**Method Used and Implementation:**

Using the MNIST dataset, the given code implements a convolutional neural network (CNN) for character recognition. The 28x28 grayscale pictures of handwritten letters from A to Z represent the MNIST dataset.

The load\_data() function reads in the images and their accompanying labels, resizes the images to a defined size (IMG\_SIZE), and converts the images to grayscale as the first part in the implementation. The pixels are resized to have values between 0 and 1 and the images are then rearranged into a 4D array (with dimensions [number of photos, IMG\_SIZE, IMG\_SIZE, 1]). Using Keras' to\_categorical function, the labels are also transformed into a category format.

The model architecture consists 2- dimensional convolutional layers (conv-2D), max pooling layers to extract the maximum features in each convolution result, dropout layers to overcome model overfitting, and a dense output layer with a softmax activation function for maximum A to Z MNIST images classification. The first convolutional layer has 32 filters with a 3x3 kernel size and 'same' padding, followed by a ReLU activation function. This is followed by another convolutional layer with 32 filters and ReLU activation, a max pooling layer with a pool size of 2x2 to downsample the output of the convolutional layers, and a dropout layer with a rate of 0.25 to reduce overfitting. Two more convolutional layers with 64 filters and ReLU activation are added, followed by another max pooling layer and dropout layer. Adding more layers help to learn more complex and hidden features in images. The output is then flattened and fed through 512 neurons in a dense layer with ReLU activation to add nonlinearity. To further prevent overfitting and increase the generalizability of the model, a second dropout layer with a rate of 0.5 is added. The output layer is then added, along with an equal number of neurons as categories and a softmax activation function.

The model is subsequently built using the compile() method together with an accuracy metric, an Adam optimizer, and a categorical crossentropy loss function. The model is trained using the fit() technique, with a batch size of 32 and a validation split of 0.2. TensorBoard is used to track the training's progress, and the logs are saved to a dedicated directory for viewing.

The implementation, as a whole, follows the conventional procedures for training a CNN model, including loading the data, specifying the model architecture, compiling the model, and training the model using the fit technique. Depending on the particular problem and dataset, different model architecture and hyperparameter choices may be made (e.g., layer count, filter size, kernel size, dropout rate, etc.).