Kraków Applied Physics and Computer Science

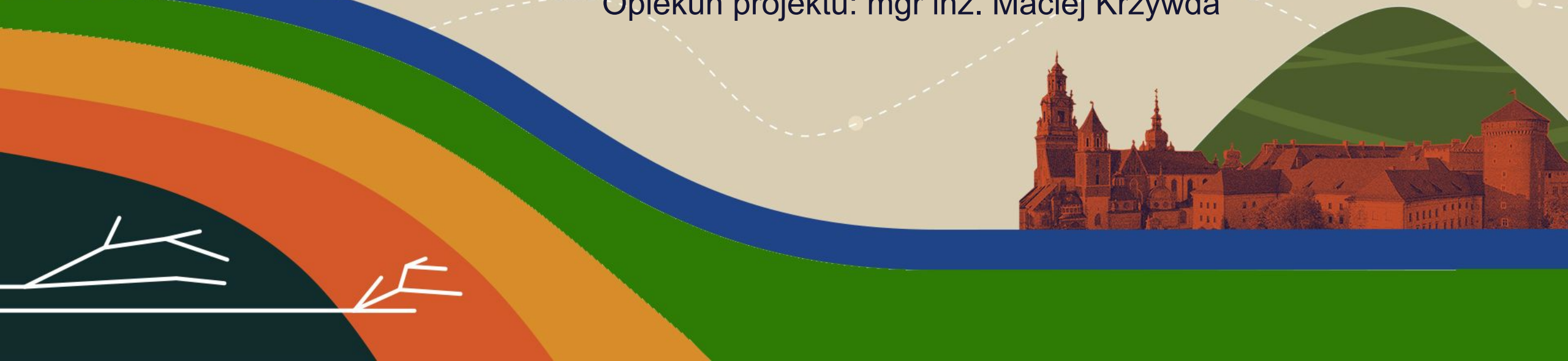
Summer School'22

29.07.2022

Wykorzystanie algorytmów genetycznych do opytmalizacji procesu trenowania grafowych sieci neuronowych

Prezenter: Paweł Rybczyński

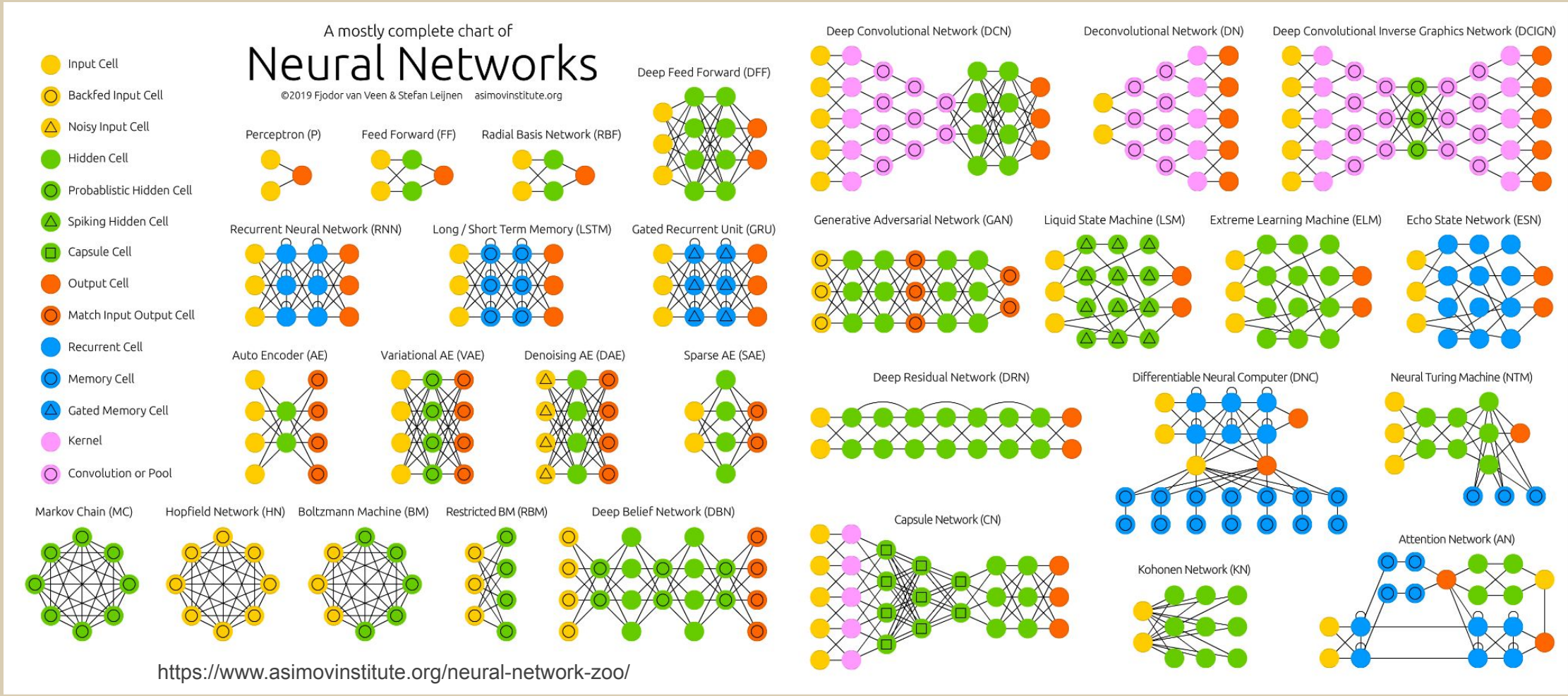
Opiekun projektu: mgr inż. Maciej Krzywda



Plan prezentacji

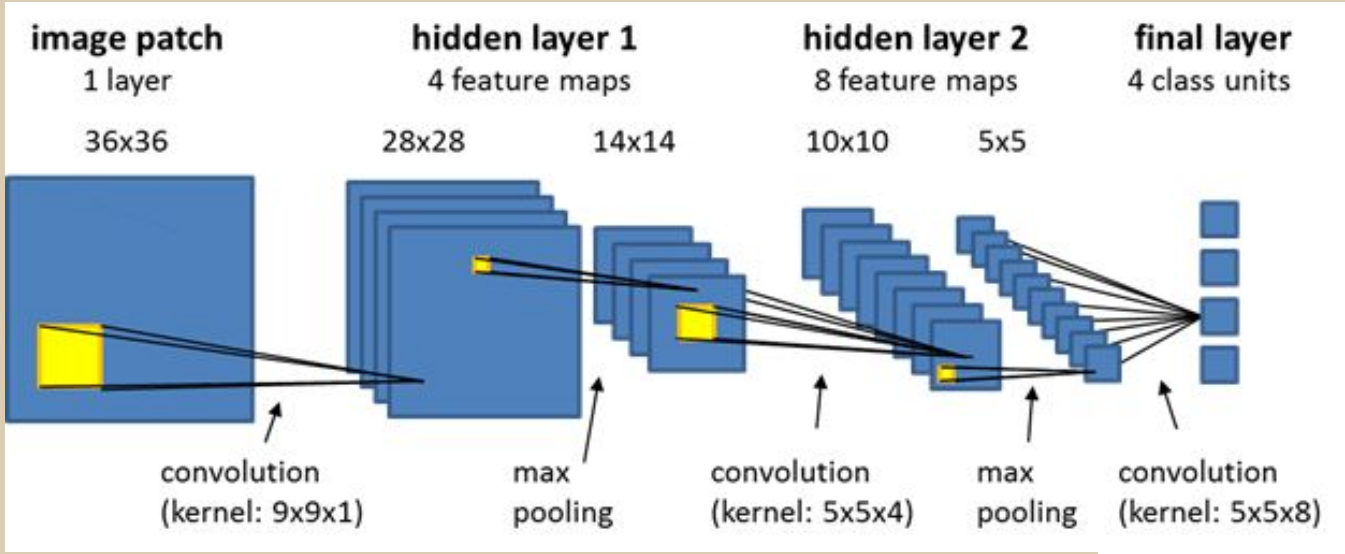
1. Sieci neuronowe.
2. CNN.
3. Budowa sieci GCN.
4. Optymalizacja.
5. Algorytmy genetyczne.
6. Omówienie wyników optymalizacji GCN za pomocą algorytmu genetycznego.
7. Porównanie wyników.

Sieci neuronowe

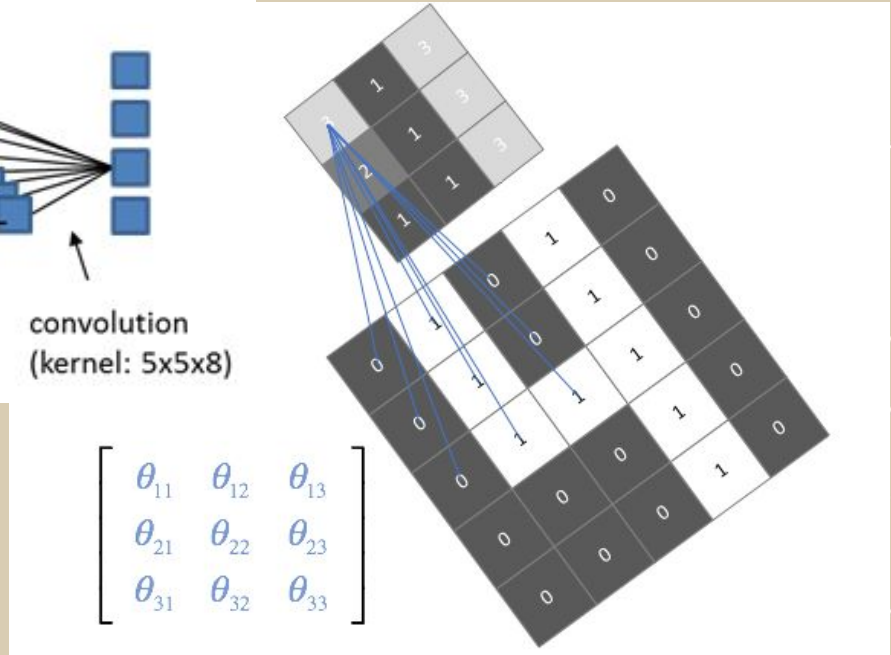


https://www.asimovinstitute.org/neural-network-zoo/

Konwolucyjne sieci neuronowe (CNN)



https://docs.ecognition.com/v10.0.2/eCognition_documentation/User%20Guide%20Developer/8%20Classification%20-%20Deep%20Learning.htm



https://www.jeremyjordan.me/convolutional-neural-networks/

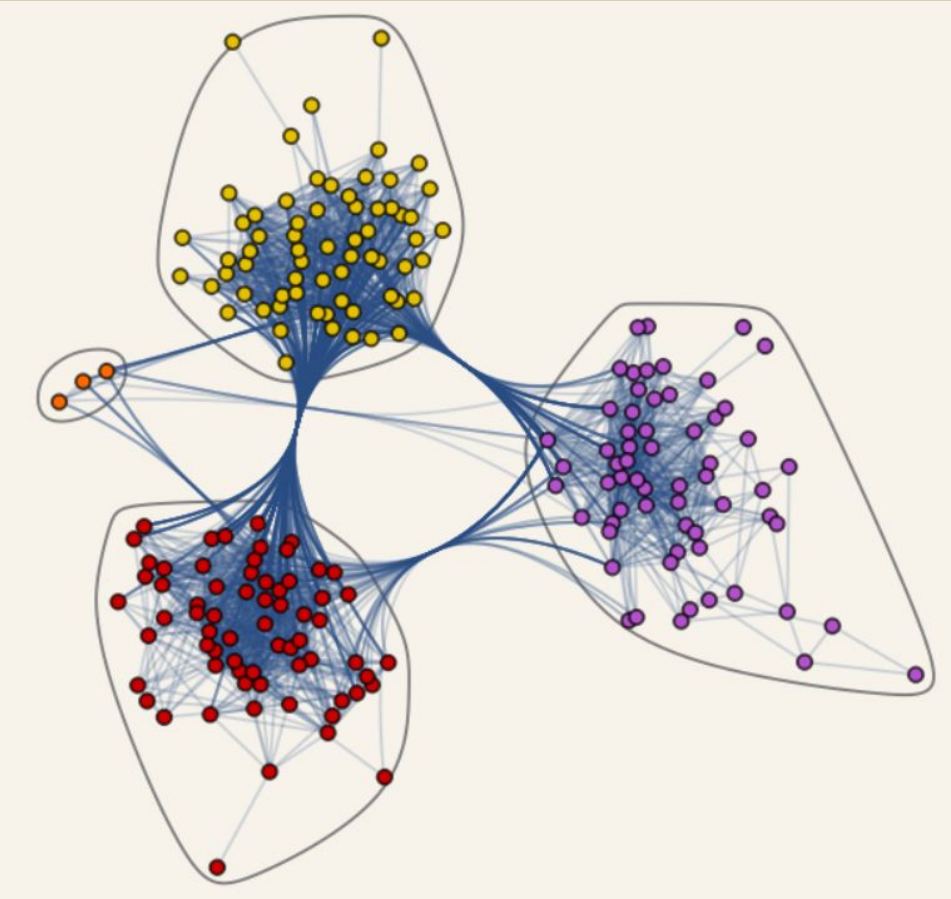
Idea konwolucji - agregacja informacji z pikseli poprzez zadany filtr (macierz).

Grafowe sieci konwolucyjne (GCN)

Budowa grafu:

- węzły (nodes),
- wektor cech (feature vector),
- krawędzie (egdes),
- etykiety (labels).

Zadaniem sieci neuronowej może być identyfikacja poszczególnych węzłów i grupowanie ich w klasy, lub interpretacja grafu jako całości.



https://www.datacamp.com/tutorial/comprehensive-introduction-graph-neural-networks-gnns-tutorial

Dataset - Cora

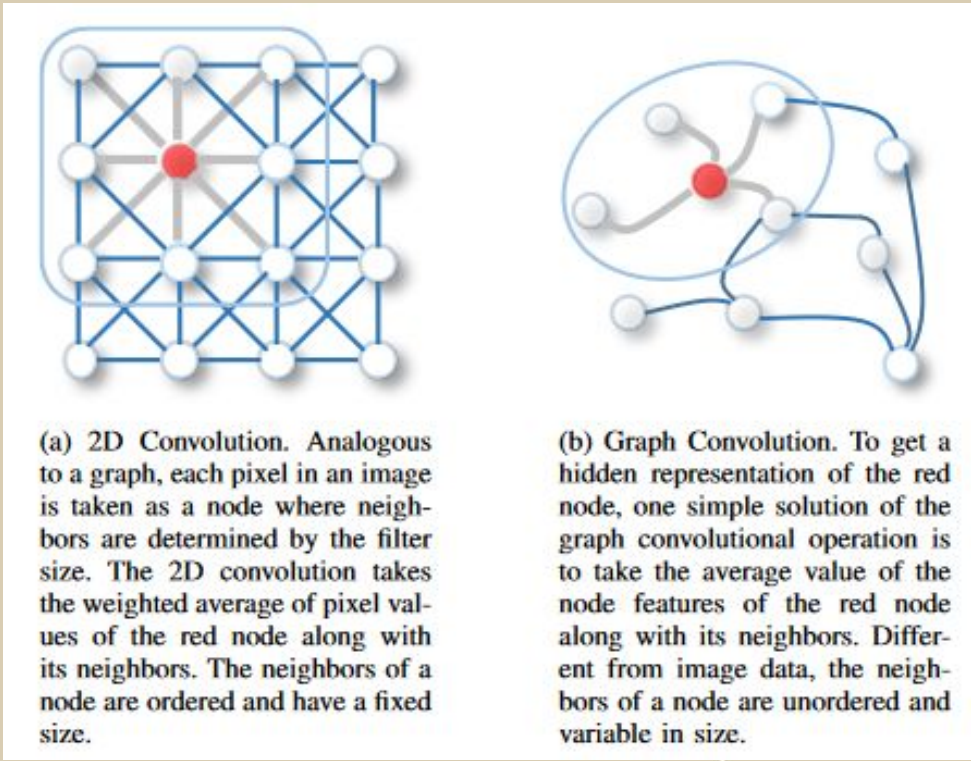
W projekcie wykorzystano zbiór danych Cora. Jest to zbiór publikacji reprezentowanych przez odpowiadające im “worki słów” oraz połączonych między sobą siecią cytowań. Każdy dokument to jeden węzeł, a worek słów to wektor cech danego węzła.

Zadaniem sieci neuronowej jest pogrupowanie 2708 węzłów, każdy o 1433 cechach, na 7 klas. Każdy z węzłów ma średnio 3.9 krawędzi, czyli sąsiadów.

Yang, Z., Cohen, W.W., & Salakhutdinov, R. (2016). Revisiting Semi-Supervised Learning with Graph Embeddings. ArXiv, abs/1603.08861.

GCN vs CNN

CNN używa się do przetwarzania ustrukturyzowanych danych takich jak obrazy. Jednak przy bardziej złożonych danych, reprezentowanych w postaci grafu, nie można zastosować tego podejścia. Pojawia się potrzeba uogólnienia pojęcia konwolucji tak aby mogła działać w domenie grafów, co nie jest możliwe przy użyciu zwykłego przestrzennego filtra.



(a) 2D Convolution. Analogous to a graph, each pixel in an image is taken as a node where neighbors are determined by the filter size. The 2D convolution takes the weighted average of pixel values of the red node along with its neighbors. The neighbors of a node are ordered and have a fixed size.

(b) Graph Convolution. To get a hidden representation of the red node, one simple solution of the graph convolutional operation is to take the average value of the node features of the red node along with its neighbors. Different from image data, the neighbors of a node are unordered and variable in size.

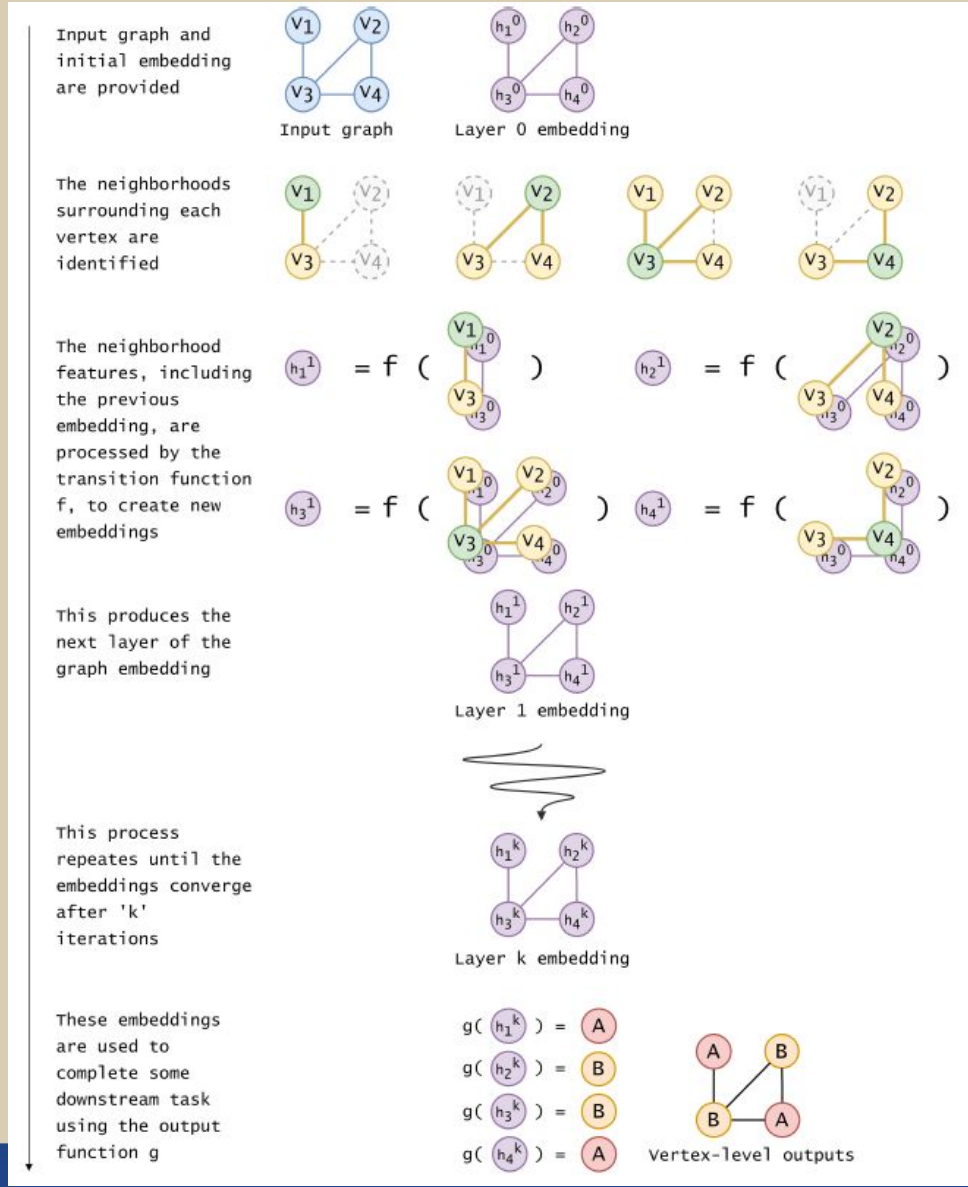
Wu, Zonghan et al. “A Comprehensive Survey on Graph Neural Networks.” *IEEE Transactions on Neural Networks and Learning Systems* 32 (2019): 4-24.

Grafowe sieci konwolucyjne

Konwolucja dla grafów:

-każda ukryta warstwa agreguje informacje z najbliższego sąsiedztwa,

-funkcja przejścia może być użyta dla dowolnego węzła, czyli nie zależy od liczby sąsiednich węzłów.



Ward, Isaac Ronald et al. “A Practical Tutorial on Graph Neural Networks.” *ACM Computing Surveys (CSUR)* (2022): n. pag.

Grafowe sieci konwolucyjne

Najłatwiejszym i najbardziej efektywnym sposobem implementacji GCN jest skorzystanie z macierzowej reprezentacji problemu:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right)$$

gdzie $H^{(l)}$ jest l-tą ukrytą warstwą, A macierzą przylegania (adjacency matrix), D macierzą stopnia (degree matrix), $W^{(l)}$ l-tą macierzą trenowalnych wag, a σ funkcją aktywacji, np. ReLU().

A może być binarna lub ważona i opisuje połączenia między węzłami, a D odpowiada za normalizację.

Optymalizator Adam (Adaptive Moment Estimation)

Adam to metoda podobna do stochastycznego gradientu z charakterystycznymi cechami:

- wykładniczo zanikająca średnia kwadratów poprzednich gradientów - adaptacyjne zmiany wag zależne od częstości występowania cech, szybkość uczenia się dla wag związanych z często pojawiającymi się cechami będzie mniejsza,
- wykładniczo zanikająca średnia poprzednich gradientów - analogia do pędu toczącej się kuli.

Jej hiperparametry są łatwe w interpretacji, learning rate to ułamek gradientu brany do poprawy parametrów, a decay rate odpowiada za wielkość poprawek do learning rate.

https://www.kdnuggets.com/2020/12/optimization-algorithms-neural-networks.html

Optymalizacja

Wydajność i czas uczenia się danej sieci neuronowej może się zmieniać w zależności od przyjętych parametrów takich jak liczba ukrytych cech, liczba epok, wielkość partii danych (batch size), szybkość uczenia się sieci (learning rate), czy weight decay - modyfikacje learning rate.

W celu stworzenia wydajnej sieci neuronowej konieczne jest znalezienie optymalnych wartości parametrów. Jednak sam proces optymalizacji może być złożony i zajmować dużo czasu i charakteryzować się różną wydajnością końcową sieci.

Optymalizacja hiperparametrów

Do zagadnienia optymalizacji można podejść na wiele sposobów. Najczęściej spotykane podejścia to:

- poszukiwanie siatkowe (grid search),
- poszukiwanie losowe (random search),
- Bayesowskie,
- gradientowe,
- genetyczne,

Dlaczego optymalizacja genetyczna?

Metody takie grid search czy Bayesowskie chociaż są wydajniejsze niż metoda random search, to wymagają jednoczesnego fitowania i optymalizacji złożonych i wielowymiarowych funkcji. W efekcie mogą pojawiać się bardzo duże szumy w otrzymywanych wynikach.

Z tego powodu poszukuje się alternatywnych metod optymalizacji, które zapewniłby, jeśli nie lepszą dokładność, to szybszy czas działania. Dobór odpowiedniej metody, może znacząco skrócić czas optymalizacji.

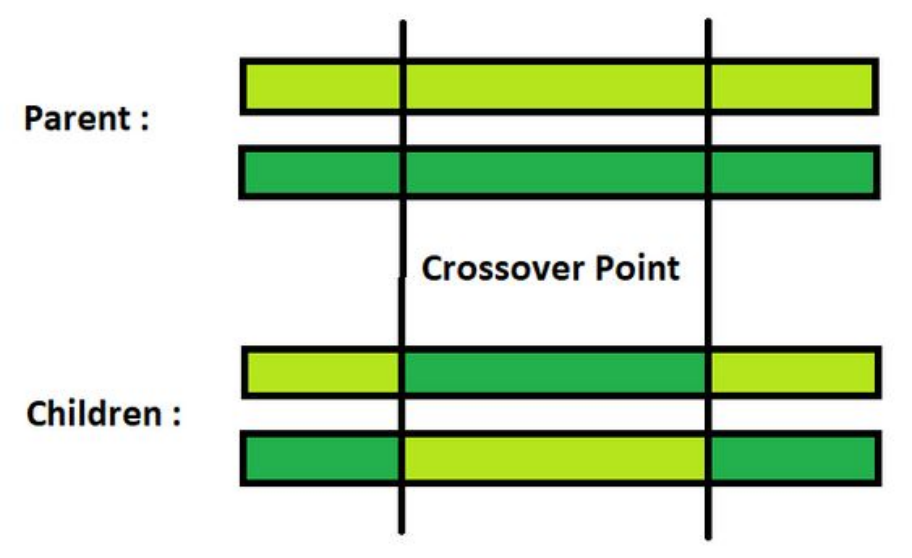
Li, Lisha, et al. "Hyperband: A novel bandit-based approach to hyperparameter optimization." *The Journal of Machine Learning Research* 18.1 (2017): 6765-6816.

Algorytmy genetyczne

Funkcjonalności budujące algorytm genetyczny:

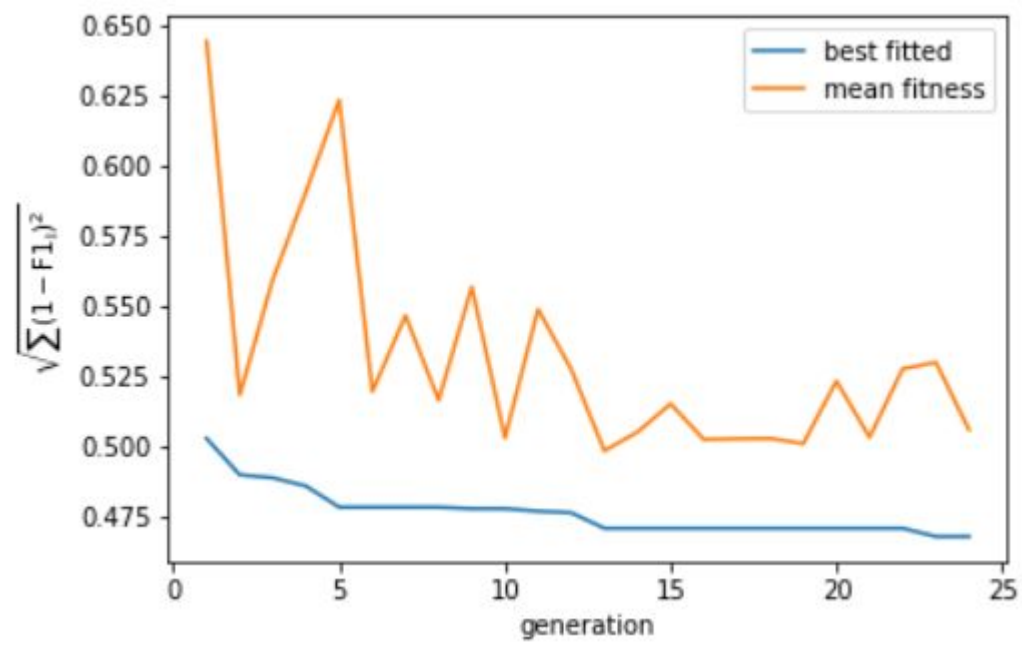
- krzyżowanie,
- mutacja,
- funkcja przystosowania.

W przypadku trenowania sieci neuronowych jako miarę przystosowania można przyjąć między innymi dokładność (accuracy) lub F1 score.



https://www.geeksforgeeks.org/crossover-in-genetic-algorithm/

Funkcja przystosowania



Klasa	początkowa wartość F1	końcowa wartość F1
1	71.86%	74.13%
2	84.21%	84.82%
3	90.59%	90.72%
4	83.96%	84.55%
5	83.33%	83.97%
6	82.98%	83.16%
7	76.06%	79.43%

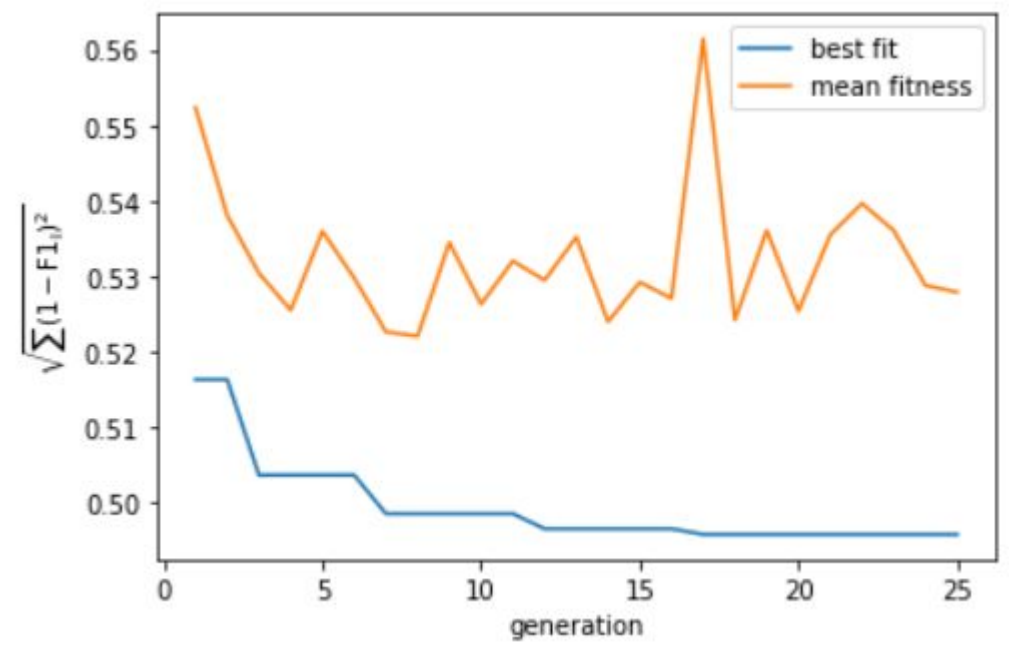
$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

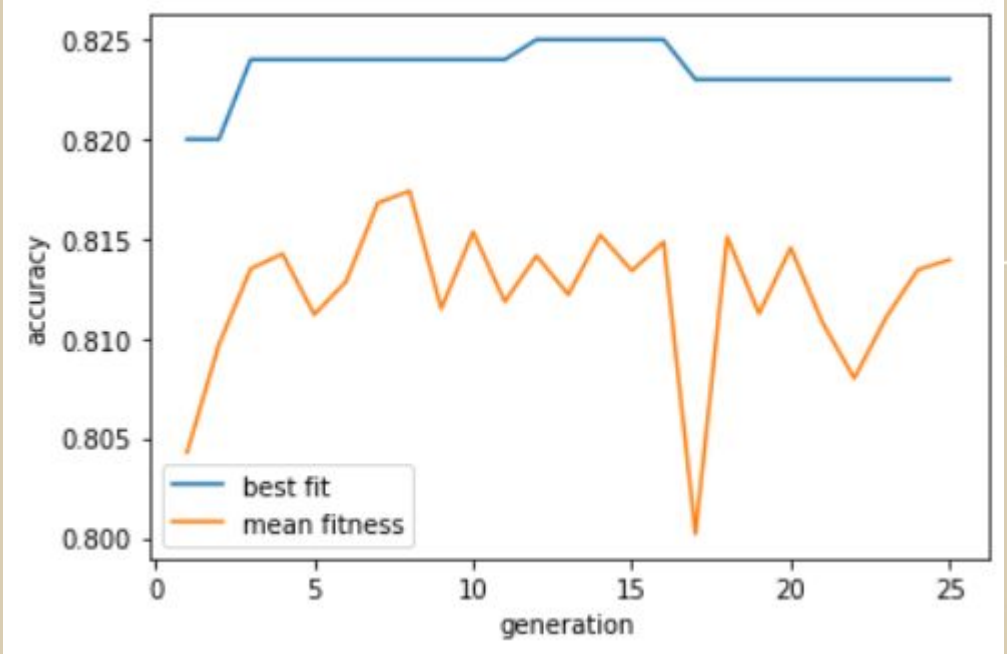
$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

Funkcja przystosowania





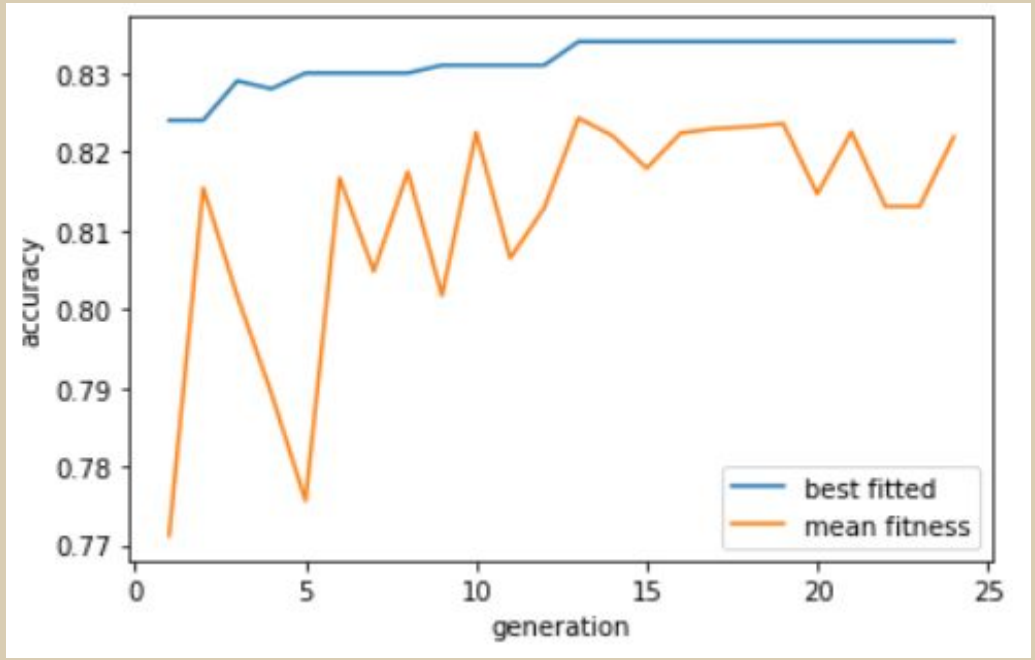
Co wybrać jako miarę przystosowania?...

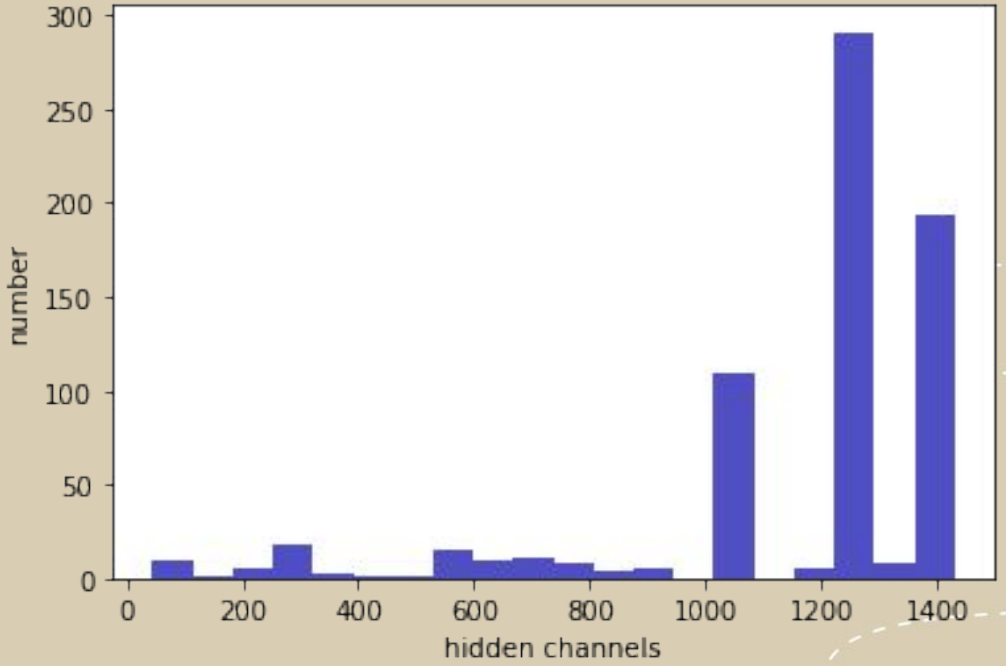
Funkcja przystosowania

W badanym przypadku optymalizacja względem F1 skupia się na poprawie wyniku F1 dla każdej z klas, natomiast optymalizacja względem dokładności poprawia ogólną dokładność, kosztem niewielkiej poprawy w wybranych klasach.

	optymalizacja dokładności		optymalizacja F1	
Klasa	początkowa wartość F1	końcowa wartość F1	początkowa wartość F1	końcowa wartość F1
1	71.79%	74.39%	71.86%	74.13%
2	84.38%	84.82%	84.21%	84.82%
3	89.51%	91.35%	90.59%	90.72%
4	82.57%	85.28%	83.96%	84.55%
5	82.39%	83.60%	83.33%	83.97%
6	81.44%	83.16%	82.98%	83.16%
7	76.71%	76.92%	76.06%	79.43%
dokładność	81.70%	83.50%	82.40%	83.40%

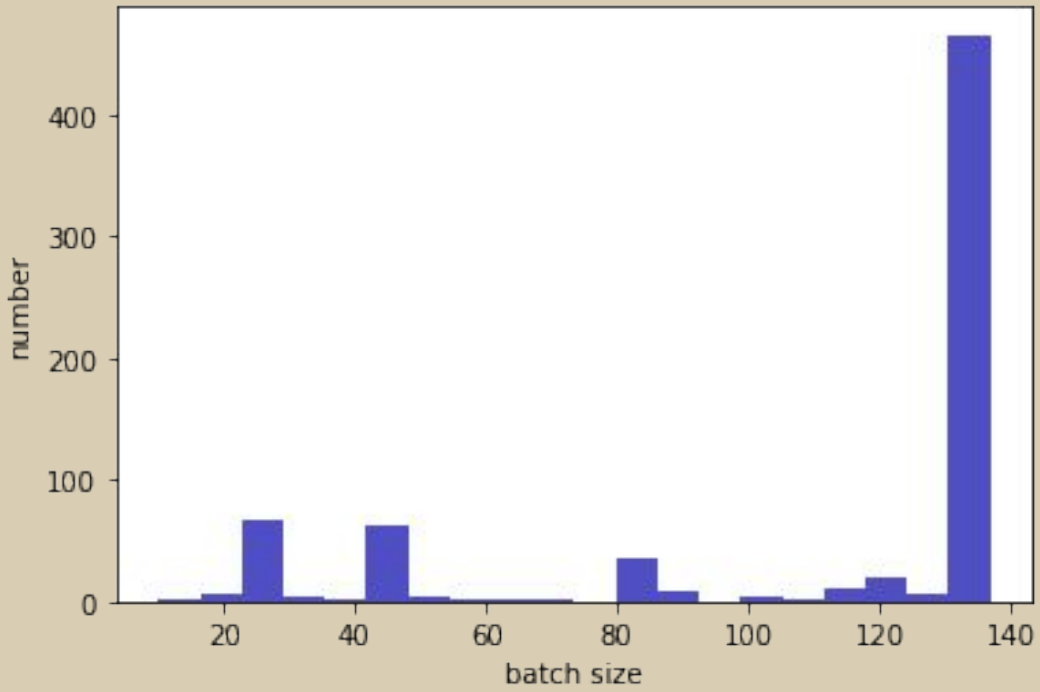
Mutacje

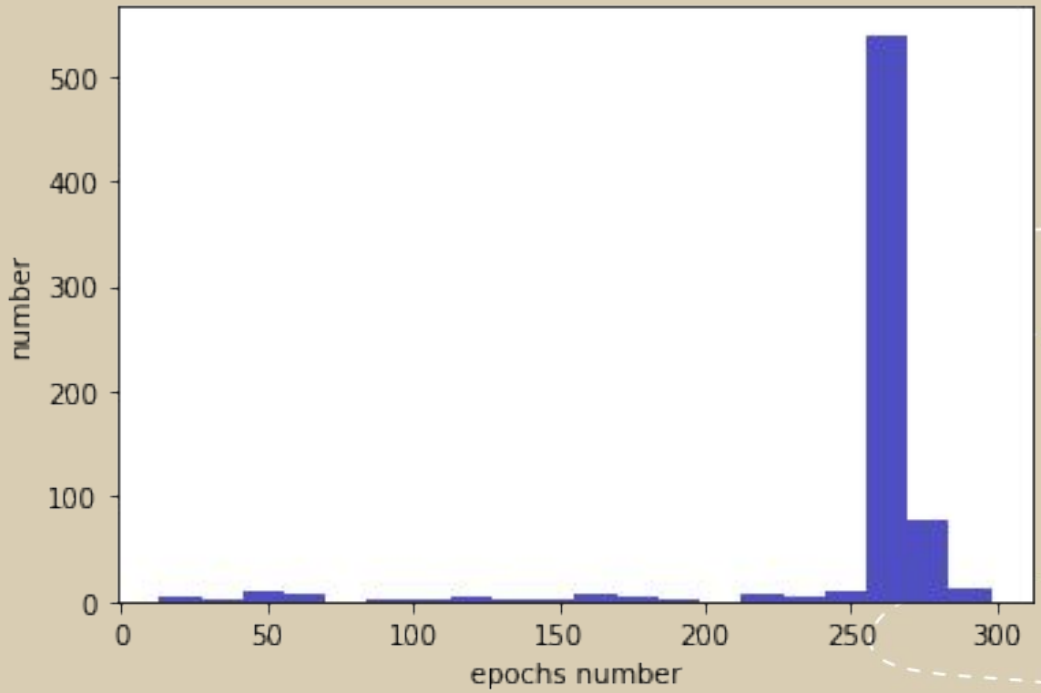




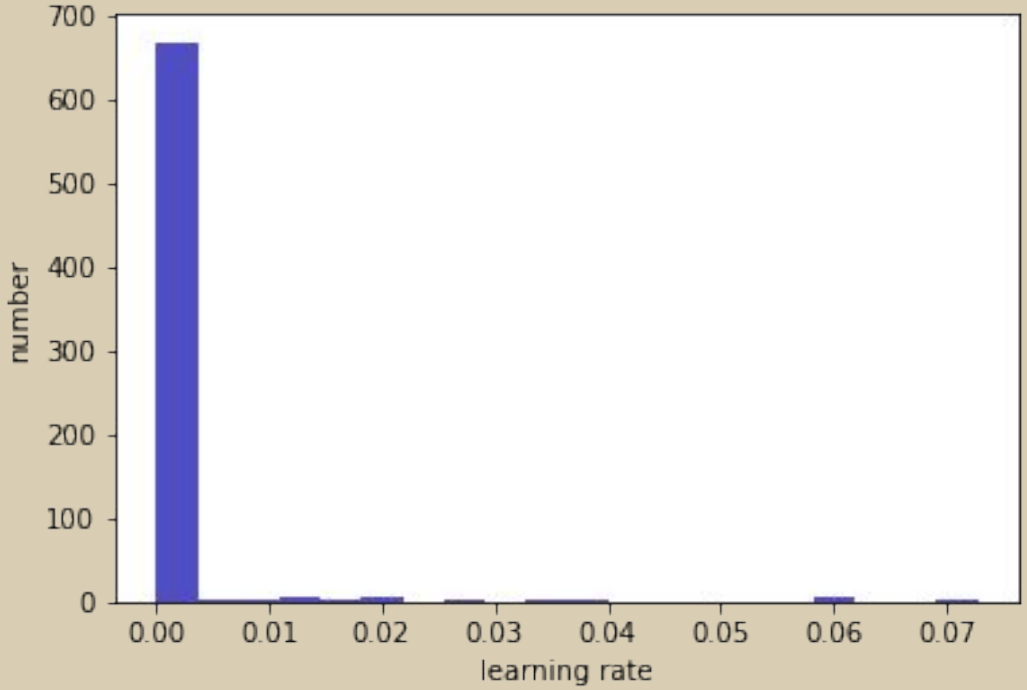
Dzięki mutacji możliwe jest wprowadzenie nowych genów do populacji. Pozwala to uniknąć ujednolicenia puli genów.

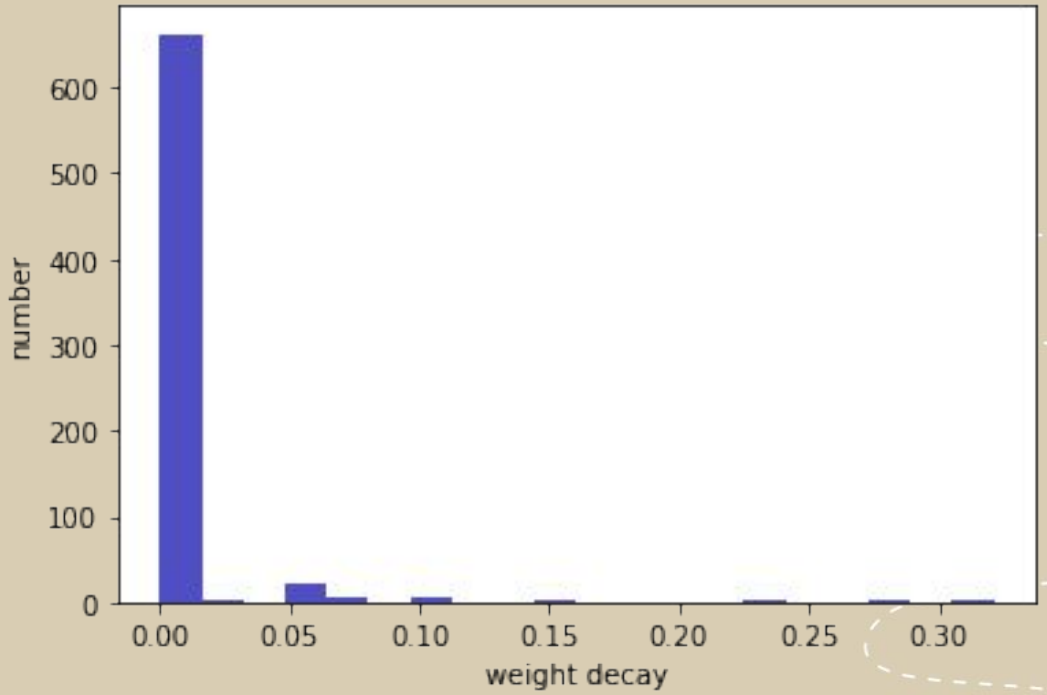
Mutacje


number
400
300
200
100
0
20
40
60
80
100
120
140
batch size


number
500
400
300
200
100
0
0
50
100
150
200
250
300
epochs number

Mutacje


700
600
500
400
300
200
100
0
number
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
learning rate


600
500
400
300
200
100
0
number
0.00
0.05
0.10
0.15
0.20
0.25
0.30
weight decay

Wyniki optymalizacji

Dla 20 pokoleń czas działania algorytmu wynosi około godziny i osiąga dokładność około 83.5%. Największy wpływ na ten czas ma liczba epok przyjęta dla osobników w danej populacji.

Wybór parametrów nie jest jednoznaczny, zatem można dokonywać prób modyfikacji algorytmu, tak aby preferowane były zestawy genów z niewielką liczbą epok.

Wyniki optymalizacji

W zestawieniu 66 publikacji, w których analizowany był ten sam zestaw danych zdecydowana większość otrzymywanych wyników dokładności mieściła się w przedziale 80-90%. Zatem otrzymany wynik 83.5% również mieści się w tym przedziale.

https://paperswithcode.com/sota/node-classification-on-cora

Method	Citeseer	Cora	Pubmed	NELL
ManiReg [3]	60.1	59.5	70.7	21.8
SemiEmb [28]	59.6	59.0	71.1	26.7
LP [32]	45.3	68.0	63.0	26.5
DeepWalk [22]	43.2	67.2	65.3	58.1
ICA [18]	69.1	75.1	73.9	23.1
Planetoid* [29]	64.7 (26s)	75.7 (13s)	77.2 (25s)	61.9 (185s)
GCN (this paper)	**70.3** (7s)	**81.5** (4s)	**79.0** (38s)	**66.0** (48s)
GCN (rand. splits)	67.9 ± 0.5	80.1 ± 0.5	78.9 ± 0.7	58.4 ± 1.7

Kipf, T. N., & Welling, M. (2016). *Semi-Supervised Classification with Graph Convolutional Networks*. doi:10.48550/ARXIV.1609.02907

Dziękuję za uwagę! :)

Google Colab Notebook:

https://colab.research.google.com/drive/1AlG9QxWqUOipyg7CQts-Hn4OOXXbcL7z?usp=sharing

Prezenter: Paweł Rybczyński

Opiekun projektu: mgr inż. Maciej Krzywda