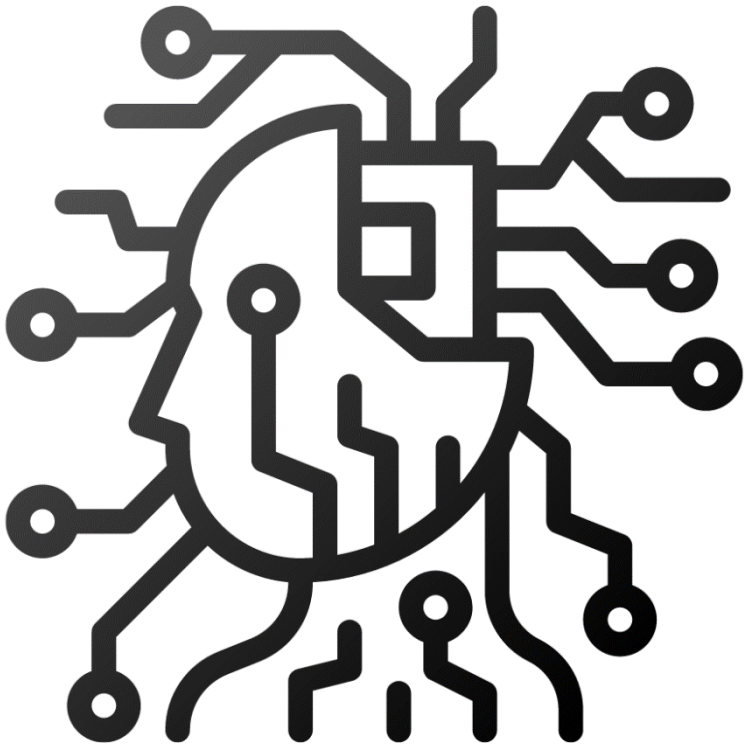


# Algorytm wstecznej propagacji błędu
# Odczytywanie liter z alfabetu

Tomasz Lewandowski
Bartosz Prządka
Grupa 3

# Spis treści

# 1.Wstęp

## Zasada Działania.

Sieć neuronowa zawiera 3 warstwy wejściową (piksele obrazu wejściowego służą jako wejście do niego), ukrytą i wyjściową. Liczba neuronów warstwy wyjściowej zależy od liczby znaków, które mają być rozpoznawane dla 26 znaków będzie to 26 neuronów wyjściowych. Liczba neuronów warstwy wejściowej zależy od iloczynu wymiarów obrazu czy szerokości i wysokości. Warstwa ukryta ma ustaloną przez użytkownika liczbę neuronów. Każdy neuron w warstwie wyjściowej jest kategoryzowany poprzez zestaw danych wejściowych czyli poprawnego rozwiązania. Oznacza to, że do wytrenowania określonego znaku neuron przyjmujący dany znak otrzymuje wartość bliską 1, natomiast reszta wartości przy 0. Dokładności sieci neuronowej jest definiowana poprzez odległość rzeczywistą od wyniku docelowego, która pomniejsza się dzięki zastosowanemu wzorowi:

$$Błąd\ całkowity = \sum \frac{1}{2}\left(\text{wynik docelowy}_x - \text{wyjście}_x\right)^2$$

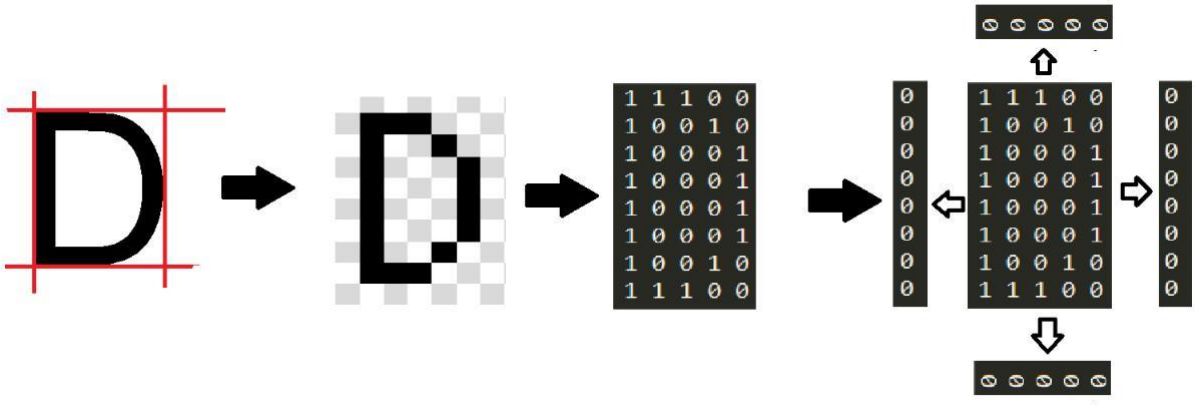
Gdzie X to warstwa wyjściowa.

Jest to podejście zmniejszania błędu średniokwadratowego nazywające się Opadaniem Gradientowym (Gradient Descent), które wraz z algorytmem Wstecznej propagacji błędu dostosowują wagi w taki sposób, aby błąd całkowity zbliżył się do globalnego minimum. Wagi i anomalie są inicjowana losowo w zakresie <-0.5 ; 0.5>. Gdyby zostały zainicjalizowane między 0 i 1 sieć neuronowa nigdy się nie zbiegnie ponieważ wejście wszystkich neuronów w warstwie ukrytej będzie wyższe niż aktywne wejście zakresu sigmoidy w wyniku czego wyjście byłoby zawsze 1.

Sieć jest zaprojektowana oraz szkolona w taki sposób że dla zestawu wejść „I” takich że dla każdego elementu „C” (pojedynczy znak) w zbiorze „I” tylko neuron „O” (neuron warstwy wyjściowej) klasyfikuje element „C” przypisując najwyższą wartość spośród innych neuronów wyjściowych. Kiedy sieć jest wytrenowana dla wszystkich elementów w zestawie „I” może być użyta do rozpoznania znaków.

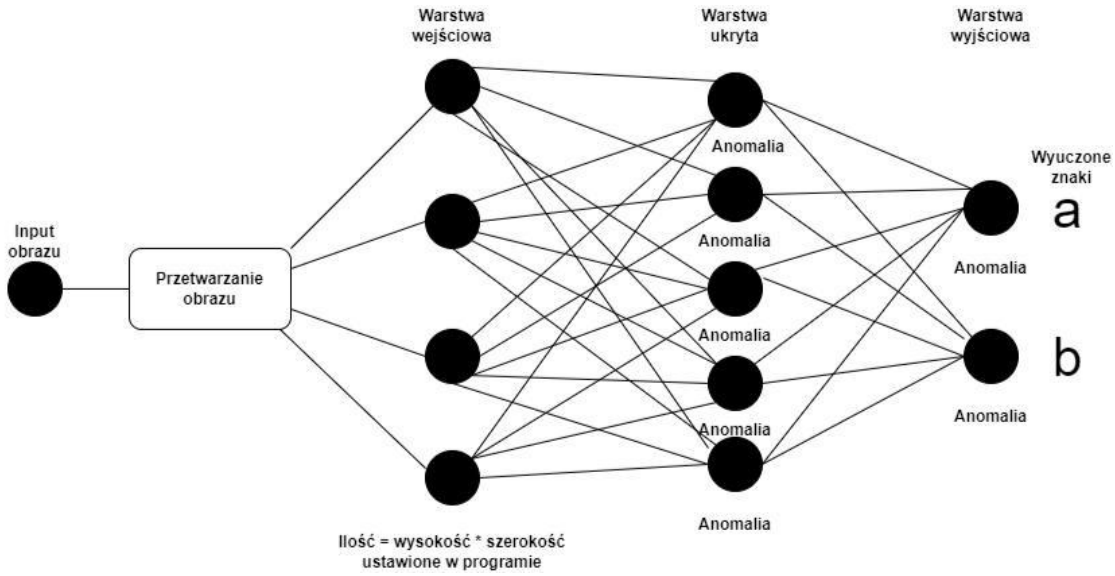
# 2. Przetwarzanie obrazu

W celu uzyskania jednolitości obrazu z neuronami warstwy wejściowej musimy zmienić jego rozmiar, aby to zrobić wykorzystujemy operacje binaryzacji czyli konwersji koloru na czarno miało. Następnie wykrywamy krawędzie obrysu dla każdego znaku czyli najbardziej wysuniętych jego końców aby uzyskać cztery krawędzie górną, dolną, lewą i prawą które składają się na kwadrat w którym znajduje się znak. Ostatni krok to operacja normalizacji czyli zmiana rozmiaru obrazu do wymiaru szerokości zdefiniowanych przy ilości neuronów warstwy wejściowej.



***Rys. 1**. Proces przetwarzania*

# 3. Architektura sieci




*Rys. 2.Architektura sieci*

*Rysunek 2* przedstawia budowę sieci. Warstwa wejściowa to macierz, która jest pomnożona przez wagi łączące wejścia do neuronów warstwy ukrytej. Zatem macierz wag między danymi na wejściowej warstwie a w warstwie ukrytej ma rozmiar równy iloczynowi liczby neuronów obu warstw. Funkcja aktywacyjna użyta dla warstwy ukrytej jak i warstwy wyjściowej to logistyczna funkcja sigmoidalna, oznacza to że ich wyjścia mieszą się w zakresie (0;1). Wyjścia warstwy ukrytej są podawana jako dane wejściowe do warstwy wyjściowej. Dane te są w następnej kolejności mnożone przez wagi łączące neurony warstwy ukrytej z neuronami warstwy wyjściowej. Neuron wyjściowy z maksymalną wartością wyjściową oznacza największe prawdopodobieństwo rozpoznania znaku. W ten sposób reszta neuronów wyjściowych osiągnie wartości około 0 co oznacza, że tylko neuron który klasyfikuje znak wejściowy uruchomi wartość około 1 i rozpozna go.

# 4. Rozpoznawanie znaków

Obraz do digitalizacji jest najpierw przeszukiwany pod kątem wystąpienia akapitów, a następnie wierszy. Znaki, które zostaną znalezione są wejściem do sieci neuronowej. Sieć wyświetli cyfrowy znak pasujący do znaku na obrazie.

Akapit jest definiowany jako kombinacja czarnych pikseli z białą przerwą. Kiedy zostanie znaleziony akapit, linia zostanie znaleziona przez wykrycie poziomego czarnego piksela definiującego górę, a ostatni napotkany poziomy piksel definiuje dno. Po znalezieniu linii znaki są znajdywane przez pierwszy wykryty pionowy czarny piksel, który definiuje lewą krawędź znaku, a następnie ostatni napotkany czarny pionowy piksel, który definiuje prawą krawędź znaku. Zabieg ten jest powtarzany, aż nie zostanie znaleziony kolejny pionowy piksel. Po znalezieniu wszystkich znaków słowo jest definiowane jako kombinacja znaków występujących po sobie aż do znalezienia większej przerwy między znakami. Po zeskanowaniu obrazu w poszukiwaniu znaków, każdy znak przechodzi fazę przetwarzania wstępnego, a następnie jest wprowadzany do sieci neuronowej w celu rozpoznania

# 5. Wyniki działania oraz testy.

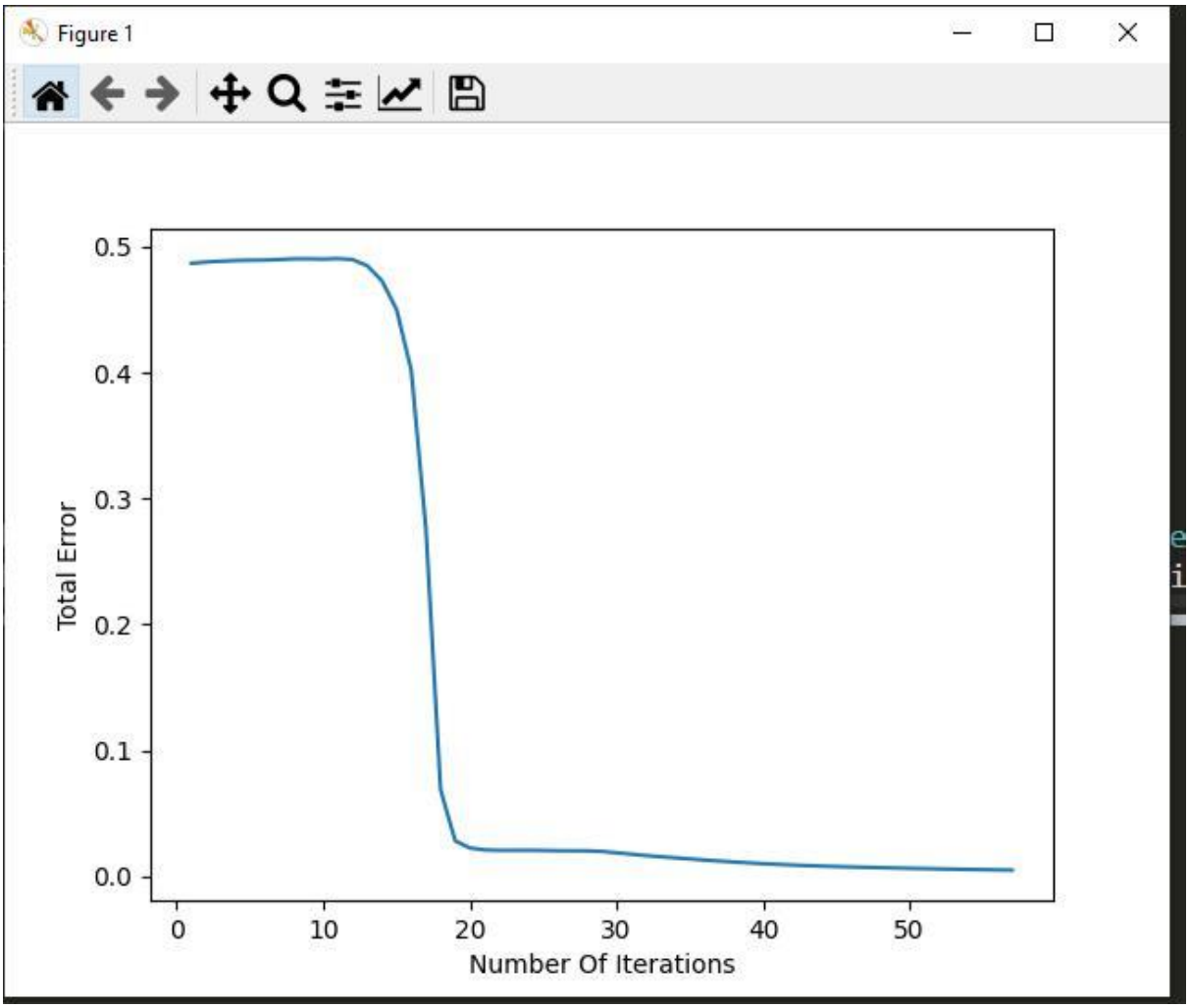
Parametry, które można ustawić w celu optymalizacji tempa oraz dokładności rozpoznawania znaków:

Szybkość uczenia się (*learning rate*)

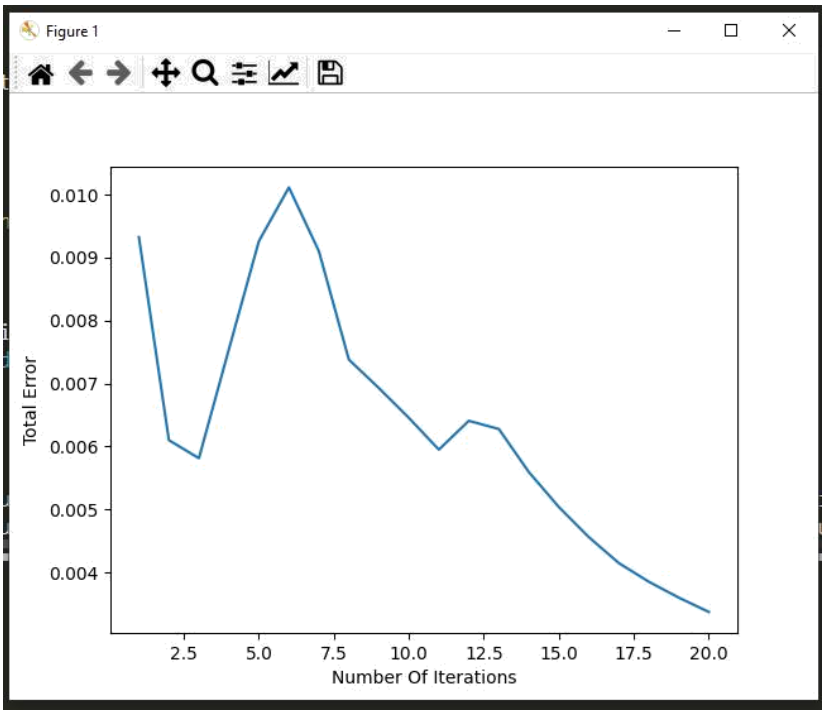
Ilość neuronów warstwy ukrytej(*hidden neurons*)

Docelowy poziom błędu(*total error*)

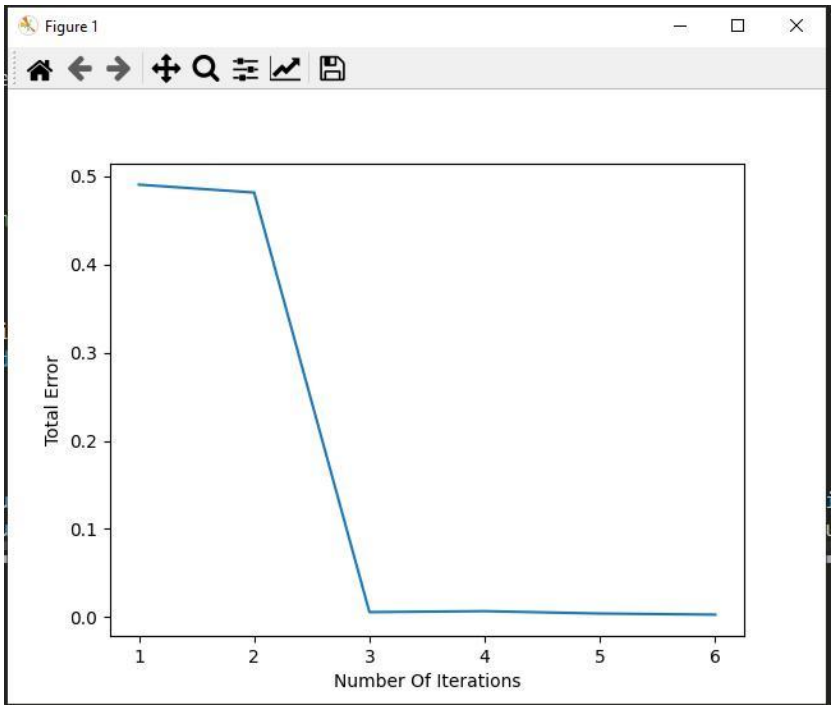
Zmiana learning rate




***Rys 3***. - Wynik eliminacji błędu przy learning rate = 0.1 momentum = 1 hidden neurons = 80 przy dobrze odczytanym tekście w 57 iteracji.

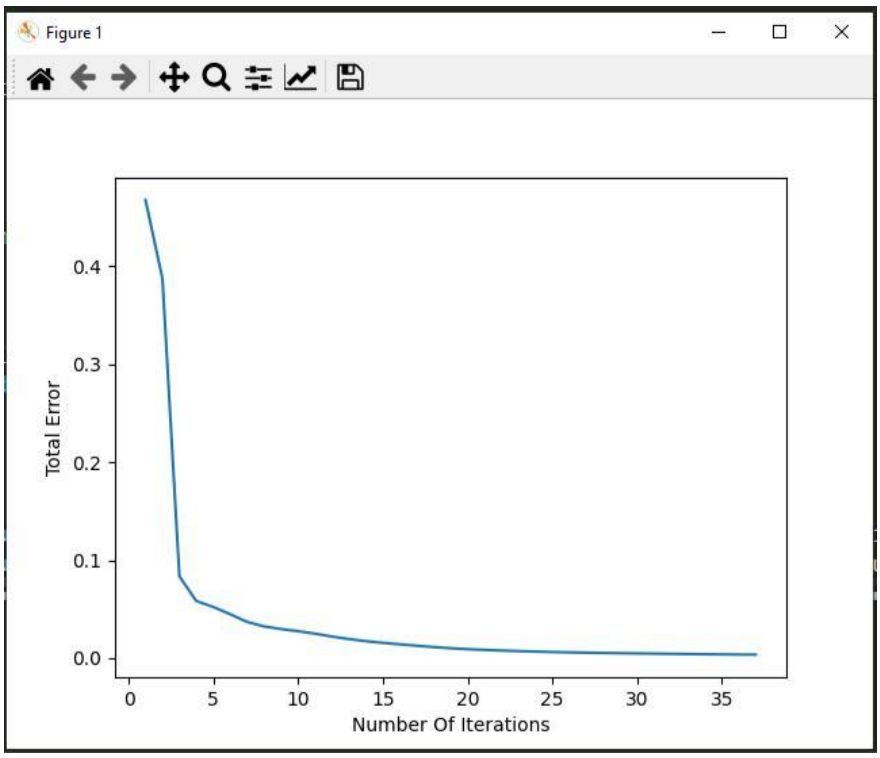



***Rys 4.*** Wynik eliminacji błędu przy learning rate = 0.5 momentum = 1 hidden neurons = 80 przy dobrze odczytanym tekście w 20 iteracji.

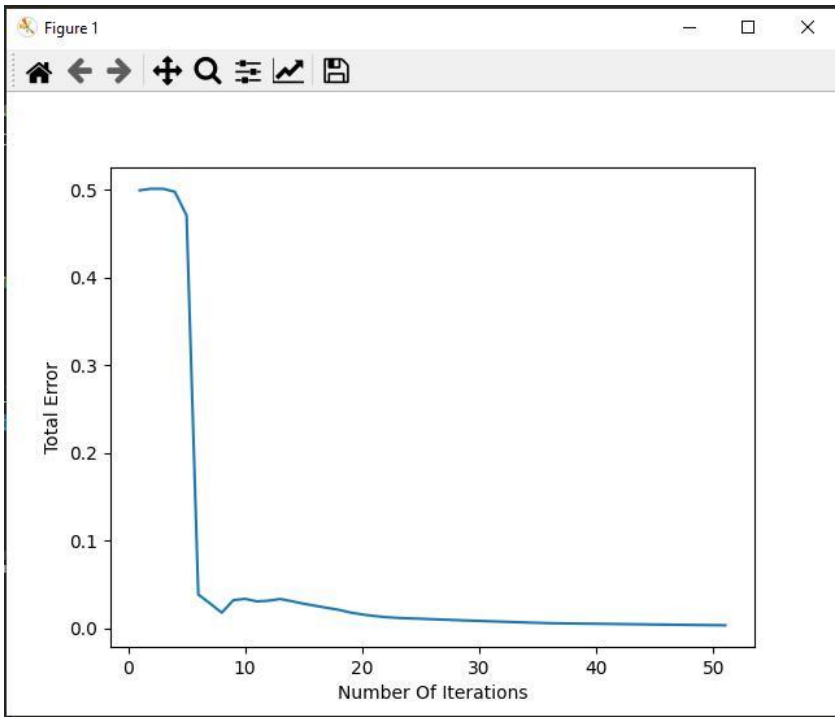



***Rys 5.*** Wynik eliminacji błędu przy learning rate = 1 momentum = 1 hidden neurons = 80 przy niepoprawnie odczytanym tekście w 6 iteracji.

**Zmiana liczby neuronów ukrytych.**




***Rys 6.*** Wynik eliminacji błędu przy learning rate = 0.3 momentum = 1 hidden neurons = 40 przy dobrze odczytanym tekście w 37 iteracji.




***Rys 7.*** Wynik eliminacji błędu przy learning rate = 0.3 momentum = 1 hidden neurons = 100 przy dobrze odczytanym tekście w 51 iteracji.

# 6. Wnioski

Program może odczytać 26 znaków (od A do Z). Najbardziej optymalne wyniki podczas uczenia maszynowego poprzez wsteczną propagacje błędu uzyskamy dobierając uśrednione parametry. Jak wynik z moich testów uśrednione parametry pozwalają na szybką naukę wraz z wysoko procentową skutecznością programu. Odczytywanie znaków podczas badań przebiegało wykorzystując ten sam obraz. Czyli zdjęcie tekstu napisanego w foncie Arial Black.


# 7. Instrukcja obsługi

Po uruchomieniu programu zapyta on użytkownika o wybranie 1-4 obrazków do rozpoznania znaków. Następnie użytkownik będzie musiał podać parametry szybkości uczenia, liczby ukrytych neuronów oraz docelowy poziom błędu. Najbardziej optymalnie jest podać

learning rate = 0.3 , hidden neurons = 80, target terror = 0.0035.

Następnie program przystąpi do przetwarzania zdjęcia i wyświetli nam oryginał. Po czym przystąpi do treningu w rozpoznawaniu na podstawie obrazu liter, może to zająć trochę w zależności od parametrów. Następnie wyświetli wykres przebiegu nauki. Po jego zbadaniu  i zamknięciu przejdzie do wypisania w konsoli poprawnych słów w tekście oraz zapisania ich do pliku „Wyjscie.tx

# 8. Źródła


https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

https://en.wikipedia.org/wiki/Backpropagation#Derivation

https://towardsdatascience.com/an-introduction-to-gradient-descent-and-backpropagation-81648bdb19b2

https://machinelearningmastery.com/implement-backpropagation-algorithm-scratch-python/

https://www.researchgate.net/publication/315512510_Optical_character_recognition_using_back_propagation_neural_network