**Training the network to read handwritten digits**

## **MNIST Handwritten Digit Classification Dataset**

The [MNIST dataset](https://en.wikipedia.org/wiki/MNIST_database) is an acronym that stands for the Modified National Institute of Standards and Technology dataset.

It is a dataset of 60,000 small square 28×28 pixel grayscale images of handwritten single digits between 0 and 9.

The task is to classify a given image of a handwritten digit into one of 10 classes representing integer values from 0 to 9, inclusively.

### **Load Dataset**

For example, we know that the images are all pre-aligned (e.g. each image only contains a hand-drawn digit), that the images all have the same square size of 28×28 pixels, and that the images are grayscale.

We can, therefore, use a one hot encoding for the class element of each sample, transforming the integer into a 10 element binary vector with a 1 for the index of the class value, and 0 values for all other classes. We can achieve this with the to\_categorical() utility function.

### **Prepare Pixel Data**

We know that the pixel values for each image in the dataset are unsigned integers in the range between black and white, or 0 and 255.

A good starting point is to [normalize the pixel values](https://machinelearningmastery.com/how-to-normalize-center-and-standardize-images-with-the-imagedatagenerator-in-keras/) of grayscale images, e.g. rescale them to the range [0,1]. This involves first converting the data type from unsigned integers to floats, then dividing the pixel values by the maximum value.

### **Define Model**

Next, we need to define a baseline convolutional neural network model for the problem.

The model has two main aspects: the feature extraction front end comprised of convolutional and pooling layers, and the classifier backend that will make a prediction.

Given that the problem is a multi-class classification task , this will also require the use of a softmax activation function. Between the feature extractor and the output layer, we can add a dense layer to interpret the features.

All layers will use the [ReLU activation function](https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/) and the He weight initialization scheme, both best practices.

The [categorical cross-entropy](https://machinelearningmastery.com/cross-entropy-for-machine-learning/) loss function will be optimized, suitable for multi-class classification, and we will monitor the classification accuracy metric, which is appropriate given we have the same number of examples in each of the 10 classes.

### **Evaluate Model**

After the model is defined, we need to evaluate it.

We will train the baseline model for a modest 30 training epochs with a default batch size of 150 examples. The test set for each fold will be used to evaluate the model both during each epoch of the training run, so that we can later create learning curves, and at the end of the run, so that we can estimate the performance of the model. As such, we will keep track of the resulting history from each run, as well as the classification accuracy of the fold.





