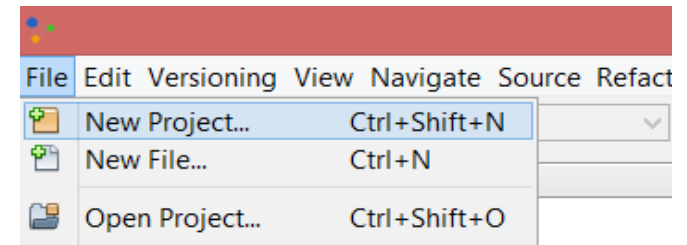


NeurophStudio

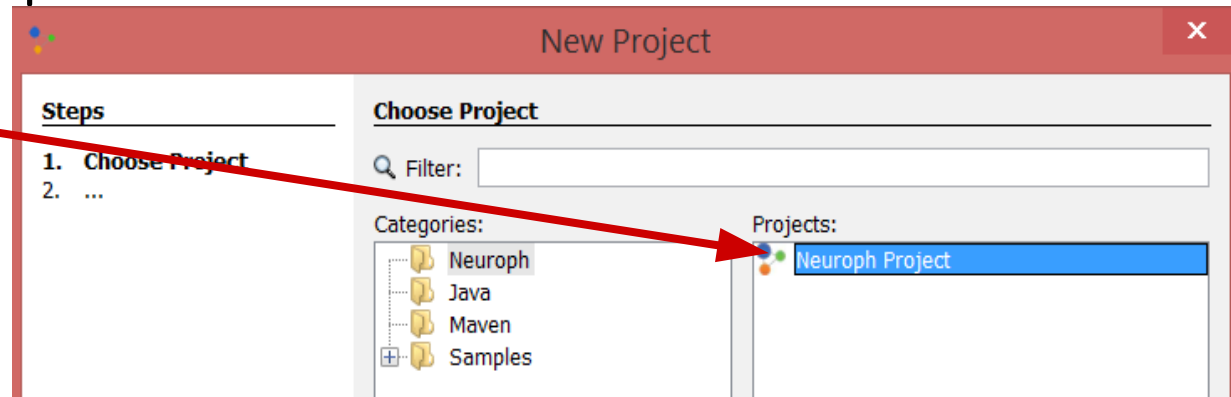
- ✓ <http://neuroph.sourceforge.net/download.html>
- ✓ wybrałam ten program ponieważ ma opcję rozpoznawania obrazów

Nowy projekt

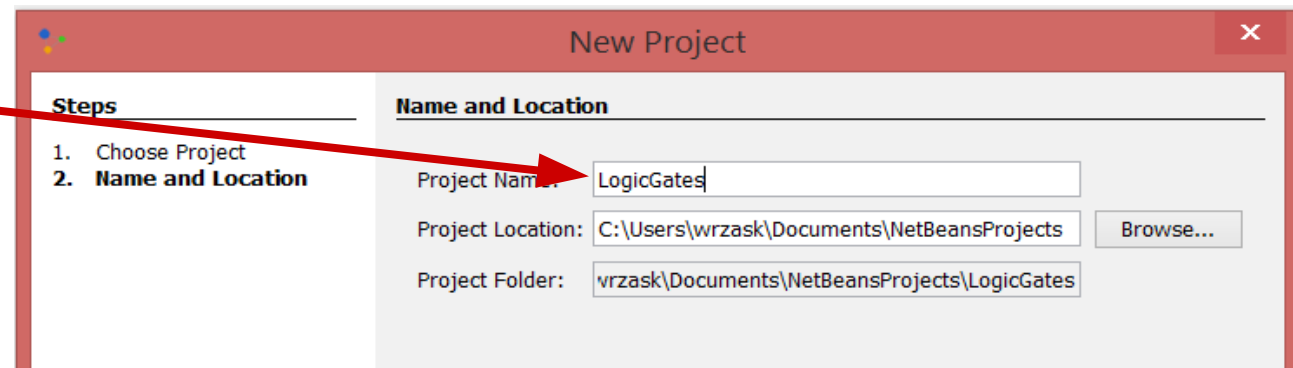
✓ Otwieramy nowy projekt



✓ Wybieramy kategorię Neuroph
i Neuroph Project

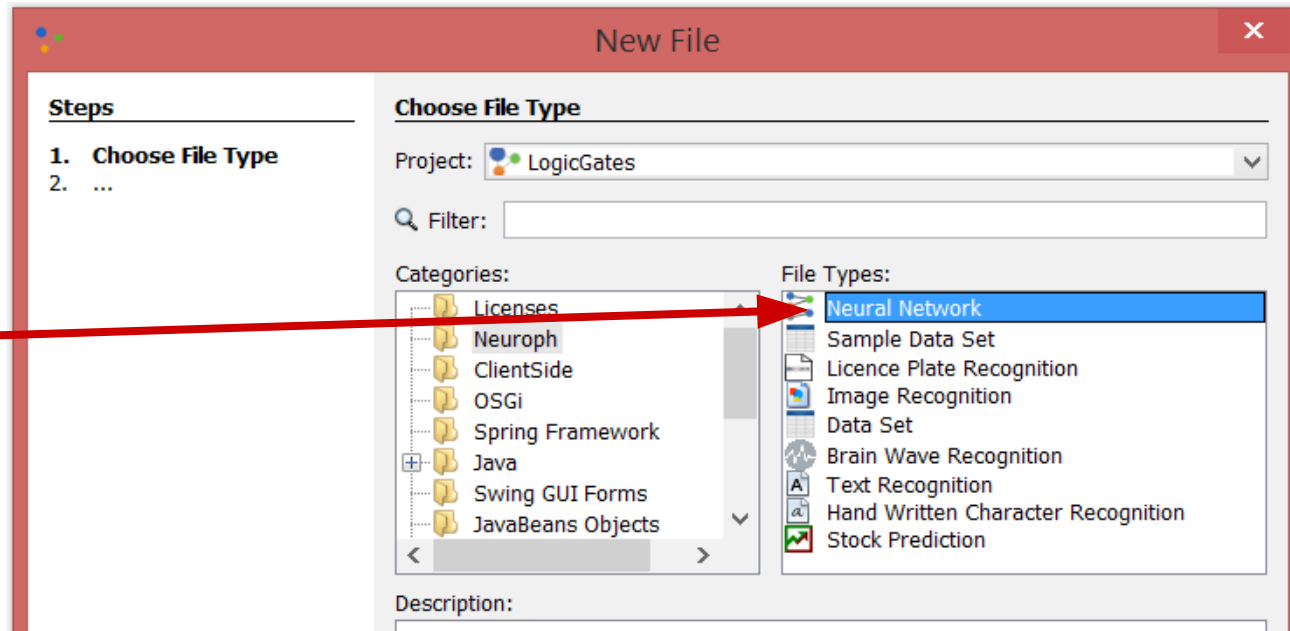


✓ Podajemy nazwę projektu
np. LogicGates



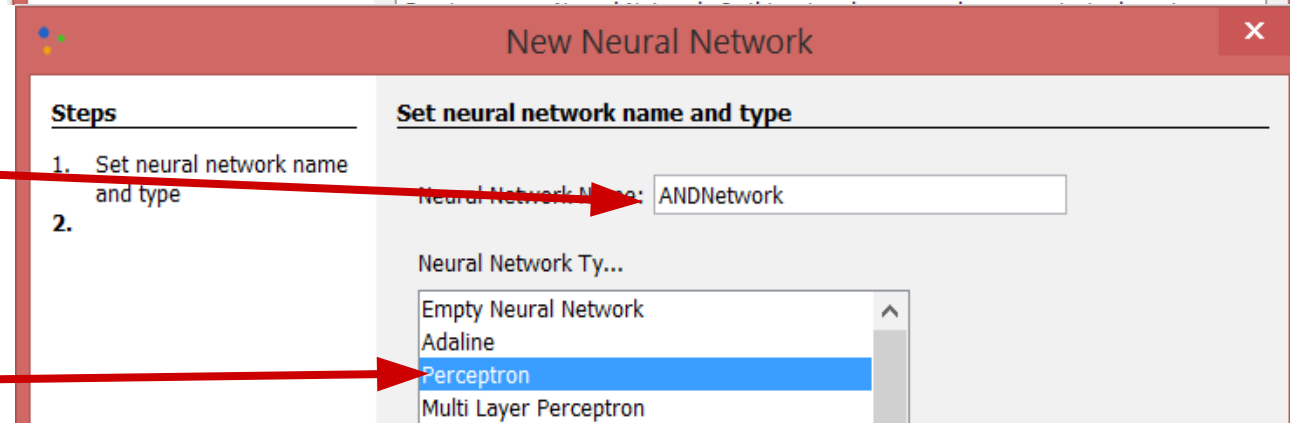
Nowa sieć

- ✓ Otwieramy nowy plik
- ✓ W projekcie Logic gates wybieramy Neuroph
 - Neural Network



- ✓ Podajemy nazwę sieci np. ANDNetwork

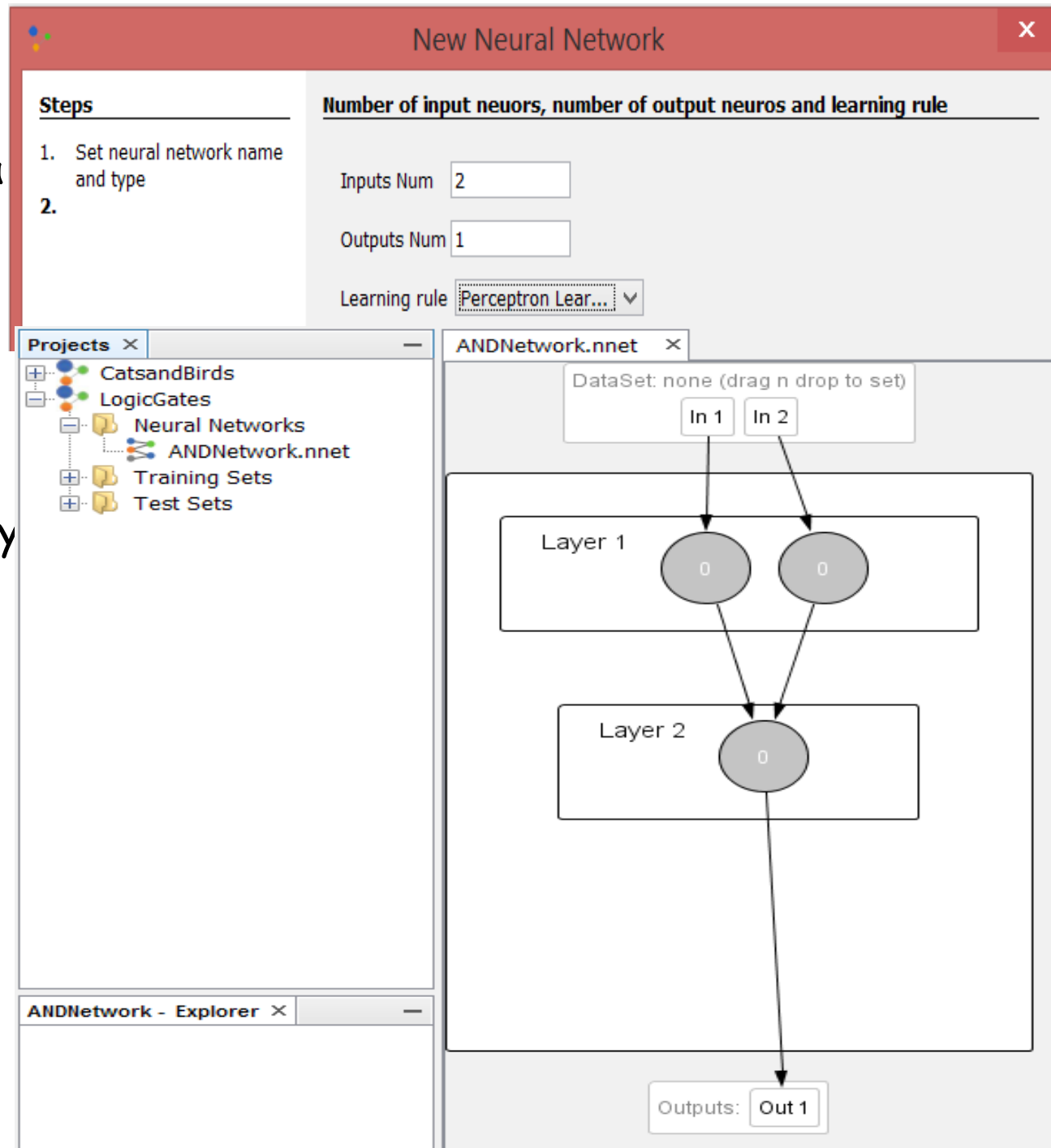
- ✓ Wybieramy rodzaj sieci
 - Perceptron



Nowa sieć

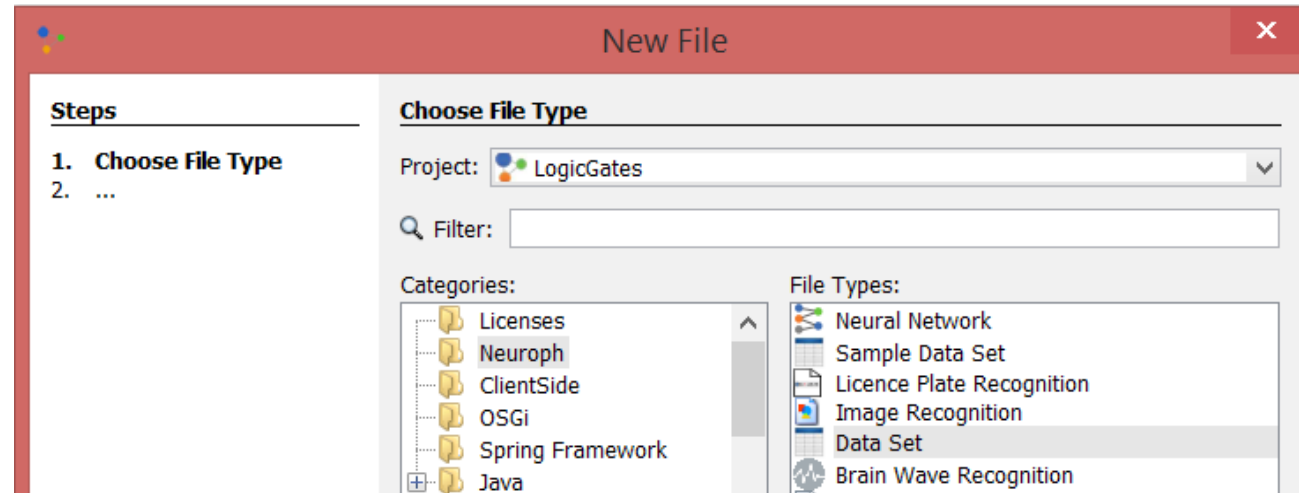
- ✓ Wybieramy liczbę wejść i Wyjść oraz regułę nauczania
→ Perceptron Learning

- ✓ W ten sposób stworzyliśmy sieć neuronową z dwoma neuronami na wejściu i jednym na wyjściu, co jest przedstawione graficznie na screenie obok

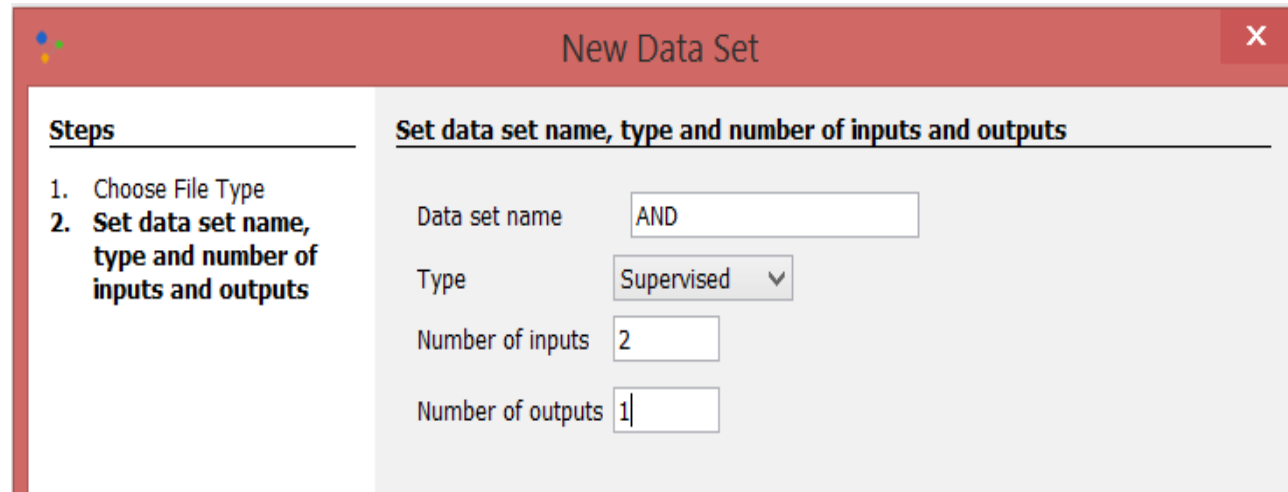


Trenowanie sieci

- ✓ Otwieramy nowy plik
- wybieramy Data Set

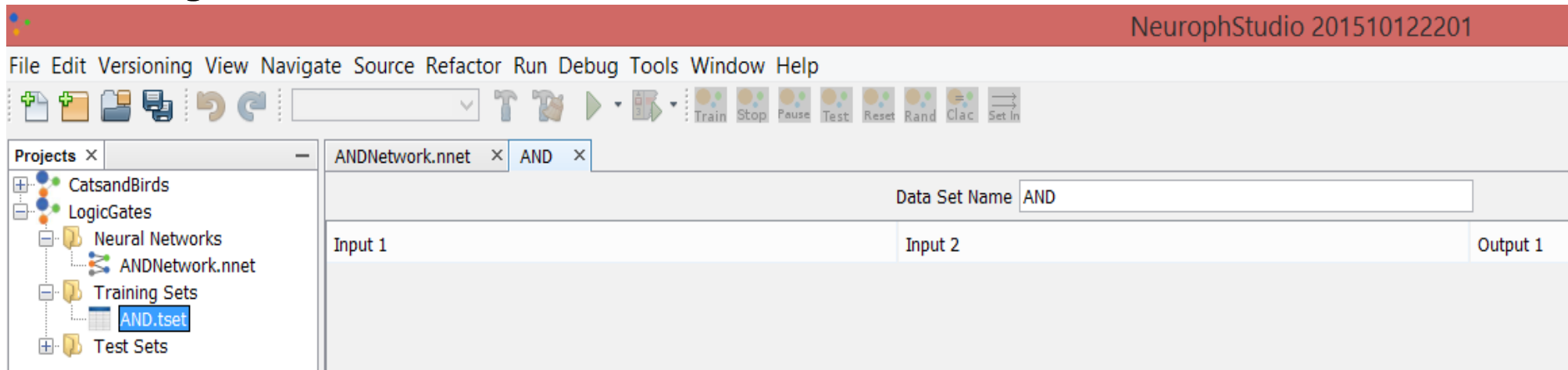


- ✓ Wybieramy nazwę tutaj AND
- liczbę wejść i wyjść
- oraz typ



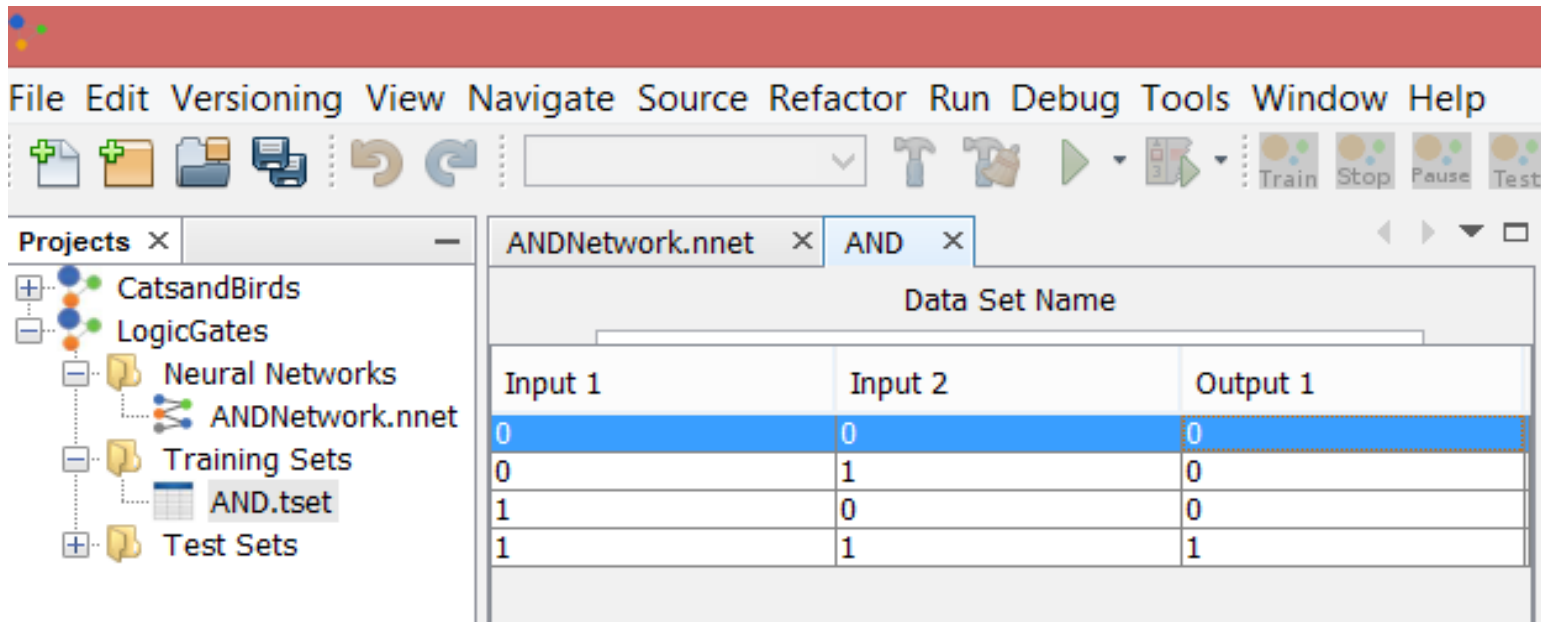
Trenowanie sieci

✓ W oknie pojawi się plik z danymi do trenowania (jeżeli nie to potrzeba dwukliknąć na ikonę obok And.tset). Dodajemy Add Row (przycisk na dole na screenie go nie widać)



Trenowanie sieci

✓ Poniżej wpisałam wejścia i wyjścia dla bramki AND, koniecznie zatwierdźcie dane przyciskiem OK na dole okna



The screenshot shows a software interface for training a neural network. The main window displays a table for defining a data set for an AND gate. The table has columns for Input 1, Input 2, and Output 1. The first row is highlighted in blue and contains the values 0, 0, and 0. The second row contains 0, 1, and 0. The third row contains 1, 0, and 0. The fourth row contains 1, 1, and 1. The table is titled "Data Set Name".

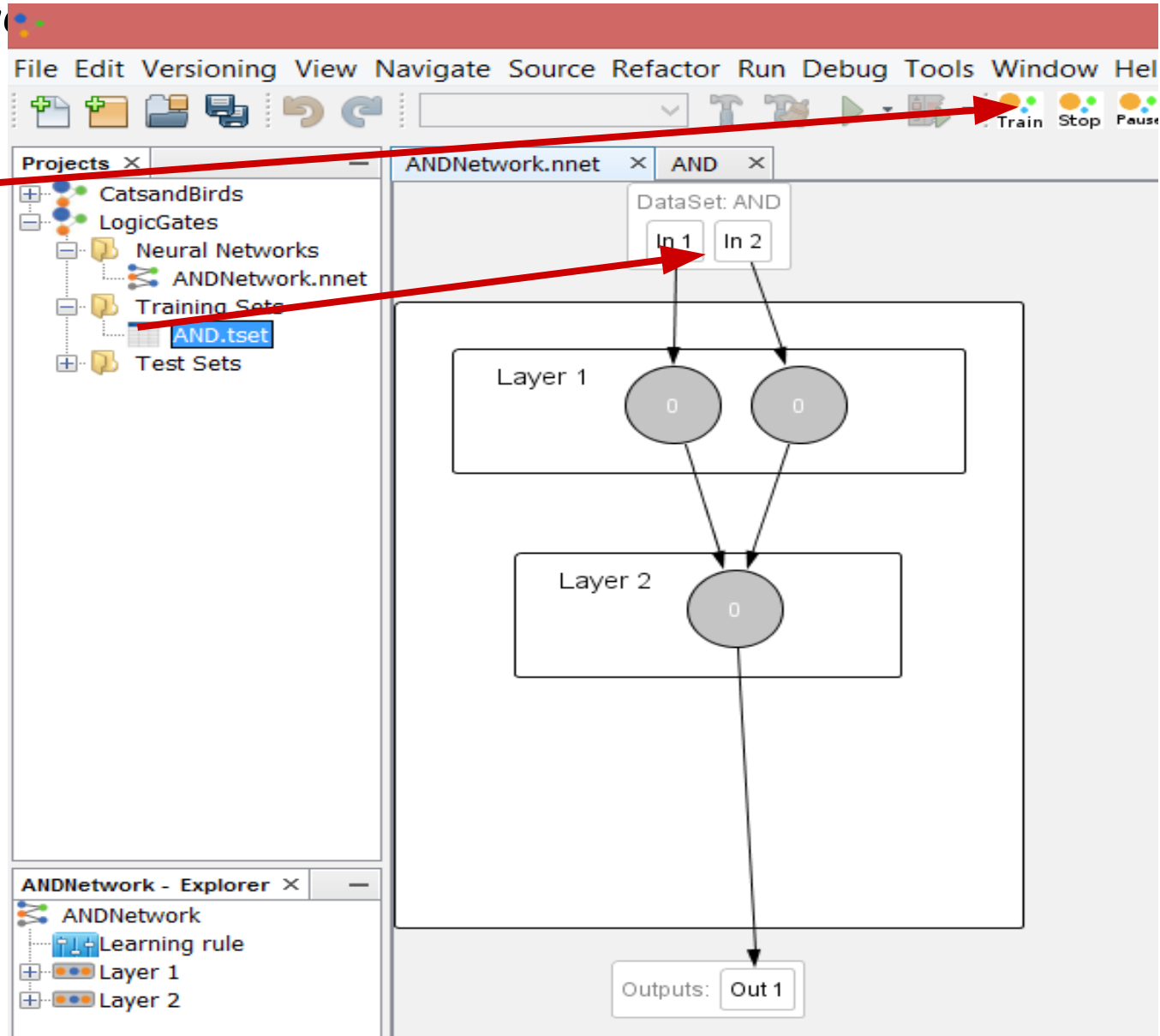
Input 1	Input 2	Output 1
0	0	0
0	1	0
1	0	0
1	1	1

Trenowanie sieci

✓ Przechodzimy do zakładki AndNetwork.nnet

✓ Przeciągam dane testowe do wejścia (drag&drop)

✓ Train



Trenowanie sieci

✓ Pojawi się okno dialogowe

✓ Celem uczenia się sieci jest zminimalizowanie funkcji błędu

✓ Pozostałe parametry:

Learning rate - parametr kontroluje

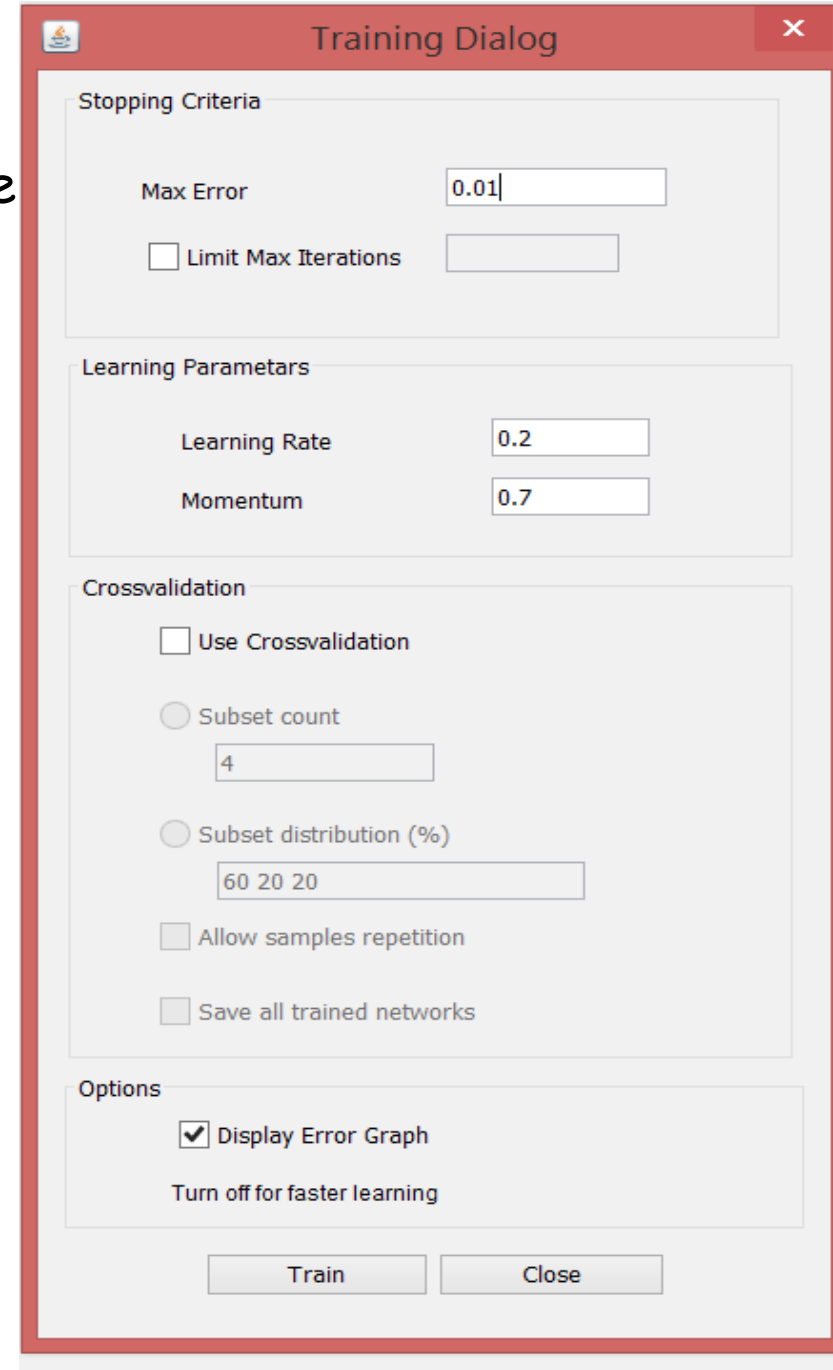
wielkość wag jest z przedziału $[0,1]$

Momentum - dodaje pewien ułamek wagi z

poprzedniej iteracji, zapobiega wpadaniu

w minima lokalne, także z przedziału $[0,1]$

✓ Zatwierdzam parametry przyciskiem Train



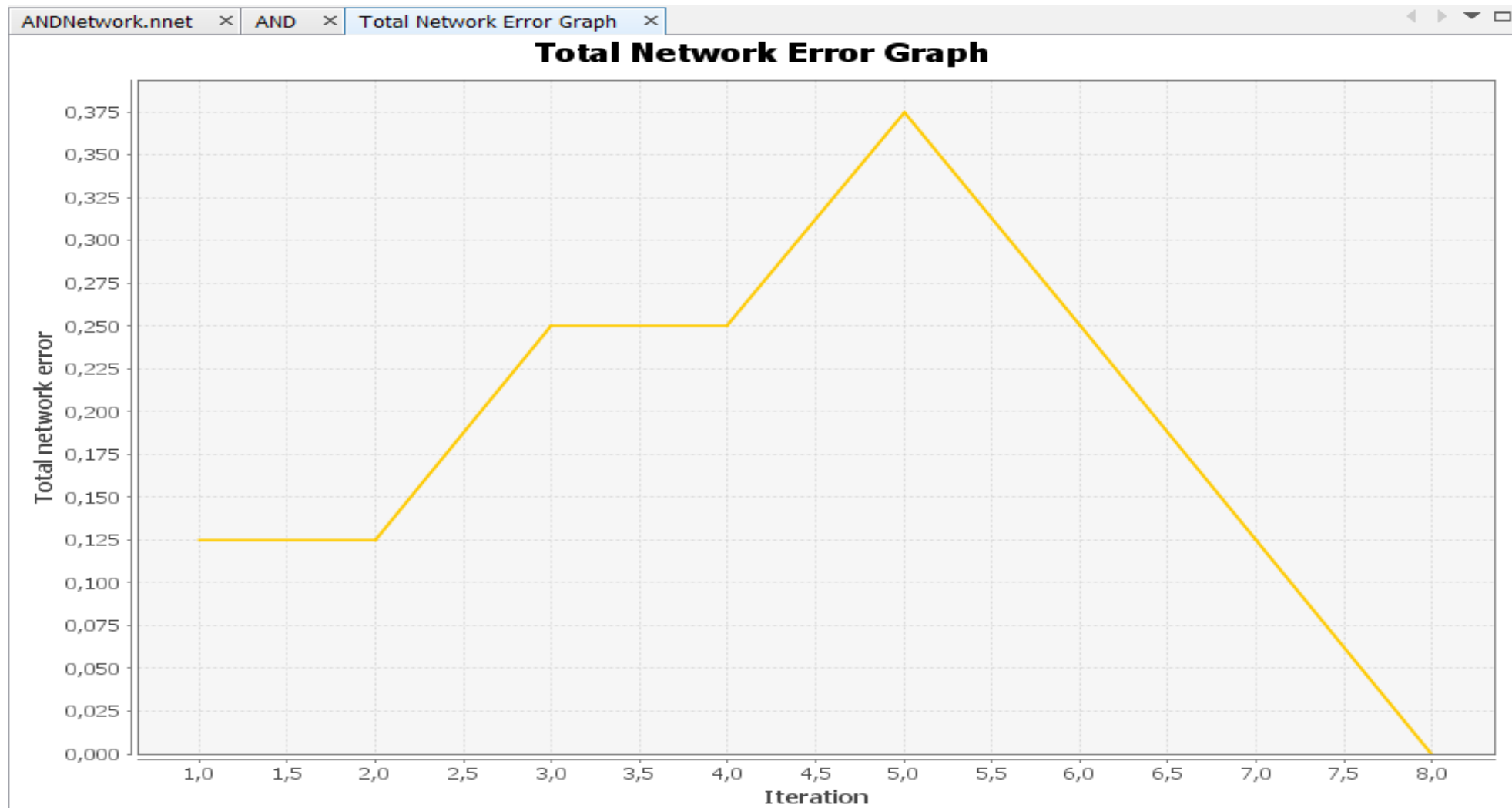
The screenshot shows a 'Training Dialog' window with the following settings:

- Stopping Criteria:**
 - Max Error: 0.01
 - Limit Max Iterations
- Learning Parameters:**
 - Learning Rate: 0.2
 - Momentum: 0.7
- Crossvalidation:**
 - Use Crossvalidation
 - Subset count: 4
 - Subset distribution (%): 60 20 20
 - Allow samples repetition
 - Save all trained networks
- Options:**
 - Display Error Graph
 - Turn off for faster learning

Buttons: Train, Close

Trenowanie sieci

✓ Po ośmiu iteracjach błąd praktycznie spadł do zera



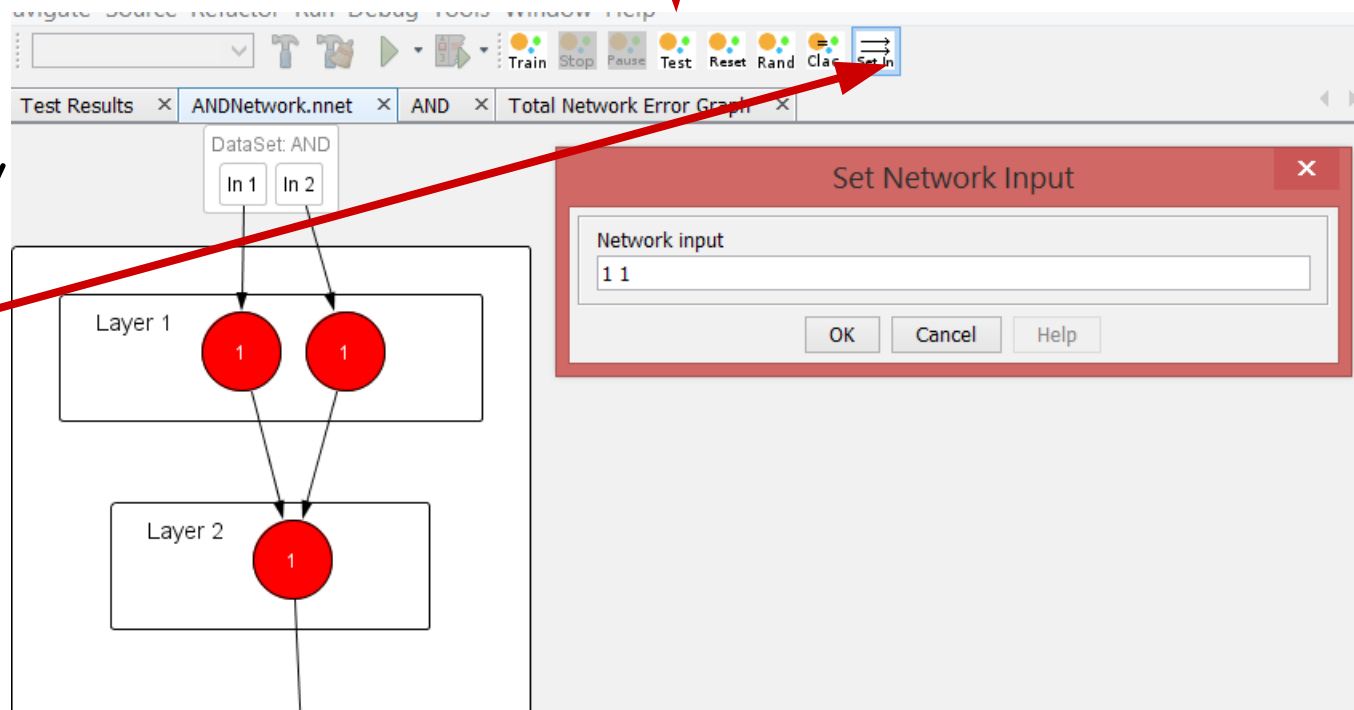
Testowanie sieci

✓ Aby przetestować sieć możemy użyć przycisku test

Wtedy w nowym oknie pojawią się wyniki testów

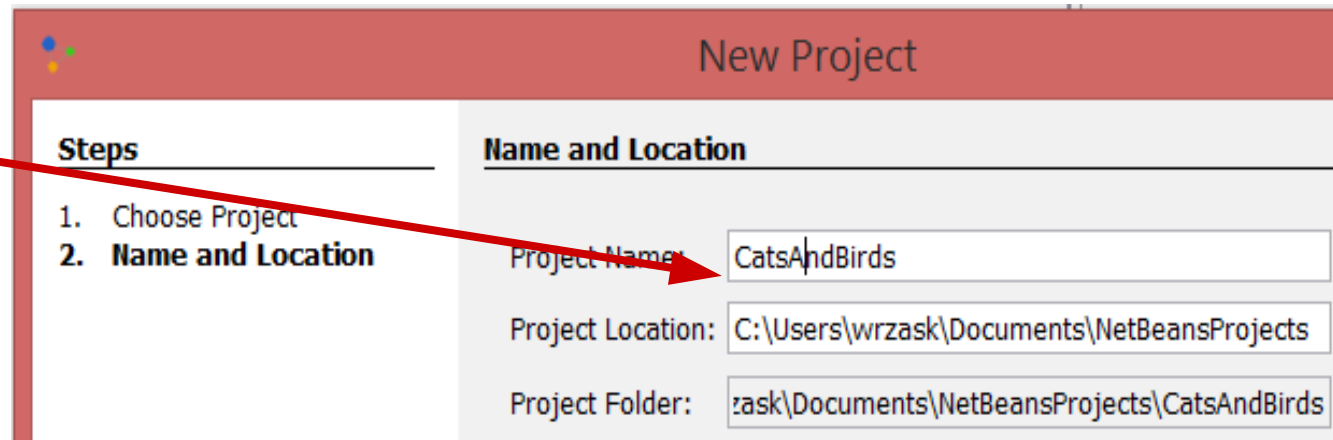
✓ Możemy także przetestować pojedynczy przykład wybierając Set input

```
Test Results x ANDNetwork.nnet x AND x Total N
Input: 0; 0; Output: 0; Desired output: 0; Error: 0;
Input: 0; 1; Output: 0; Desired output: 0; Error: 0;
Input: 1; 0; Output: 0; Desired output: 0; Error: 0;
Input: 1; 1; Output: 1; Desired output: 1; Error: 0;
Total Mean Square Error: 0.0
```



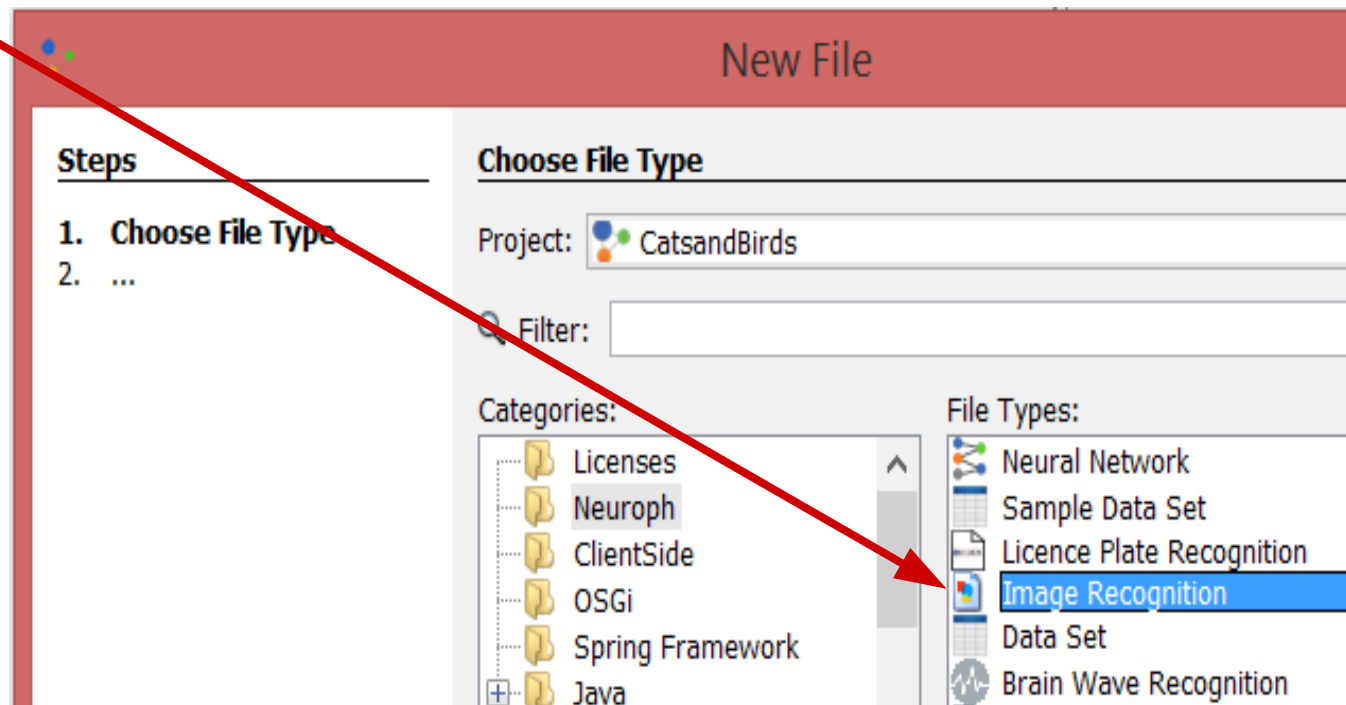
Rozpoznawanie obrazów

✓ Tworzymy nowy projekt



✓ Wybieramy typ pliku

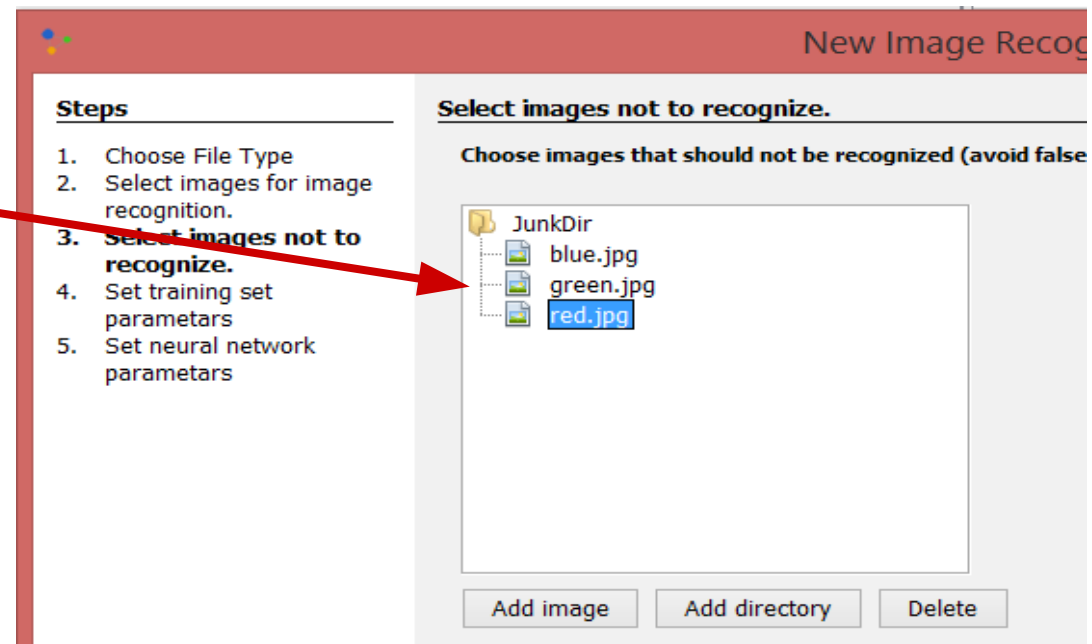
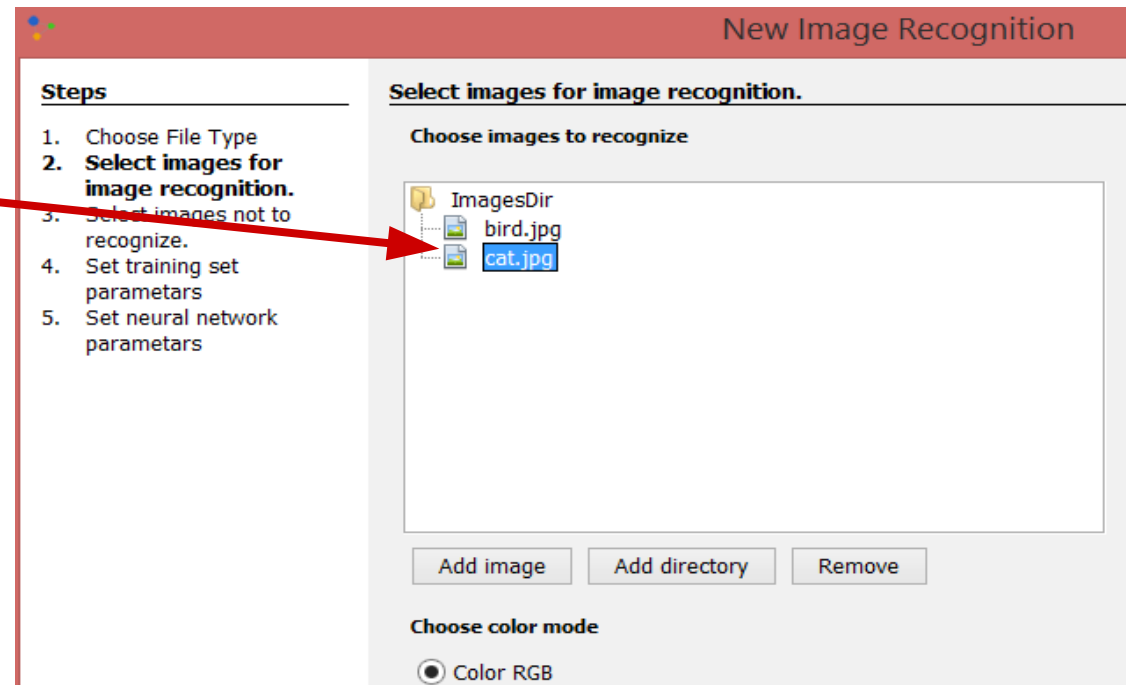
Image Recognition



Rozpoznawanie obrazów

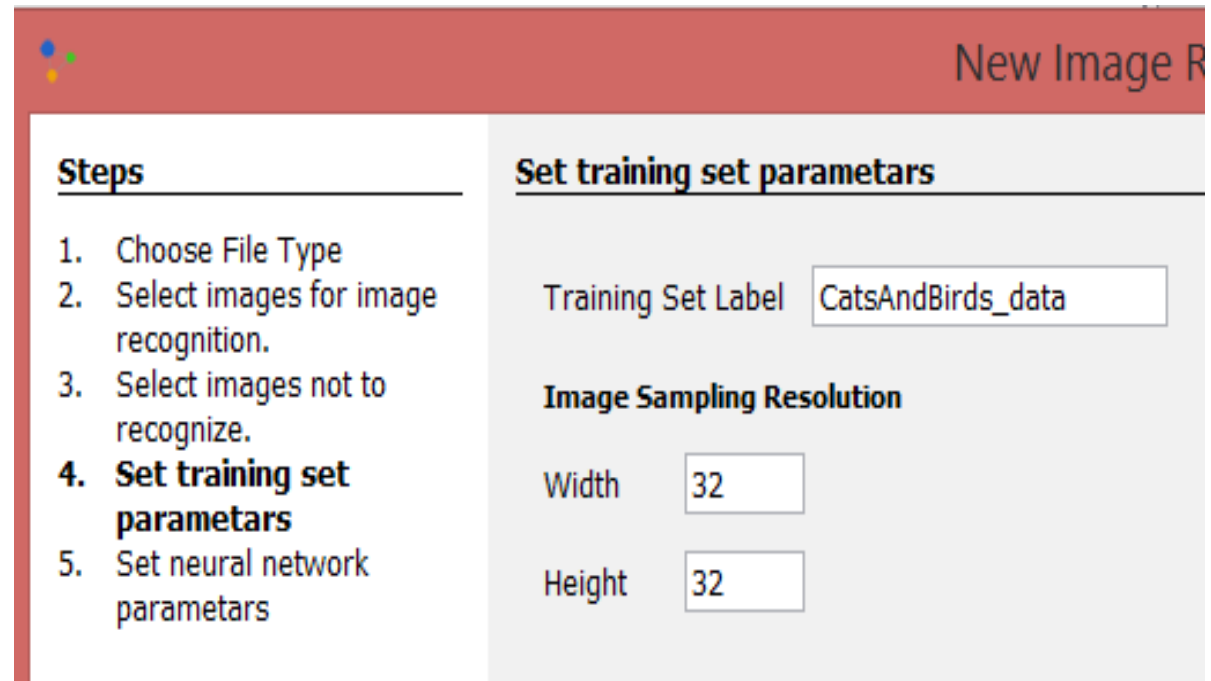
✓ Wybieramy obrazy do trenowania sieci: ptak i kot, przyciskiem Add image

✓ a także obrazy, które z pewnością nie przedstawiają kota czy ptaka

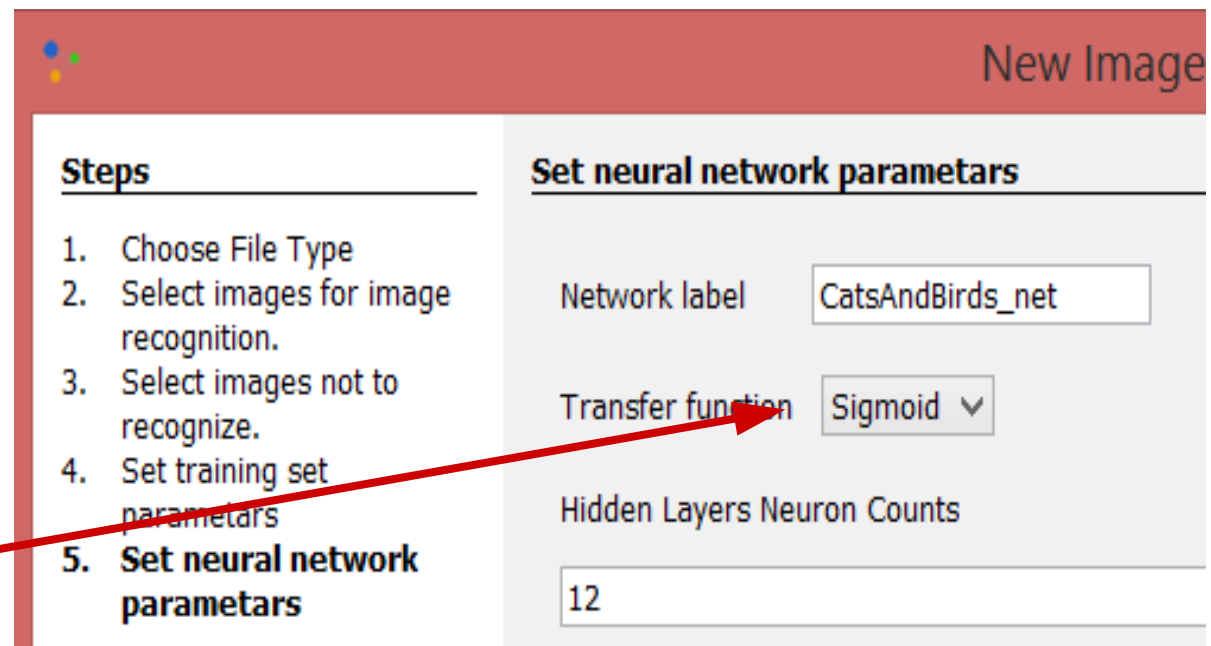


Rozpoznawanie obrazów

✓ W tym kroku należy podać nazwę danych do testów oraz liczbę próbek obrazu (w innym projekcie wzięłam 16 na 16, po prostu przy tylu danych program się wieszał)

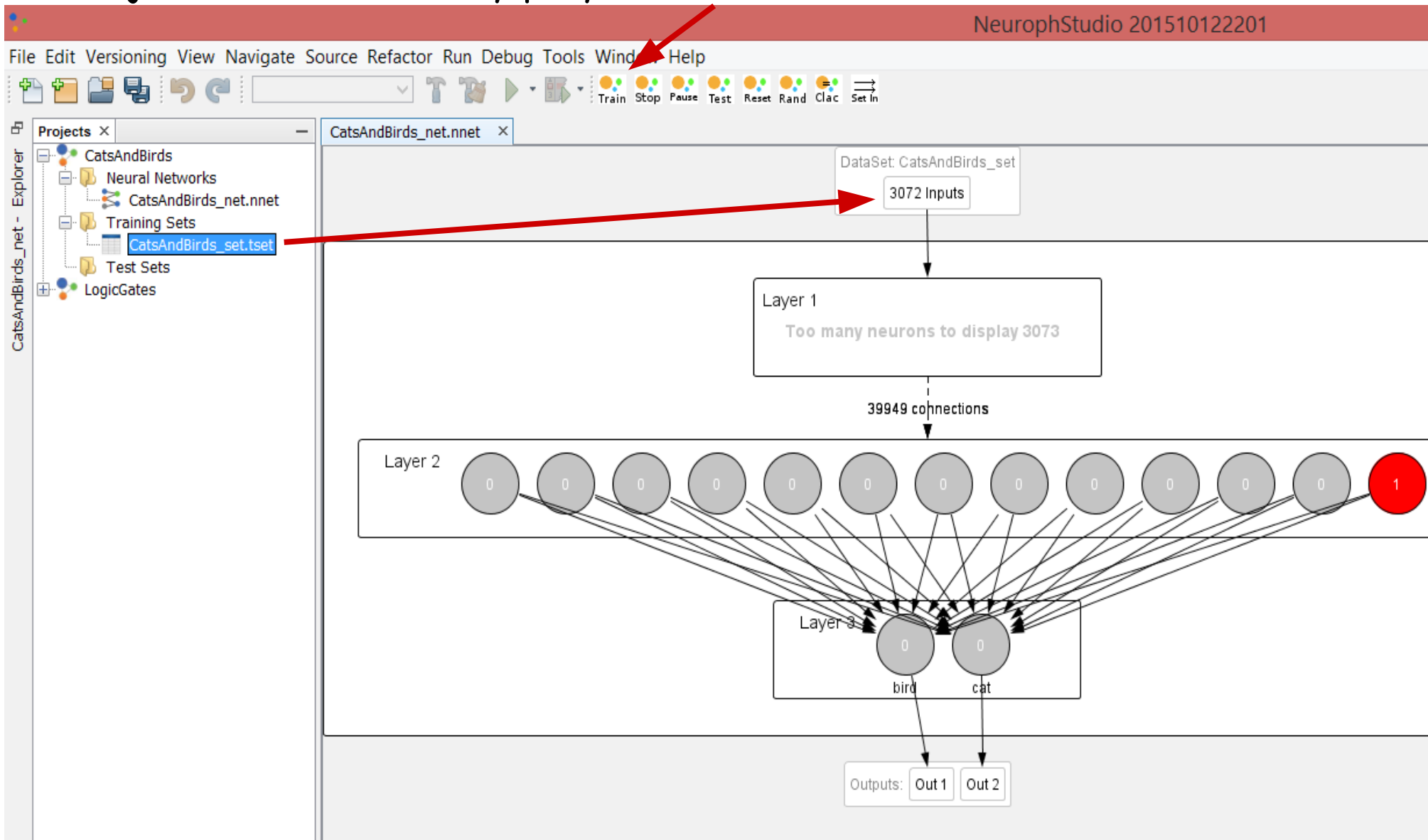


✓ Ustawiamy parametry Sieci - wpisałam 12 neuronów w warstwie ukrytej, ustalamy także funkcję aktywacji



Rozpoznawanie obrazów

- ✓ Przechodzimy do trenowania sieci: dane testowe przeliczamy (drag&drop) do wejścia sieci i naciskamy przycisk Train



Rozpoznawanie obrazów

✓ Przechodzimy do trenowania sieci. Celem uczenia się sieci jest zminimalizowanie funkcji błędu

✓ Pozostałe parametry:

Learning rate - parametr kontroluje

wielkość wag jest z przedziału $[0,1]$

Momentum - dodaje pewien ułamek wagi z

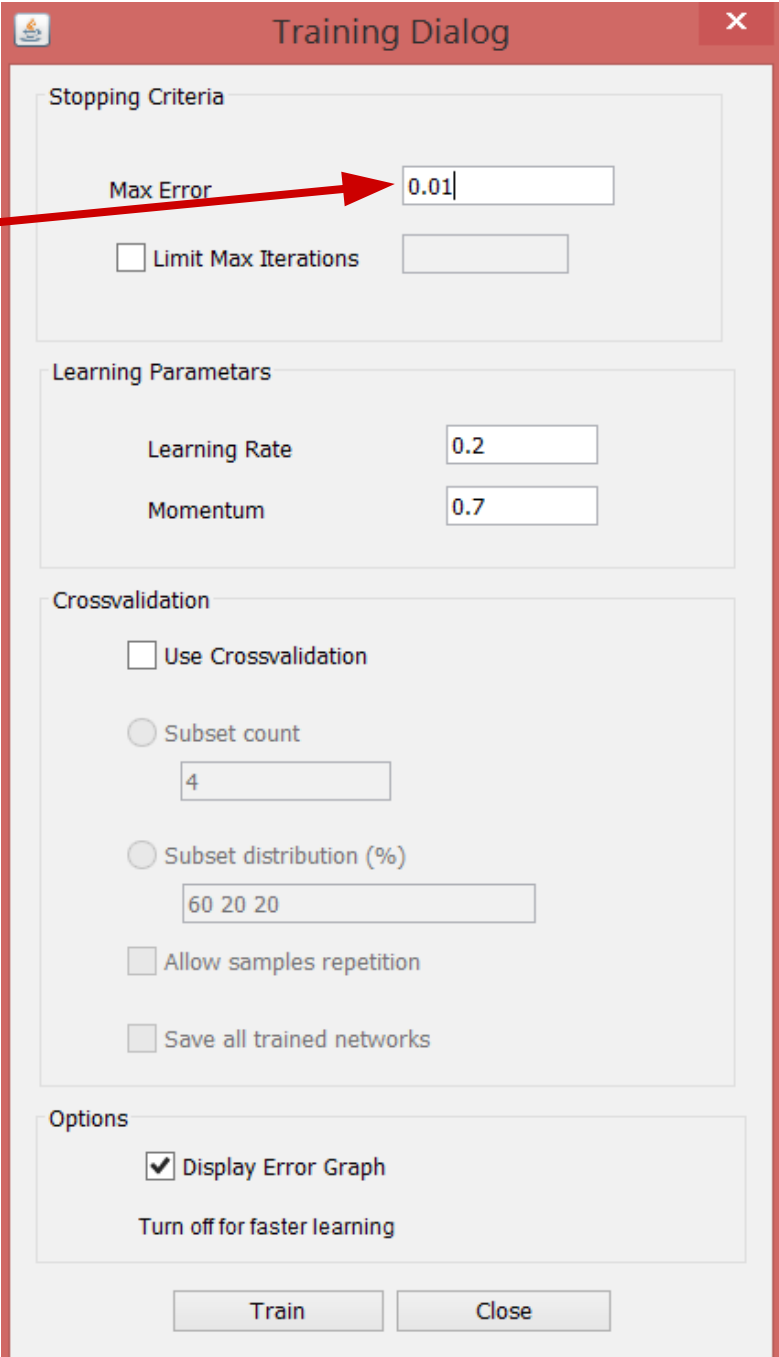
poprzedniej iteracji, zapobiega wpadaniu

w minima lokalne, także z przedziału $[0,1]$. Zbyt

niska wartość nie zapobiega wpadnięciu w

minimum lokalne, zbyt wysoka powoduje

niestabilności



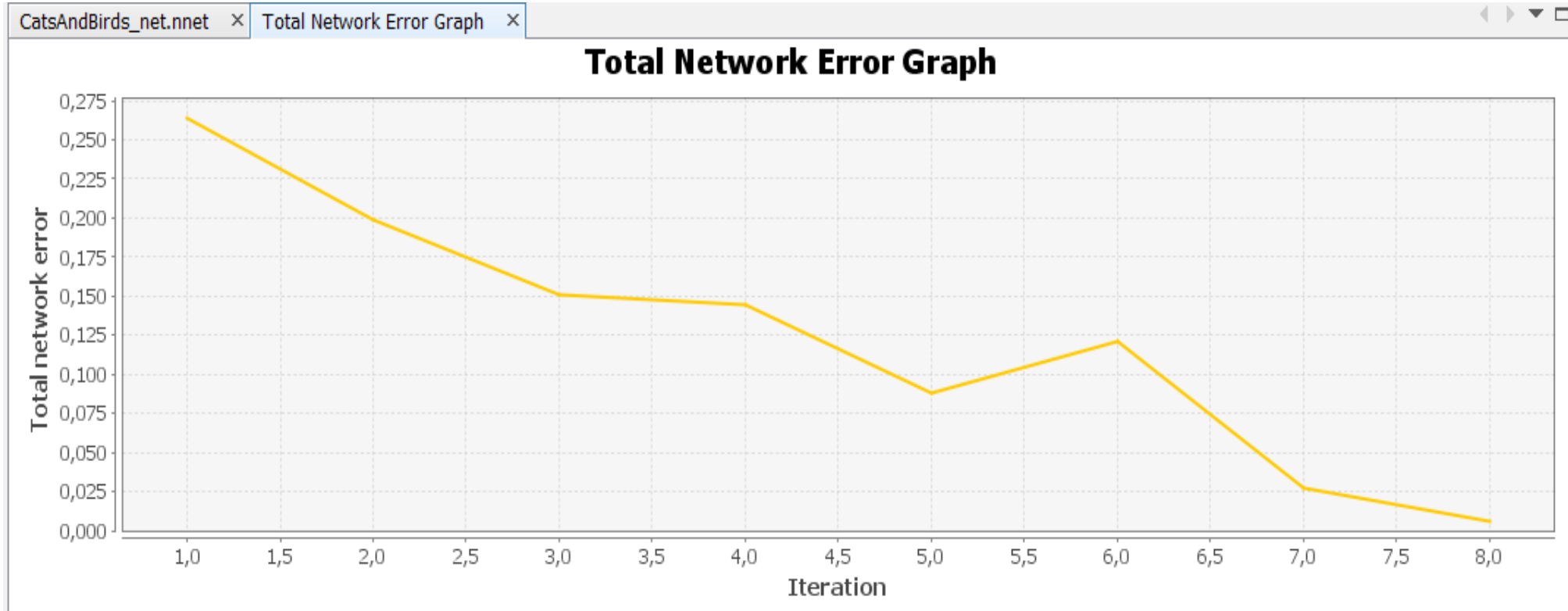
The image shows a 'Training Dialog' window with the following settings:

- Stopping Criteria:**
 - Max Error: 0.01
 - Limit Max Iterations: (empty)
- Learning Parameters:**
 - Learning Rate: 0.2
 - Momentum: 0.7
- Crossvalidation:**
 - Use Crossvalidation:
 - Subset count: 4
 - Subset distribution (%): 60 20 20
 - Allow samples repetition:
 - Save all trained networks:
- Options:**
 - Display Error Graph:
 - Turn off for faster learning: (checkbox)

Buttons: Train, Close

Rozpoznawanie obrazów

✓ Po ośmiu iteracjach funkcja błędu spadła poniżej ustalonego wcześniej parametru



Rozpoznawanie obrazów

✓ W zakładce Image Recognition Test testuję także inne obrazy kotów i ptaków - na przykładzie widać,

że grafika tego ptaszka bardziej przypomina sieci ptaka niż kota. Ładujemy nowe obrazy do testów przy pomocy Select Test Image

Image Recognition Test × Total Network Error Graph × CatsAndBirds_net.nnet × Total Network Error Gra

Testing network Test data set

Select Test Image Test whole data set

Output ×

JMonkeyEngine Logs × Neuroph × Image Recognition Test × Image Recognition Results ×

```
bird : 0,0257
cat : 0
|
```