Applied Physics and Computer Science Summer School '22

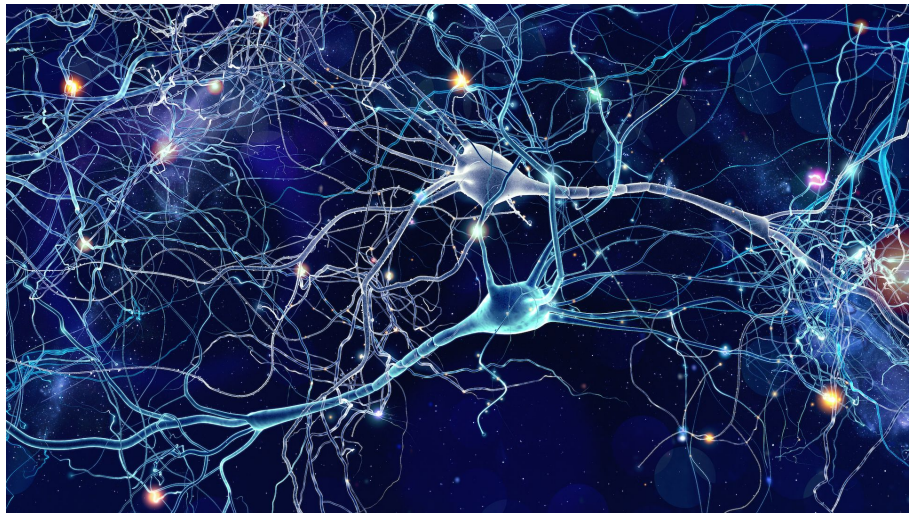
29 lipca 2022

Symulator impulsowych sieci neuronowych EDHA

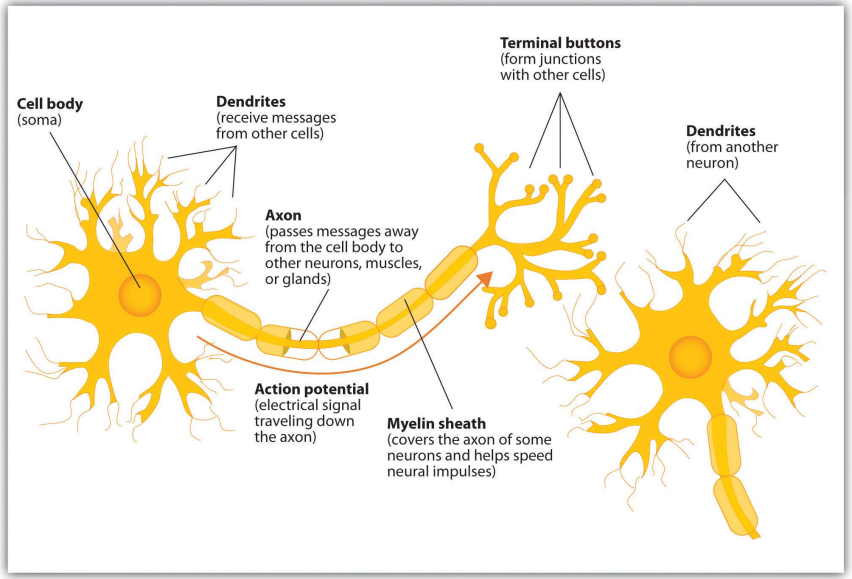
Piotr Mirosław
dr inż. Andrzej Skoczeń

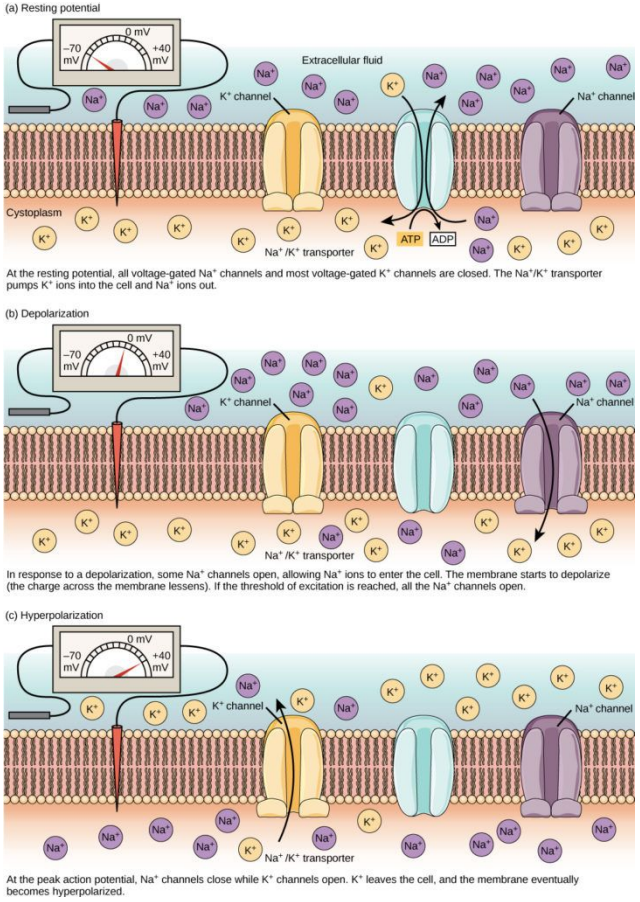
Wprowadzenie

- Trzecia generacja sieci neuronowych
- informacje w postaci impulsów (spikes)
- symulacja oparta o zdarzenia
- brak wydajnych symulatorów SNN

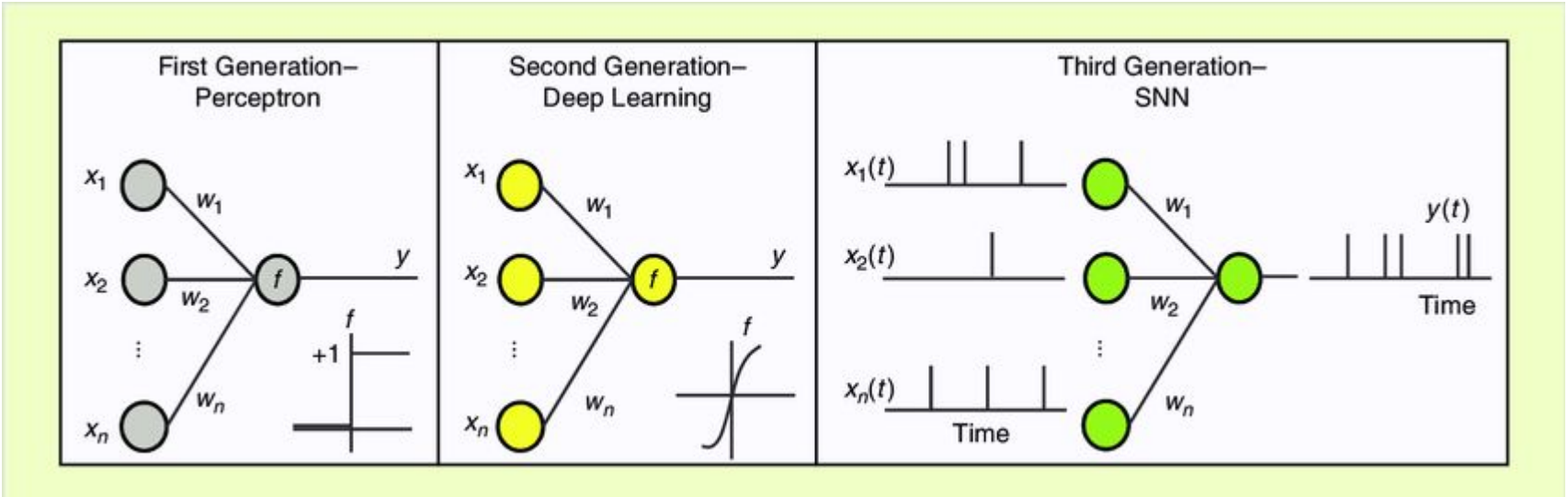


Budowa neuronu



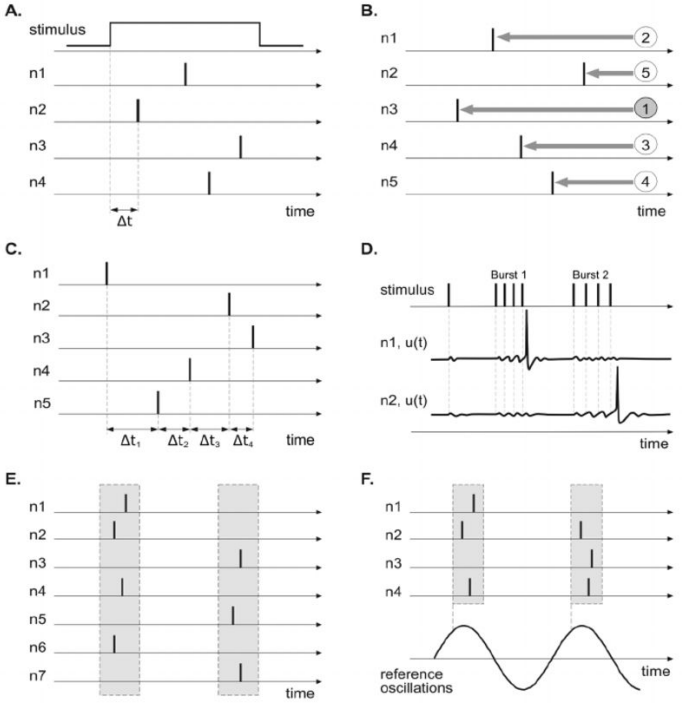


Trzecia generacja sieci neuronowych



Informacje w postaci impulsów

(A) Czas do pierwszego impulsu;
(B) kodowanie kolejności impulsów;
(C) kodowanie oparte o dokładne czasy impulsów;
(D) resonant burst coding;
(E) kodowanie przez synchronizację;
(F) kodowanie fazy



Symulacja oparta o zdarzenia

Event-driven

Każdy impuls jest zdarzeniem mającym miejsce w określonej chwili czasu

Pomiędzy impulsami nic się nie dzieje, więc nie ma potrzeby aktualizowania obliczeń.

Powoduje to mniejsze zapotrzebowanie energetyczne sieci zbudowanej na symulatorze opartym o zdarzenia, niż bazującym na zegarze.

Clock-driven

W systemach opartych o zegar, aktualizacja stanu dzieje się co dany przedział czasu.

Im mniejszy jest, tym większą precyzję wyników można uzyskać.

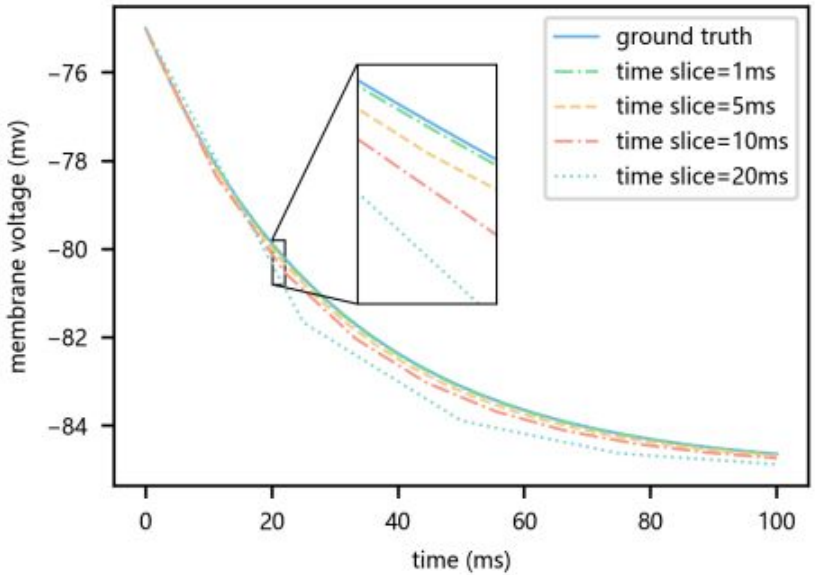
Problemem jest jednak to, że obliczenia są wykonywane często i wtedy, gdy nie jest to potrzebne (nie zachodzą żadne zmiany).

Dostępne symulatory

Simulator	Open Source	Simulation	GPU	Programming Language
Brian2 [17]	Yes	clock-driven	Yes	C++ with Python wrapper
Bindsnet [25]	Yes	clock-driven	Yes	C++ with Python wrapper
PyNest [20]	Yes	clock/event-driven	Yes	C++ with Python interface
Nengo [26]	Yes	clock-driven	Yes	C++ with Python wrapper
NEURON [27]	Yes	clock-driven	No	C++ with Python interface
CARLsim [28]	Yes	clock-driven	Yes	C/C++ with Python wrapper
EDHA (ours)	Yes	event-driven	No	Java

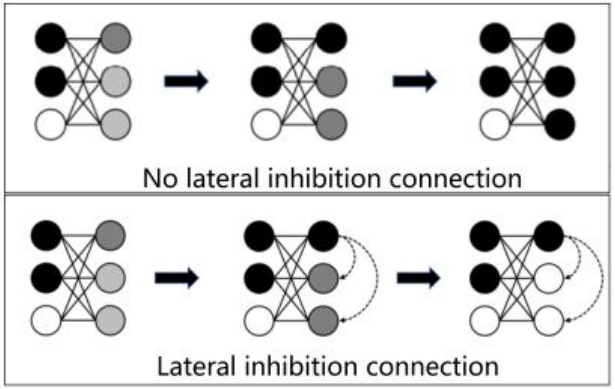
Clock vs event driven

Porównanie dokładności napięcia membrany pod wpływem impulsu, dla różnej wielkości przedziału czasowego w systemach clock-driven.

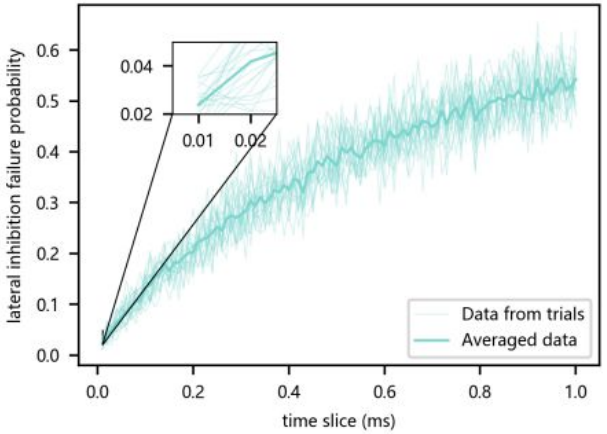


Clock vs event-driven

W przypadku systemów clock-driven zjawisko hamowania bocznego może nie działać tak jak powinno. Jeśli w jednym czasie na wyjściu zostaną wygenerowane dwa impulsy, to zahamują siebie nawzajem, co spowoduje wzajemne stłumienie sygnałów na wyjściu.

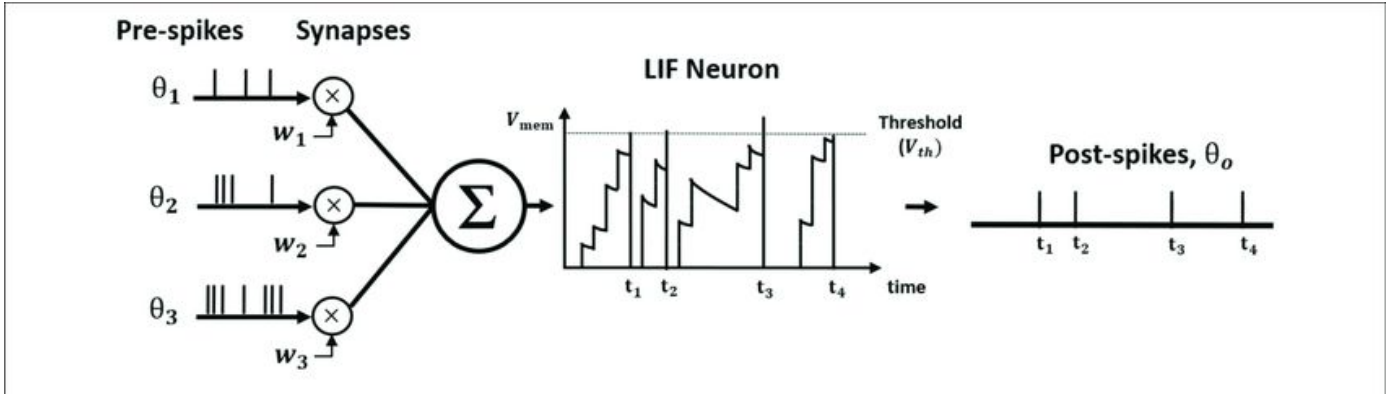


$$f_r = \frac{n_r}{n}$$



Model neuronów LIF

Neuron przyjmuje impulsy od synaps, które zwiększają potencjał membrany, gdy potencjał przekroczy pewien próg, wtedy neuron generuje impuls wyjściowy do aksonu. Plusem tego modelu jest łatwość jego implementacji, z każdym impulsem obliczona jest nowa wartość potencjału membrany i sprawdzenie czy zostanie wygenerowany impuls wyjściowy.



STDP - reguła uczenia

W symulatorze EDHA zastosowano zasadę uczenia się STDP (spike time-dependent plasticity).

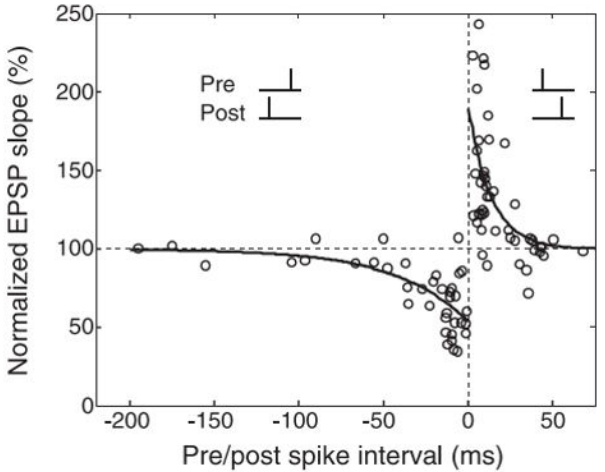
Zgodnie z regułą Hebba, która mówi “neurons that fire together, wire together”, zasada STDP stwierdza, że:

- połączenie synapsy jest wzmocnione, gdy aktywność postsynaptyczna następuje po aktywności presynaptycznej
- podobnie jest osłabiona, gdy aktywność presynaptyczna i postsynaptyczna przebiega w odwrotnej kolejności czasowej

STDP - reguła uczenia

$$\Delta w = \begin{cases} +A_+ \exp\left(-\frac{\Delta t}{\tau_{pre}}\right) & if\ \Delta t > 0 \\ -A_- \exp\left(+\frac{\Delta t}{\tau_{post}}\right) & if\ \Delta t \leq 0 \end{cases}$$

$$\Delta t = t_{post} - t_{pre}$$



A, τ_{pre} - stałe kontrolujące szybkość uczenia się i wrażliwość na czas

Δt - różnica czasu pomiędzy impulsem wychodzącym a wchodzącym

Struktura symulatora EDHA

Moduły w niebieskich ramkach można modyfikować, a w żółtych ramkach nie.

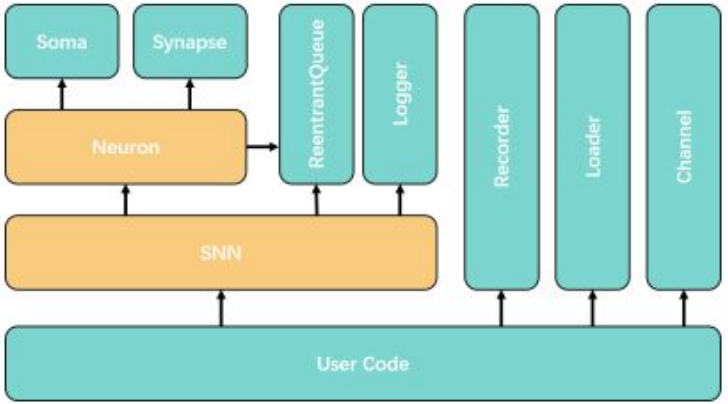
Neuron - funkcje neuronu

SNN - koordynacja neuronów

ReentrantQueue - kolejka impulsów

Logger - informacje o błędach

Recorder, Loader, Channel - Zapis informacji o impulsach, wczytywanie oraz zapis do pliku



Spike Queue

Skoro nie czas to co?

Impulsy przechowywane są w strukturze danych jaką jest sterta zaimplementowana jako drzewo binarne. Kluczami po których impulsy są sortowane są czasy wysłania impulsu.

Czas najbliższego impulsu staje się czasem aktualnym.

Table 3. Comparison of time complexity of spike queue constructed by three data structures.

Data Structure	Insert	Update	Deletee	Get
Minimum heap	O(log(n))	O(log(n))	O(1)	O(1)
Array	O(n)	O(n)	O(n)	O(1)
Linked list	O(n)	O(n)	O(1)	O(1)

Porównanie złożoności czasowej implementacji kolejki impulsów przez trzy struktury danych

Jak stworzyć sieć na symulatorze EDHA?

Na przykładzie danych ze zbioru ręcznie pisanych cyfr MNIST - http://yann.lecun.com/exdb/mnist/

1. Inicjalizacja wejściowej warstwy neuronów - 784 neurony (obrazy 28x28)
2. Inicjalizacja wyjściowej warstwy neuronów - 400 neuronów - uczenie nienadzorowane; powstaje coś w stylu sieci Kohonena
3. Połączenie synapsami neuronów wejściowych z wyjściowymi, przydzielenie losowych wag początkowych
4. Dodanie warstwy hamowania cech - połączenie neuronów wyjściowych ze sobą synapsami z ujemnymi wagami, w ten sposób dwa neurony nie nauczą się tej samej cechy.

Jak stworzyć sieć na symulatorze EDHA?

5. Inicjalizacja kolejki impulsów
6. W pętli wczytanie impulsów z pliku
7. Uruchomienie symulatora
8. Zapisanie wag synaps i progów napięciowych aktywacji neuronów do plików

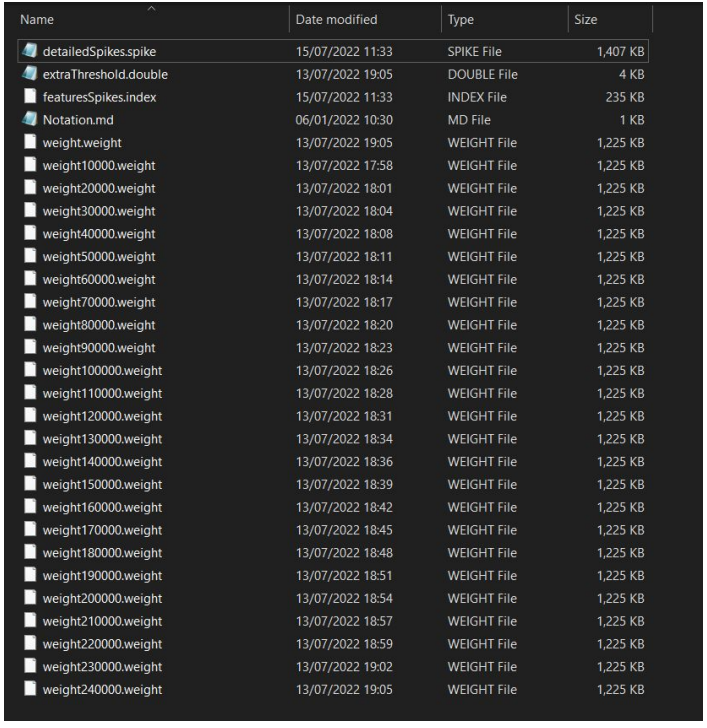
Uruchomienie symulatora

Wynikiem trenowania sieci jest zbiór plików zawierających wagi synaps. Wagi zapisywane są do pliku co 10000 obserwacji. Do testów pobrana jest ostateczna wersja wag znajdująca się w pliku weight.weight

Uczenie sieci:

(niestety brak informacji o dokładności sieci w danej epoce)

```
"C:\Users\Piotr Mirosla
10000
20000
30000
```



Name	Date modified	Type	Size
detailedSpikes.spike	15/07/2022 11:33	SPIKE File	1,407 KB
extraThreshold.double	13/07/2022 19:05	DOUBLE File	4 KB
featuresSpikes.index	15/07/2022 11:33	INDEX File	235 KB
Notation.md	06/01/2022 10:30	MD File	1 KB
weight.weight	13/07/2022 19:05	WEIGHT File	1,225 KB
weight10000.weight	13/07/2022 17:58	WEIGHT File	1,225 KB
weight20000.weight	13/07/2022 18:01	WEIGHT File	1,225 KB
weight30000.weight	13/07/2022 18:04	WEIGHT File	1,225 KB
weight40000.weight	13/07/2022 18:08	WEIGHT File	1,225 KB
weight50000.weight	13/07/2022 18:11	WEIGHT File	1,225 KB
weight60000.weight	13/07/2022 18:14	WEIGHT File	1,225 KB
weight70000.weight	13/07/2022 18:17	WEIGHT File	1,225 KB
weight80000.weight	13/07/2022 18:20	WEIGHT File	1,225 KB
weight90000.weight	13/07/2022 18:23	WEIGHT File	1,225 KB
weight100000.weight	13/07/2022 18:26	WEIGHT File	1,225 KB
weight110000.weight	13/07/2022 18:28	WEIGHT File	1,225 KB
weight120000.weight	13/07/2022 18:31	WEIGHT File	1,225 KB
weight130000.weight	13/07/2022 18:34	WEIGHT File	1,225 KB
weight140000.weight	13/07/2022 18:36	WEIGHT File	1,225 KB
weight150000.weight	13/07/2022 18:39	WEIGHT File	1,225 KB
weight160000.weight	13/07/2022 18:42	WEIGHT File	1,225 KB
weight170000.weight	13/07/2022 18:45	WEIGHT File	1,225 KB
weight180000.weight	13/07/2022 18:48	WEIGHT File	1,225 KB
weight190000.weight	13/07/2022 18:51	WEIGHT File	1,225 KB
weight200000.weight	13/07/2022 18:54	WEIGHT File	1,225 KB
weight210000.weight	13/07/2022 18:57	WEIGHT File	1,225 KB
weight220000.weight	13/07/2022 18:59	WEIGHT File	1,225 KB
weight230000.weight	13/07/2022 19:02	WEIGHT File	1,225 KB
weight240000.weight	13/07/2022 19:05	WEIGHT File	1,225 KB

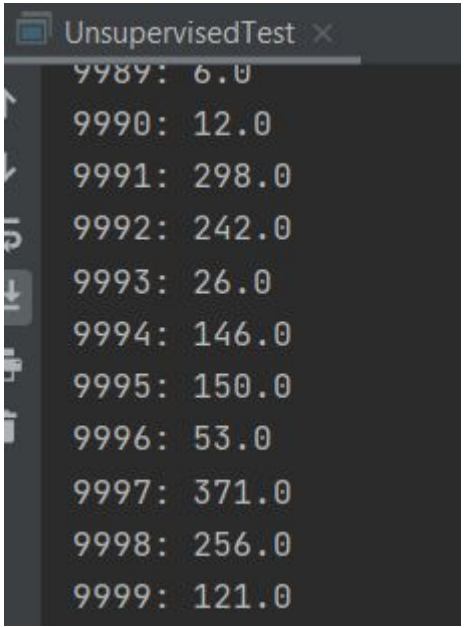
Uruchomienie symulatora

W pliku extraThreshold znajdują się informacje o progach napięciowych aktywacji neuronów - 400 wartości typu double.

```
"C:\Users\Piotr Miroslaw\.jdks\
0: 198.20520048574636
1: 184.77551953745152
2: 212.751733141171
3: 155.716225077009
4: 163.41354211062298
5: 208.06016839894278
6: 204.1797546133781
7: 227.287713251695
8: 318.5944941770835
9: 304.2962708997511
10: 250.50049056930192
```

Uruchomienie symulatora

Po procesie testowania w pliku featuresSpikes.index znajdują się “klasy” do których dana obserwacja została przypisana. Jako, ze jest to uczenie nienadzorowane, zamiast 10 klas, wyekstrahowane zostało 400 cech i względem tych cech obrazy zostały podzielone.



Porównanie prędkości

Table 4. Attributes that were used in the speed comparison network. EDHA with asterisk is the EDHA run on the optimized model.

Attribute	Brian2	Bindsnet	EDHA	EDHA *
Input layer size	28 × 28	28 × 28	28 × 28	28 × 28
Output layer size	20 × 20	20 × 20	20 × 20	20 × 20
Inhibited layer size	20 × 20	20 × 20	0 [1]	0 [1]
Number of smaples in training set	60,000	60,000	60,000	60,000
Number of samples in testing set	10,000	10,000	10,000	10,000
Averaged number of spikes per sample	2284.38	2284.38	2284.38	136.54
Training Epoch	3	3	3	4

No inhibited layer is required to implement lateral inhibition in EDHA.

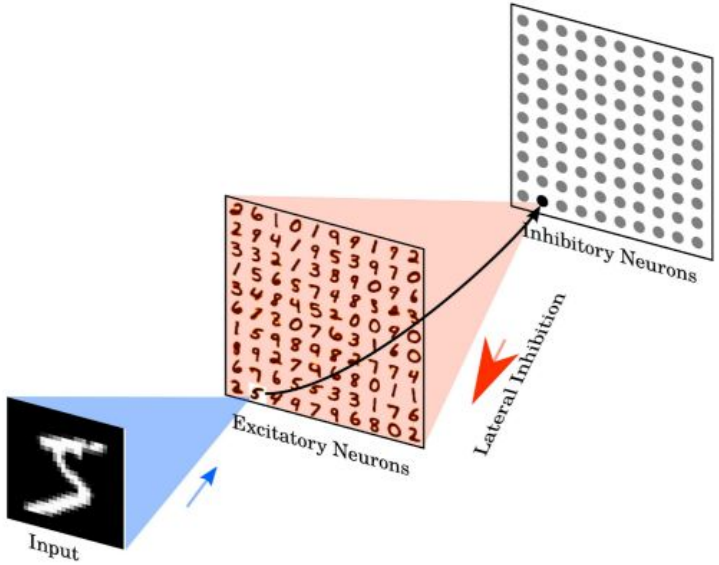
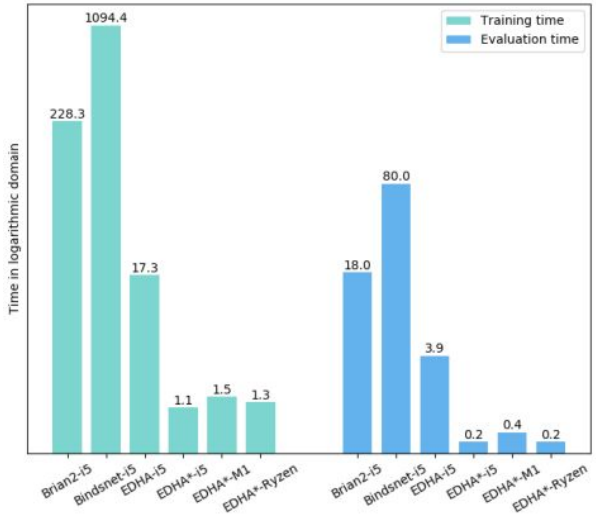


Figure 9. Network structure of Deihl's work [31].

Porównanie prędkości

Simulator	CPU	RAM	OS	Accuracy
Brian2	Intel i5-6500 (3.2 GHz × 4)	16G	win10	87.7%
Bindsnet	Intel i5-6500 (3.2 GHz × 4)	16G	win10	86.7%
EDHA	Intel i5-6500 (3.2 GHz × 4)	16G	win10	87.9%
EDHA *	Intel i5-6500 (3.2 GHz × 4)	16G	win10	88.0%
EDHA *	Apple M1(3.2 GHz × 8)	8G	BigSur	87.9%
EDHA *	AMD Ryzen 5 2500U (3.6 GHz × 4)	6.3G	win10	88.0%



Benchmarking

2 warstwy po n neuronów

Warstwa wejściowa - n wejść typu Poisson z ustawioną częstotliwością wysyłania impulsu

Wyjście zgodnie z modelem LIF i zasadą STDP

Warstwy połączone każda z każdą,

Wagi połączeń 1. stałe, 2. zmienne (uczące się)

Parametry platformy: Intel i5-6500 CPU, 16 GB memory

Benchmarking

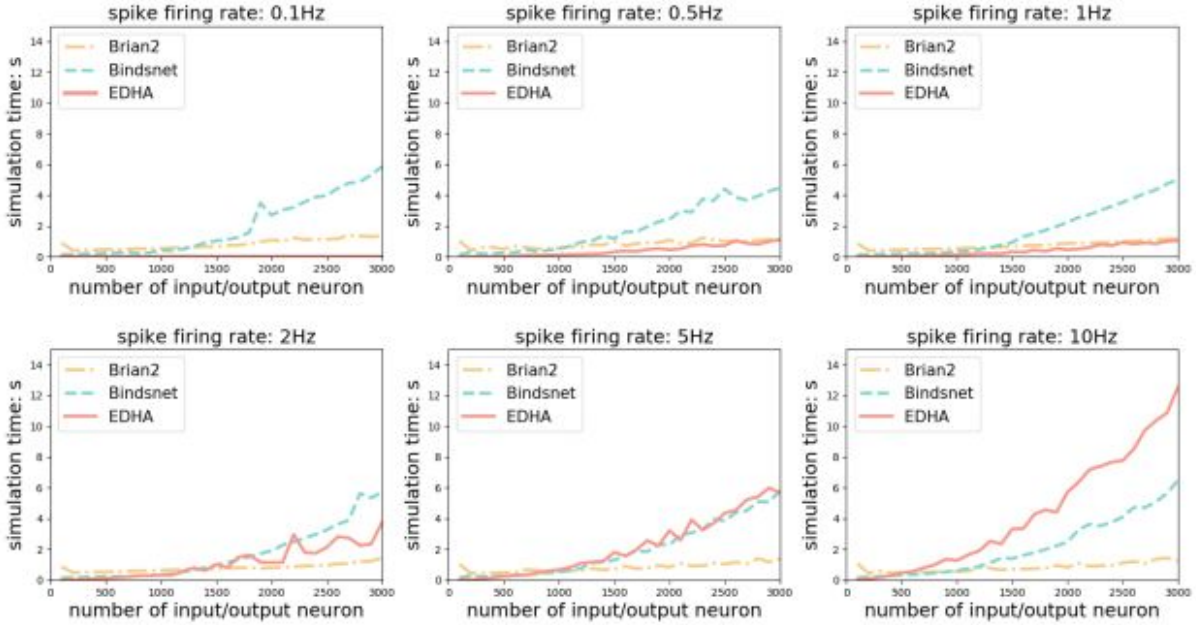


Figure 11. Benchmark simulation with fixed weight.

Benchmarking

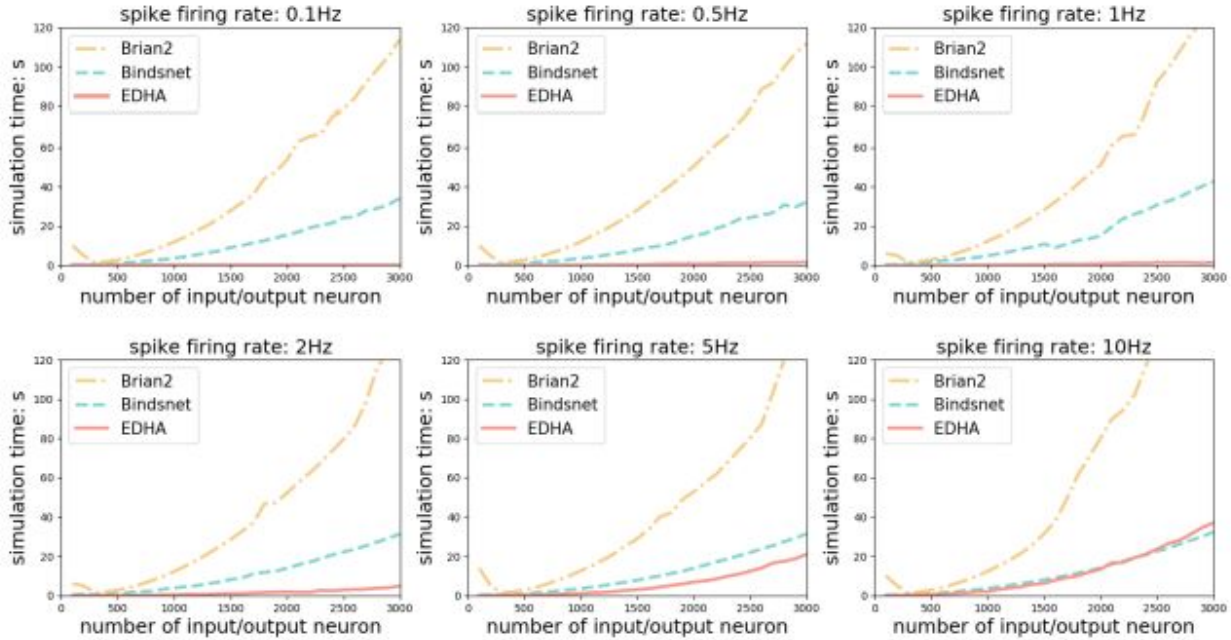


Figure 12. Benchmark simulation with learnable weight.

Dodatkowe uwagi

1. Dokładność modelu

   Przy ewaluacji modelu brak jakiegokolwiek dostępu do metryk dokładności, funkcji straty.

2. Wielowątkowość

   System EDHA jest jednowątkowy i pomimo tego bardzo wydajny. Istnieje zatem możliwość ulepszenia systemu poprzez wprowadzenie wielowątkowości.

3. Symulacje na większą skalę

   W przeprowadzonej przeze mnie symulacji liczba neuronów to 784 neurony wejściowe i 400 neuronów wyjściowych, podczas gdy mózg to ok. 100 miliardów neuronów.

Źródła

- EDHA/code - snnhub
- EDHA: Event-Driven High Accurate Simulator for Spike Neural Networks - Lingfei Mo , Xinao Chen and Gang Wang
- Contributed article Networks of spiking neurons: The third generation of neural network models Wolfgang Maass
- Event- and Time-Driven Techniques Using Parallel CPU-GPU Co-processing for Spiking Neural Networks - Francisco Naveros1, Jesus A. Garrido1, Richard R. Carrillo1, Eduardo Ros1* and Niceto R. Luque
- An Efficient and Perceptually Motivated Auditory Neural Encoding and Decoding Algorithm for Spiking Neural Networks - Zihan Pan1, Yansong Chua2*, Jibin Wu1, Malu Zhang1, Haizhou Li1 and Eliathamby Ambikairajah

Dziękuję za uwagę

Piotr Mirosław