# 1 Generating Test Data

## 1.1 Running the generation script

For convenience a script was included to automate the generation of test data called generate_test_data.py. Invoke the following command to generate the test data.

```
$ chmod +x generate_test_data.py
$ ./generate_test_data.py
```

Listing 1: Generating the test data

A class dedicated to manipulate CSV files was created called csv_utils. This class also has a method for generating a matrix representing a combination input bits.

```python
def to_matrix(n: int):
    """
    A helper function that generates a matrix of bit combinations
    :param n: size of the matrix
    :return: a matrix with the input combinations
    """
    def gen(n: int):
        for i in range(1, 2 ** n - 1):
            yield '{:0{n}b}'.format(i, n=n)

    matrix = [[0 for i in range(n)]]
    for perm in list(gen(n)):
        matrix.append([int(s) for s in perm])
    matrix.append([1 for i in range(n)])
    return matrix
```

Listing 2: Input bit generation

The following method is responsible for generating the data set.

```python
def generate_data_to_csv(matrix_size: int, file_name: str = 'hard_problem', transformation_function=transformation):
    """
    A helper function to aid in generating csv data
    :param transformation_function: The transformation function for generating the output bits
    :param matrix_size: The size of the matrix
    :param file_name: The file name to produce
    :return: Input matrix and output matrix
    """
    input_array = np.asarray(to_matrix(matrix_size))
    output = np.apply_along_axis(transformation_function, 1, input_array)
    data_frame = pd.DataFrame(np.concatenate((input_array, output), axis=1))
    data_frame.to_csv(os.path.join('resources', f'{file_name}.csv'), header=None, index=None)
    return input_array, output
```

Listing 3: Feed forward implementation

```python
def feedforward(self, inp: np.ndarray = None) -> (np.ndarray, np.ndarray):
    """
    A feed forward method that allows the neural net to 'think'.
    :param inp: a numpy array representing the inputs
    :return: a tuple representing the output of the hidden and final output
    """
    if inp is None:
        inp = self.inputs
    net_h = np.dot(inp, self.wh)
    out_h = self.activation(net_h)
    net_o = np.dot(out_h, self.wo)
    out_o = self.activation(net_o)
    return out_h, out_o
```

Listing 4: Feed forward implementation