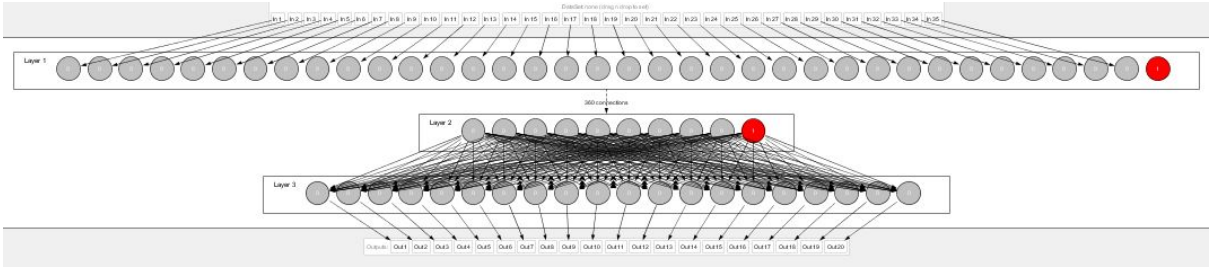
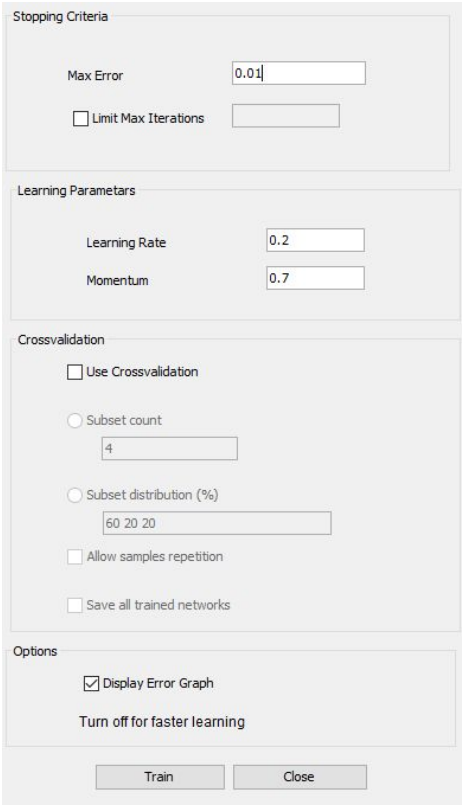
Sprawozdanie PSI 3:

Celem ćwiczenia było poznanie budowy i działania wielowarstwowych sieci neuronowych poprzez uczenie z użyciem algorytmu wstecznej propagacji błędu rozpoznawania konkretnych liter alfabetu. Pierwszym krokiem było wygenerowanie danych uczących dwudziestu liter, 10 małych i 10 dużych. Jest to 10 pierwszych liter alfabetu. Oprócz liter musiałem przygotować zestaw sprawdzający czy wynikiem jest dokładnie ta litera, o którą prosiliśmy. Następnie przygotowałem wielowarstwową sieć neuronową według algorytmu wstecznej propagacji włączając to neuron BIAS'u. Kolejną czynnością było testowanie sieci dla różnych współczynników uczenia i bezwładności.

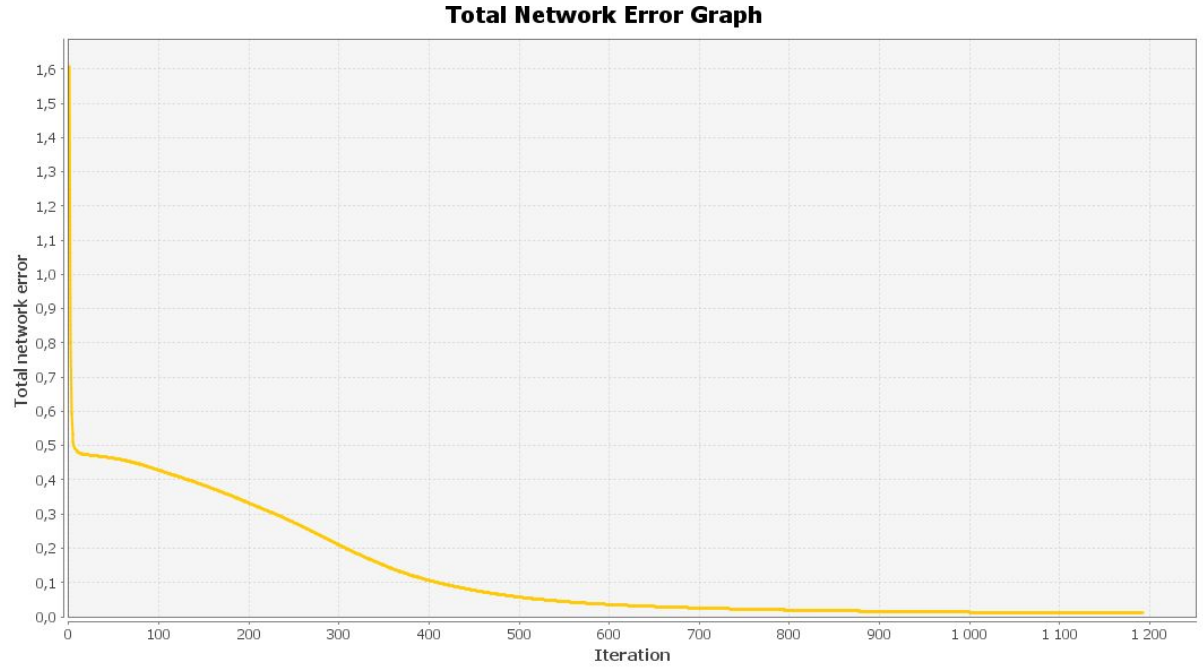


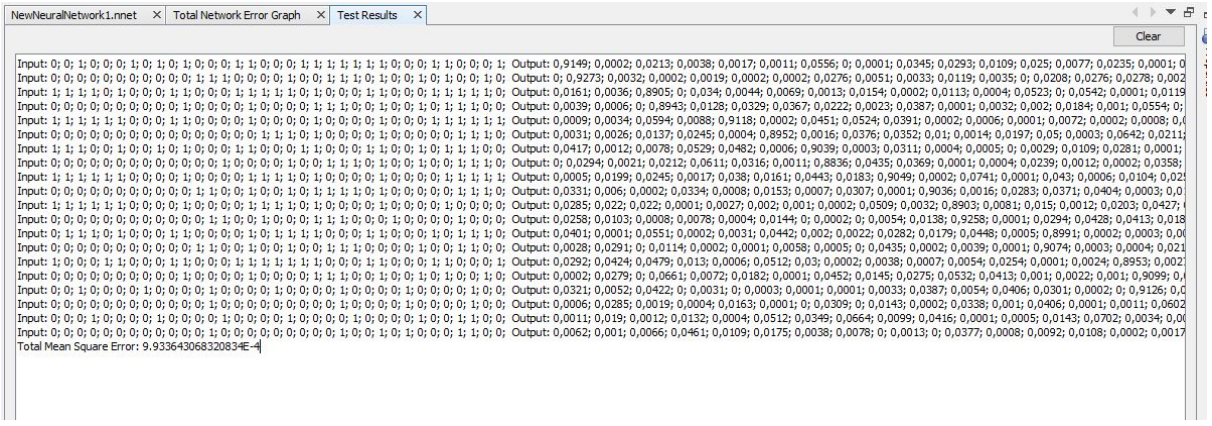
W warstwie ukrytej (warstwa nr 2) umieściłem 9 neuronów, oraz dodatkowo jeden neuron BIAS'u. Warstwa pierwsza zawiera zawiera 35 neuronów + neuron BIAS (zaznaczony na czerwono). Dla tych neuronów nie są liczone wagi, więc trzeba je traktować jedynie jako wejścia (inputs). Warstwa 3 jest warstwą wyjściową, i zawiera 20 neuronów, ze względu na 20 różniących się od siebie liter, które trzeba rozpatrywać jako osobny przypadek aby je potem rozróżnić.

Okno wyboru współczynników sieci neuronowej:



Max error: 0.01, Współczynnik uczenia: 0.2, Momentum: 0.7





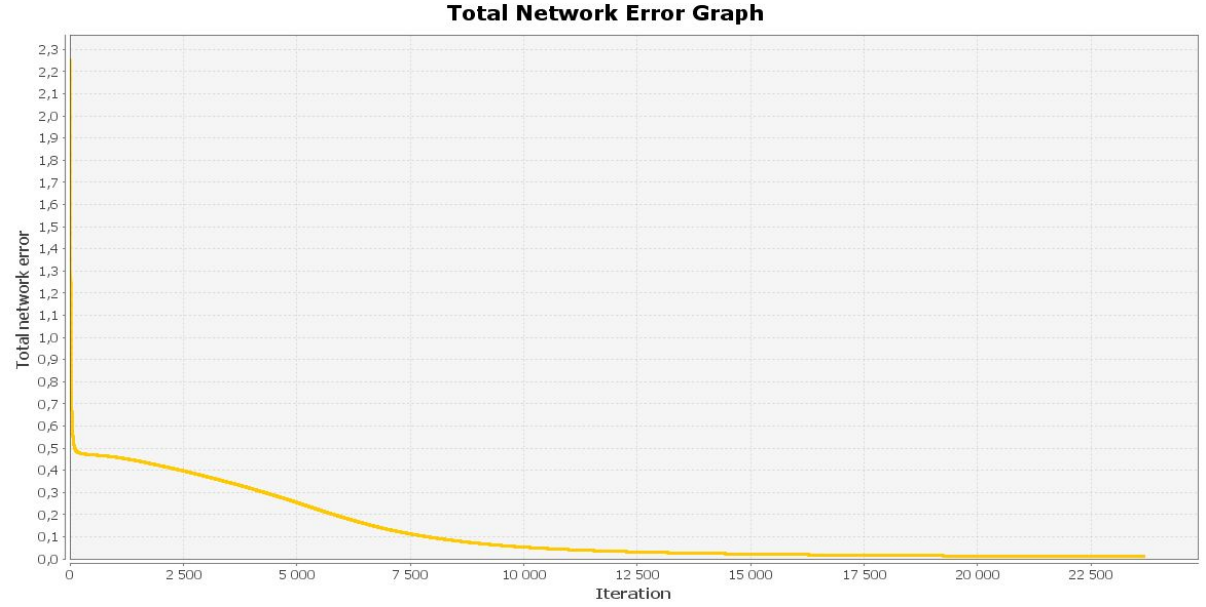
Niekonwertowane dane:

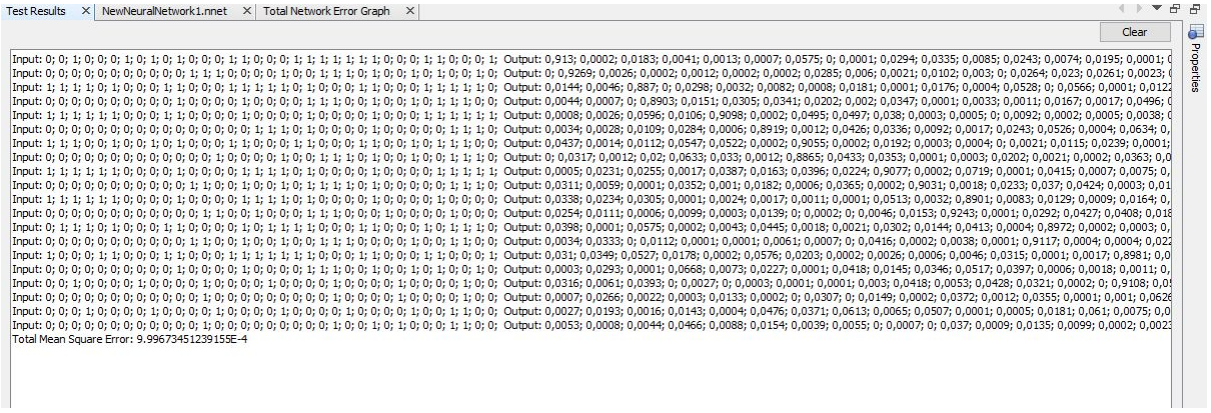
Input: 0; 0; 1; 0; 0; 0; 1; 0; 1; 0; 1; 0; 0; 0; 1; 1; 0; 0; 0; 1; 1; 1; 1; 1; 1; 1; 0; 0; 0; 1; 1; 0; 0; 0; 1;
Output: 0,9149; 0,0002; 0,0213; 0,0038; 0,0017; 0,0011; 0,0556; 0; 0,0001; 0,0345; 0,0293; 0,0109; 0,025; 0,0077; 0,0235; 0,0001; 0,0271; 0; 0,0005; 0,0082; Desired output: 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; Error: -0,0851; 0,0002; 0,0213; 0,0038; 0,0017; 0,0011; 0,0556; 0; 0,0001; 0,0345; 0,0293; 0,0109; 0,025; 0,0077; 0,0235; 0,0001; 0,0271; 0; 0,0005; 0,0082;
Input: 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1; 0; 0; 0; 0; 1; 0; 0; 1; 1; 1; 0; 1; 0; 0; 1; 0; 1; 0; 0; 1; 0;
Output: 0; 0,9273; 0,0032; 0,0002; 0,0019; 0,0002; 0,0002; 0,0276; 0,0051; 0,0033; 0,0119; 0,0035; 0; 0,0208; 0,0276; 0,0278; 0,002; 0,0094; 0,0063; 0,0002; Desired output: 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; Error: 0; -0,0727; 0,0032; 0,0002; 0,0019; 0,0002; 0,0002; 0,0276; 0,0051; 0,0033; 0,0119; 0,0035; 0; 0,0208; 0,0276; 0,0278; 0,002; 0,0094; 0,0063; 0,0002;
Input: 1; 1; 1; 1; 0; 1; 0; 0; 0; 1; 1; 0; 0; 0; 1; 1; 1; 1; 1; 0; 1; 0; 0; 0; 1; 1; 0; 0; 0; 1; 1; 1; 1; 1; 0;
Output: 0,0161; 0,0036; 0,8905; 0; 0,034; 0,0044; 0,0069; 0,0013; 0,0154; 0,0002; 0,0113; 0,0004; 0,0523; 0; 0,0542; 0,0001; 0,0119; 0,0001; 0,0012; 0,0066; Desired output: 0; 0; 1; 0; 0; 0; 0; 0; 0; 0;

0; 0; 0; 0; 0; 0; 0; 0; 0; 0; Error: 0,0161; 0,0036; -0,1095; 0; 0,034; 0,0044; 0,0069; 0,0013; 0,0154; 0,0002; 0,0113; 0,0004; 0,0523; 0; 0,0542; 0,0001; 0,0119; 0,0001; 0,0012; 0,0066;
Input: 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 1; 1; 0; 0; 1; 0; 0; 1; 0; 1; 1; 1; 0; 0;
Output: 0,0039; 0,0006; 0; 0,8943; 0,0128; 0,0329; 0,0367; 0,0222; 0,0023; 0,0387; 0,0001; 0,0032; 0,002; 0,0184; 0,001; 0,0554; 0; 0,0011; 0,0043; 0,0381; Desired output: 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; Error: 0,0039; 0,0006; 0; -0,1057; 0,0128; 0,0329; 0,0367; 0,0222; 0,0023; 0,0387; 0,0001; 0,0032; 0,002; 0,0184; 0,001; 0,0554; 0; 0,0011; 0,0043; 0,0381;
Input: 1; 1; 1; 1; 1; 1; 0; 0; 0; 1; 1; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 1; 1; 1; 1; 1;
Output: 0,0009; 0,0034; 0,0594; 0,0088; 0,9118; 0,0002; 0,0451; 0,0524; 0,0391; 0,0002; 0,0006; 0,0001; 0,0072; 0,0002; 0,0008; 0,003; 0,0003; 0,0057; 0,0002; 0,0144; Desired output: 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; Error: 0,0009; 0,0034; 0,0594; 0,0088; -0,0882; 0,0002; 0,0451; 0,0524; 0,0391; 0,0002; 0,0006; 0,0001; 0,0072; 0,0002; 0,0008; 0,003; 0,0003; 0,0057; 0,0002; 0,0144;
Input: 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 1; 0; 1; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 1; 1; 1; 0;
Output: 0,0031; 0,0026; 0,0137; 0,0245; 0,0004; 0,8952; 0,0016; 0,0376; 0,0352; 0,01; 0,0014; 0,0197; 0,05; 0,0003; 0,0642; 0,0211; 0,0006; 0,0001; 0,0427; 0,021; Desired output: 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; Error: 0,0031; 0,0026; 0,0137; 0,0245; 0,0004; -0,1048; 0,0016; 0,0376; 0,0352; 0,01; 0,0014; 0,0197; 0,05; 0,0003; 0,0642; 0,0211; 0,0006; 0,0001; 0,0427; 0,021;
Input: 1; 1; 1; 0; 0; 1; 0; 0; 1; 0; 1; 0; 0; 0; 1; 1; 0; 0; 0; 1; 1; 0; 0; 0; 1; 1; 0; 0; 1; 0; 1; 1; 1; 0; 0;
Output: 0,0417; 0,0012; 0,0078; 0,0529; 0,0482; 0,0006; 0,9039; 0,0003; 0,0311; 0,0004; 0,0005; 0; 0,0029; 0,0109; 0,0281; 0,0001; 0,0005; 0; 0,0249; 0,0184; Desired output: 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; Error: 0,0417; 0,0012; 0,0078; 0,0529; 0,0482; 0,0006; -0,0961; 0,0003; 0,0311; 0,0004; 0,0005; 0; 0,0029; 0,0109; 0,0281; 0,0001; 0,0005; 0; 0,0249; 0,0184;
Input: 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 0; 0; 1; 1; 1; 0; 1; 0; 0; 1; 0; 0; 1; 1; 1; 0;
Output: 0; 0,0294; 0,0021; 0,0212; 0,0611; 0,0316; 0,0011; 0,8836; 0,0435; 0,0369; 0,0001; 0,0004; 0,0239; 0,0012; 0,0002; 0,0358; 0,0002; 0,0591; 0,0598; 0,01; Desired output: 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; Error: 0; 0,0294; 0,0021; 0,0212; 0,0611; 0,0316; 0,0011; -0,1164; 0,0435; 0,0369; 0,0001; 0,0004; 0,0239; 0,0012; 0,0002; 0,0358; 0,0002; 0,0591; 0,0598; 0,01;
Input: 1; 1; 1; 1; 1; 1; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 1; 1; 1; 0; 1; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 1; 1; 1; 1;
Output: 0,0005; 0,0199; 0,0245; 0,0017; 0,038; 0,0161; 0,0443; 0,0183; 0,9049; 0,0002; 0,0741; 0,0001; 0,043; 0,0006; 0,0104; 0,025; 0,0024; 0; 0,0062; 0,0001; Desired output: 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; Error: 0,0005; 0,0199; 0,0245; 0,0017; 0,038; 0,0161; 0,0443; 0,0183; -0,0951; 0,0002; 0,0741; 0,0001; 0,043; 0,0006; 0,0104; 0,025; 0,0024; 0; 0,0062; 0,0001;
Input: 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 0; 0; 1; 0; 0; 1; 0; 1; 1; 1; 1; 0; 1; 0; 0; 0; 0; 0; 1; 1; 1; 0;
Output: 0,0331; 0,006; 0,0002; 0,0334; 0,0008; 0,0153; 0,0007; 0,0307; 0,0001; 0,9036; 0,0016; 0,0283; 0,0371; 0,0404; 0,0003; 0,0126; 0,0045; 0,0336; 0,0267; 0,0052; Desired output: 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; Error: 0,0331; 0,006; 0,0002; 0,0334; 0,0008; 0,0153; 0,0007; 0,0307; 0,0001; -0,0964; 0,0016; 0,0283; 0,0371; 0,0404; 0,0003; 0,0126; 0,0045; 0,0336; 0,0267; 0,0052;
Input: 1; 1; 1; 1; 1; 1; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 1; 1; 1; 0; 1; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 0; 0; 0; 0;
Output: 0,0285; 0,022; 0,022; 0,0001; 0,0027; 0,002; 0,001; 0,0002; 0,0509; 0,0032; 0,8903; 0,0081; 0,015; 0,0012; 0,0203; 0,0427; 0,0337; 0; 0,0001; 0; Desired output: 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; Error: 0,0285; 0,022; 0,022; 0,0001; 0,0027; 0,002; 0,001; 0,0002; 0,0509; 0,0032; -0,1097; 0,0081; 0,015; 0,0012; 0,0203; 0,0427; 0,0337; 0; 0,0001; 0;
Input: 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 0; 0; 1; 0; 0; 0; 1; 1; 1; 0; 0; 0; 1; 0; 0; 0; 0; 1; 0; 0; 0;
Output: 0,0258; 0,0103; 0,0008; 0,0078; 0,0004; 0,0144; 0; 0,0002; 0; 0,0054; 0,0138; 0,9258;

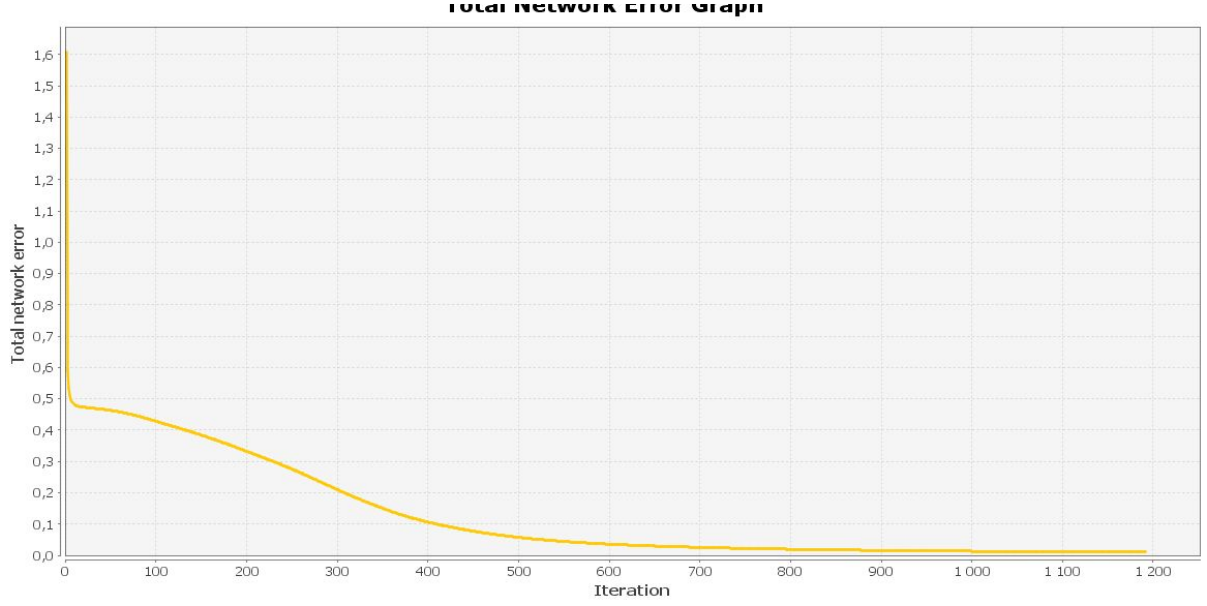
0,0001; 0,0294; 0,0428; 0,0413; 0,0187; 0,0449; 0; 0,036; Desired output: 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; Error: 0,0258; 0,0103; 0,0008; 0,0078; 0,0004; 0,0144; 0; 0,0002; 0; 0,0054; 0,0138; -0,0742; 0,0001; 0,0294; 0,0428; 0,0413; 0,0187; 0,0449; 0; 0,036;
Input: 0; 1; 1; 1; 0; 1; 0; 0; 0; 1; 1; 0; 0; 0; 0; 1; 0; 1; 1; 1; 1; 0; 0; 0; 1; 1; 0; 0; 0; 1; 0; 1; 1; 1; 0;
Output: 0,0401; 0,0001; 0,0551; 0,0002; 0,0031; 0,0442; 0,002; 0,0022; 0,0282; 0,0179; 0,0448; 0,0005; 0,8991; 0,0002; 0,0003; 0,0008; 0,051; 0,0001; 0,0023; 0,0003; Desired output: 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; Error: 0,0401; 0,0001; 0,0551; 0,0002; 0,0031; 0,0442; 0,002; 0,0022; 0,0282; 0,0179; 0,0448; 0,0005; -0,1009; 0,0002; 0,0003; 0,0008; 0,051; 0,0001; 0,0023; 0,0003;
Input: 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 1; 0; 0; 1; 0; 0; 1; 0; 0; 1; 1; 1; 0; 0; 0; 0; 1; 0; 0; 1; 1; 0; 0;
Output: 0,0028; 0,0291; 0; 0,0114; 0,0002; 0,0001; 0,0058; 0,0005; 0; 0,0435; 0,0002; 0,0039; 0,0001; 0,9074; 0,0003; 0,0004; 0,0218; 0,0221; 0,0553; 0,0046; Desired output: 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; Error: 0,0028; 0,0291; 0; 0,0114; 0,0002; 0,0001; 0,0058; 0,0005; 0; 0,0435; 0,0002; 0,0039; 0,0001; -0,0926; 0,0003; 0,0004; 0,0218; 0,0221; 0,0553; 0,0046;
Input: 1; 0; 0; 0; 1; 1; 0; 0; 0; 1; 1; 0; 0; 0; 1; 1; 1; 1; 1; 1; 1; 0; 0; 0; 1; 1; 0; 0; 0; 1; 1; 0; 0; 0; 1;
Output: 0,0292; 0,0424; 0,0479; 0,013; 0,0006; 0,0512; 0,03; 0,0002; 0,0038; 0,0007; 0,0054; 0,0254; 0,0001; 0,0024; 0,8953; 0,0027; 0,0004; 0; 0,0072; 0,0328; Desired output: 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; Error: 0,0292; 0,0424; 0,0479; 0,013; 0,0006; 0,0512; 0,03; 0,0002; 0,0038; 0,0007; 0,0054; 0,0254; 0,0001; 0,0024; -0,1047; 0,0027; 0,0004; 0; 0,0072; 0,0328;
Input: 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 1; 1; 0; 0; 1; 0; 0; 1; 0; 1; 0; 0; 1; 0;
Output: 0,0002; 0,0279; 0; 0,0661; 0,0072; 0,0182; 0,0001; 0,0452; 0,0145; 0,0275; 0,0532; 0,0413; 0,001; 0,0022; 0,001; 0,9099; 0,0001; 0,0037; 0; 0,0001; Desired output: 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; Error: 0,0002; 0,0279; 0; 0,0661; 0,0072; 0,0182; 0,0001; 0,0452; 0,0145; 0,0275; 0,0532; 0,0413; 0,001; 0,0022; 0,001; -0,0901; 0,0001; 0,0037; 0; 0,0001;
Input: 0; 0; 1; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 0; 0;
Output: 0,0321; 0,0052; 0,0422; 0; 0,0031; 0; 0,0003; 0,0001; 0,0001; 0,0033; 0,0387; 0,0054; 0,0406; 0,0301; 0,0002; 0; 0,9126; 0,059; 0,0007; 0,0011; Desired output: 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; Error: 0,0321; 0,0052; 0,0422; 0; 0,0031; 0; 0,0003; 0,0001; 0,0001; 0,0033; 0,0387; 0,0054; 0,0406; 0,0301; 0,0002; 0; -0,0874; 0,059; 0,0007; 0,0011;
Input: 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 0; 0;
Output: 0,0006; 0,0285; 0,0019; 0,0004; 0,0163; 0,0001; 0; 0,0309; 0; 0,0143; 0,0002; 0,0338; 0,001; 0,0406; 0,0001; 0,0011; 0,0602; 0,8918; 0,001; 0,0504; Desired output: 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; Error: 0,0006; 0,0285; 0,0019; 0,0004; 0,0163; 0,0001; 0; 0,0309; 0; 0,0143; 0,0002; 0,0338; 0,001; 0,0406; 0,0001; 0,0011; 0,0602; -0,1082; 0,001; 0,0504;
Input: 0; 0; 0; 1; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 0; 0; 0; 0; 1; 0; 0; 1; 1; 0; 0;
Output: 0,0011; 0,019; 0,0012; 0,0132; 0,0004; 0,0512; 0,0349; 0,0664; 0,0099; 0,0416; 0,0001; 0,0005; 0,0143; 0,0702; 0,0034; 0,0003; 0,0047; 0,0023; 0,8998; 0,0392; Desired output: 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; Error: 0,0011; 0,019; 0,0012; 0,0132; 0,0004; 0,0512; 0,0349; 0,0664; 0,0099; 0,0416; 0,0001; 0,0005; 0,0143; 0,0702; 0,0034; 0,0003; 0,0047; 0,0023; -0,1002; 0,0392;
Input: 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 1; 0; 1; 0; 0; 0; 1; 1; 0; 0;
Output: 0,0062; 0,001; 0,0066; 0,0461; 0,0109; 0,0175; 0,0038; 0,0078; 0; 0,0013; 0; 0,0377; 0,0008; 0,0092; 0,0108; 0,0002; 0,0017; 0,0566; 0,0076; 0,9099; Desired output: 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; Error: 0,0062; 0,001; 0,0066; 0,0461; 0,0109; 0,0175; 0,0038; 0,0078; 0; 0,0013; 0; 0,0377; 0,0008; 0,0092; 0,0108; 0,0002; 0,0017; 0,0566; 0,0076; -0,0901;
Total Mean Square Error: 9.933643068320834E-4

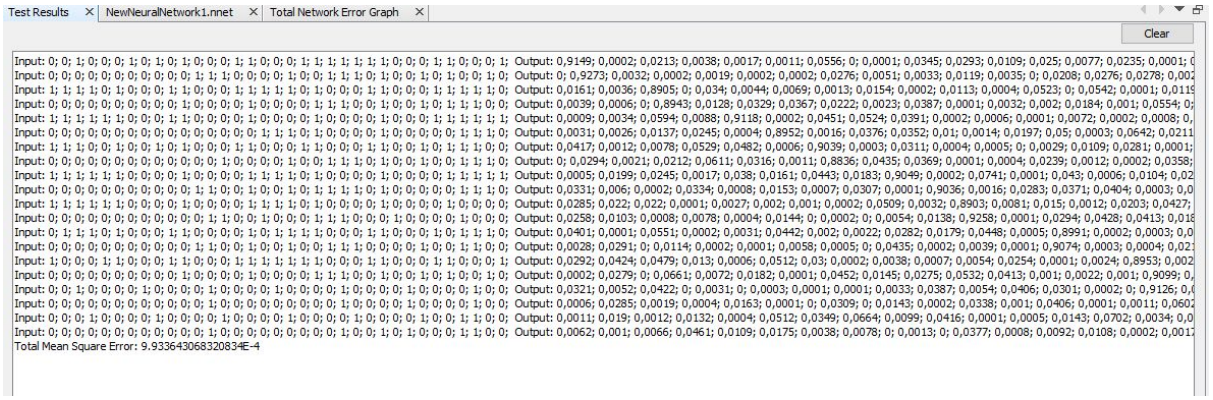
Współczynnik uczenia: 0.01, Momentum: 0.7



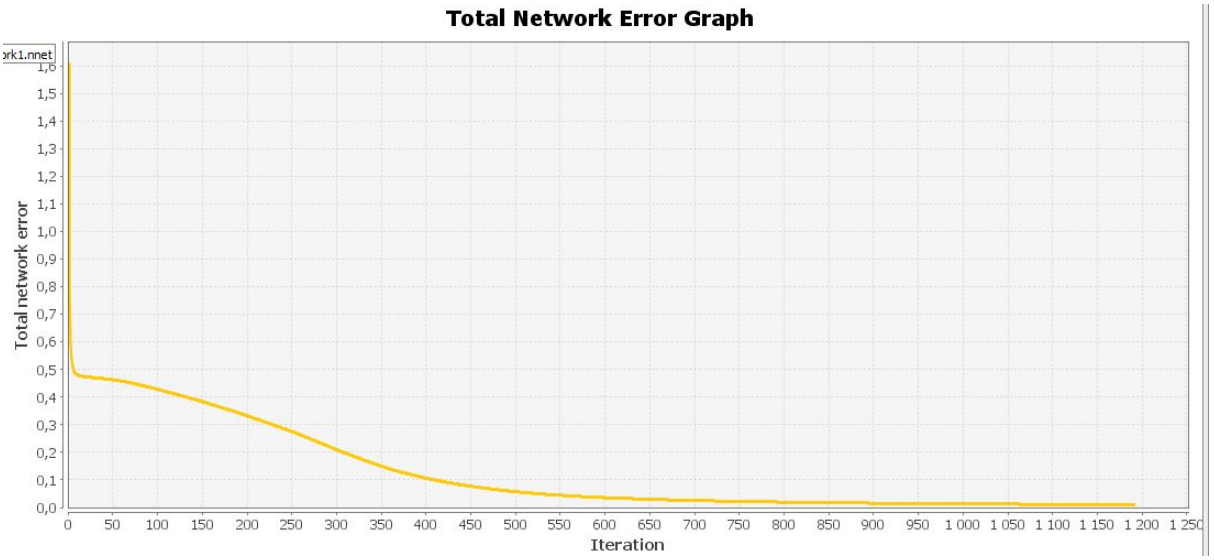


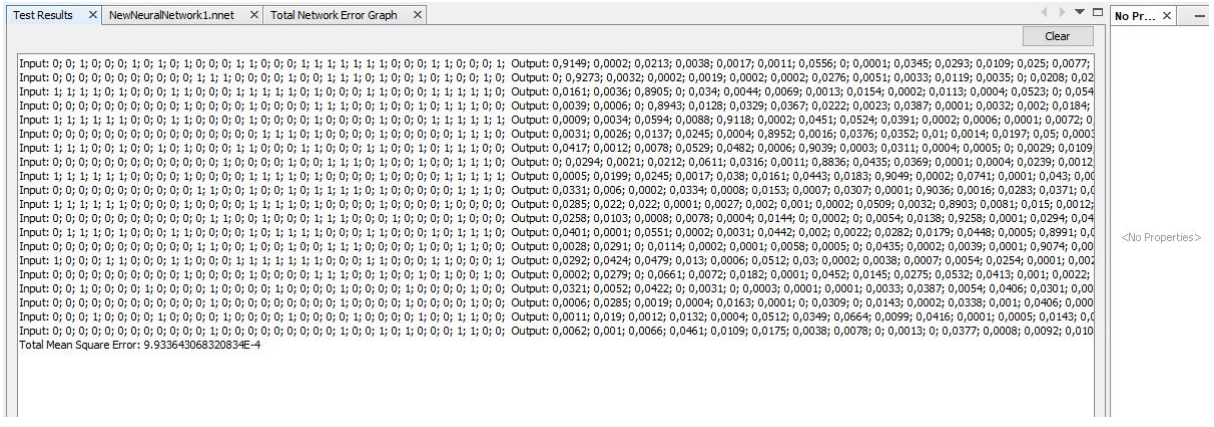
Współczynnik uczenia: 0.2, Momentum: 0.1





Współczynnik uczenia: 0.2, Momentum: 2





Ogólny schemat procesu trenowania sieci wygląda następująco:

1.Ustalamy topologię sieci, tzn. liczbę warstw, liczbę neuronów w warstwach.

2.Inicjujemy wagi losowo (na małe wartości).

3.Dla danego wektora uczącego obliczamy odpowiedź sieci (warstwa po warstwie).

4.Każdy neuron wyjściowy oblicza swój błąd, oparty na różnicy pomiędzy obliczoną odpowiedzią y oraz poprawną odpowiedzią t.

5.Błędy propagowane są do wcześniejszych warstw.

6.Każdy neuron (również w warstwach ukrytych) modyfikuje wagi na podstawie wartości błędu i wielkości przetwarzanych w tym kroku sygnałów.
7.Powtarzamy od punktu 3. dla kolejnych wektorów uczących. Gdy wszystkie wektory zostaną użyte, losowo zmieniamy ich kolejność i zaczynamy wykorzystywać powtórnie.
8.Zatrzymujemy się, gdy średni błąd na danych treningowych przestanie maleć. Możemy też co jakiś czas testować sieć na specjalnej puli nieużywanych do treningu próbek testowych i kończyć trenowanie, gdy błąd przestanie maleć.

Wzór na zmianę konkretnej wagi wygląda następująco:

$$\Delta w_j = q \cdot (t - y) \cdot f'(s) \cdot z_j$$

Powyższe wyprowadzenie jest prawidłowe tylko dla ostatniej warstwy, kiedy wiemy, jaka powinna być prawidłowa wartość wyjścia (wówczas możemy policzyć błąd). Dla wcześniejszych warstw takiej informacji bezpośrednio nie mamy. Zamiast tego błąd ten będziemy przybliżać, przenosząc go (propagując) z kolejnych warstw. Oznaczmy: w - waga wejścia neuronu, z - sygnał wchodzący do neuronu danym wejściem, d - współczynnik błędu obliczony dla danego neuronu, s - wartość wzbudzenia (suma wartości wejściowych pomnożonych przez wagi) dla danego neuronu. Pomocniczy współczynnik błędu d zdefiniujemy dla ostatniej warstwy jako:

$$d = f'(s) \cdot (t - y)$$

a dla pozostałych warstw:

$$d = f'(s) \cdot \sum_{i=1}^{n} w_i d_i$$

czyli neuron w warstwie ukrytej "zbiera" błędy d i z neuronów, z którymi jest połączony. Zmiana wag połączeń następuje jednocześnie we wszystkich warstwach, po fazie propagacji błędu (tak, aby wszelkie obliczenia na wszystkich warstwach odbywały się jeszcze na starych wartościach wag) i odbywa się według wzoru:

$$\Delta w = q \cdot d \cdot z$$

Wnioski:

Im mniejszy współczynnik uczenia się sieci, tym więcej czasu sieć potrzebuje na nauczenie się, przez co zwiększa się ilość iteracji. Na podstawie wyników można powiedzieć, że optymalną wartością momentum jest wartość 0.7 – zarówno mniejsze jak i większe wartości tego współczynnika dawały w rezultacie większą ilość iteracji potrzebną do nauczenia się sieci. Zauważono również, że wartość tego współczynnika nie ma wpływu na wartość średniego błędu kwadratowego. Podczas zmniejszania współczynnika uczenia sieci rośnie wartość średniego błędu kwadratowego.