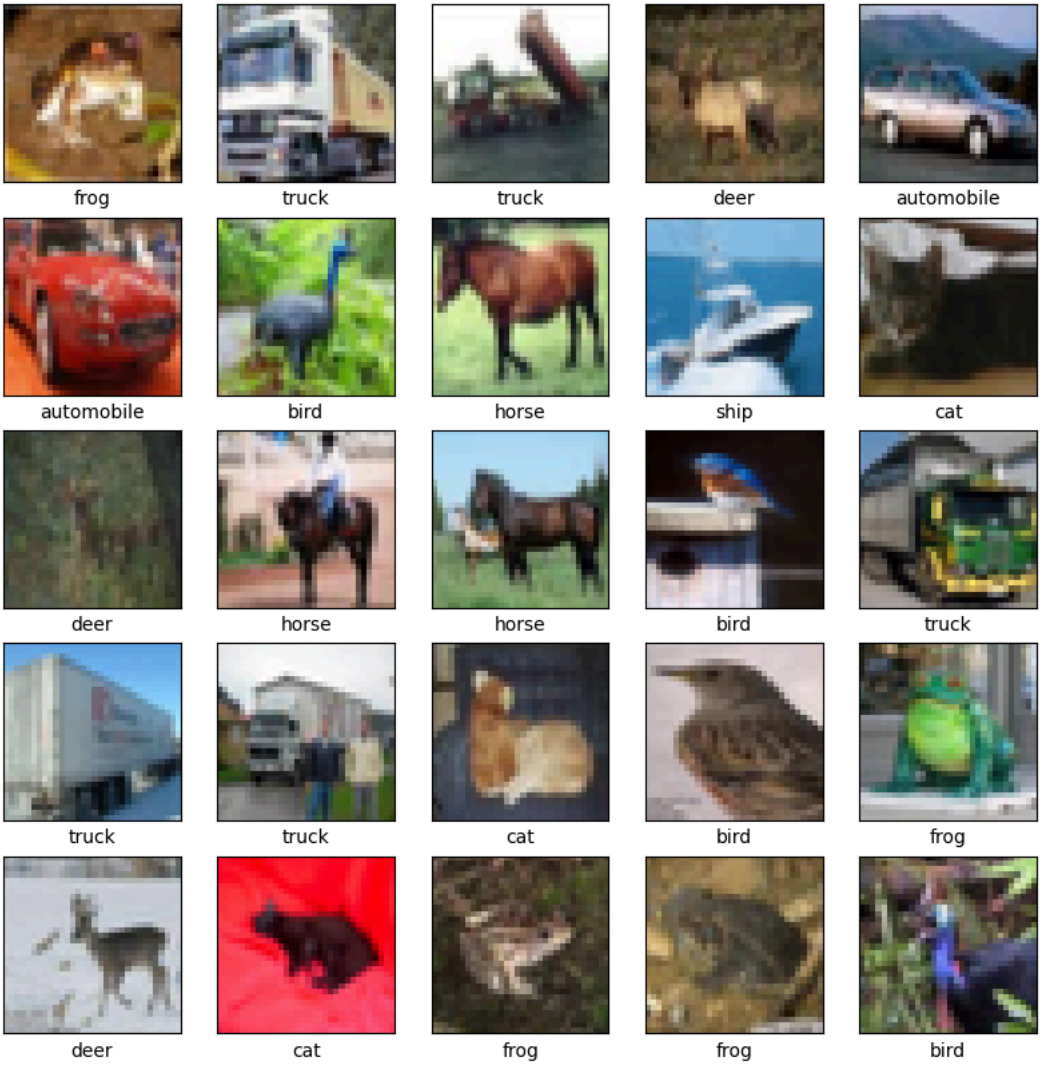
```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt


(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
```

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

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

plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i])
    # The CIFAR labels happen to be arrays,
    # which is why you need the extra index
    plt.xlabel(class_names[train_labels[i][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.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 32)        896

 max_pooling2d (MaxPooling2  (None, 15, 15, 32)        0
 D)

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

 max_pooling2d_1 (MaxPoolin  (None, 6, 6, 64)          0
 g2D)

 conv2d_2 (Conv2D)           (None, 4, 4, 64)          36928

=================================================================
Total params: 56320 (220.00 KB)
Trainable params: 56320 (220.00 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 32)        896

 max_pooling2d (MaxPooling2  (None, 15, 15, 32)        0
 D)

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

 max_pooling2d_1 (MaxPoolin  (None, 6, 6, 64)          0
 g2D)

 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: 122570 (478.79 KB)
Trainable params: 122570 (478.79 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

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

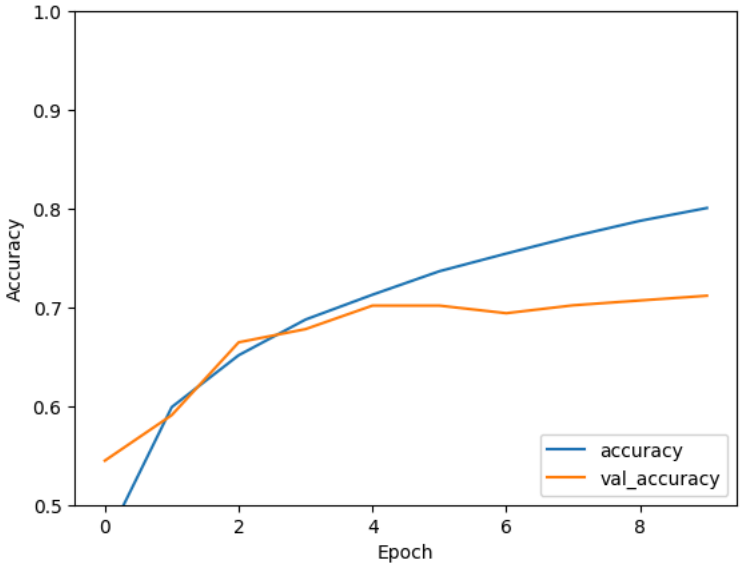
```
history = model.fit(train_images, train_labels, epochs=10,
                    validation_data=(test_images, test_labels))
```

```
Epoch 1/10
1563/1563 [==============================] - 87s 55ms/step - loss: 1.4878 - accuracy: 0.4613 - val_loss: 1.2596 - val_accuracy: 0.54
Epoch 2/10
1563/1563 [==============================] - 73s 46ms/step - loss: 1.1362 - accuracy: 0.5990 - val_loss: 1.1582 - val_accuracy: 0.59
Epoch 3/10
1563/1563 [==============================] - 74s 48ms/step - loss: 0.9895 - accuracy: 0.6515 - val_loss: 0.9481 - val_accuracy: 0.66
Epoch 4/10
1563/1563 [==============================] - 72s 46ms/step - loss: 0.8911 - accuracy: 0.6877 - val_loss: 0.9214 - val_accuracy: 0.67
Epoch 5/10
1563/1563 [==============================] - 73s 47ms/step - loss: 0.8158 - accuracy: 0.7128 - val_loss: 0.8732 - val_accuracy: 0.76
Epoch 6/10
1563/1563 [==============================] - 77s 49ms/step - loss: 0.7563 - accuracy: 0.7366 - val_loss: 0.8722 - val_accuracy: 0.76
Epoch 7/10
1563/1563 [==============================] - 70s 45ms/step - loss: 0.6972 - accuracy: 0.7545 - val_loss: 0.9004 - val_accuracy: 0.69
Epoch 8/10
1563/1563 [==============================] - 70s 45ms/step - loss: 0.6436 - accuracy: 0.7719 - val_loss: 0.8903 - val_accuracy: 0.76
Epoch 9/10
1563/1563 [==============================] - 70s 45ms/step - loss: 0.5998 - accuracy: 0.7877 - val_loss: 0.8858 - val_accuracy: 0.76
Epoch 10/10
1563/1563 [==============================] - 70s 45ms/step - loss: 0.5626 - accuracy: 0.8007 - val_loss: 0.8932 - val_accuracy: 0.71
```

```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')

test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)
```

```
313/313 - 3s - loss: 0.8932 - accuracy: 0.7118 - 3s/epoch - 11ms/step
```




```
print(test_acc)
```

```
0.7117999792098999
```