PROJECT REPORT – DIGIT RECOGNITION

1. Introduction

Since the AI is swiftly making its presence felt in the everyday world, we are trying to observe some of its basic functionalities and catch a glimpse of how it is working in the background. Thus, we are approaching a rather common task in Machine Learning, digit recognition, in the most widely-spread manner: through Supervised Learning.

2. Problem Definition and Algorithm

2.1. Task Definition

Can a machine properly detect a digit from an image?

2.2. Algorithm Definition

-> Collect a given image

-> Resize it as a 28 x 28 matrix

-> Make it Grayscale

-> Adjust the contrast so that the digit is easier to recognize

-> Flatten the matrix into an array

-> Use a trained neural network to decide what digit was initially drawn

3. Experimental Evaluation

The neural network is trained and tested on the usual MNIST database, which is also a module in python. We adjusted the structure of the neural network so it won’t have low accuracy or so it won’t be so computationally expensive.

By default, we are going to have 784 (28 x 28) neurons on the input layer and 10 on the output layer. Now, we are going to adjust the hidden layers.

a)

O imagine care conține text, portocaliu

Descriere generată automat



b) 10 epochs

O imagine care conține text

Descriere generată automat



c) 20 epochs



d) 40 epochs



e) 10 epochs

O imagine care conține text

Descriere generată automat



f) 40 epochs



g) 10 epochs

O imagine care conține text

Descriere generată automat



Discussion for the number of layers chosen:

Usually, the larger the number of layers, the better and we observed that during the trials. Those were only the threshold ones.

Up to some point, the larger the number of epochs the network trained for, the better the results. That was obvious for the smaller networks, but for the larger ones they just became computationally expensive. For instance, for the last example, after other tests, after the 6th epoch, the difference is minimal, thus more than 10 epochs would be futile.

Initially we tested the network on 28 x 28 pixels Paint image and the results were ok, some numbers were easier to recognise than others. After that, the problem that we had to convert a real image into what we need arose.

Discussion for parsing the image:

Since the dataset is not necessarily the most convenient one, the given image must be in a certain format. Thus, we have to somehow get the image to be 28x28 and black and white.

We used the basic resize and grayscale methods, but the digit was not exactly like those in the dataset and the predictions were a little bit off. After some photoshop editing, the conclusion was that we must increase the contrast of the photo:

Thus, the pixels that are at least 97% of the pixel with the maximum intensity will also become maximum, and the other way around. After several tries, those were the percentages that gave the best aspect to the compressed, grayscaled image.

4. Conclusion

As long as the images are in the right format or maybe from the same source, a machine can detect a digit with 97% accuracy, but, when the format is slightly different, it starts to behave a little bit off since some digits are more distinctive than others. It is not really “intelligent”.