ПРОГРАМНИ СРЕДСТВА ЗА ОБРАБОТКА НА ДАННИ ЗА МОЗЪЧНА АКТИВНОСТ

ас. д-р Александър Иванов
Бургаски свободен университет

***Abstract**: Brain activity data is used for medical purposes and brain-computer interfaces. Processing this type of data is not trivial matter and it requires special tools. In this article an overview of some programming solutions for neural data processing is provided. The article focuses on the Python programming language and specific libraries for neural data handling and machine learning.*

***Keywords:** neural data, python, machine learning.*

I. Въведение

Данни за мозъчна активност се използват не само в тесните рамки на медицината. В последното десетилетие популярност добиват т.нар. Мозъчно-компютърни интерфейси (Brain-Computer Interfaces – BCI). Тези интерфейси представляват технологични системи, свързващи мозъчната активност с машина. Данни могат да се събират от разнообразни източници. Неинвазивните технологии не изискват хирургия и са предпочитани в немедицински разработки. Такива са Електроенцефалография (ЕЕГ), Електроокулография (ЕОГ), Електромиография (ЕМГ), Магнитоенцефалография (МЕГ), Ядрено-магнитен резонанс (ЯМР), Компютърна томография (КТ) и други. При инвазивните методи се изисква хирургия за свързване на записващите устройства към неврони. Могат да се използват масиви от микроелектроди или Електрокортикография. Една от най-използваните технологии е ЕЕГ. Тази технология има добра времева резолюция и може да засича бързи промени в мозъчната дейност. Записите представляват честоти на мозъчни вълни в интервал от време, генерирани от специфична област на мозъка. Засечените шаблони на активация, може да се обвържат с дейност/стимул, който ги предизвиква. Обработката на тези данни е от ключово значение за качеството на мозъчно-компютърните устройства.

II. Библиотеки за прочит и обработка на данни за мозъчна активност

Една от основните задачи при анализа на мозъчна активност е събирането, визуализирането и обработката на данни. Съществуват множество библиотеки за изпълнение на тези задачи. Езикът Python се е наложил като де факто стандарт за научни изследвания и машинно обучение. Библиотеката MNE-Python [1][2] е предназначена за обработка на данни за мозъчна активност, събирани чрез технологиите ЕЕГ и МЕГ. Сред изследваните величини са невронни трептения, потенциали, свързани със събития (Event-Related Potentials – ERP) и функционална свързаност между отделни области на мозъка. MNE-Python обработва различни типове данни, включително сурови данни, данни, свързани със събития, време-честотни данни, локализация на източник на сигнал и други. В таблица 1 са представени някои поддържани файлови формати за мозъчни данни.

Формат	**Описание**
BrainVision Формат	За данни от електроенцефалография (ЕЕГ)
FIF Формат	Основен формат за МЕГ и ЕЕГ данни
Neuroscan CNT Формат	За данни от ЕЕГ в Neuroscan CNT формат
CTF Формат	За данни от МЕГ
NIfTI Формат	За MRI данни, но също се поддържа за изображение на източници
Nedf Формат	За данни от МЕГ
EGI Формат	За данни от EGI системи, често използвани в ЕЕГ научни изследвания
EGI MFF Формат	За данни от МЕГ
BESA Формат	За данни от ЕЕГ и МЕГ
ASALab Формат	За данни от EEG, съхранени във формата на софтуера ASALab
KIT Формат	За данни от МЕГ
Формат NeuroMag Raw	За данни от МЕГ
EDF/EDF+ Формат	За ЕЕГ данни
CTF 4D Формат	За данни от МЕГ
Комерсиални EEG Формати	За ЕЕГ данни от Philips, Nihon Kohden и Biosemi системи

Таблица 1. Файлови формати за мозъчни данни

MNE-Python предоставя обширни инструменти за предварителна обработка на данни, включително филтриране, епохиране и премахване на артефакти, за да се гарантира качеството на данните. Локализацията на източника позволява да се изолират невронните източници, отговорни за записаните данни. Времево-честотният анализ изследва как мозъчната активност се променя с времето в отговор на различни стимули или задачи. MNE-Python предлага статистически инструменти за тестване на хипотези, което позволява да се идентифицират значими невронни реакции; библиотеката предоставя също и възможности за визуализация, за създаване на топографски карти, изходни оценки и различни типове графики за подпомагане на интерпретацията на данни. Интегрира се с популярни библиотеки за машинно обучение, което я прави подходяща за класифициране и прогнозно моделиране с помощта на невронни данни. MNE-Python има активна потребителска общност и обширна документация, което я прави достъпен инструмент както за начинаещи, така и за опитни изследователи. Изследователи и учени използват MNE-Python за широк спектър от приложения, включително когнитивни невронауки, клинични проучвания и BCI.

MNE предлага набор от примерни данни от МЕГ и ЕЕГ записи. МЕГ данните са от система Neuromag VectorView, а ЕЕГ данните съдържат записи от 60 електрода. Наборът от данни съдържа записи за слухови и визуални стимули, представени на случаен принцип. Визуалните стимули са поредици от шаблони за шахматна дъска, прекъсвани от емотикони. [1] Включени са и записи от ЯМР. Наборът данни може да се използва за тестове и обучение в MNE-Python, като гарантира последователност и възпроизводимост на резултатите. Той също така служи като стандартизиран набор от данни за валидиране за МЕГ/ЕЕГ методи. В библиотеката има и инструменти за изтегляне на други набори данни, като Brainstorm, SPM faces, EEGBCI motor imagery, Somatosensory, Multimodal, mTRF, Visual 92, Kiloword, PolySomnoGraphic, OPM, Bti и др. [1]

Следва примерен код за обработка на данни за мозъчна активност. Кодът е част от един файл, но за прегледност е представен на отдлени секции с пояснения към тях.

Инсталация

Библиотекатаl MNE-Python може да се инсталира чрез пакетния мениджър pip:
pip install mne

Зареждане на данни

В карето е представен код за прочит на сурови данни, получени чрез МЕГ, които са част от вграден в библиотеката набор. При първо пускане на кода, необходимите данни ще бъдат заредени.

```
import mne
# Load sample data (MEG data)
raw = mne.io.read_raw_fif(mne.datasets.sample.data_path() +
'/MEG/sample/sample_audvis_raw.fif',                preload=True)
```

Обработка на данните

Предварителната обработка е решаваща стъпка за гарантиране на качеството на данните. Обичайните операции включват филтриране, епохиране и премахване на артефакти.

a. Филтриране

Филтрирането помага за премахване на шума и изолиране на интересуващите ни честотни ленти. В примера вградената функция filter изрязва диапазона 1-40 Хц.

```
# Apply bandpass filter (for example, 1-40 Hz)

raw.filter(1, 40)
```

b. Епохиране

Епохирането сегментира данните в по-малки времеви прозорци въз основа на маркери за събития. Метод .find_events(raw).

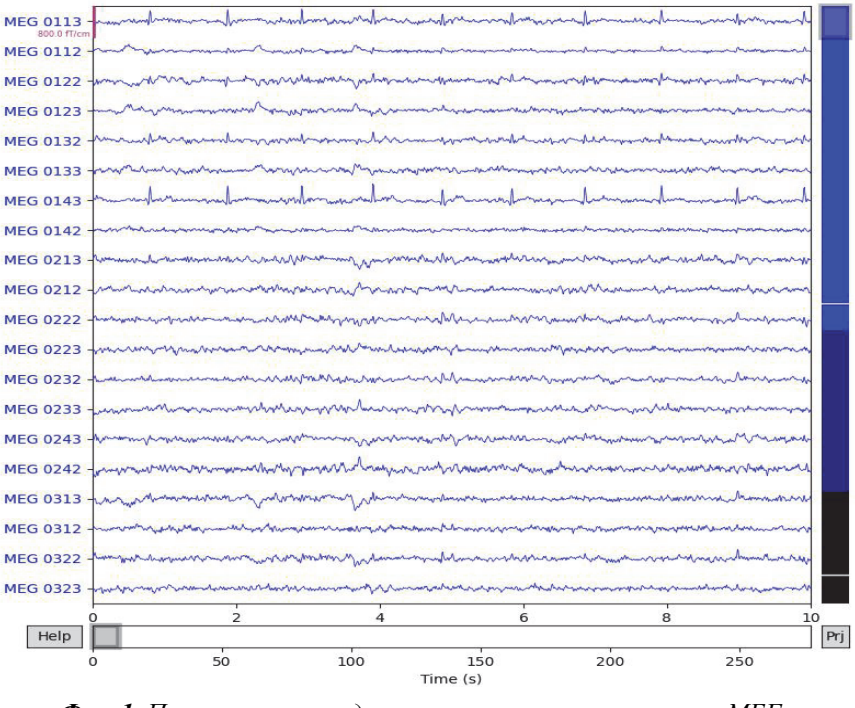
```
# Define events based on triggers

events = mne.find_events(raw)
# Create epochs around specific events (e.g., auditory stimuli)

epochs = mne.Epochs(raw, events, event_id=1, tmin=-0.2, tmax=0.5,
baseline=(None, 0))
```

4. Визуализиране на данните

MNE-Python съдържа вградени инструменти за визуализация на данни. Кодът в карето изчертава графики на сурови и обработени данни и ги запазва във външни файлове.

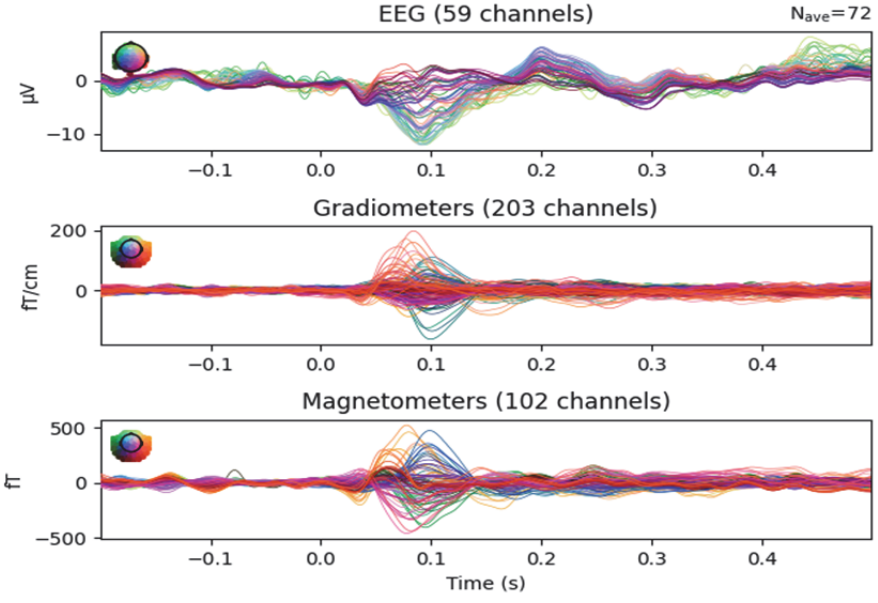
```
# Plot raw data

r=raw.plot()
r.savefig("raw.jpg")

# Plot an averaged evoked response

evoked = epochs.average()
fig=evoked.plot()
fig.savefig('sample.png')
```

На следващите фигури са показани визуализации на примерни данни, вградени в библиотеката. Изображенията са генерирани от кода в карето.



Фиг. 1. *Примерни сурови данни за мозъчна активност от МЕГ. На всеки ред са данните от различен канал/сензор.*

На Фиг. 1 са визуализирани сурови данни за записана мозъчна активност. На всеки ред е записаната активност от различен номериран сензор. По абсцисата е отразено времето. Данните са от вградения набор в библиотеката MNE, файлът raw.jpg, генериран от посоченият в предното каре код. На Фиг.2 са визуализирани данни от предизвикани потенциали, съответстващи на конкретен стимул. Данните са от същия набор.



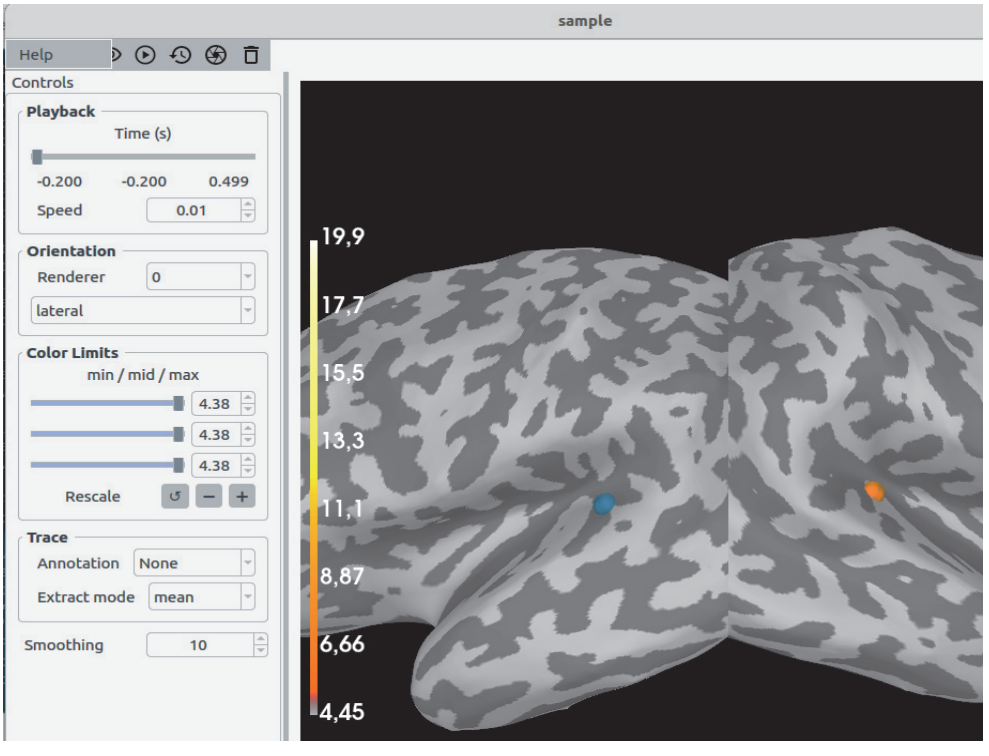
Фиг. 2. *Примерни данни за предизвикани невронни потенциали от примерния набор от библиотеката MNE. Файл sample.png*

Локализиране на източник

Локализацията на източника е процес за установяване на областите на мозъка, отговорни за записаните данни. Права задача е проблемът за предсказване как невронната активност в мозъка генерира електромагнитни сигнали, които се записват от ЕЕГ и/или МЕГ сензорите, поставени върху или около скалпа. Правата задача обикновено включва решаване на сложни уравнения на електромагнитното поле за симулиране на данните от сензора за дадена конфигурация на източника. Обратната задача при ЕЕГ и МЕГ е обратният процес – целта е да се оцени разпределението на невронните източници в мозъка, които са генерирали записаните сензорни данни. Това е сложен проблем, тъй като много различни конфигурации от източници могат да доведат до едни и същи сензорни данни. Обратен оператор е изчислителен алгоритъм или метод, използван за оценка на активността на невронния източник. Операторът използва записаните данни като вход и прилага математически трансформации, за да оцени вероятните невронни източници в мозъка. Няколко метода и алгоритми се използват в BCI за решаване на обратната задача. Те включват, но не се ограничават до Оценка на минималната норма (MNE) [3], Динамично статистическо параметрично картографиране (dSPM) [4], анализ на независими компоненти (ICA). В следващото каре е представен код за решаване на обратната задача.

```
data_path = mne.datasets.sample.data_path()
data_path_str = str(data_path)  # Convert PosixPath to a string
raw_fname = data_path_str + '/MEG/sample/sample_audvis_raw.fif'
evoked_fname = data_path_str + '/MEG/sample/sample_audvis-ave.fif'
subjects_dir = data_path_str + '/subjects'

from mne.minimum_norm import make_inverse_operator, apply_inverse
import pyvistaqt

data_path = mne.datasets.sample.data_path()
data_path_str = str(data_path)

# Load the forward solution
fname_fwd = data_path_str + '/MEG/sample/sample_audvis-meg-oct-6-fwd.fif'
fwd = mne.read_forward_solution(fname_fwd)

# Load the noise covariance matrix
fname_cov = data_path_str + '/MEG/sample/sample_audvis-cov.fif'
cov = mne.read_cov(fname_cov)

# Load the inverse operator
fname_inv = data_path_str + '/MEG/sample/sample_audvis-meg-oct-6-meg-inv.fif'
inv = mne.minimum_norm.read_inverse_operator(fname_inv)
```

За триизмерна визуализация на локализираните източници се изискват допълнителни библиотеки като pyvistaqt [5], nibabel [6]. Могат да се инсталират чрез pip. Визуализацията от кода е показана на Фиг. 3



Фиг. 3. *Визуализация на източник на сигнали*

Статистика

MNE-Python съдържа инструменти за статистически анализи. В настоящата публикация тези инструменти не се разглеждат.

Освен за обработка на съществуващи данни, съществуват и библиотеки за невронни симулации. Чрез библиотеката NEURON+ на Python могат да се правят симулации на биологични невронни мрежи. [7] Разработена от Стандфорд; съдържа среда за симулации, графичен интерфейс и модула H. На Фиг. 4 е представена графика на активацията на 2 симулирани неврона чрез библиотеката.

Освен на Python, библиотеки за обработка на мозъчни данни има и в Matlab – Neurophysiological Biomarker Toolbox [8], EEGLAB [9].

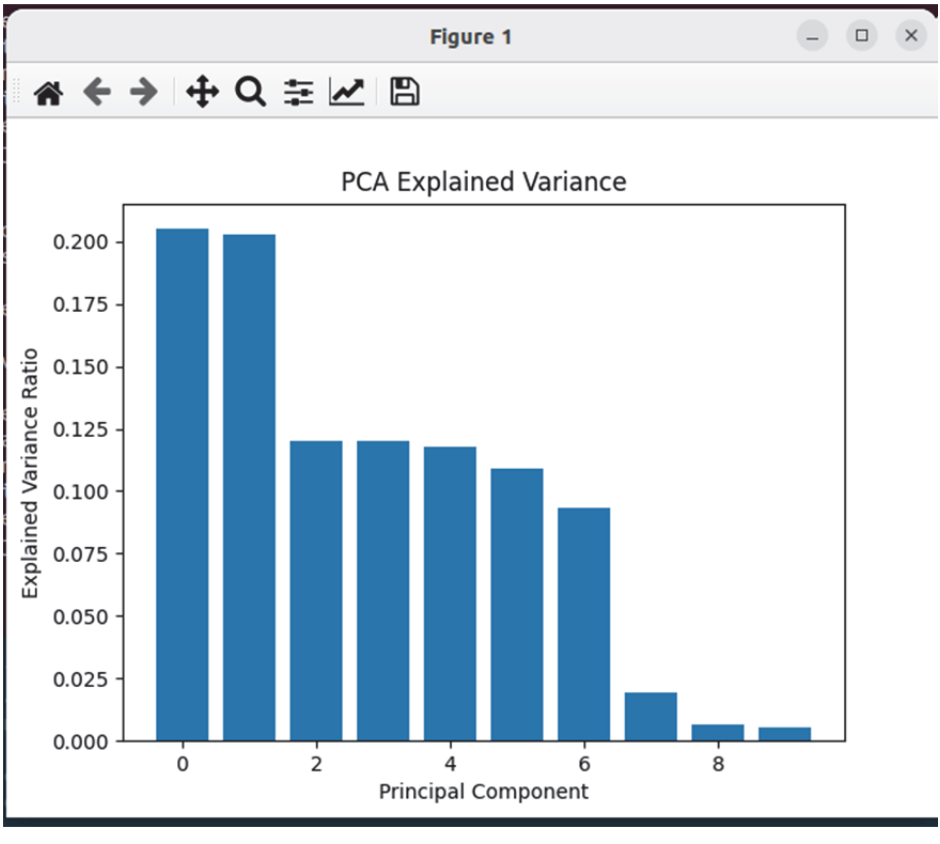
III. Технологии от машинното обучение за обработка на данни за мозъчна активност

Машинното обучение е водещият съвременен подход за обработка на данни. В голямата си част задачите за обработка на данни изискват поточна линия, започваща с неконтролирани алгоритми за почистване на данните, намаляване на размер (Анализ

на главни/независими компоненти) и групиране (DBSCAN, k-средни). Обработените данни се предават на алгоритми за контролирано обучение – класификатори. Най-често се използват изкуствени невронни мрежи, но също Бейсови класификатори, Поддържащи векторни машини, Дървета на решенията и др. MNE-Python лесно се интегрира с библиотеки за машинно обучение за по-напреднали анализи. Scikit-learn [10] е библиотека за машинно обучение на езика Python. Съдържа богат набор от алгоритми, включително изброените по-горе. Включени са и спомагателни инструменти – за етикиране на данни, за премащабиране на стойности, матрица на объркването, крос-валидация и др. По-долу е посочен пример за анализ на главни компоненти чрез използване на scikit-learn:

```
from sklearn.decomposition import PCA

# Epoch Data

events = mne.find_events(raw)
event_id = {"auditory/left": 1, "auditory/right": 2}  # Customize based on labels
epochs = mne.Epochs(raw, events, event_id, tmin=-0.2, tmax=0.5, baseline=(None, 0), preload=True)

# Reshape data to (trials, channels * time points)

X = epochs.get_data().reshape(len(epochs), -1)

# Apply PCA

n_components = 10
pca = PCA(n_components=n_components)
X_pca = pca.fit_transform(X)

# Visualize PCA Results

explained_variance_ratio = pca.explained_variance_ratio_
plt.bar(range(n_components), explained_variance_ratio)
plt.xlabel("Principal Component")
plt.ylabel("Explained Variance Ratio")
plt.title("PCA Explained Variance")
```

На Фигура 5 са визуализирани главните компоненти от примерния файл.



Фиг. 5 *. Главни компоненти от файлът с примерни данни от вградения набор в библиотеката*

Върху получения резултат може да се обучи класификатор, например поддържаща векторна машина (Support Vector Machine). Този вид класификатори изчисляват права линия, разделяща оптимално набори от точки. Въпреки, че е линеен модел, чрез нелинейно полагане може да извършва нелинейна класификация. За тест се може да се извърши крос-валидация.

Друга популярна платформа за машинно обучение е TensorFlow. Тя е разработена от Google с фокус върху невронните подходи. Съдържа модели като LSTM и конволюционни мрежи. [11] Първите са подходящи за обработка на циклични данни (с рекурсивен характер, например времеви редове), а вторите изпълняват конволюционни стъпки върху входно пространство.

Данни за невронна активност могат да се представят под формата на графи. Графите описват мозъчния конектом – шаблонът на свързване на отделните зони на мозъка (разбира се, с ограничена резолюция). Популярни модели за обработка на данни, структурирани като граф, са Графовите конволюционни мрежи (Graph Convolutional Networks – GCNs). [12] Вместо да оперират върху решетка (като в стан-

дартните конволюционни невронни мрежи), те работят с връзките между елементите (ребрата на графите). Основната идея на GCNs е да научат как информацията се разпространява по графа. Те използват съседство и свързаност между върховете, за да извличат информация от съседите и да обработват върховете на графа. Това ги прави изключително полезни за анализ на социални мрежи, геномни данни, географски данни и други. GCNs позволяват на моделите да прихванат контекста на върховете в графа и да предсказват свойствата или връзките между тях. Те са важен инструмент за развиването на алгоритми за анализ на графови данни и за решаване на разнообразни задачи върху графи.

Следва неизчерпателен списък с библиотеки за обработка на данни в графови структури.

PyTorch Geometric: PyTorch Geometric (PyG) е популярна библиотека за дълбоко обучение, базирано на графи, която включва широка гама от графични конволюционни слоеве, набори от данни с графова структура и помощни програми за изграждане и обучение на GCN. Интегрирана е с PyTorch, което я прави предпочитан избор за изследователи и практици. [13]

DGL (Deep Graph Library): DGL е многофункционална библиотека за дълбоко обучение върху графи. Поддържа различни модели на графови невронни мрежи, включително GCN, и предоставя лесен за използване програмен интерфейс (API) за манипулиране на графи. DGL работи с множество платформи за дълбоко обучение, включително PyTorch и TensorFlow. [14]

StellarGraph: StellarGraph е библиотека на Python за граф-ориентирано машинно обучение. Осигурява абстракции на високо ниво за изграждане и обучение на графови конволюционни мрежи върху широкомащабни графи. StellarGraph поддържа GCN. [15]

Spektral: Spektral е библиотека за дълбоко обучение върху графи, изградена върху TensorFlow и Keras. Предлага прост и интуитивен интерфейс за създаване и обучение на GCN и други базирани на графи модели. Spektral е известна със своя удобен за потребителя интерфейс. [16]

GraphSAGE: GraphSAGE предоставя инструменти за обучение за индуктивно представяне на графи. Може да се използва за класификация на графи и задачи за класификация на възли и е използваема за широкомащабни графи. [17]

Графови мрежи на OpenAI: Библиотеката Graph Nets на OpenAI е гъвкава платформа за изграждане и обучение на базирани на графи модели. Включва реализации на различни графови невронни мрежи, включително GCN. Интегрира се с TensorFlow. [18]

CuGraph (RAPIDS): CuGraph е опция, предоставяща GPU-ускорен анализ на графи. Част е от пакета от библиотеки RAPIDS и осигурява GPU-задвижвана реализация на GCN и други графови алгоритми. [19]

IV. Заключение

Обработката на данни за мозъчна активност е важна област на практиката, свързана както с медицината, така и с областта на мозъчно-компютърните интерфейси. Съществува широко разнообразие от програмни и математически подходи за обработка на такъв тип данни, като повечето са базирани на машинно обучение. Чрез пред-

ставените програмни средства може да се постигне добро качество на програмните продукти, които ги използват.

Тази статия е публикувана и подготвена по време на изследователската практика на автора в Университета на Крайова, Румъния, финансиран по проект BG05M2OP001-2.016-0030-C01 „Модернизация на образователните решения за кръгова икономика, стратегически инфраструктури и производства (МИКС-ИП))", финансиран по ОП „Наука и образование за интелигентен растеж 2014-2020 г."

Литература и интернет източници:

1. https://mne.tools/stable/index.html
2. Gramfort A, Luessi M, Larson E, Engemann DA, Strohmeier D, Brodbeck C, Goj R, Jas M, Brooks T, Parkkonen L, Hämäläinen M. MEG and EEG data analysis with MNE-Python. Front Neurosci. 2013 Dec 26;7:267. doi: 3389/fnins.2013.00267. PMID: 24431986; PMCID: PMC3872725.
3. Olaf Hauk, Keep it simple: a case for using classical minimum norm estimation in the analysis of EEG and MEG data, NeuroImage, Volume 21, Issue 4, 2004, Pages 1612-1621, ISSN 1053-8119, https://doi.org/10.1016/j.neuroimage.2003.12.018
4. J. Nawel, H. Abir, B. Ichrak, N. Amal and B. A. Chokri, „A Comparison of Inverse Problem Methods for Source Localization of Epileptic Meg Spikes", *2019 IEEE 19th International Conference on Bioinformatics and Bioengineering (BIBE)*, Athens, Greece, 2019, pp. 867-870, doi: 10.1109/BIBE.2019.00161.
5. https://docs.pyvista.org/version/stable/
6. https://nipy.org/nibabel/
7. https://neuron.yale.edu/neuron/static/docs/neuronpython/firststeps.html
8. https://en.wikipedia.org/wiki/Neurophysiological_Biomarker_Toolbox
9. https://en.wikipedia.org/wiki/EEGLAB
10. https://scikit-learn.org/stable/
11. https://www.tensorflow.org/
12. Chaoqiang Liu, Hui Ji, Anqi Qiu, Fast vertex-based graph convolutional neural network and its application to brain images, Neurocomputing, Volume 434, 2021, Pages 1-10, ISSN 0925-2312, https://doi.org/10.1016/j.neucom.2020.12.097.
13. https://github.com/rusty1s/pytorch_geometric
14. https://github.com/dmlc/dgl
15. https://www.stellargraph.io/
16. https://graphneural.network/
17. https://github.com/williamleif/GraphSAGE
18. https://github.com/deepmind/graph_nets
19. https://github.com/rapidsai/cugraph