Describe how you set up and trained your model, and explain the parameters you chose and why?

The MNIST dataset was loaded using the mnist.load_data() function. This dataset contains 60,000 training images and 10,000 testing images, each of which is a 28x28 grayscale image of a handwritten digit.

The images were reshaped to have a single color channel (since they're grayscale), and their pixel intensities were normalized to the range [0, 1] by dividing by 255. This is a common preprocessing step that can help the model learn more effectively.

The labels, which are the actual digits each image represents, were one-hot encoded. This means they were converted from a single number to a 10-dimensional vector with a 1 in the position of the correct digit and 0s elsewhere. For example, the digit 3 would be represented as [0, 0, 0, 1, 0, 0, 0, 0, 0, 0].

The model was built using Keras's Sequential API, which allows you to build models layer-by-layer in a straightforward way. The model consists of a convolutional layer, a max pooling layer, a flattening layer, and a dense (fully connected) layer.

The convolutional layer has 32 filters, each of which is 3x3. The filters are slid over the input image, and each computes a weighted sum of the pixels it covers. The weights are learned during training. The 'relu' activation function is used to introduce non-linearity into the model.

The max pooling layer reduces the spatial dimensions of the input by taking the maximum value over 2x2 patches. This helps to make the model invariant to small translations and reduces the computation needed for subsequent layers.

The flattening layer reshapes the 3D output of the previous layer into a 1D vector, so it can be input to the dense layer.

The dense layer has 10 units and uses the 'softmax' activation function, which outputs a probability distribution over the 10 digit classes. This means the output of the model for each input image is a vector of 10 probabilities that sum to 1, and the digit with the highest probability is the model's prediction.

The model was compiled with the ‘adam’ optimizer, the ‘categorical_crossentropy’ loss function, and it was instructed to compute ‘accuracy’ as a metric during training. The ‘adam’ optimizer is a variant of gradient descent that adapts the learning rate for each weight individually, which can lead to more effective learning. The ‘categorical_crossentropy’ loss function is appropriate for multiclass classification problems like this one, and measures the dissimilarity between the predicted and true distributions.

The model was trained for 10 epochs with a batch size of 64, using 20% of the training data as a validation set. An epoch is one pass through the entire training dataset, and the batch size is the number of training examples used to compute the gradient on each step. The validation set is used to monitor the model’s performance on unseen data during training.