**Basic\_Classification**

Loading the dataset returns four NumPy arrays:

* The train\_images and train\_labels arrays are the *training set*—the data the model uses to learn.
* The model is tested against the *test set*, the test\_images, and test\_labels arrays.

**Build the model**

* Building the neural network requires configuring the layers of the model, then compiling the model.

### Setup the layers

The basic building block of a neural network is the **layer**. Layers extract representations from the data fed into them. And, hopefully, these representations are more meaningful for the problem at hand.

Most of deep learning consists of chaining together simple layers. Most layers, like tf.keras.layers.Dense, have parameters that are learned during training.

The first layer in this network, tf.keras.layers.Flatten, transforms the format of the images from a 2d-array (of x by y pixels), to a 1d-array of x \* y = z pixels. Think of this layer as unstacking rows of pixels in the image and lining them up. This layer has no parameters to learn; it only reformats the data.

After the pixels are flattened, the network consists of a sequence of two tf.keras.layers.Dense layers. These ar**e densely-connected, or fully-connected, neural layers.** The first Dense layer has 128 nodes (or neurons). The second (and last) layer is a **10-node softmax layer**—this returns an array of 10 probability scores that sum to 1. Each node contains a score that indicates the probability that the current image belongs to one of the 10 classes.

### Compile the model

Before the model is ready for training, it needs a few more settings. These are added during the model's compile step:

* **Loss function** —This measures how accurate the model is during training. We want to minimize this function to "steer" the model in the right direction.
* **Optimizer** —This is how the model is updated based on the data it sees and its loss function.
* **Metrics**—Used to monitor the training and testing steps. The following example uses accuracy, the fraction of the images that are correctly classified.

**Train the model**

Training the neural network model requires the following steps:

1. Feed the training data to the model—in this example, the train\_images and train\_labels arrays.
2. The model learns to associate images and labels.
3. We ask the model to make predictions about a test set—in this example, the test\_images array. We verify that the predictions match the labels from the test\_labels array.

To start training, call the model.fit method—the model is "fit" to the training data:

As the model trains, the loss and accuracy metrics are displayed.

## Evaluate accuracy

Next, compare how the model performs on the test dataset: model.evaluate

## Make predictions

With the model trained, we can use it to make predictions about some images. model.predict

model.predict returns a list of lists, one for each image in the batch of data. Grab the predictions for our (only) image in the batch: