**Machine Learning and Neural Networks Theory**

**Title**

Options:

* Option1
* Option2

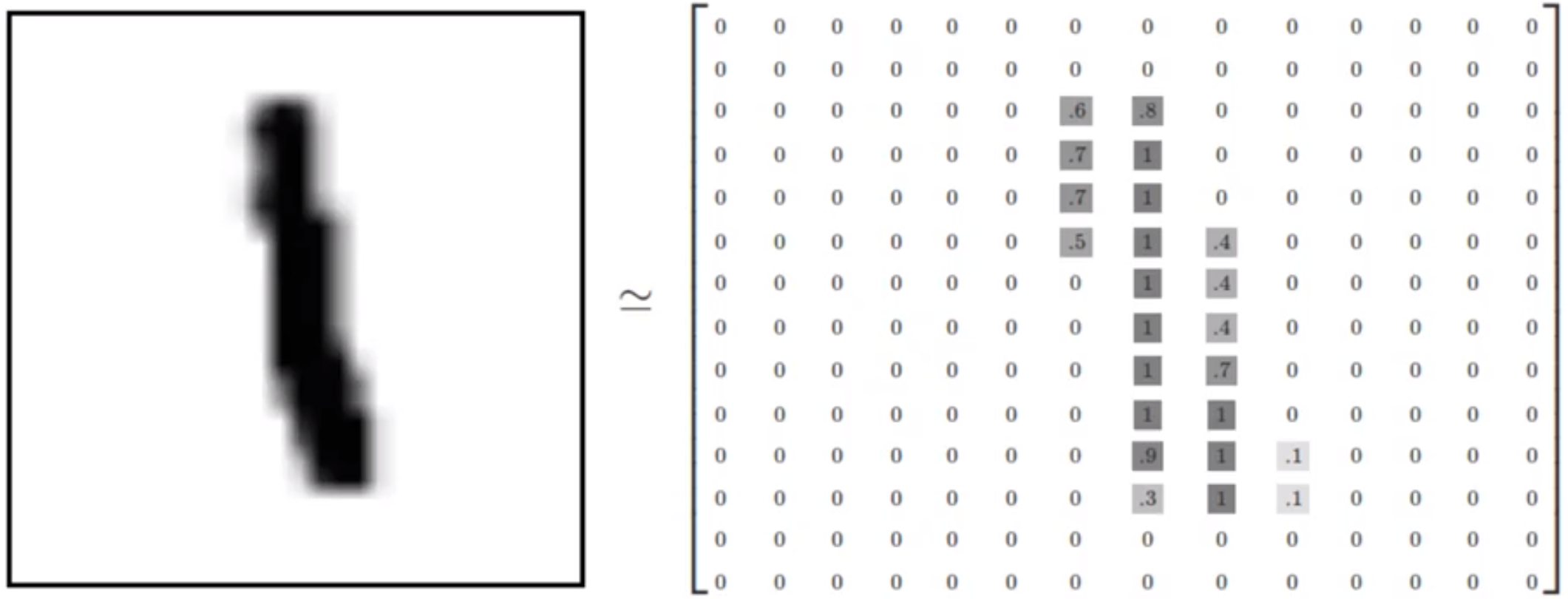
Text

1. **Convolutional Neural Networks**
   1. **MNIST database**

The MNIST database of handwritten digits from 0 to 9, available from their webpage, has a training set of 55.000 examples, a test set of 10.000 examples and a validation set of 5.000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

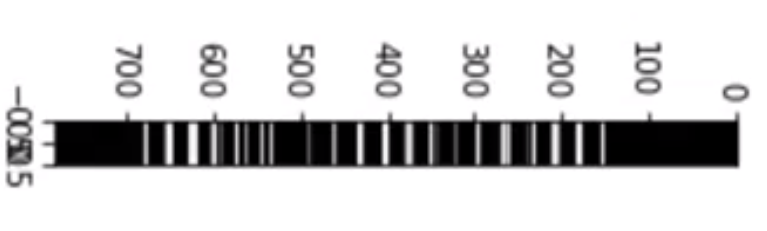
It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting.

A single digit image can be represented as an array of 28x28 pixels. The values of the array represent the grayscale image.



*Image 1. Representation of an image as a 2-D array.*

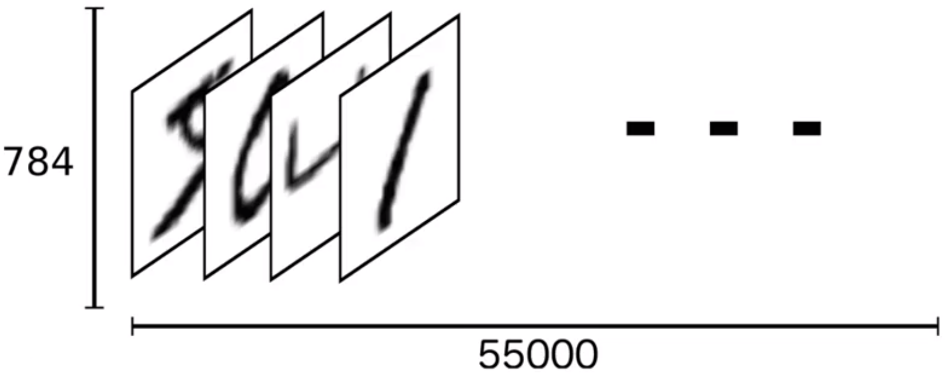
We can flatten this array to a 1-D vector of 784 numbers (28x28). Either (784,1) or (1,784) is fine, as long as the dimensions are consistent.



*Image 2. Representation of an image as a 1-D array.*

Flattering out the image, ends up removing some 2-D information, such as the relationship of a pixel to its neighboring pixels.

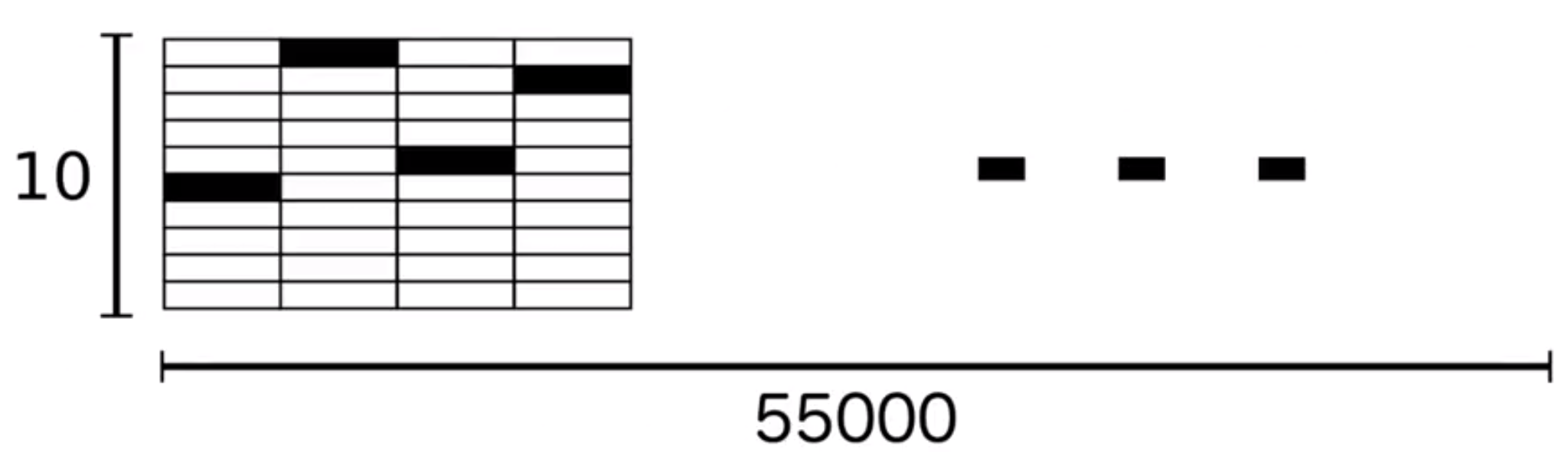
We can think the entire group of the 55.000 images as a tensor (an N-dimensional array). For labeling purposes, there’s something called the **One-Hot Encoding**. This means that instead of having string labels, we’ll have a single array for each image.



*Image 3. A tensor or N-dimensional array of images.*

The label is represented based off the index position in the label array. The corresponding label will be a 1 at the index location and zero every where else. For example, an image of “4” would have this label: [0,0,0,0,1,0,0,0,0,0].

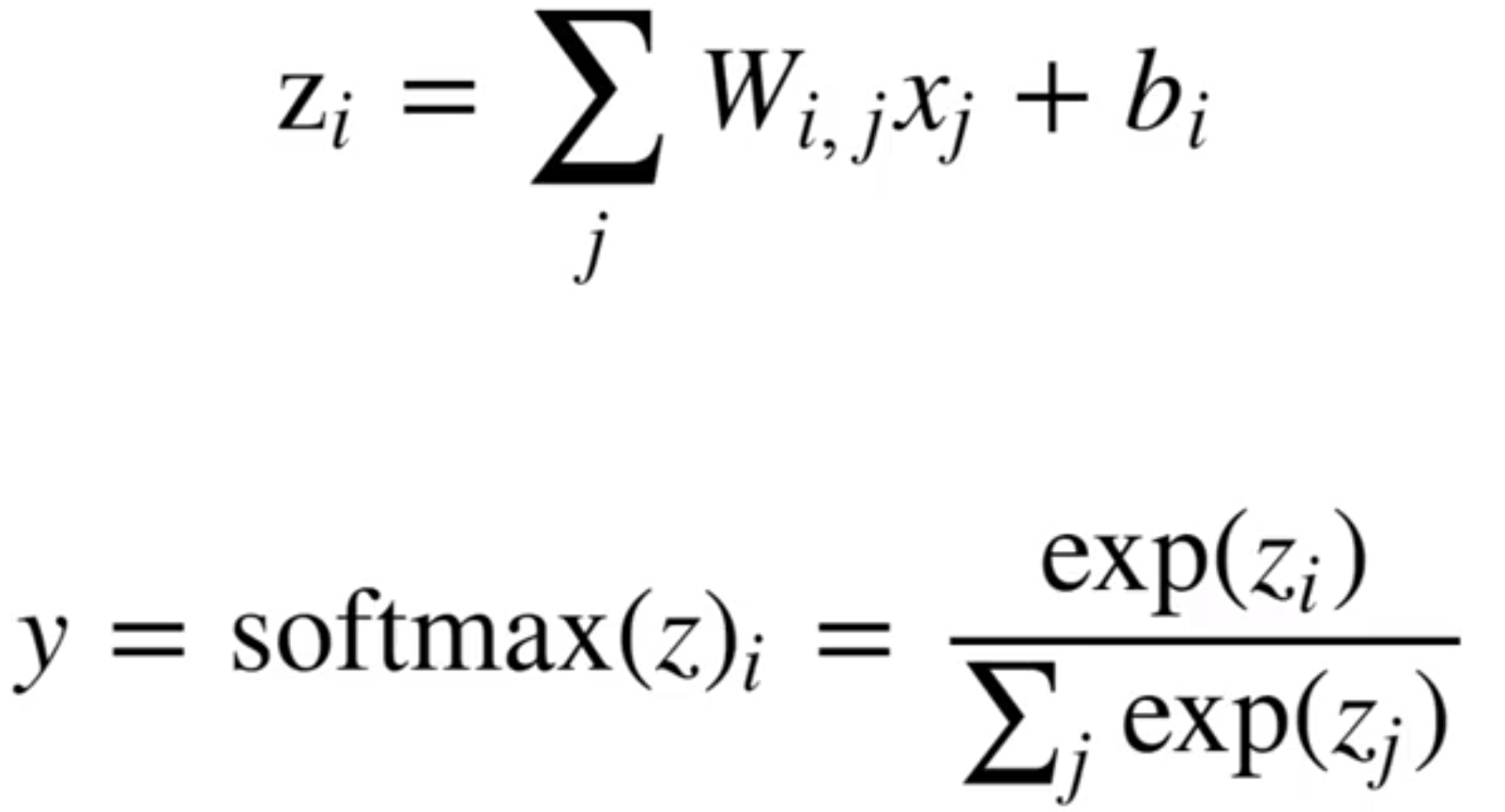
As a result, the labels for the training data end up being a large 2-D array. Where 10 are the possible digits (0 to 9) and 55.000 are all the images.



*Image 4. Large 2-D array of labels.*

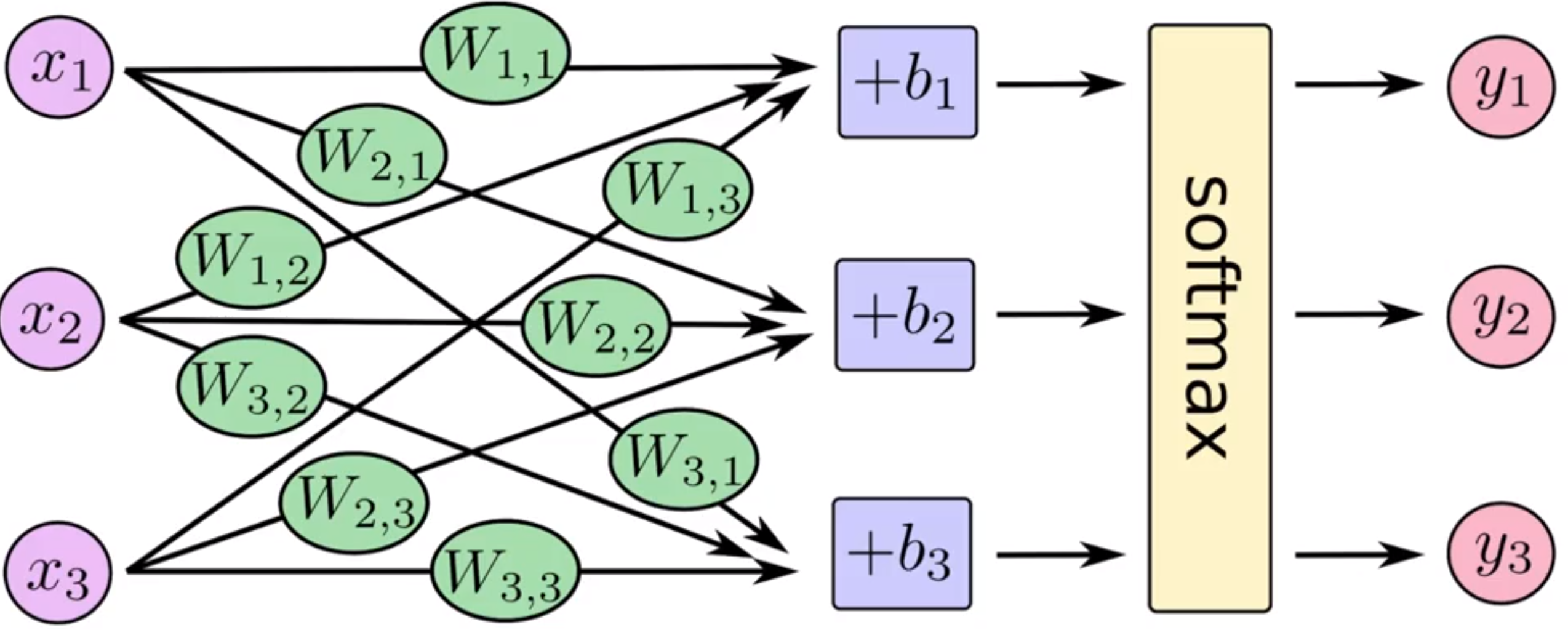
* 1. **MNIST basic approach**

We’ll use a Softmax Regression Approach, which returns a list of values between 0 and 1 that add up to one, in other words, a list of probabilities. Also, we’ll use Softmax as our activation function.



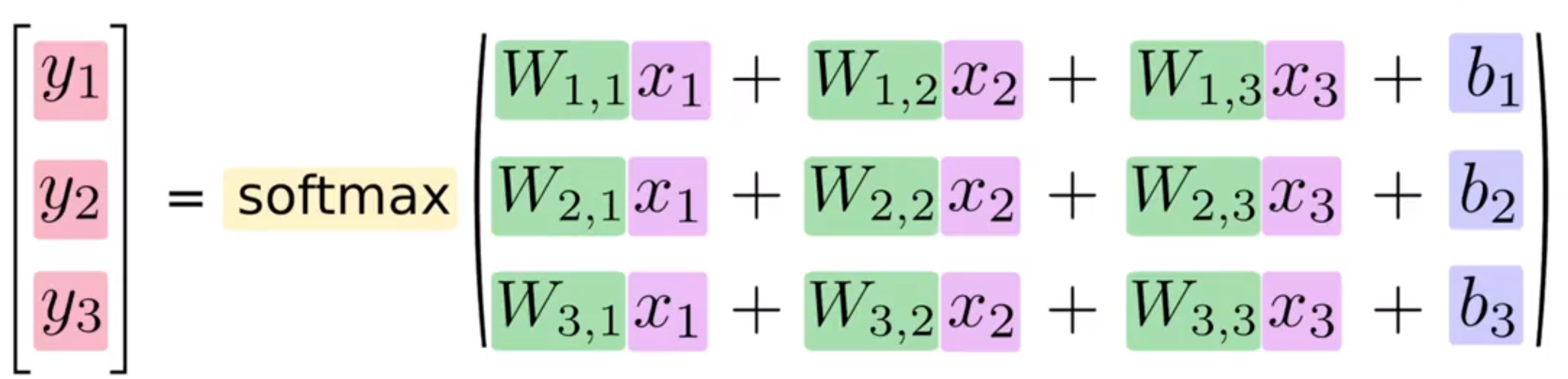
*Image 5. Softmax activation function.*

In a network format, we’ll get something like this:



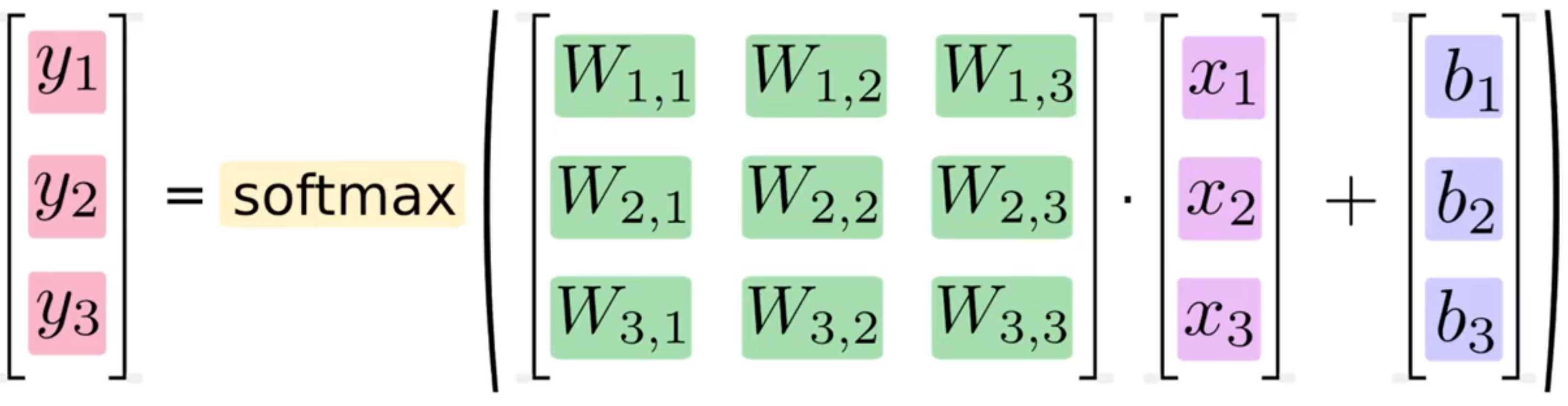
*Image 6. Softmax in a NN graphic.*

If we put it as an equation:



*Image 7. Softmax matrix equation for NN.*

Turning it into a matrix multiplication, which is computationally more efficient:



*Image 8. Softmax matrix equation as multiplication for NN.*

With that, we should be able to get our predictions and compare it to the true data in order to see the accuracy. We will see that this can be improved with CNN.

* 1. **MNIST basic approach**

We’ll use a Softmax Regression Approach, which returns a list of values between 0 and 1 that add up to one, in other words, a list of probabilities. Also, we’ll use Softmax as our activation function.