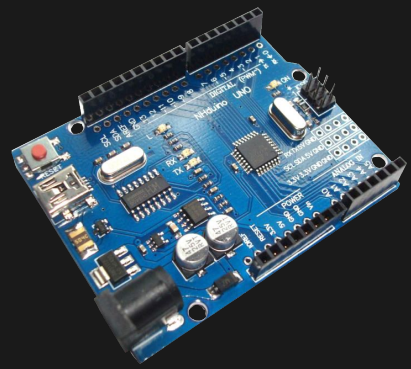
Deep neural network on Arduino

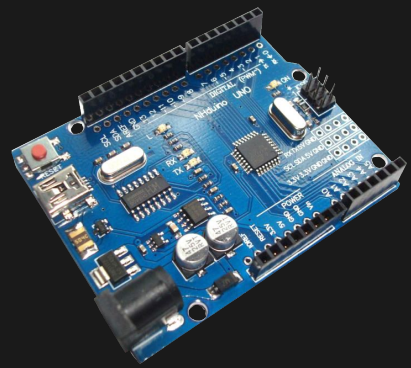
Michal CHOVANEC, PhD



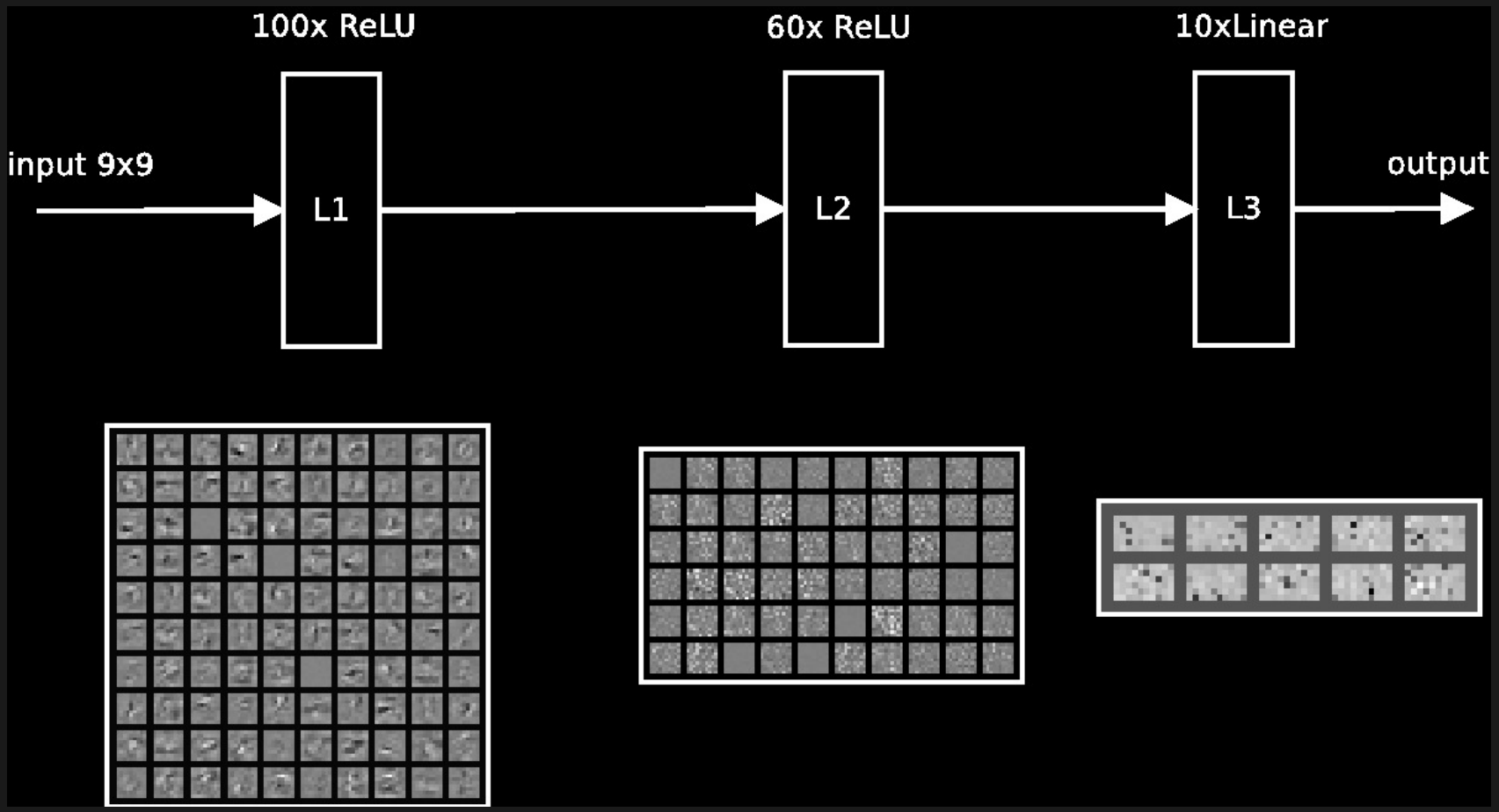
Goal

Train deep neural network on GPU and run it on Arduino

- Dataset MNIST handwritten digits down sampled from 28x28 to 9x9 pixels
- Fully connected network with two hidden ReLU layers



Network topology



Training on GPU (1080ti)

Converting weights

use simple linear mapping :

$$w_{int} = 127 \frac{w_{float}}{w_{max}}$$
$$scaling\_factor = 1000 w_{max}$$

```
#define layer_1_scaling_factor  515

signed char weights_layer_1[8200]={
-34, -17, -4, 18, 24, -9, 8, -7 ... };
```

Layer kernel

```
void matrix_vector_dot_kernel(t_nn_buffer *output, t_nn_buffer *input,
                              signed char *weights,
                              unsigned int input_size,
                              unsigned int output_size,
                              unsigned int weights_scaling)
{
  unsigned int w_ptr = 0;
  for (unsigned int j = 0; j < output_size; j++) {
    unsigned int input_ptr  = 0;
    unsigned int size       = input_size+1;

    long int sum = 0;

    while (size >= 4) {
      sum+= (weights[w_ptr]*input[input_ptr]); w_ptr++; input_ptr++;
      sum+= (weights[w_ptr]*input[input_ptr]); w_ptr++; input_ptr++;
      sum+= (weights[w_ptr]*input[input_ptr]); w_ptr++; input_ptr++;
      sum+= (weights[w_ptr]*input[input_ptr]); w_ptr++; input_ptr++;

      size-= 4;
    }

    while (size) {
      sum+= (weights[w_ptr]*input[input_ptr]); w_ptr++; input_ptr++;
      size--;
    }

    sum = (sum*weights_scaling)/(127*1000);
    output[j] = sum;
  }
}
```

Testing network on CPU

Floating point vs fixed point result

32 bit floating point confusion matrix :

967	0	6	0	0	3	6	1	6	4
0	1126	0	0	1	2	3	12	2	5
1	3	1000	5	3	1	1	13	4	0
2	2	3	979	0	17	1	1	5	7
0	0	2	0	942	2	2	3	6	15
2	0	2	9	0	852	5	1	7	6
6	1	4	0	10	8	938	0	2	1
1	1	10	9	1	1	0	986	5	6
1	2	5	7	1	3	1	1	933	3
0	0	0	1	24	3	0	10	4	962
98.673	**99.207**	**96.899**	**96.931**	**95.927**	**95.516**	**98.015**	**95.914**	**95.791**	**95.342**

result= **96.86%**

8 bit fixed point confusion matrix :

967	0	5	0	1	3	6	1	6	4
0	1126	0	0	1	1	3	11	2	5
1	3	1001	5	3	1	1	13	2	0
2	2	3	976	0	15	1	1	5	7
0	0	2	0	943	2	1	3	5	16
2	0	2	11	0	856	6	1	9	7
6	1	4	0	8	6	938	0	2	1
1	1	10	9	1	1	0	989	3	7
1	2	5	9	1	4	1	1	936	4
0	0	0	0	24	3	0	8	4	958
98.673	**99.207**	**96.996**	**96.634**	**96.029**	**95.964**	**98.015**	**96.206**	**96.099**	**94.945**

result = **96.91%**

Running on Arduino

Running on Arduino

1000 random samples from testing set

97	0	0	0	0	0	0	0	0	1
0	117	0	0	0	0	0	3	0	0
0	1	96	0	0	0	0	2	0	0
0	1	1	95	0	2	0	0	0	0
0	0	0	0	82	0	0	0	0	1
0	0	1	2	0	81	0	0	1	2
1	0	1	0	0	1	95	0	0	1
0	0	0	1	0	0	0	111	0	0
0	0	0	0	0	0	0	0	97	0
0	0	0	0	3	1	0	0	0	102
98.98	98.319	96.97	96.939	96.471	95.294	100	95.69	98.98	95.327

result = 97.3%

time per sample 0.487s