Politechnika Warszawska

WYDZIAŁ ELEKTRYCZNY


Instytut Sterowania i Elektroniki Przemysłowej

# Praca dyplomowa inżynierska

na kierunku Informatyka Stosowana

w specjalności

Porównanie wydajności wybranych języków programowania w realizacji sieci neuronowych do przetwarzaniu obrazów

Piotr Heinzelman
numer albumu 146703

promotor
dr inż. Witold Czajewski

WARSZAWA 2024

## Porównanie wydajności wybranych języków programowania w realizacji sieci neuronowych do przetwarzaniu obrazów

### Streszczenie

W niniejszej pracy porównano języki programowania wraz z ich środowiskami w zastosowaniach tworzenia głębokich sieci neuronowych w dziedzinie widzenia komputerowego, czyli implementacji sieci CNN oraz MLP. Omówiono wybrane języki: Python, Maltab, Java, C++. Do budowania sieci wykorzystano biblioteki: YOLO11, (TensoRT? PyTorch) oraz TensorFlow, Scikit-Learn dla Python. LibTorch dla C++, Deep Learning Toolbox dla Matlab oraz własne implementacje dla języka Java. W porównaniu ujęto również takie własności języków jak: popularność, dostępność bibliotek, wygoda instalacji, wsparcie obliczeń na kartach graficznych, stopień trudności języka, koszt licencji.

**Słowa kluczowe:** uczenie głębokich sieci neuronowych, sieci splotowe CNN, YOLO, wydajność, klasyfikacja obrazów, obliczenia równoległe

## Comparison of the performance of selected programming languages in the implementation of neural networks for image processing

**Abstract**

This paper compares programming languages and their environments in applications of creating deep neural networks in the field of computer vision, i.e. implementing CNN and MLP networks. Selected languages are discussed: Python, Maltab, Java, C++. The following libraries were used to build the network: YOLO11, (TensoRT? PyTorch) and TensorFlow, Scikit-Learn for Python.
LibTorch for C++, Deep Learning Toolbox for Matlab and our own implementations for Java.
The comparison also includes such language properties as: popularity, availability of libraries, convenience of installation, support for computing on graphics cards, level of language difficulty, and license cost.

**Keywords:** deep learning, convolution network, image clasification, efficiency, parallel computing

# Spis treści

# Rozdział 1

# Wprowadzenie

Zainteresowanie sztucznymi sieciami neuronowymi w środowisku naukowców jak i inżynierów wynika z potrzeby budowania bardziej efektywnych i niezawodnych systemów przetwarzania informacji wzorując się na metodach jej przetwarzania w komórkach nerwowych. Fascynacje mózgiem człowieka i jego właściwościami w latach 40-stych dały początek pracom w zakresie syntezy matematycznej modelu pojedynczych komórek nerwowych, a później na tej podstawie struktur bardziej złożonych w formie regularnych sieci. Zespół kierowany przez prof. Ryszarda Tadeusiewicza z AGH w Krakowie prowadził już w latach '80 badania nad m.in. rozpoznawaniem ręcznego pisma, czy sterowaniem ruchem robota, a pierwsza książka ukazała się w 1993r.[2]

Pierwsze prace prof. Kunihiko Fukushimy nad głębokią siecią pojawiły się w 1975 roku. Jednak prawdziwy rozwój sieci głębokich zawdzięczamy profesorowi Yann A. LeCun, który zdefiniował podstawową strukturę i algorytm uczący sieci konwolucyjnej CNN. Aktualnie sieci CNN stanowią podstawową strukturę stosowaną na szeroką skalę w przetwarzaniu sygnałów i obrazów. W międzyczasie powstało wiele odmian sieci będącej modyfikacją struktury podstawowej (R-CNN, AlexNET, GoogLeNet, ResNet, U-Net, YOLO)[9]

Ważnym rozwiązaniem jest sieć YOLO (ang. You Only Look Once), wykonująca jednocześnie funkcje klasyfikatora i systemu regresyjnego, który służy do wykrywania określonych obiektów w obrazie i określaniu ich współrzędnych. Obecnie dostępna jest już wersja 12.

Biblioteki dostarczające implementacje modeli sieci neuronowych powstały dla większości języków programowania ogólnego przeznaczenia. Niektóre z nich wykorzystują procesory graficzne do wysokowydajnych równoległych obliczeń, co powoduje gwałtowny wzrost wydajności i drastyczne obniżenie czasu uczenia sieci. Budowanie własnych modeli sieci jest dziś w zasięgu osób prywatnych, hobbystów, studentów czy małych zespołów badawczych i nie wymaga ogromnych nakładów finansowych.

## 1.1 Cel pracy

Podstawowym celem pracy jest ułatwienie podjęcia decyzji o wyborze języka i środowiska w fazie projektowej dla realizacji aplikacji wykorzystujących głębokie sieci neuronowe CNN.

Celem dydaktycznym jest dogłębne zapoznanie się z tematyką sieci MLP [2], [12] oraz CNN [5] [10] poprzez realizacje i testy własnego rozwiązania zwłaszcza z wykorzystaniem możliwości obliczeniowych karty graficznej.

## 1.2 Układ pracy

Problem obliczeniowy - W części pierwszej opisano stan wiedzy z zakresu działania głębokich sieci neuronowych tj. Perceptronu wielowarstwowego (ang. Multilayer Perceptron, MLP) oraz Konwolucyjnej sieci neuronowej (ang. Convolutional Neural Network, CNN). Zaprezentowano propagację sygnałów przez sieć, propagację wsteczną i oparty na niej proces uczenia sieci.

Problem wydajnościowy - Warunkiem niezbędnym do prowadzenia efektywnych badań nad głębokimi sieciami i dużymi modelami jest zdolność efektywnego wykorzystania systemów o dużych mocach obliczeniowych. W drugiej części opisano metody zwiększania wydajności systemów cyfrowych i zagadnienia przetwarzania równoległego.

W drugiej części przedstawiono języki, opisano wybrane cechy, informacje o wykorzystanych bibliotekach, pokazano fragmenty kodu.

W trzeciej części zastawiono cechy i wyniki pomiarów z podziałem na języki.

Wnioski na temat wyższości jednego rozwiązania nad drugim będą należeć od konkretnych twórców sieci i będą zależały od ich wymagań i możliwości.

## 1.3 Kod źródłowy i dane uczące

Przykłady rozwiązań w Python i Matlab zaczerpnięto z [9] [6] [7] [8], a także z instrukcji i przykładów załączonych do wykorzystanych bibliotek.

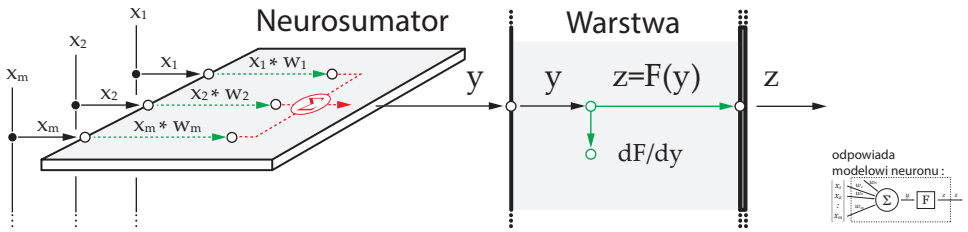
Obrazy pisma odręcznego pochodzą z bazy MNIST (yann.lecun.com), Zdjęcia twarzy wykorzystane w treningu sieci osób pochodzą z internetu - głownie z serwisów google.com oraz filmweb.pl

Pełen kod dostępny na github: https://github.com/piotrHeinzelman/inz/tree/main/MixedProj
W analizie nie brano pod uwagę czasów czytania plików oraz przygotowania danych.

Bibiografia [2] [12] [3] [9] [6] [5] [10] [7] [8] [13] [1] [11] [4]

# Rozdział 2

# Perceptron wielowarstwowy - MLP



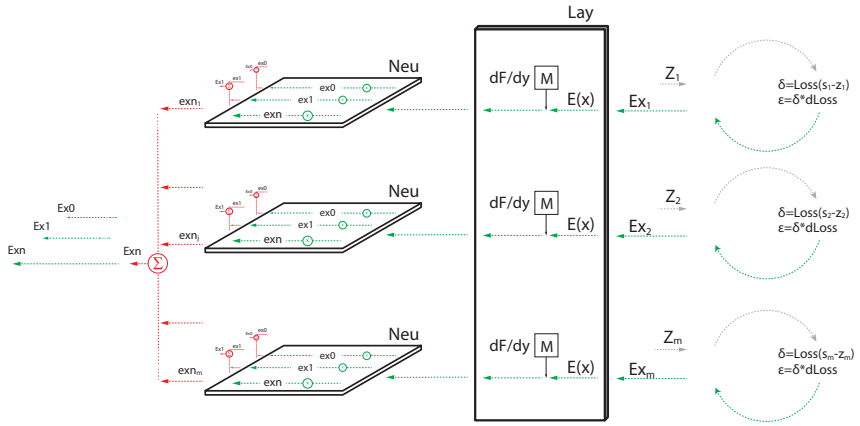

**Rysunek 1.** Model sztucznego neuronu - propagacja sygnału.

W tym rozdziale Autor przedstawi opis modelu Perceptronu wielowarstwowego MLP. Jednego z pierwszych działających modeli sieci neuronowych. Po pierwsze model matematyczny - propagacji sygnału przez pojedynczą warstwę sieci neuronowej. Po drugie odpowiadający modelowi matematycznemu model przesyłania sygnałów, z podziałem na 3 operacje: 1 - mnożenie »xiwi = xi * wi« (równoległe), 2 - sumowanie »y = x1w1 + xiwi + ... + xnwn« ( quasi równoległe), 3 - obliczenia wartości funkcji aktywacji z=F(y) - oraz dodatkowo obliczenie pochodnej funkcji w punkcie z w zależności od rodzaju funkcji aktywacji :

| rodzaj warstwy | funkcja aktywacji $z = F(y)$ | wartość pochodnej $\frac{\partial F}{\partial y}$ |
|---|---|---|
| logistyczna (sigmoid) | $\frac{1}{1+e^{-y}}$ | $(z)(1-z)$ |
| ReLU | jeśli y<0 to z=y, jeśli y>=0 to z=0 | jeśli y<0 to 1, jeśli y>=0 to 0 |
| softmax | $z_i = \frac{e^{x_i}}{\sum e^{x_i}}$ | dla prawidłowej klasy k: $y_k(1-y_k)$, |
| softmax | | dla pozostałych klas: $-y_i * y_k$ |

metody obliczenia błędu,

a także realizację procesu uczenia przez zastosowanie metody wstecznej propagacji błędu.

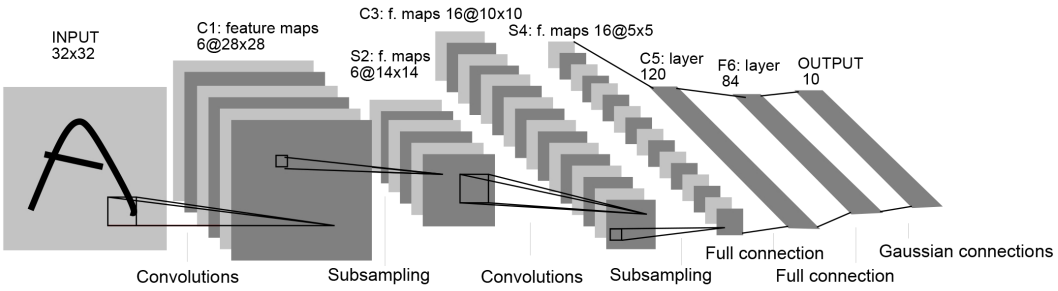



**Rysunek 2.** propagacja wstecz - uczenie sieci. (Operacje niezależne (mnożenie) oznaczone kolorem zielonym i operacje wymagające synchronizacji (dodawanie) oznaczone kolorem czerwonym)

# Rozdział 3

# Sieć splotowa CNN

W drugim rozdziale Autor opisze działanie warstwy splotowej CNN, oraz warstw pomocniczych - warstwy redukującej rozmiar, warstwy filtrującej wartości - pozostawiającej sygnały dodatnie, i odrzucającej ujemne. Warstwy filtrującej wartości maksymalne lub wyliczającą wartość średnią sygnału wartości. Przedstawiona będzie także metodyka obliczenia poprawek w filtrze podczas w procesie uczenia, a także propagacji sygnału błędu wstecz przez warstwę splotową i warstwy pomocnicze. [13]




**Rysunek 3.** Architektura LaNet-5 [13]

# Rozdział 4

# Przetwarzanie równoległe

W tym rozdziale Autor opisze sposoby zwiększania wydajności procesorów dla wykonywania obliczeń wektorów o 3, 4, 5 i więcej niezależnych wymiarach w procesorach od 80386 - czyli od rejestrów xmm i rozkazów MMX przez AVX do AVX256 i rejestrów ymm i zmm. A także możliwości równoległego mnożenia i quasi równoległego dodawania przy zastosowaniu możliwości obliczeniowej kart graficznych. A także ich wpływ na wydajność realizacji modeli sieci neuronowych i głębokich sieci neuronowych.

# Rozdział 5

# Przedstawienie języków

W tym rozdziale autor opisze własności języków które będą brane pod uwagę przy porównaniu. Będą to: 1) Popularność języka, 2) Dostępność bibliotek, 3) Wygoda instalacji, 4) Wykorzystanie GPU, 5) Koszt licencji, 6) Łatwość użytkowania (próg wejścia)

Następnie każdy język będzie przedstawiony ogólnie, oraz będą jego własności.

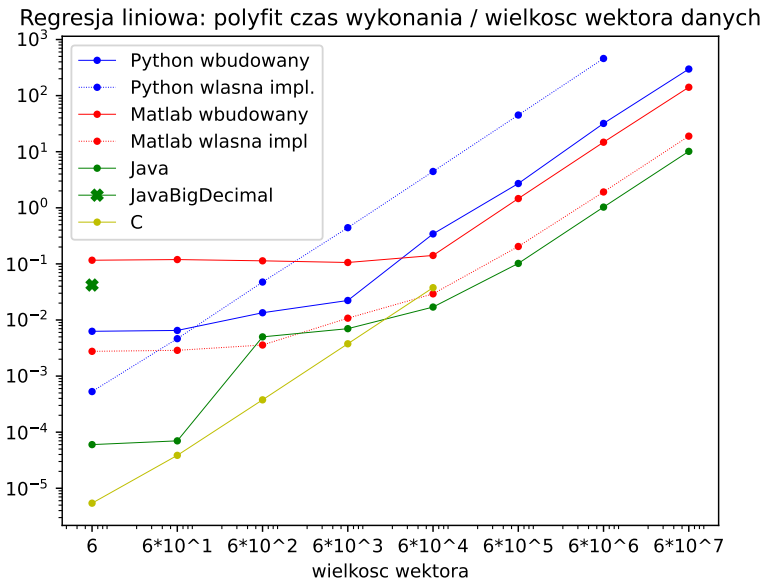
Przy opisie bibliotek i języków w haśle Łatwość użytkowania pojawią się fragmenty kodu - prezentujące wykorzystanie poszczególnych bibliotek. Języki to C++, Python, Matlab( to nie język - raczej środowisko ). W porównaniu szybkości / wydajności pojawi się także język Java - w pierwszych 3 zastosowaniach - jednak bez wykorzystania GPU.

PS.raczej nie uda mi się zdążyć z przygotowaniem kodu GPU + Java :-(

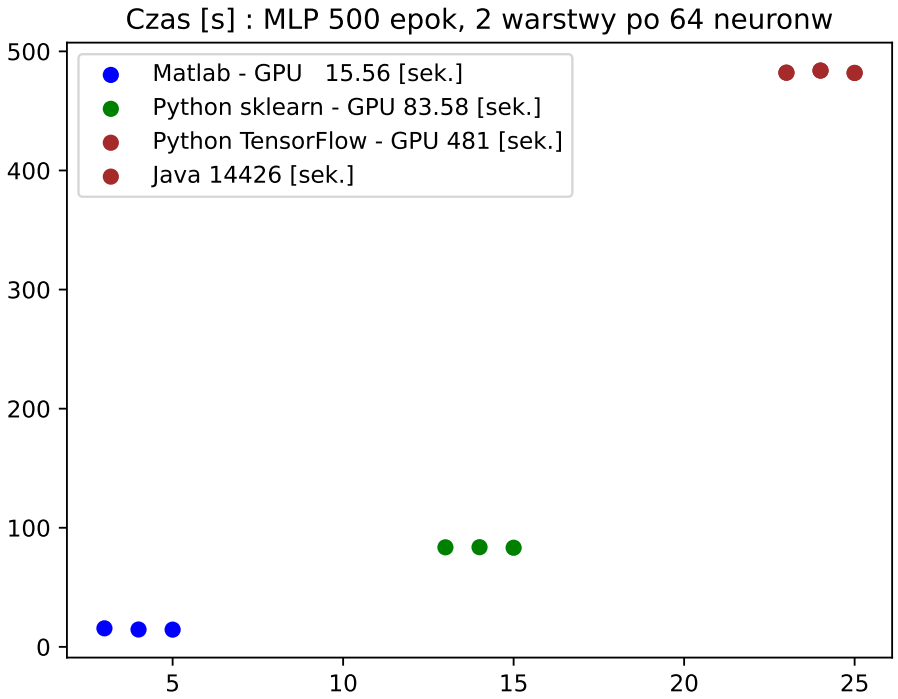
https://stackoverflow.com/questions/64895480/simplest-way-to-use-gpu-in-java

a także zestawienie czasów wykonania poszczególnych zadań :

1) Jednowymiarowa regresja liniowa (ogólna sprawność języka)



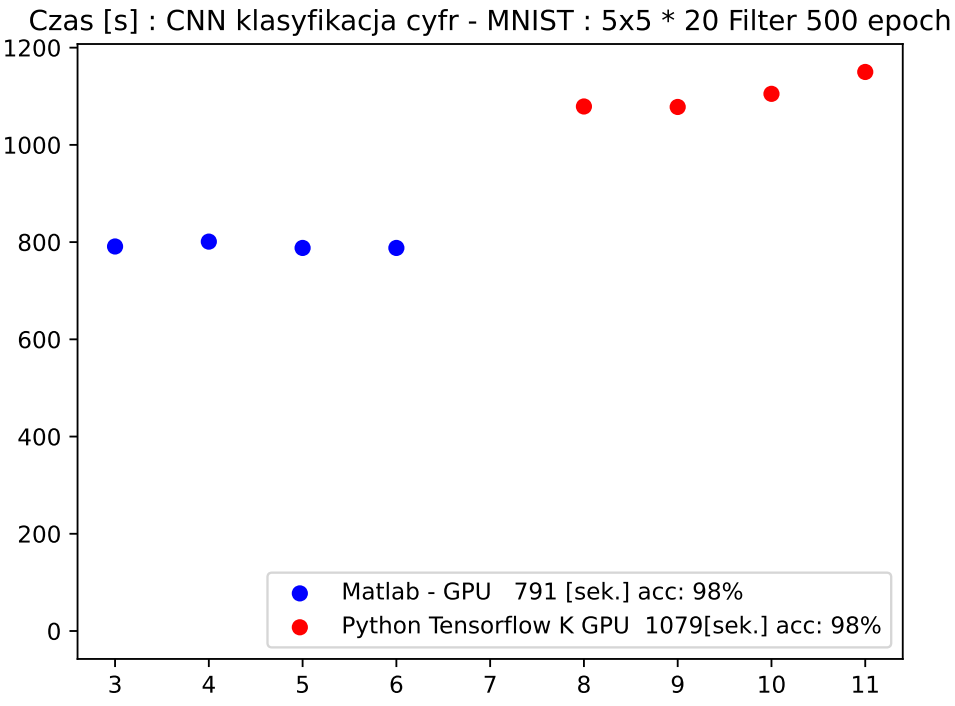
2) Klasyfikowanie pisma odręcznego MLP




3) CNN klasyfikacja cyfr - MNIST

4) CNN klasyfikacja cyfr - Klasytfikacja twarzy

5) Detekcja i segmentacji obiektów sieci głębokiej CNN z wykorzytsaniem b) Python tensorflow.keras c) C++ libtorch


Czas [s] : CNN klasyfikacja cyfr - MNIST : 5x5 * 20 Filter 500 epoch
1200
1000
800
600
400
200
0
3
4
5
6
7
8
9
10
11
Matlab - GPU 791 [sek.] acc: 98%
Python Tensorflow K GPU 1079[sek.] acc: 98%

# Rozdział 6

# Podsumowanie

W podsumowaniu pojawi się porównanie języków w formie wykresu w zależności od 3 wektorów : wydajności, wygody, ceny. A także wnioski zależne od czasów wykonania programów.

```
1
2
3  @book{korbicz1994,
4      author    = {J zef Korbicz, Andrzej Obuchowicz, Dariusz Uci ski},
5      title     = {Sztuczne sieci neuronowe: podstawy i zastosowania},
6      publisher = {Akademicka Oficyna wydawnicza PLJ},
7      year      = {1994},
8      ISBN      = {83-7101-197-0},
9  }
10
11
12
13 @book{wit1986,
14     author    = {R. Wit},
15     title     = {Metody programowania nieliniowego},
16     publisher = {Warszawa: WNT},
17     year      = {1986},
18 }
19
20
21 @misc{WKasprzak,
22   author    = {W Ćodzimierz Kasprzak},
23   title     = {Metody sztucznej inteligencji C6.pdf},
24   publisher = {wyk Ćad Okno PW},
25   year = {2024},
26   note    = {materia Ć dydaktyczny},
27 }
28
29 @book{ossowski2023,
30     author    = {Stanis Ćaw Osowski,Robert Szmur Ćo},
```

```
    title     = {Matematyczne modele uczenia maszynowego w j Źzykach
   MATLAB i PYTHON},
    publisher = {Oficyna Wydawnicza Politechniki Warszawskiej},
    year      = {2023},
    ISBN      = {978-83-8156-597-4},
}


@book{ossowski2020,
    author    = {Stanis Ćaw Osowski},
    title     = {Sieci neuronowe do przetwarzania informacji},
    publisher = {Oficyna Wydawnicza Politechniki Warszawskiej},
    year      = {2020},
    ISBN      = {978-83-7814-923-1},
}


@book{MMuraszkiewiczRobertNowak,
    author    = {praca zbiorowa pod redakcj   Mieczys Ćawa
   Muraszkiewicza i Roberta Nowaka},
    title     = {Sztuczna inteligencja dla in ynier w},
    publisher = {Oficyna Wydawnicza Politechniki Warszawskiej},
    year      = {2023},
    ISBN      = {978-83-8156-584-4 },
}




@book{russell2023,
    author    = {Stuart Russell, Peter Norvig},
    title     = {Artificial Intelligence: A Modern Aproach, 4th Edition
   },
    editor    = {T Ćumaczenie: Andrzej Gra y ski },
    publisher = {Pearson Education, Inc., Polish language by Helion S.A.
    2023 },
    year      = {2023},
    ISBN      = {978-83-283-7773-8},
}




@book{guido2021,
    author    = {Sarah Guido, Andreas M{\"u}ller},
    title     = {Machine learning, Python i data science},
```

```
    orgintitle     = {Introduction to Machine Learning with Python: A
    Guide for Data Scientists},
    publisher = {Helion S.A.},
    year      = {2021, 2023},
    ISBN      = {978-83-8322-751-1},
}

@book{raschka2021,
    author    = {Sebastian Raschka, Vahid Mirjalili},
    title     = {Machine learning, Python i data science},
    orgintitle     = {Python Machine Learning: Machine Learning and Deep
    Learning with Python, scikit-learn, and TensorFlow 2, 3rd Edition},
    publisher = {Helion S.A.},
    year      = {2021},
    ISBN      = {978-83-283-7001-2},
}

@misc{YannCnn,
    author  = {Y. Bengio - Universit  de Montr al, Yann Lecun - New
    York University},
    title   = {Convolutional Networks for Images, Speech, and Time-
    Series},
    howpublished    = {\href{https://www.researchgate.net/publication
    /2453996_Convolutional_Networks_for_Images_Speech_and_Time-Series}},
    year      = {1997},
}


@book{asmx64,
    author    = {Jo Van Hoey},
    editor    = {T Ćumaczenie: Grzegorz Werner },
    title     = {Programowanie w asemblerze x64},
    orgintitle     = {Beginning x64 Assembly Programming: From novice to
    AVX Professional},
    publisher = {Helion S.A.},
    year      = {2024},
    orginyear = {2019},
    ISBN      = {978-83-289-0109-4},
}

@book{tad93,
    author = {Ryszard Tadeusiewicz},
    title = {Sieci neuronowe},
    publisher = {Akademicka Oficyna Wydawnicza P/M},
    year = {1993},
    ISBN = {83-85769-03-X},
```

```
113 }
114
115 @book{kos18,
116     author = {Robert A. Kosi ski},
117     title = {Sztuczne sieci neuronowe, Dynamika nieliniowa i chaos},
118     publisher = {AWydawnictwo WNT},
119     year = {2018},
120     ISBN = {978-83-01-19732-2},
121 }
```

# Bibliografia


[1] Hoey, J. V., *Programowanie w asemblerze x64*, Werner, T. G., red. Helion S.A., 2024, ISBN: 978-83-289-0109-4.

[2] Józef Korbicz Andrzej Obuchowicz, D. U., *Sztuczne sieci neuronowe: podstawy i zastosowania*. Akademicka Oficyna wydawnicza PLJ, 1994, ISBN: 83-7101-197-0.

[3] Kasprzak, W., *Metody sztucznej inteligencji C6.pdf*, materiał dydaktyczny, 2024.

[4] Kosiński, R. A., *Sztuczne sieci neuronowe, Dynamika nieliniowa i chaos*. AWydawnictwo WNT, 2018, ISBN: 978-83-01-19732-2.

[5] Mieczysława Muraszkiewicza i Roberta Nowaka, praca zbiorowa pod redakcją, *Sztuczna inteligencja dla inżynierów*. Oficyna Wydawnicza Politechniki Warszawskiej, 2023, ISBN: 978-83-8156-584-4.

[6] Osowski, S., *Sieci neuronowe do przetwarzania informacji*. Oficyna Wydawnicza Politechniki Warszawskiej, 2020, ISBN: 978-83-7814-923-1.

[7] Sarah Guido, A. M., *Machine learning, Python i data science*. Helion S.A., 2021, 2023, ISBN: 978-83-8322-751-1.

[8] Sebastian Raschka, V. M., *Machine learning, Python i data science*. Helion S.A., 2021, ISBN: 978-83-283-7001-2.

[9] Stanisław Osowski, R. S., *Matematyczne modele uczenia maszynowego w językach MATLAB i PYTHON*. Oficyna Wydawnicza Politechniki Warszawskiej, 2023, ISBN: 978-83-8156-597-4.

[10] Stuart Russell, P. N., *Artificial Intelligence: A Modern Aproach, 4th Edition*, Grażyński, T. A., red. Pearson Education, Inc., Polish language by Helion S.A. 2023, 2023, ISBN: 978-83-283-7773-8.

[11] Tadeusiewicz, R., *Sieci neuronowe*. Akademicka Oficyna Wydawnicza P/M, 1993, ISBN: 83-85769-03-X.

[12] Wit, R., *Metody programowania nieliniowego*. Warszawa: WNT, 1986.

[13] Y. Bengio - Université de Montréal, Y. L. .-. N. Y. U., *Convolutional Networks for Images, Speech, and Time-Series*, 1997.

# Spis rysunków