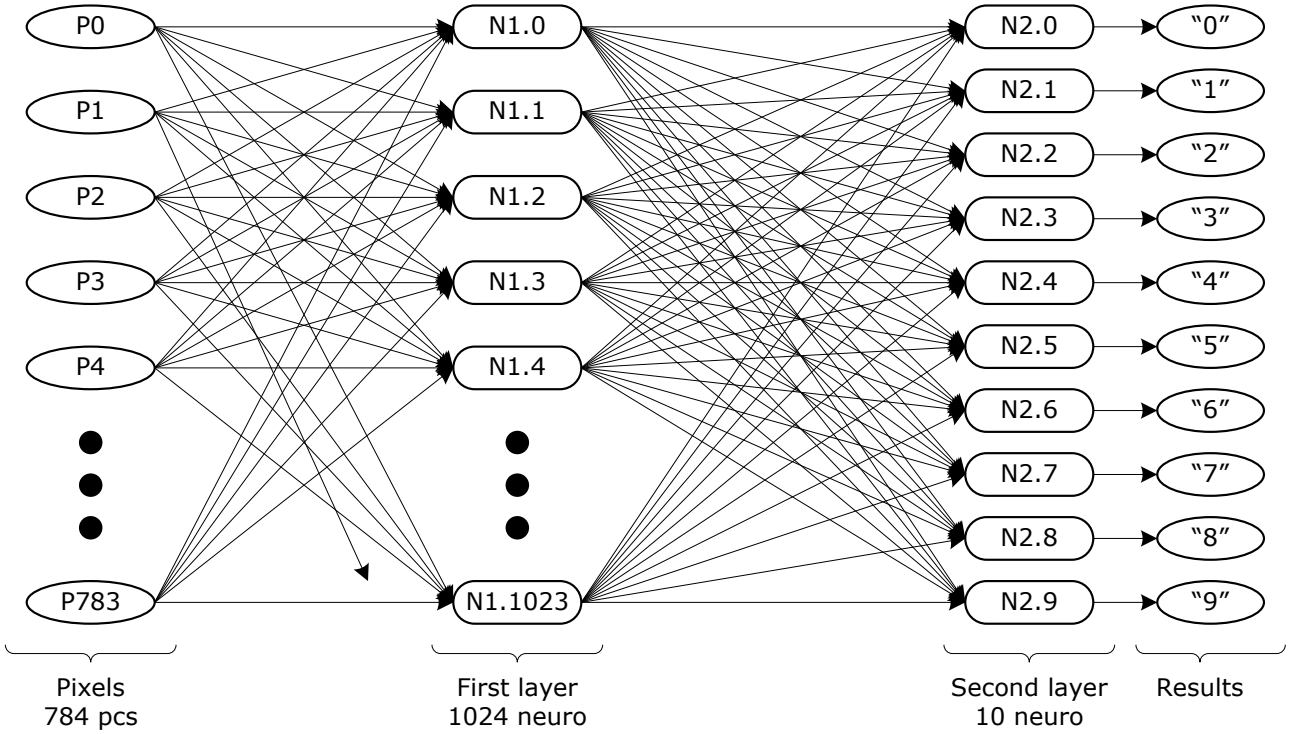
Perceptron structure.

The problem of digit recognition was solved using a perceptron, shown in the figure below, loaded into an experimental neuroblock.



The original images of 28*28 pixels in size were not preliminarily processed, the pixel values (0.0 or 1.0) were applied directly to the inputs of the first layer of neurons. Each neuron of the first layer has 784 inputs.

The number of neurons in the first layer was chosen to be 1024. This number was chosen on the assumption that 100 spelling variations should be provided for each digit from 0 to 9. Then each neuron of the first layer after training will respond to one style of writing a certain number. It has been rounded up to a "beautiful" 1024 just for ease of viewing and analyzing the perceptron table.

The results of the work of 1024 neurons of the first layer were fed to the inputs of 10 neurons of the second layer. And these 10 neurons made a decision about which number was presented for identification. At the output of each of the 10 neurons of the output layer, values from 0 to 1.0 appear. It was assumed that the second layer would accumulate the reactions of the neurons of the first layer, evaluating the images of which numbers prevailed according to the neurons of the first layer.

The final decision as to which digit is shown in the input is made based on which of the 10 outputs will have the highest number. This was done in software, using the instructions of the main core unit.

Base bias constants for all network neurons were set equal to 0.

The sigmoid was chosen as the output functions of all neurons of the perceptron.

The calculation of errors for the output layer was carried out according to the formula:

$$\delta_n = \begin{Bmatrix} 0 - y, & if\ n\ \neq label \\ 1 - y, & if\ n = label \end{Bmatrix}$$

Where:

n – is the number of the output layer neuron (from 0 to 9).

label – the value of the digit shown in the presented picture (from 0 to 9).

y – output value of the neuron.

The calculation of the errors of the first layer of neurons was carried out according to the formula:

$$\delta_m = \sum_{n=0}^{9} W_n^m * \delta_n$$

Where:

δ_m- a set of errors (1024 numbers) of neurons of the first layer, where m varies from 0 to 1023.

W – weight coefficients at the inputs of neurons of the second layer.

δ_n- output layer neuron errors.

Further, the calculation of new weight coefficients for the first layer of neurons was carried out according to the formula:

$$W_x^m = W_x^m + \eta * \delta^m * \frac{df_1^m(e)}{de} * x_x$$

Where:

index x changes from 0 to 783.

x_x- input values, 784 numbers.

m – index of input neurons, varies from 0 to 1023.

η – coefficient regulating the learning rate. In this case, it was chosen to be 0.05, one might say at random. It did not change during the learning process.

δ^m- errors of neurons of the first layer.

$\frac{df_1^m(e)}{de}$ - derivative of the neuron activation function. For a sigmoid, it is equal to:

$$\frac{df_1^m(e)}{de} = S(x) * (1 - S(x))$$

Where:

S(x) – value at the output of the neuron.

The calculation of new weight coefficients for the output layer of neurons was carried out according to the formula:

$$W_m^n = W_m^n + \eta * \delta^n * \frac{df_2^n(e)}{de} * y_m$$

Where:

n – varies from 0 to 9 (according to the number of output layer neurons).

m – numbers of inputs of each neuron from 0 to 1023.

δ^n- output layer neuron errors.

y_m- the result of the m-th neuron of the first layer.