Classification

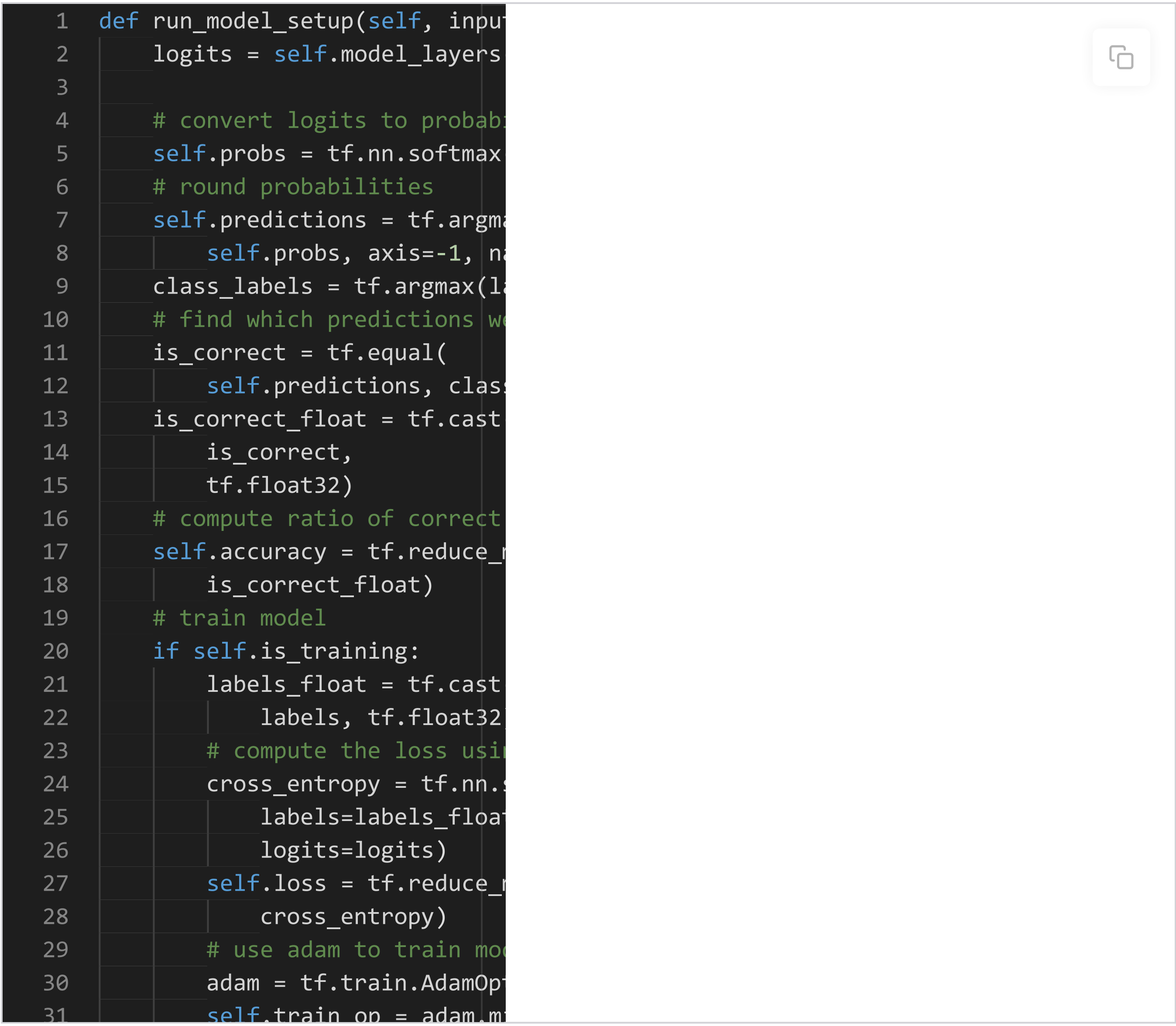
Learn how to train and use the CNN model for MNIST datasets.

Chapter Goals:

- Understand how hand-drawn digits are processed and passed into the model for classification

A. Model logistics

The `run_model_setup` function below shows how to set up and train the CNN we've coded:



```
def run_model_setup(self, inpu
    logits = self.model_layers

    # convert logits to probab
    self.probs = tf.nn.softmax
    # round probabilities
    self.predictions = tf.argm
        self.probs, axis=-1, n
    class_labels = tf.argmax(l
    # find which predictions w
    is_correct = tf.equal(
        self.predictions, clas
    is_correct_float = tf.cast
        is_correct,
        tf.float32)
    # compute ratio of correct
    self.accuracy = tf.reduce_
        is_correct_float)
    # train model
    if self.is_training:
        labels_float = tf.cast
            labels, tf.float32
        # compute the loss usi
        cross_entropy = tf.nn.
            labels=labels_floa
            logits=logits)
        self.loss = tf.reduce_
            cross_entropy)
        # use adam to train mo
        adam = tf.train.AdamOp
        self.train_op = adam.m
```

For more explanation of the code, see the Machine Learning for Software

Engineers course on Educative.

B. Real data

After training a model on the MNIST dataset, it is ready to classify real hand-drawn digits. Using the techniques from the **Image Processing** section, we can decode the hand-drawn image to obtain its pixel data (in grayscale format) and then resize it to the same dimensions as the MNIST image data. Since our model inputs have shape `(batch_size, input_dim**2)`, we flatten the image's pixel data and reshape it to `(1, input_dim**2)`.

C. Classifying hand-drawn digits

The code below runs a digit classifier implemented in the backend. It will prompt you to draw a digit. The model will predict which digit you drew.

```
run_digit_recognizer()
```