, 0.19607843],
         ...,
         [0.56078434, 0.45882353, 0.32156864],
         [0.56078434, 0.45490196, 0.32941177],
         [0.5647059 , 0.45490196, 0.3372549 ]]], dtype=float32)
[[0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]]
```

2. Build the model

In [6]:

```
input_layer = layers.Input((32, 32, 3))

x = layers.Conv2D(filters=32, kernel_size=3, strides=1, padding="same")(
    input_layer
)
x = layers.BatchNormalization()(x)
x = layers.LeakyReLU()(x)

x = layers.Conv2D(filters=32, kernel_size=3, strides=2, padding="same")(x)
x = layers.BatchNormalization()(x)
x = layers.LeakyReLU()(x)

x = layers.Conv2D(filters=64, kernel_size=3, strides=1, padding="same")(x)
x = layers.BatchNormalization()(x)
x = layers.LeakyReLU()(x)
```

```
x = layers.Conv2D(filters=64, kernel_size=3, strides=2, padding="same")(x)
x = layers.BatchNormalization()(x)
x = layers.LeakyReLU()(x)

x = layers.Flatten()(x)

x = layers.Dense(128)(x)
x = layers.BatchNormalization()(x)
x = layers.LeakyReLU()(x)
x = layers.Dropout(rate=0.5)(x)

x = layers.Dense(NUM_CLASSES)(x)
output_layer = layers.Activation("softmax")(x)

model = models.Model(input_layer, output_layer)

model.summary()
```

Model: "functional"

Layer (type)	Output Shape	Par
input_layer (InputLayer)	(None, 32, 32, 3)	
conv2d (Conv2D)	(None, 32, 32, 32)	
batch_normalization (BatchNormalization)	(None, 32, 32, 32)	
leaky_re_lu (LeakyReLU)	(None, 32, 32, 32)	
conv2d_1 (Conv2D)	(None, 16, 16, 32)	9
batch_normalization_1 (BatchNormalization)	(None, 16, 16, 32)	
leaky_re_lu_1 (LeakyReLU)	(None, 16, 16, 32)	
conv2d_2 (Conv2D)	(None, 16, 16, 64)	18
batch_normalization_2 (BatchNormalization)	(None, 16, 16, 64)	
leaky_re_lu_2 (LeakyReLU)	(None, 16, 16, 64)	
conv2d_3 (Conv2D)	(None, 8, 8, 64)	36
batch_normalization_3 (BatchNormalization)	(None, 8, 8, 64)	
leaky_re_lu_3 (LeakyReLU)	(None, 8, 8, 64)	
flatten (Flatten)	(None, 4096)	
dense (Dense)	(None, 128)	524
batch_normalization_4 (BatchNormalization)	(None, 128)	
leaky_re_lu_4 (LeakyReLU)	(None, 128)	
dropout (Dropout)	(None, 128)	
dense_1 (Dense)	(None, 10)	1
activation (Activation)	(None, 10)	

Total params: 592,554 (2.26 MB)

Trainable params: 591,914 (2.26 MB)

Non-trainable params: 640 (2.50 KB)

3. Train the model

In [7]:
```
opt = optimizers.Adam(learning_rate=0.0005)
model.compile(
```

```
    loss="categorical_crossentropy", optimizer=opt, metrics=["accuracy"]
)
```

In [8]:
```
model.fit(
    x_train,
    y_train,
    batch_size=32,
    epochs=10,
    shuffle=True,
    validation_data=(x_test, y_test),
)
```

```
Epoch 1/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 22s 14ms/step - accuracy: 0.4491 - loss: 1.57
17 - val_accuracy: 0.5734 - val_loss: 1.1989
Epoch 2/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 22s 14ms/step - accuracy: 0.5885 - loss: 1.16
19 - val_accuracy: 0.6432 - val_loss: 1.0085
Epoch 3/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 22s 14ms/step - accuracy: 0.6507 - loss: 1.00
18 - val_accuracy: 0.6388 - val_loss: 1.0335
Epoch 4/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 21s 14ms/step - accuracy: 0.6812 - loss: 0.91
23 - val_accuracy: 0.6693 - val_loss: 0.9263
Epoch 5/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 22s 14ms/step - accuracy: 0.7072 - loss: 0.84
32 - val_accuracy: 0.6922 - val_loss: 0.8831
Epoch 6/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 22s 14ms/step - accuracy: 0.7188 - loss: 0.80
36 - val_accuracy: 0.6488 - val_loss: 1.0208
Epoch 7/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 21s 13ms/step - accuracy: 0.7362 - loss: 0.75
46 - val_accuracy: 0.7145 - val_loss: 0.8231
Epoch 8/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 21s 13ms/step - accuracy: 0.7497 - loss: 0.71
74 - val_accuracy: 0.7202 - val_loss: 0.8095
Epoch 9/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 22s 14ms/step - accuracy: 0.7615 - loss: 0.68
45 - val_accuracy: 0.7223 - val_loss: 0.8001
Epoch 10/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 22s 14ms/step - accuracy: 0.7730 - loss: 0.65
26 - val_accuracy: 0.7277 - val_loss: 0.7982
```

Out[8]:
```
<keras.src.callbacks.history.History at 0x14f0ea680>
```

4. Evaluation

In [9]:
```
model.evaluate(x_test, y_test, batch_size=1000)
```

```
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 57ms/step - accuracy: 0.7277 - loss: 0.7982
```

Out[9]:
```
[0.7981552481651306, 0.7276999950408936]
```

In [10]:
```
CLASSES = np.array(
    [
        "airplane",
```

```
        "automobile",
        "bird",
        "cat",
        "deer",
        "dog",
        "frog",
        "horse",
        "ship",
        "truck",
    ]
)

preds = model.predict(x_test)
preds_single = CLASSES[np.argmax(preds, axis=-1)]
actual_single = CLASSES[np.argmax(y_test, axis=-1)]
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
```

In [11]:

```
import matplotlib.pyplot as plt

n_to_show = 10
indices = np.random.choice(range(len(x_test)), n_to_show)

fig = plt.figure(figsize=(15, 3))
fig.subplots_adjust(hspace=0.4, wspace=0.4)

for i, idx in enumerate(indices):
    img = x_test[idx]
    ax = fig.add_subplot(1, n_to_show, i + 1)
    ax.axis("off")
    ax.text(
        0.5,
        -0.35,
        "pred = " + str(preds_single[idx]),
        fontsize=10,
        ha="center",
        transform=ax.transAxes,
    )
    ax.text(
        0.5,
        -0.7,
        "act = " + str(actual_single[idx]),
        fontsize=10,
        ha="center",
        transform=ax.transAxes,
    )
    ax.imshow(img)
```

In [ ]: