Neural Network

Thomas Laverghetta (tlave002@odu.edu),
Old Dominion University – Computational Modeling and Simulation Engineering

Please read over Assignment 5 System Requirements document (which accompanies this report) first to gain understanding of program overview.

Algorithms

For this assignment, I used MATLAB's deep-learning toolbox to train and run a neural network provided input and target data. To produce the input and target data, I used Python script which would take the raw-data and parse it. Then to reduce the input data size, I added the ability for the user to compress input data by factor n such that $\{2^n | \ n \in \mathbb{N}[0,10]\}$ (grouping by powers of two). The pseudocode for data parser is below:

```
DataParser(filename, factor n):
    Read fille(filename)
    Create two output files: target and input files
    Sum = 0          // used for compression to sum values
    Counter = 0      // used for compression to count num iterations
    Foreach line from file:
        If line is target:
            Output target in row format to target file
        Else:
            Foreach char in line:
                Increment counter to count number of iterations (Count++)
                Sum += int(char)
                If not Count % 2^n: // have grouped 2^n inputs
                    Output (Sum / 2^n) in CSV format to input file
                    Sum = 0
    Close target and input files
```

Results

Using Python data parser and MATLAB deep-learning toolbox, I ran 9-tests with 3-factors (0, 1, 4) and 3-levels of hidden neurons (5, 10, 50). Each test will have different combination of parameters. All tests have validation and testing at 15% out of sample data. The figures below show the confusion matrices outputted from the various tests.

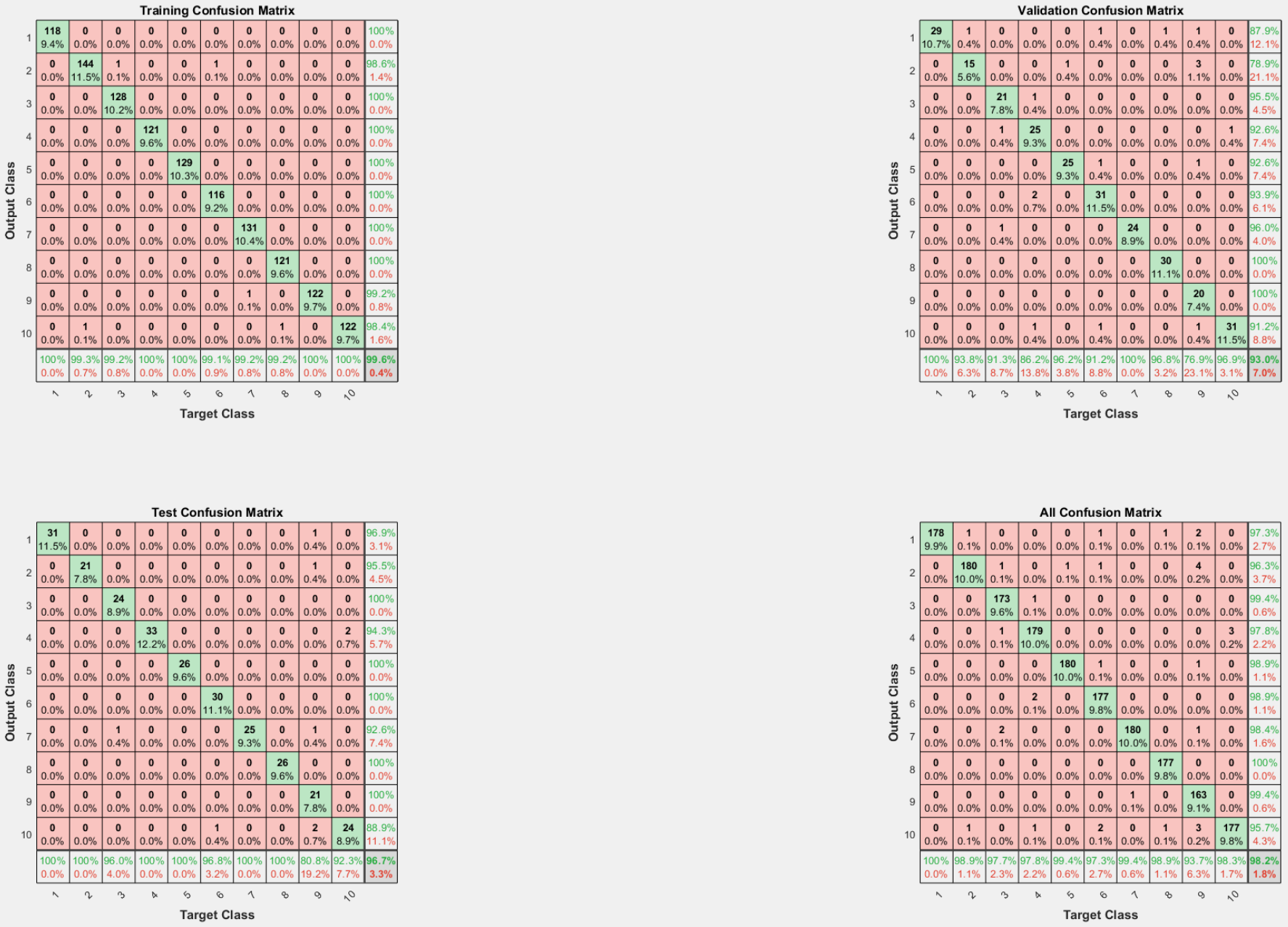

Figure 1. Factor 0 and 10 Hidden Neurons

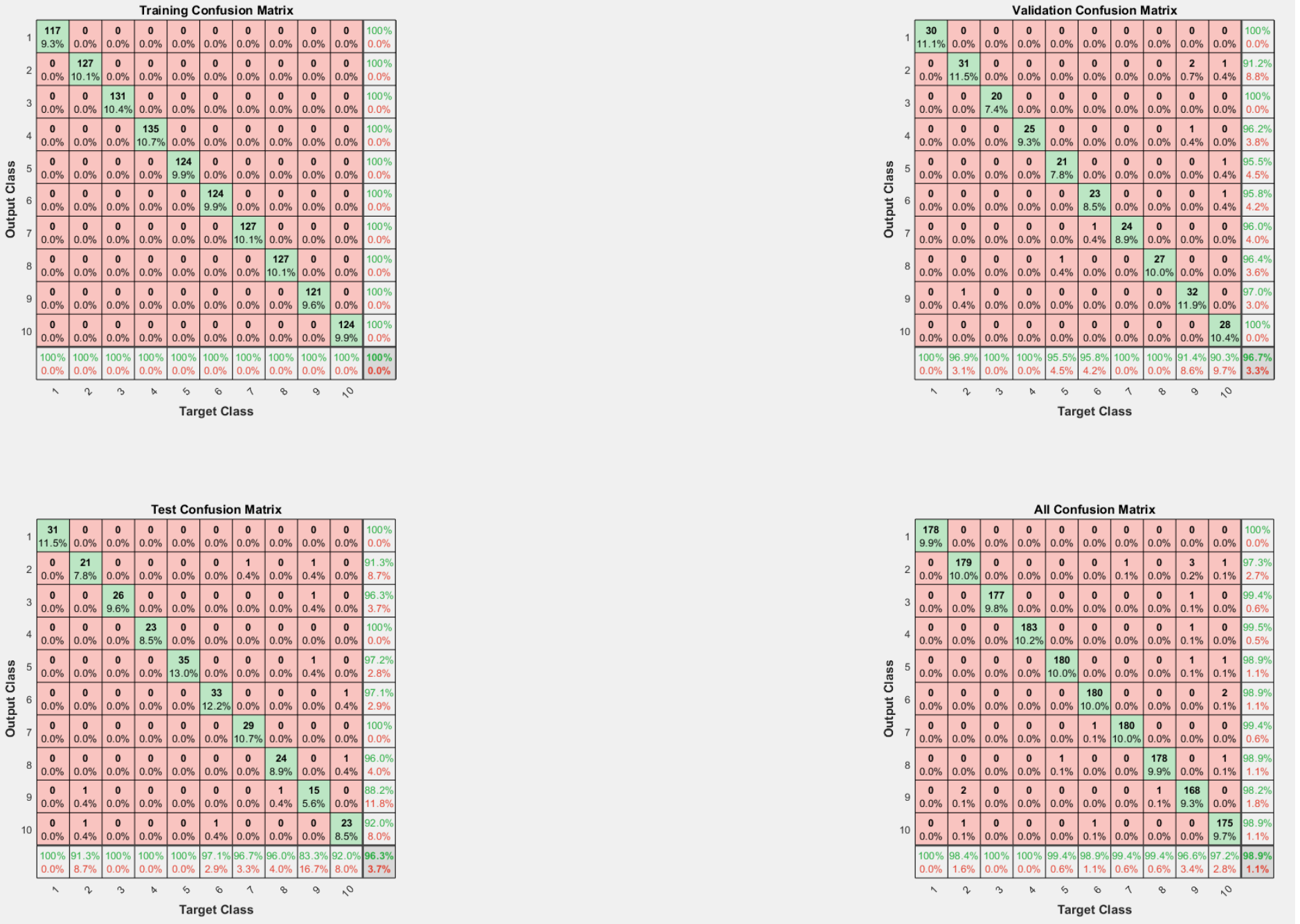

Figure 2. Factor 0 and 50 hidden neurons

Figure 3. factor 0, 5 hidden neurons

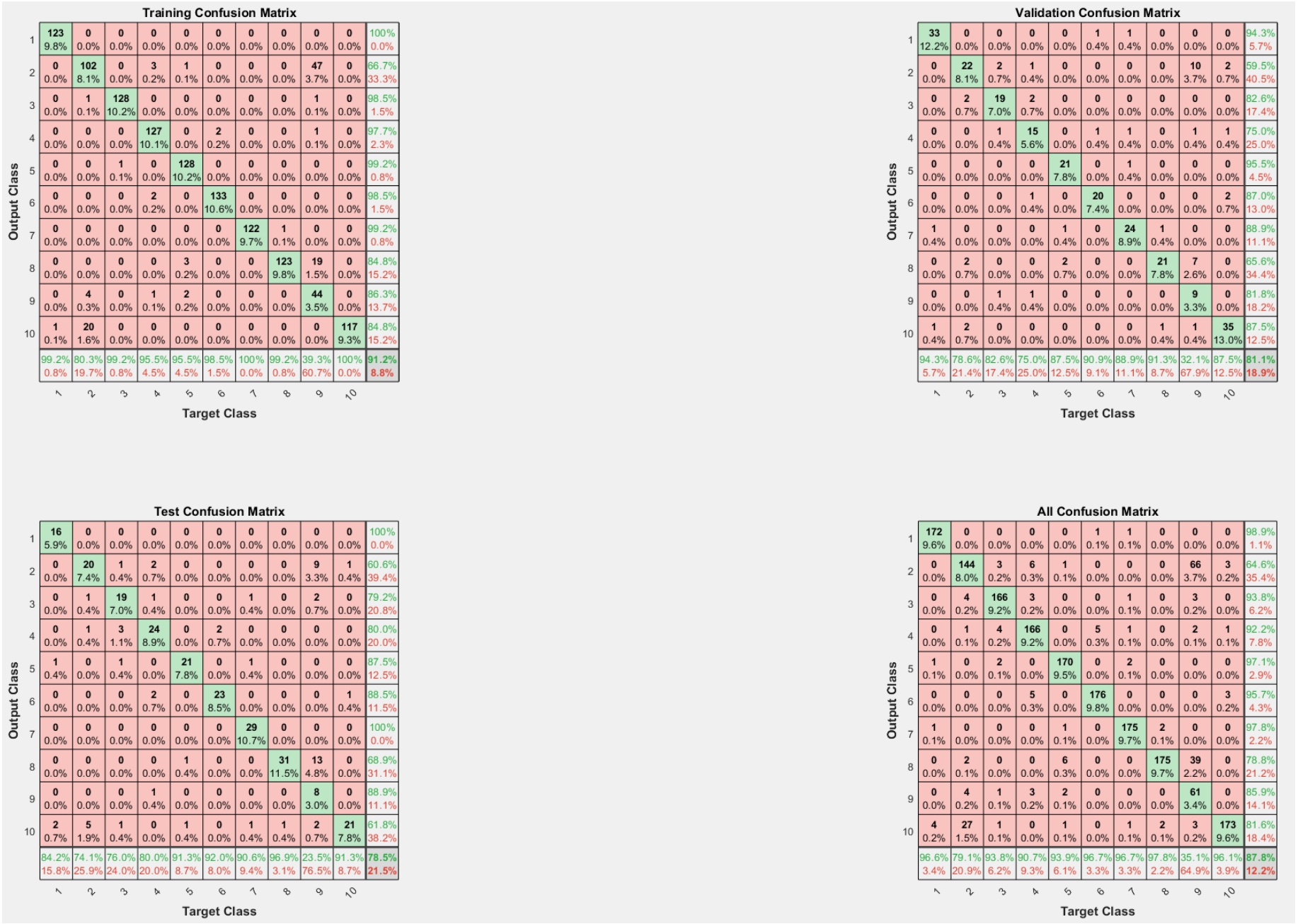

Figure 4. Factor 1 and 5 hidden neurons

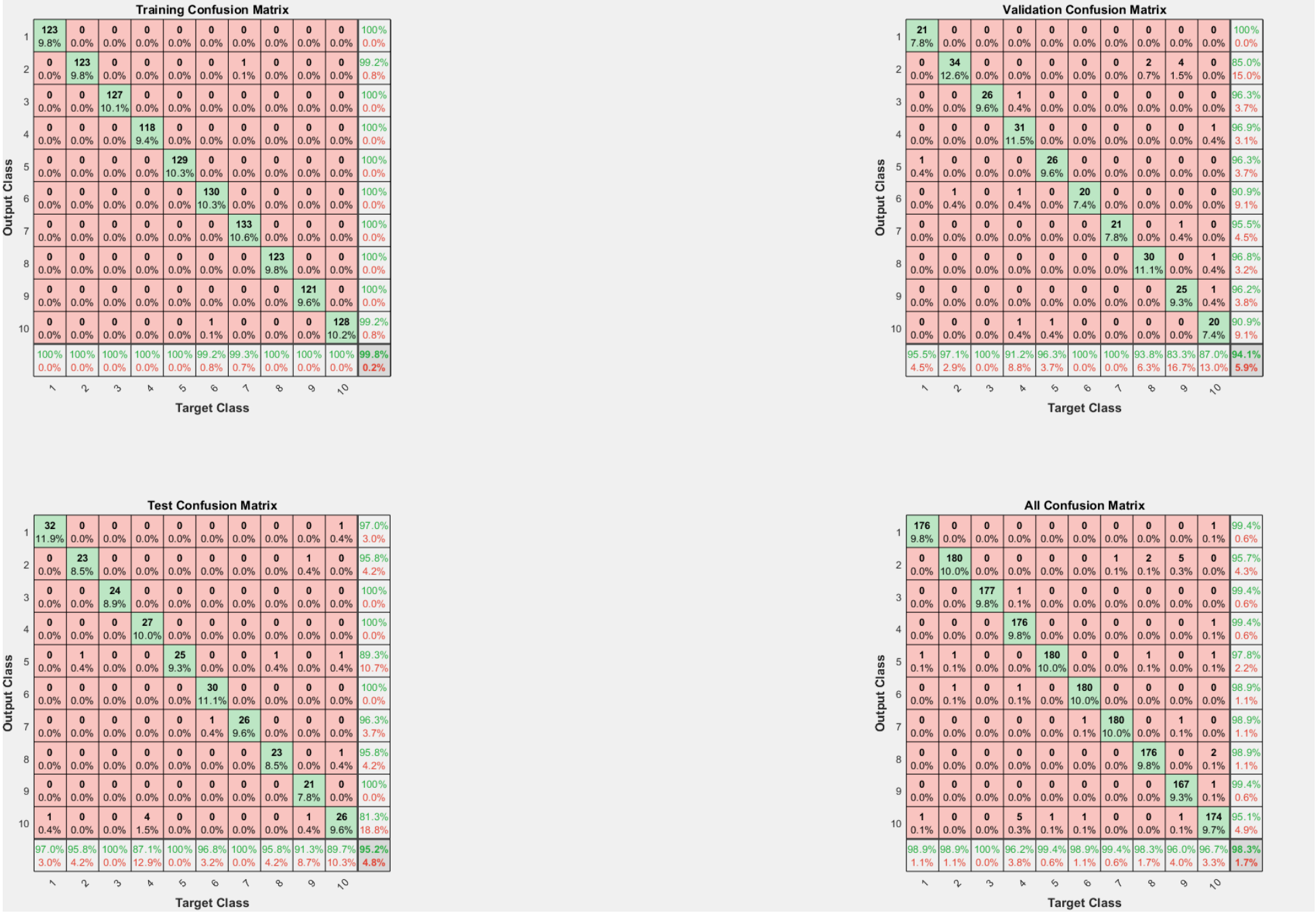

Figure 5. Factor 1 and 10 hidden neurons

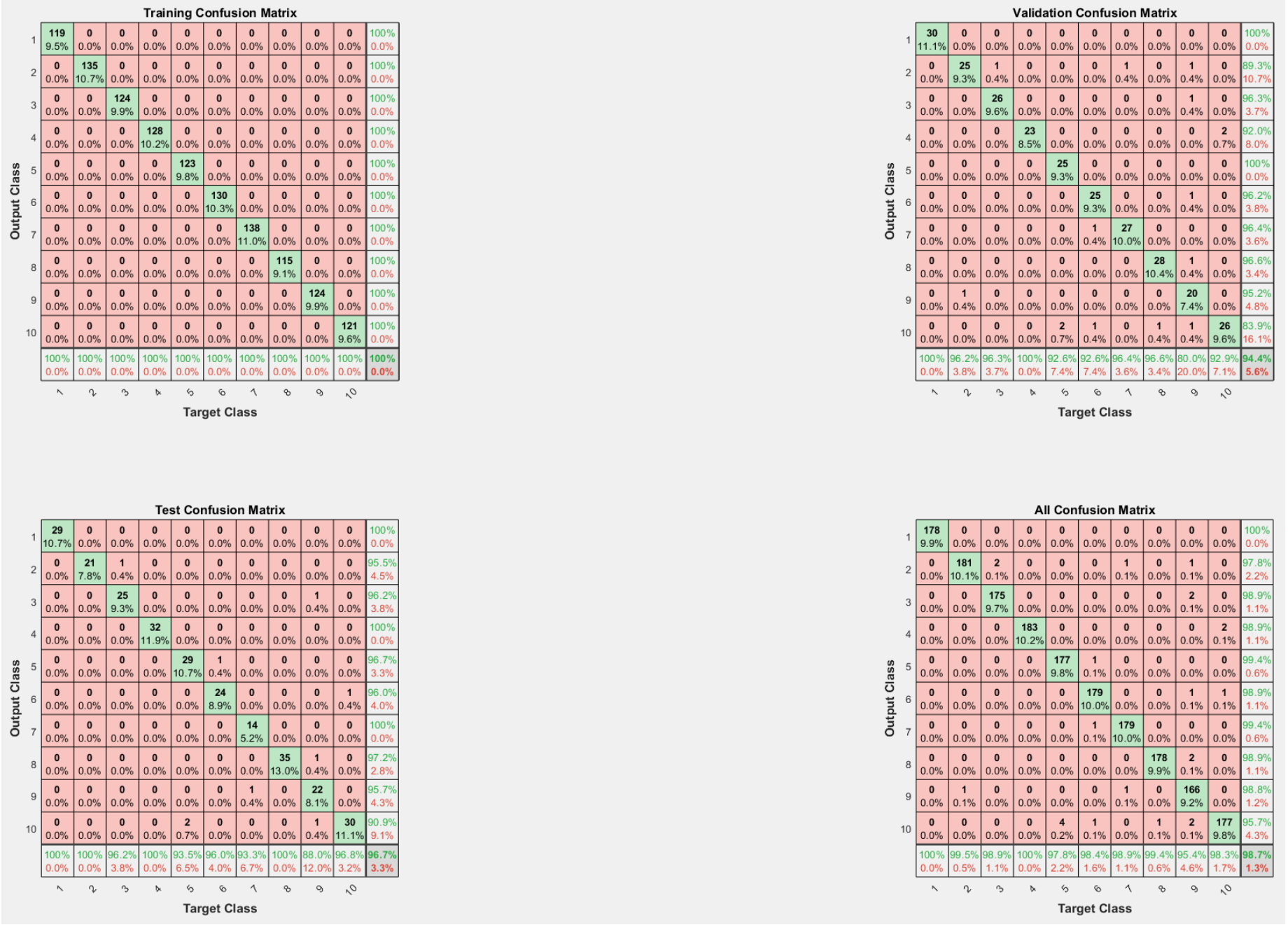

Figure 6. Factor 1 and 50 hidden neurons

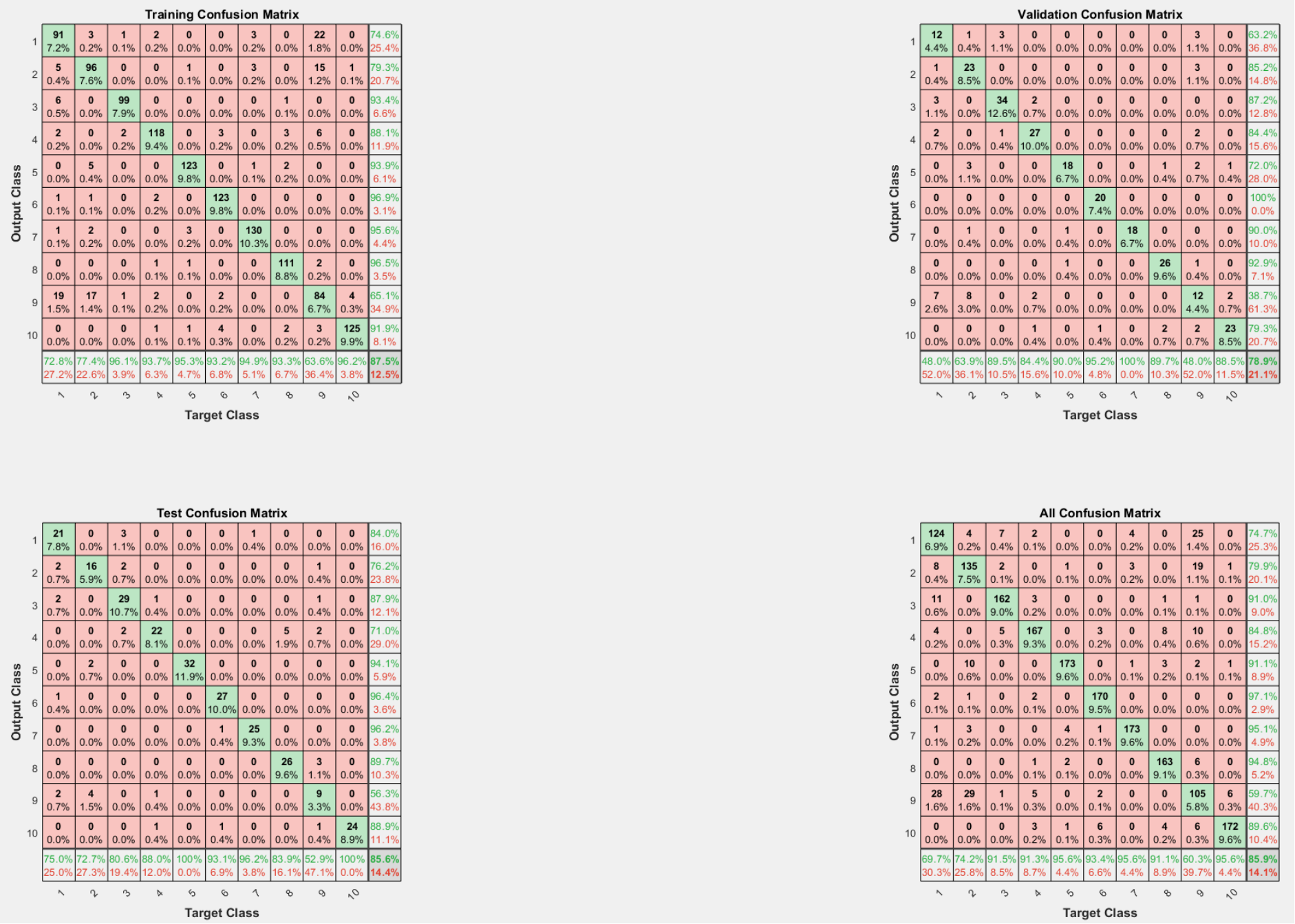

Figure 7. Factor 4 and 5 hidden neurons

Figure 8. Factor 4 and 10 hidden neurons

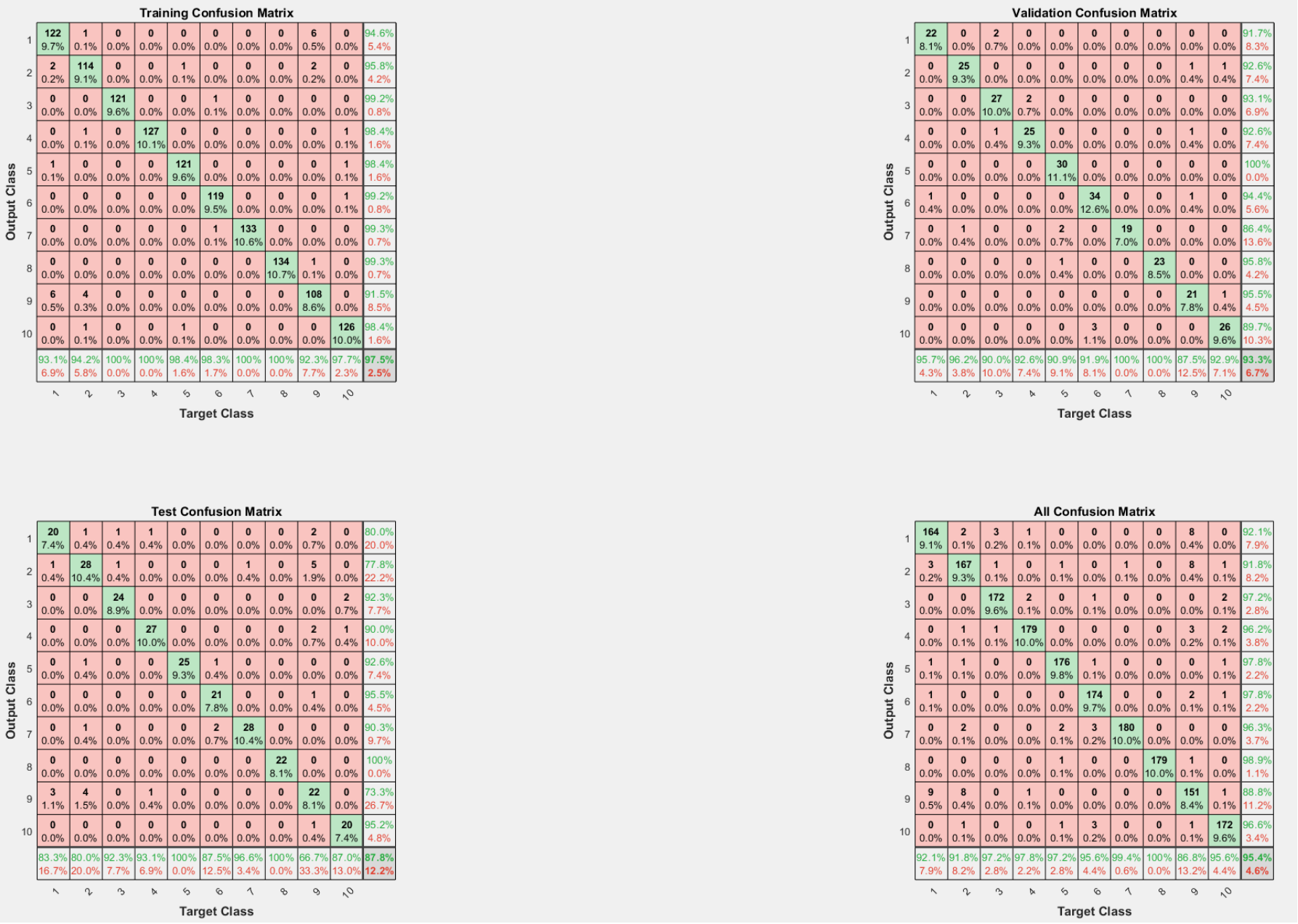

Figure 9. Factor 4 and 50 hidden neurons

Analysis

Given confusion matrices in previous section, I gathered all test confusion matrices overall accuracy values and put them into Table 1 below. From the table, the difference between 10 and 50 hidden neurons is minute, and for compression factor 0, 50 is worse than 10. Also, compression factors 0 and 1 produce similar results when hidden neurons are greater than 5. On the other hand, compression factor 4 has worse values across the board. In conclusion, the neural networks that produce the best output are those with reasonable number of hidden neurons (between 10 and 50) and compression factors between 0 and 2.

Table 1. Overall Test Accuracy from Test Confusion Matrices.

		Compression Factor		
		0	1	4
Hidden Neurons	5	85.9	78.5	85.6
	10	96.7	95.2	86.7
	50	96.3	96.7	87.8