

```
In [1]: import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
```

```
In [2]: (train_images, train_labels), (test_images, test_labels) = datasets.cifar100.load_data()
train_images, test_images = train_images / 255.0, test_images / 255.0
```

Downloading data from <https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz> (https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz)
170498071/170498071 [=====] - 383s 2us/step

```
In [5]: class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
plt.figure(figsize=(10,10))
for i in range(10):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i])
    plt.xlabel(class_names[train_labels[i][0]])
    plt.show()
```



frog



truck



truck



deer



automobile



automobile



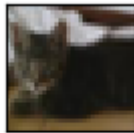
bird



horse



ship



cat

```
In [7]: 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()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
=====		
conv2d_3 (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d_2 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_4 (Conv2D)	(None, 13, 13, 64)	18496
max_pooling2d_3 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_5 (Conv2D)	(None, 4, 4, 64)	36928
flatten_1 (Flatten)	(None, 1024)	0
dense_2 (Dense)	(None, 64)	65600
dense_3 (Dense)	(None, 10)	650
=====		
Total params: 122570 (478.79 KB)		
Trainable params: 122570 (478.79 KB)		
Non-trainable params: 0 (0.00 Byte)		

```
In [ ]: model.compile(optimizer='adam', loss=tf.keras.losses.
↳SparseCategoricalCrossentropy(from_logits=True), metrics=['accuracy'])
epochs = 1
h = model.fit(train_images, train_labels, epochs=epochs,
↳validation_data=(test_images, test_labels))
```