



UNIVERSITATEA TEHNICĂ

DIN CLUJ-NAPOCĂ

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

DEPARTAMENTUL CALCULATOARE

ABORDĂRI PENTRU GRUPAREA POTENȚIALELOR FOLOSIND AUTOCODIFICATOARE

LUCRARE DE DISERTAȚIE

	Absolvent masterand:	Eugen-Richard ARDELEAN
	Coordonator științific:	Prof. Dr. Eng. Mihaela DÎNȘOREANU

2021



UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

DECAN,		DIRECTOR DEPARTAMENT,
Prof. dr. ing. Liviu MICLEA		Prof. dr. ing. Rodica POTOLEA

Eugen-Richard ARDELEAN

**ABORDĂRI PENTRU GRUPAREA POTENȚIALELOR
FOLOSIND AUTOCODIFICATOARE**

1. **Enunțul temei:** În domeniul Grupării Potențialelor, gruparea este dificil de făcut datorită problemelor care pot să apară la înregistrare și la preprocesare și datorită diferitelor feluri de neuroni. Aceasta teza propune metode de Extragere a Trăsăturilor pentru a îmbunătăți rezultate grupării.
2. **Conținutul lucrării:** Introducere, Obiectivele cercetării, Studiu bibliografic, Prezentarea proiectului, Rezultate experimentale și Interpretări, Concluzii, Bibliografie, Anexa
3. **Locul documentării:** Universitatea Tehnică din Cluj-Napoca, Departamentul Calculatoare, Tehnologia Informației în Economie
4. **Consultanți:** Dr. Eng. Raul Cristian Mureșan și Dr. Eng. Vasile Vlad Moca (TINS, Transylvanian Institute of Neuroscience), Prof. Dr. Eng. Rodica Potolea și Prof. Dr. Eng. Camelia Lemnaru (UTCN)
5. **Data emiterii temei:** 1 octombrie 2019
6. **Data predării:** 5 iulie 2021

Student masterand:	<i>ACK</i>
Coordonator științific:	

**UNIVERSITATEA TEHNICĂ**

DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

**Declarație pe proprie răspundere privind
autenticitatea lucrării de disertație**

Subsemnatul(a) ARDELEAN EUGEN-RICHARD

legitimat(ă) cu CI seria XH nr. 878025,
 CNP 196122055072, autorul lucrării
ABORNAȚI PENTRU GRUPAREA POTENȚIALELOR FOLOSIND
AUTO CODIFICATOARE
 elaborată în vederea susținerii
 examenului de finalizare a studiilor de licență la Facultatea de Automatică și Calculatoare,
 Specializarea TEHNOLOGIA INFORMAȚIEI ÎN ECONOMIE din cadrul Universității
 Tehnice din Cluj-Napoca, sesiunea IULIE a anului universitar 2021,
 declar pe proprie răspundere, că această lucrare este rezultatul propriei activități
 intelectuale, pe baza cercetărilor mele și pe baza informațiilor obținute din surse care au
 fost citate, în textul lucrării, și în bibliografie.

Declar, că această lucrare nu conține porțiuni plagiate, iar sursele bibliografice au
 fost folosite cu respectarea legislației române și a convențiilor internaționale privind
 drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în fața unei alte
 comisii de examen de disertație.

In cazul constatării ulterioare a unor declarații false, voi suporta sancțiunile
 administrative, respectiv, *anularea examenului de disertație*.

Data		Nume, Prenume
<u>05.07.2021</u>		<u>ARDELEAN EUGEN-RICHARD</u>
		<u>ER</u>
		Semnătura

Cuprins

Capitolul 1. Introducere.....	1
1.1. Contextul proiectului.....	1
1.1.1. Neuronul.....	1
1.1.2. Neuroștiința.....	2
1.1.3. Gruparea Potențialelor.....	2
1.2. Conținutul lucrării.....	4
1.3. Colaborări.....	4
Capitolul 2. Obiectivele cercetării.....	5
2.1. Scopul lucrării.....	5
2.2. Obiective specifice.....	5
Capitolul 3. Studiu Bibliografic și Stadiul actual în domeniu.....	7
3.1. Bazele Grupării Potențialelor.....	7
3.2. Algoritmi de Grupare.....	7
3.2.1. K-Means.....	7
3.2.2. DBSCAN.....	8
3.2.3. SBM.....	8
3.3. Evaluarea performanței.....	9
3.3.1. Evaluarea performanței folosind metrii externe.....	9
3.3.2. Evaluarea performanței folosind metrii interne.....	10
3.4. Metode actuale de Extragere a Trăsăturilor.....	12
3.4.1. Principal Component Analysis.....	12
3.4.2. Metode bazate pe derivate.....	13
3.5. Metode bazate pe procesarea de semnal.....	13
3.5.1. Magnitudine și Fază.....	13
3.5.2. Short-Time Fourier Transform.....	13
3.5.3. Superlet Transform.....	14
3.5.4. Hilbert Transform.....	15
3.5.5. Fast Fourier Transform.....	15
3.5.6. Multitaper: Discrete Prolate Spherical Surfaces (DPSS).....	17
3.6. Metode bazate pe învățare.....	18
3.6.1. Autocodificatoare.....	18
3.6.2. Ensemble.....	20

3.6.3. Long Short-Term Memory (LSTM).....	21
Capitolul 4. Prezentarea proiectului.....	22
4.1. Seturi de date.....	22
4.1.1. Structura setului de date.....	22
4.1.2. Caracteristicile setului de date.....	23
4.1.3. Formatul setului de date.....	23
4.2. Preprocesarea potențialelor de acțiune.....	24
4.3. Extragerea Trăsăturilor.....	25
4.3.1. Principal Component Analysis.....	25
4.3.2. Extragerea Trăsăturilor folosind derivatele de grad 1 si 2.....	25
4.4. Pipeline.....	27
4.5. Autocodificator.....	31
4.5.1. Modelul arhitecturii.....	31
4.5.2. Antrenarea Autocodificatorului.....	33
4.5.3. Aplicarea PCA.....	35
4.5.4. Pre-antrenare.....	37
4.5.5. Expansiune.....	38
4.6. SBM îmbunătățit.....	39
4.7. Autocodificator LSTM.....	40
Capitolul 5. Rezultate teoretice și Interpretări.....	43
5.1. Rezultate inițiale.....	43
5.2. Evaluarea performanței pipeliine-ului.....	44
5.3. Evaluarea performanțelor autocodificatorului.....	46
5.4. Evaluarea performanței autocodificatoarelor LSTM.....	50
5.5. Evaluarea SBM-ului îmbunătățit.....	54
Capitolul 6. Concluzii.....	56
6.1. Obiectivul.....	56
6.2. Contribuții.....	56
6.3. Rezultate Obținute.....	56
6.4. Dezvoltări Ulterioare.....	56
Bibliografie.....	58
Anexa 1 – Informații Relevante.....	60
Anexa 2.....	65

Capitolul 1. Introducere

1.1. Contextul proiectului

1.1.1. Neuronul

Neuronul este celula specifică sistemului nervos. Neuronii sunt celule excitabile din punct de vedere electric care comunică între ele prin conexiuni numite sinapse. Din puncte de vedere structural, neuronul este compus din soma, axon și dendrite. Informația este recepționată și procesată în soma și dendrite, în această zonă se face contactul prin sinapse cu alți neuroni. Axonul face legatura de la corpul unei celule la dendritele altor celule formând o sinapsă. Schimbarea de potențial este făcută de-a lungul axonului. Astfel, un număr mare de neuroni interconectați sunt capabili să comunice la viteze foarte mari prin descărcări electrice, denumite potențiale de acțiune.

O descărcare electrică transmite informație în sistemul nervos. În general, descărcările electrice ale unui neuron sunt similare din punct de vedere a duratei, amplitudinii și a frecvenței. Aceste părți ale unei descărcări electrice reprezintă codificarea informației transmise.

Neuronul în totalitatea sa este acoperit de o membrană. Această membrană conține căi pentru a permite trecerea ionilor în interiorul sau în afara celulei. În interiorul celulei există o concentrație mai mare de ioni de potasiu, în timp ce în exterior de sodiu. Spațiul extracelular este încărcat mai pozitiv decât cel intracelular, creând astfel o diferență de potențial. Astfel cand informația este transmisă prin sinapsă, canalele pentru ionii de sodiu sunt deschise, rezultând o schimbare de potențial. Odată ce un anumit prag este atins, se deschid mai multe canale de sodiu ceea ce duce la o schimbare și mai mare de potențial. Acest proces se numește depolarizare și se întâmplă într-un interval foarte scurt de timp. Procesul este ilustrat în figura 1.1, se poate observa faza de creștere (rising phase) care duce la depășirea superioară (overshoot) odată ce potențialul trece de 0mV. [Bear2007]

După închiderea canalelor pentru sodiu, începe deschiderea celor de potasiu, procesul de repolarizare. Multitudinea de ioni de potasiu cauzează o scădere bruscă a potențialului membranei sub valorile inițiale, care după inchiderea canalelor de potasiu este adusă la normal. De asemenea, și acest lucru se poate observa în figura 1.1, faza de descreștere (falling phase) care duce la depășirea inferioară (undershoot), urmată de revenirea la starea normală. [Bear2007]

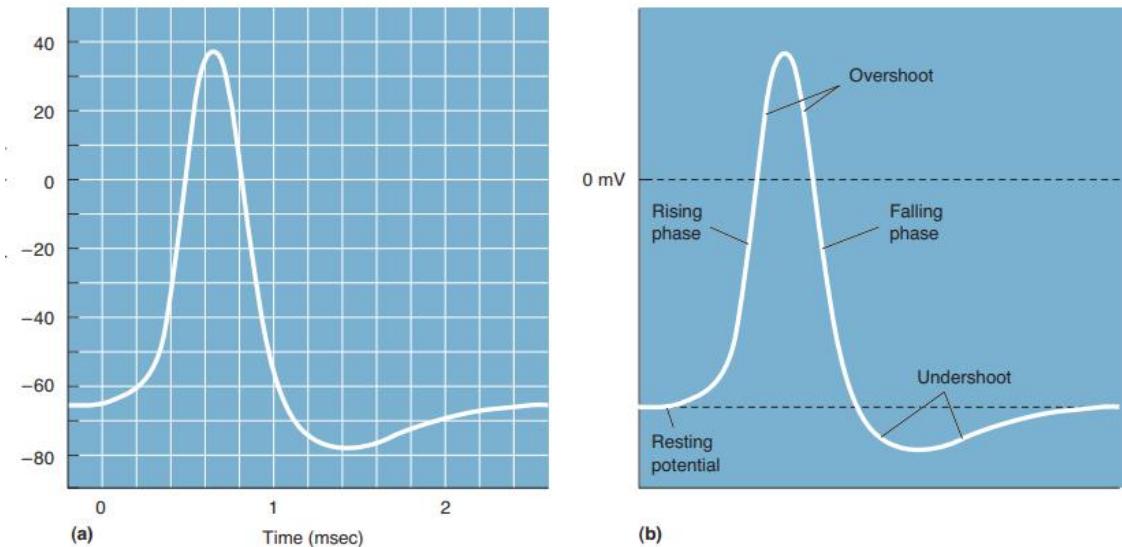


Figure 1.1 – O descărcare electrică (a) - reprezintă o descărcare electrică afişată de un osciloscop (b) – reprezintă părțile unei descărcări electrice [Bear2007]

1.1.2. Neuroștiința

Neuroștiința este ramura care se ocupă de studiul creierului și a sistemului nervos. Obiectivul acestei științe este de a înțelege modul de funcționare a creierului prin diferite metode și abordări. Neuroștiința combina mai multe ramuri precum, fiziologie, biologie moleculară, psihologie, modelare matematică. Abordarea acestui proiect asupra domeniului este din punctul de vedere al Tehnologiei Informației și ramura numește Neuroștiința Computațională.

Pentru a studia creierul, oamenii de știință își concentrează atenția asupra celulei care reprezintă cea mai mare parte a functionalitatii creierului, neuronul. Una din metodele prin care se poate studia comportamentul neuronului, este gruparea potențialelor.

1.1.3. Gruparea Potențialelor

În literatură, gruparea potențialelor este procesul prin care din înregistrări neuronale se ajunge la identificarea neuronului care a produs fiecare potențial de acțiune. Scopul procesului este de a ajunge la o formă în care potențialele de acțiune unui neuron sunt grupați între ei și separați de potențialele altor neuroni. În general, potențialele de acțiune ale aceluiași neuron sunt similară, astfel o grupare ar trebui să reprezinte toate potențialele unui neuron. Pașii procesului sunt ilustrați în figura 1.1. [Quiroga2007]

Înregistrarea datelor se face prin electrozi introdusi în creierul subiecților. Pentru a face acest lucru există 2 metode: înregistrare intracelulară și extracelulară. În înregistrările intracelulare, electrodul este introdus într-un singur neuron, și sunt înregistrate doar descărcările electrice ale aceluiași neuron. În înregistrările extracelulare, electrodul este introdus în spațiul dintre neuroni, astfel descărcările electrice înregistrate aparțin mai multor neuroni.

Deoarece acest proces este dificil în general înregistrările se fac extracelular. Din acest motiv, înregistrarea conține combinații de semnale a mai multor neuroni din

vecinătatea vârfului electrodului. Astfel, în înregistrare din cauza distanței față de electrod, unele din descărurile electrice ale neuronilor care vor fi înregistrate ca și zgomot, care va trebui extras. În procesul clasic ilustrat în figura 1.2, înregistrarea sau datele brute sunt datele de intrare iar identificarea apartenenței descărărilor electrice la un neuron datele de ieșire.

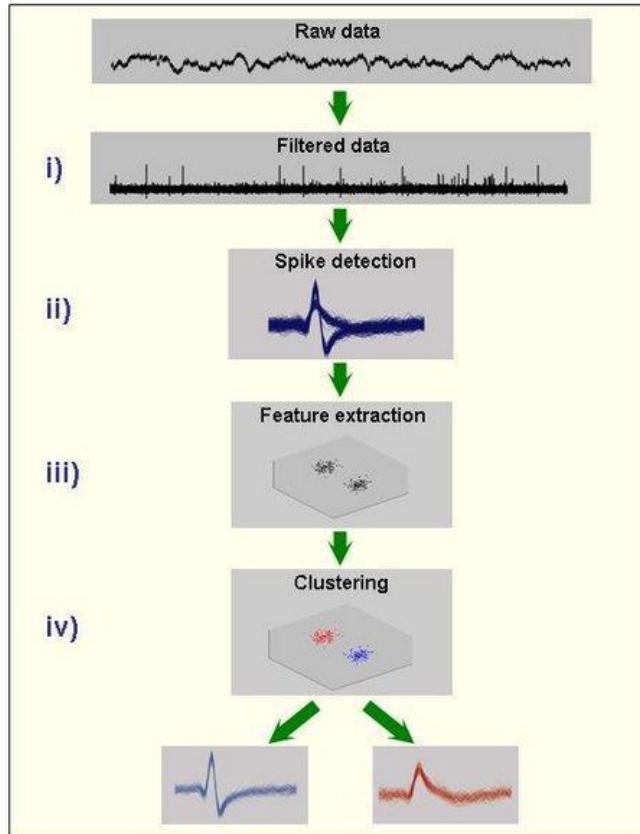


Figure 1.2 – Cei 4 pași ai Grupării Potențialelor (i) Filtrarea înregistrării (ii) Detectarea potențialelor de acțiune din datele filtrate (iii) Extragerea trăsăturilor relevante (iv) Gruparea [Quiroga2007]

Primul pas al procesului este filtrarea datelor inițiale. Acest pas se face prin folosirea unui filtru de tip bandpass. Filtrul bandpass permite trecerea semnalelor care se află între 2 frecvențe specifice în timp ce ignoră alte frecvențe. În acest pas, prin filtrare se păstrează majoritatea potențialelor de acțiune și se îndepărtează zgomotul.

În al doilea pas se face detectarea potențialelor de acțiune. Pentru acest pas, în general, se folosește un prag al amplitudinii. Pentru valoarea pragului trebuie găsit un compromis deoarece o valoare prea mare duce la pierderea potențialelor cu amplitudine redusă iar o valoare prea mică duce la identificarea zgomotului ca potențial de acțiune. [Rey2015] Valoarea pragului determină amplitudinile mai mari sa fie considerate potențiale.

$$\text{Prag} = 5\sigma_n \quad \sigma_n = \text{median}\{|x|\} / 0.6745$$

Tehnologia informației poate să aducă îmbunătățiri următorilor 2 pași. Al treilea pas, Extragerea Trăsăturilor, constă în transformarea unei forme de undă într-un set de caracteristici cât mai mic, care păstrează destulă informație. Inițial, în forma de undă, o

descărcare electrică este reprezentat ca un set de valori într-un interval de timp, echivalent cu un vector. Din acest set de puncte, nu toate sunt la fel de importante în identificarea apartenenței unui potențial de acțiune. Astfel, extragerea trăsăturilor încearcă să determine care sunt cele mai importante sau să creeze trăsături care păstrează informația cea mai importantă, cea care distinge o descărcare electrică a unui neuron de cea a altui neuron. Un alt fapt de luat în considerare este că majoritatea algoritmilor de grupare au dificultăți cu datele cu multe dimensiuni. Din acest motiv se folosesc algoritmi ca PCA [Mishra2017] pentru a reduce dimensionalitatea, rezultând într-un set de date cu 2-3 dimensiuni. Acest pas ajută și la vizualizarea rezultatelor.

Pasul final este cel de grupare. Datele de intrare pentru grupare sunt punctele rezultate din pasul anterior. Gruparea se poate face și manual de către un observator, dar se preferă folosirea unui algoritm de machine learning. Scopul grupării este de a separa punctele în grupări bazate pe similaritate. Rezultatul grupării este încă o caracteristică, numită eticheta sau “label” – în general un număr, care reprezintă apartenența unui punct la o grupare care este identificată prin acest număr.

Deoarece Gruparea face parte din algoritmii unsupervised, acuratețea este greu de identificat neavând etichetarea reală (*ground truth*) pentru a compara. Din acest motiv, acuratețea este calculată prin metrii care iau în considerare diferite aspecte precum forma grupului, distanțele dintre punctele din același grup și distanțele dintre grupuri.

1.2. Conținutul lucrării

Restul lucrării este organizat în modul următor:

- Capitolul 2 conține descrierea detaliată a temei de cercetare și a obiectivelor acesteia
- Capitolul 3 prezintă stadiul actual a domeniului și o prezentare a cercetărilor similare
- Capitolul 4 cuprinde prezentarea conceptelor teoretice și analiza diferențelor metode folosite de-a lungul dezvoltării proiectului
- Capitolul 5 prezintă metodele teoretice și practice și analiza evaluării performanței
- Capitolul 6 cuprinde analiza critică a rezultatelor obținute și descrierea posibilelor dezvoltări ulterioare

1.3. Colaborări

Proiectul a fost dezvoltat pe parcursul a doi ani și este unul de ampoloare. Persoanele care au lucrat pe acest proiect sunt Andreea Maria Gui, Diana-Georgeta Lazea și Alexandru Hristache în anul academic 2019/2020 și Vlad Negru și Roxana Aldea în 2020/2021. Fiecare grup a lucrat pe propria ramură a proiectului, fiecare individ fiind axat pe dezvoltarea unei subramuri. Ramura prezentată în acest document este dezvoltată împreună cu Coporile Andreea. În decursul proiectului s-a colaborat cu Transylvanian Institute of Neuroscience (TINS) cu îndrumarea Dr. Eng. Raul Muresan și Dr. Eng. Vasile V. Moca.

Capitolul 2. Obiectivele cercetării

2.1. Scopul lucrării

Acest proiect este continuarea proiectului de licență. [Ardelean2019] Proiectul de licență s-a concentrat asupra ultimului pas al grupării potențialelor, și anume Gruparea. A fost dezvoltat un algoritm de Grupare care să rezolve dificultățile datelor neuronale. Limitările observate sunt datorate pașilor anteriori grupării în pipeline-ul grupării potențialelor, din acest motiv acest proiect se axează pe extragerea trăsăturilor.

Extragerea trăsăturilor un pas al grupării potențialelor. Acest pas produce setul de caracteristici pe care se aplică Gruparea, astfel fiind o influență puternică a calității rezultatelor. Lipsa unui set calitativ de caracteristici nu poate fi compensată de un algoritm cu o performanță ridicată. În acest pas se procesează formele de undă a potențialelor de acțiune pentru a găsi sau de a crea trăsăturile cele mai informative și discriminante.

2.2. Obiective specifice

Obiectivele specifice ale proiectului pot fi împărțite în următoarele categorii:

1. Cercetarea domeniului
 - Înțelegerea caracteristicilor potențialelor de acțiune și cum se produc
 - Înțelegerea posibilelor diferențe dintre potențiale de acțiune de la neuroni diferiți
 - Studierea literaturii domeniului Grupării Potențialelor, în special partea de extragere a trăsăturilor
 - Analiza metodelor de extragere a trăsăturilor existente:
 - Tehnici statistice
 - Tehnici bazate pe Învățare automată
 - Tehnici bazate pe Rețele Neuronale
2. Proiectarea metodelor, evaluarea și interpretarea rezultatelor
 - Crearea unor caracteristici care să ofere separabilitate între grupuri
 - Metode bazate pe derive
 - Aplicarea și crearea unor metode de Extragere a Trăsăturilor și validarea rezultatelor
 - PCA
 - Discrete Wavelet Transform
 - Fourier Transform
 - Hilbert Transform
 - Autoencodere
 - Pipeline
 - Dezvoltarea și validarea a diferitelor abordări pentru Autocodificatoare
 - Autocodificatoare simple
 - Autocodificatoare preantrenate
 - Autocodificatoare expandante

- Autocodificatoare cu Fourier Transform
 - Blackman Window
 - Gaussian Window
 - DPSS
 - Ansamblu de Autocodificatoare cu Fourier Transform
 - Autocodificatoare cascadate
- Autocodificatoare cu LSTM

3. Colaborarea în echipă

- Atribuirea sarcinilor, organizarea muncii individuale în ramuri folosind Github
- Prezentarea rezultatelor muncii la intervale de două săptămâni
- Documentarea rezultatelor

Capitolul 3. Studiu Bibliografic și Stadiul actual în domeniu

3.1. Bazele Grupării Potențialelor

Primul pas în acest domeniu este familiarizarea cu conceptele de bază ale Neuroștiinței și mai ales a Grupării Potențialelor. Înțelegerea conceptelor de bază a Grupării Potențialelor, pașii și metodele de funcționare împreună cu înțelegerea și vizualizarea funcționării neuronului au fost înțelese din multiple surse. [Bear2007] [Quiroga2007]

Gruparea Potențialelor și implementările sale din ultimii ani sunt prezentate de Quiroga în articolul său [Quiroga2015]. În acest articol sunt prezențați pașii de funcționare a Grupării Potențialelor, moduri de dobândire a datelor, avantajele, dezavantajele și provocările întâlnite în acest domeniu luând în considerare hardware-ul și algoritmii existenți. Astfel, am luat hotărârea de a interveni în pașii Grupării Potențialelor cu verificări empirice pentru a ne asigura că rezultatele găsite sunt cele căutate.

3.2. Algoritmi de Grupare

Primul pas al lucrării a fost utilizarea algoritmului Principal Component Analysis (PCA) [Mishra2017] asupra formei de undă pentru a reduce dimensionalitatea spațiului, ca Extragere a Trăsăturilor. Urmat de pasul de grupare folosind algoritmii K-Means [MacQueen1967], DBSCAN [Ester1996] și SBM [Ardelean2019].

3.2.1. *K-Means*

K-Means partătionează setul de date în K grupuri (K este un parametru). Fiecare punct din set primește eticheta grupului de care a fost anexat. Un punct face parte din grupul care are cel mai apropiat centru.

Algoritmul primește ca parametru K numărul de grupuri de identificat. Inițializează K puncte alese în mod aleator ca centroizii grupurilor. Folosind distanța Euclidiană atribuie punctele setului de date celui mai apropiat centroid. După ce toate punctele din setul de date au fost atribuite unui grup, se calculează media pătratică a distanțelor într-un grup și punctul care are media minimă devine noul centroid al grupului. După repetate iterații ale acestor pași se ajunge la punct de convergență. Convergența, în cel mai simplu caz, înseamnă ca între 2 iteratii consecutive nu se schimbă eticheta nici unui punct.

Unul dintre dezavantajele algoritmului este faptul ca trebuie estimat numărul de grupuri înainte de a folosi algoritmul. Fiind bazat pe centroizi, K-Means are de asemenea probleme cu identificarea grupurilor cu o formă aleatoare.

Pentru versiunea folosită a algoritmului, complexitatea medie este $\Theta(n k d I)$, unde n reprezintă mărimea setului de date, k reprezintă numărul de grupuri, d numărul de dimensiuni a setului de date și I numărul de iterații.

3.2.2. DBSCAN

Density Based Spatial Clustering of Applications with Noise (DBSCAN) este un algoritm bazat pe densitate care consideră punctele într-o zona de densitate ridicată ca fiind într-un grup, iar punctele care se află în zone de densitate joasă ca fiind zgomot (noise). Este capabil de a identifica grupuri cu forme aleatoare.

DBSCAN folosește 2 parametrii: eps și minPts . Folosind acești 2 parametrii decide dacă un punct aparține unui grup sau este zgomot. Algoritmul se folosește de distanțele dintre puncte și de un număr minim de puncte dintr-o vecinătate pentru a lua aceste decizii.

Eps este parametrul care reprezintă distanța dintre puncte, din punct de vedere empiric poate fi comparat cu raza. Acest parametru este foarte sensibil, trebuie ales destul de mare pentru a nu face supragrupare (*overclustering*), dar destul de mic încât să separe grupuri diferite. MinPts reprezintă numărul minim de puncte care trebuie să fie în aceeași vecinătate ca aceasta vecinătate să fie considerată grup, în cazul în care un punct nu face parte din nicio vecinătate este considerat zgomot.

DBSCAN nu are nevoie de numărul de grupuri ca date de intrare (*input*) și este capabil să identifice grupuri de forme aleatoare. Dar, are dificultăți cu seturi de date care au grupuri cu densități diferite. Acest fapt se datorează modului de funcționare a algoritmului. Un grup cu densitate mică va fi identificat ca zgomot de către DBSCAN.

Performanța algoritmului variază după modul de selecție a vecinilor și variază între $\Theta(n \log n)$ și $\Theta(n^2)$. Algoritmul nu este deterministic, dar acest lucru se observă foarte rar la punctele de margine (border points), aceste cazuri fiind excepții.

3.2.3. SBM

Space Breakdown Method (SBM) este un algoritm de grupare care se bazează pe prezumția de distribuție gaussiană a grupurilor. Distribuția gaussiană a grupurilor reprezintă faptul că densitatea cea mai ridicată se va afla în centrul grupului iar îndepărțarea de acest centru rezulta în scăderea densității. SBM este diferit de algoritmii prezențați anteriori prin faptul că folosește un grilaj (*grid*) pentru a reduce spațiul, ținând cont de densitățile diferitelor zone ale setului de date.

Algoritmul folosește 2 parametri: numărul de partitioare (PN) și pragul minim al unui centroid. Numărul de partitioare reprezintă numărul de coșuri (“*chunk*”-uri) în care sunt împărțite datele pentru o dimensiune. Pragul minim al unui centroid este valoarea minimă necesară pentru un chunk pentru a putea fi considerat un centroid al unui grup, chiar dacă conține valori mai mari decât toți vecinii. Algoritmul începe prin a normaliza setul de date în intervalul [0, PN]. Pentru un set de date cu 2 dimensiuni, se creează o matrice de dimensiuni PNxPN în care fiecare celulă reprezintă numărul de puncte în intervalul corespondent în setul de date. De exemplu celula 0x0 din matrice conține numărul de puncte în intervalul [0, 1] pe axa X și Y. Un centru al unui grup este reprezentat de faptul că toate celulele vecine din matrice au valori mai mici. Pentru a elimina cazul în care zgomotul ajunge să fie considerat un posibil centru de grup s-a integrat pragul minim al unui centroid. Acest prag reprezintă numărul minim de puncte necesar al unei celule pentru a putea fi considerat un centru. Pașii explicați pot fi extrapolati pentru n dimensiuni.

Precum DBSCAN algoritmul nu are nevoie de numărul de grupuri, complexitatea crește liniar cu mărimea setului de date, dar exponențial cu numărul de dimensiuni. Un pas de extragerea a trăsăturilor adecvat aduce setul de caracteristici la un număr redus de dimensiuni ceea ce ar îmbunătăți complexitatea. Este limitat în identificarea grupurilor datorită cerintei de distribuție gaussiana. Este capabil de a identifica grupuri suprapuse cu densități diferite.

Acest algoritm constă în 5 pași: normalizare, împărțirea în bucăți (*chunkification*), căutarea centroizilor, expansiunea centroizilor și adunarea bucăților (*dechunkification*). Pașii de normalizare, chunkification și dechunkification au o complexitate de $\Theta(n)$, iar pașii de căutare și expansiune a centroizilor de $\Theta(PN^N)$, unde n reprezintă mărimea setului de date, PN numărul de partitionare și N numărul de dimensiuni. Astfel, complexitatea finală este de $\Theta(n+PN^N)$.

3.3. Evaluarea performanței

Datorită faptului că datele folosite sunt sintetice ele conțin și etichetarea reală a fiecărui punct. Știind asta, putem să evaluăm acuratețea algoritmilor din ambele puncte de vedere, supervizat și nesupervizat. Informațiile prezentate în următoarele pagini despre metrii sunt sintetizate din următoarele resurse: [Deborah2010], [Pedregosa2011], [Santos2009], [Vinh2009], [Rendon2011].

3.3.1. Evaluarea performanței folosind metrii externe

a. Adjusted Rand Index (ARI)

ARI este o metrică bazată pe numărarea perechilor de puncte. Se verifică dacă punctele din același grup au aceeași etichetă (*agreement*) sau etichete diferite (*disagreement*) între rezultatul grupării și etichetarea reală (*ground truth*). Această metrică se folosește de relația dintre puncte. Din acest motiv, metrica penalizează overclustering-ul.

Avantajele folosirii acestei metrii sunt faptul ca se poate identifica atribuirea aleatoare a etichetelor prin valori apropiate de 0. Valorile rezultate sunt încadrate în intervalul [-1, 1] cu valori negative fiind atribuirii independente, iar valori apropiate de 1 fiind grupări corecte.

ARI este o îmbunătățire adusă Rand Index-ului, care nu garantează ca asignarea aleatoare a etichetelor rezulta într-o valoare apropiată de 0.

Formulele matematice ale metrii:

$$RI = \frac{\text{agreements}}{\text{agreements} + \text{disagreements}}$$

$$ARI = \frac{RI - \text{ExpectedRI}}{\text{MaxRI} - \text{ExpectedRI}}$$

b. Adjusted Mutual Information (AMI)

AMI este o metrică bazată pe teoria informației și pe masurarea entropiei. Mutual information-ul a 2 grupuri U și V poate fi calculat folosind formulele următoare:

$$MI(U, V) = \sum_{i=0}^{|U|} \sum_{j=0}^{|V|} \frac{|U_i \cap V_j|}{N} \log \frac{N|U_i \cap V_j|}{|U_i| |V_j|}$$

$$AMI = \frac{MI(U,V) - E(MI(U,V))}{average(H(U),H(V)) - E(MI(U,V))}$$

Unde, N este numărul de puncte din setul de date, |X| numărul de puncte în grupul X, iar H(X) entropia asociată grupului X.

AMI este o îmbunătățire adusă Mutual Information-ului, care nu garanta ca asignarea aleatoare a etichete rezulta într-o valoarea apropiata de 0.

c. Purity

Purity este o metrică de evaluare externă a calității grupării. Se calculează procentul de puncte clasificate corect. Prin corect, ne referim la faptul ca fiecare grup identificat apare și în etichetarea reală. Folosind etichetarea reală a punctelor putem găsi corectitudinea atribuirilor, cu toate acestea etichetele din etichetarea reală și din rezultatul grupării trebuie mapate. Din acest motiv, formula purity-ului folosește maximul pentru a găsi cel mai bun grup de mapat, cel care are cel mai multe puncte din grupul din etichetarea reală.

Rezultatul aparține intervalului [0, 1]. Această metrică are avantajul de a nu penaliza overclustering-ul. Un dezavantaj este faptul că pentru o grupare în care fiecare punct este un grup se evaluează cu scor maxim, de asemenea grupurile foarte nebalanșate dau un scor ridicat.

Formula purity-ului este:

$$Purity = \frac{1}{N} \sum_{i=1}^k \max |C_i \cap L|$$

Unde N reprezintă mărimea setului de date, k numărul de grup rezultate din Grupare, C un grup iar L numărul maxim de puncte pentru grupul C.

d. Homogeneity, Completeness si V-Measure

Aceste metriki sunt legate între ele, V-Measure fiind media armonică între celelalte două. Un grup este considerat omogen când toate punctele dintr-un grup fac parte din aceeași clasă. Prin inversarea etichetelor, cele rezultate din grupare și etichetare reală, se obține Completeness-ul. Completeness-ul este realizat când toate punctele dintr-o clasă fac parte din același grup.

V-Measure este echivalent cu Normalized Mutual Information. Această metrică este simetrică, astfel poate fi folosită și pentru a interpreta acordul dintre rezultatele a două metode de Grupare. Formula este următoarea:

$$V - Measure = \frac{(1 + \beta) * (Homogeneity * Completeness)}{(\beta * Homogeneity + Completeness)}$$

Homogeneity, Completeness și V-Measure rezultă în valori în intervalul [0, 1], dorite fiind valorile mai ridicate.

3.3.2. Evaluarea performanței folosind metriki interne

a. Silhouette

Coefficientul Silhouette evaluează grupurile după cât de bine sunt definite. Un scor mai mare înseamnă că grupul este mai bine definit.

Avantajul acestei metriki este faptul că scorul se găsește în intervalul [-1, 1], unde -1 reprezintă o grupare incorectă, 0 reprezintă o grupare suprapusă iar 1 o grupare foarte

densă. Un grup dens și bine separat reprezintă conceptul standard de grup și rezultă într-un scor mai mare pentru aceasta metrică. Scorul tinde să fie mai mare pentru grupuri convexe decât alte concepte.

$$s = \frac{b - a}{\max(a, b)}$$

Unde, a este distanța medie între un punct și restul punctelor din grup, iar b distanța medie dintre punct și toate punctele din cel mai apropiat grup diferit.

b. Davies-Bouldin

Indexul Davies-Bouldin reprezintă similaritatea medie între grupuri, unde similaritatea este măsura care compara distanțe între grupuri și mărimea lor. Valoare minimă a indexului este 0, cu cât este mai apropiat de 0 cu atâta există o separare mai bună.

Acest index este mai ușor de calculat decât Silhouette dar are valori mai mari pentru grupuri convexe și este limitat doar pentru distanța euclidiană.

Acest index poate fi definit ca similaritatea medie dintre 2 grupuri, acest fapt se poate observa și în formule:

$$R_{ij} = \frac{s_i - s_j}{d_{ij}}$$

$$DB = \frac{1}{k} \sum_{i=1}^k \max R_{ij}$$

Unde, s reprezintă distanța medie între fiecare punct din grupul i și centroidul grupului, iar d reprezintă distanța între centroidele grupurilor i și j .

c. Calinski-Harabasz

Cunoscut și sub numele de Variance Ratio Criterion, pentru acest index un rezultat mai mare reprezintă grupuri definite mai bine. Formula indexului este rația dintre dispersia intragrup și intergrup pentru toate grupurile. Unde dispersia este suma distanțelor ridicate la pătrat.

Avantajul acestui index este faptul că un scor mare indică faptul că grupul în cauză se apropie de conceptul standard al unui grup, este dens și separat. Alt avantaj este că este rapid de calculat. La fel ca și la Davies-Bouldin are valori mai mari pentru grupuri convexe.

Formula indexului este următoarea:

$$s = \frac{\text{tr}(B_k)}{\text{tr}(W_k)} \frac{n - k}{k - 1}$$

Unde, n reprezintă mărimea setului de date, k numărul de grupuri identificat, iar B și W sunt urma dispersiei intergrup și respectiv intragrup.

3.4. Metode actuale de Extragere a Trăsăturilor

3.4.1. Principal Component Analysis

Metoda standard de extragere a trăsăturilor este Principal Component Analysis (PCA) [Mishra2017]. PCA este cea mai folosită metodă de reducere a dimensionalității prin transformarea a unui set mare de caracteristici într-un set considerabil mai mic cu o pierdere minimă de informație. În mod evident, aceasta reducere are ca preț, acuratețea. Dar, în Gruparea Potențialelor, este nevoie de acest compromis între acuratețe și simplitate, deoarece reducerea dimensionalitatii permite vizualizare și aduce rapiditate în calculele algoritmilor.

Caracteristicile rezultate din PCA sunt numite Principal Components și ele reprezintă combinații liniare ale varianțelor ridicate ale setului original de date. Dezavantajul acestei metode este faptul că se folosește de asumția că varianța aduce separabilitate care nu se încadrează pentru toate seturile de date.

PCA translatează datele pentru a ajunge media în origine și căuta o linie care trece prin origine care aproximează cel mai bine setul de date. Prin rotire se calculează suma pătratelor distanțelor la punctele din setul de date și se alege minimul. Prima linie este prima componentă principală care contine cea mai mare varianță, fiecare componentă găsită va avea o varianță mai mică decât cea anterioară.

Astfel, folosind PCA se pot alege primele două sau trei componente principale pentru a păstra informația (ca varianța) din setul original de date pentru a crea un set nou diminuat.

În cazul nostru, PCA nu aduce reducerea necesară datorită funcționalității lui. PCA căuta dimensiuni noi prin calcularea combinațiilor liniare ale setului original. În cazul în care caracteristicile setului de date nu sunt corelate liniar, PCA nu va ajunge la soluții semnificative.

Actual, în Gruparea Potențialelor, PCA este folosit pentru vizualizarea datelor și pentru reducerea dimensionalității. Potențialele de acțiune setului de date folosit fiind părți ale unui semnal au multe dimensiuni, astfel algoritmii de grupare nu sunt fiabili fără un pas în prealabil. Din acest motiv, s-a încercat folosirea PCA-ului care păstreaza majoritatea varianței datelor, dar s-a observat că nu rezultă suficientă separabilitate din această metodă.

De cele mai multe ori PCA este folosit ca și un algoritm de Reducere a Dimensionalității, dar prin ceea ce face se încadrează și în Extragerea Trăsăturilor. Componentele create de acesta nefiind utile doar pentru vizualizare ci fiind noi trăsături care pot fi folosite pentru grupare.

PCA prin repede iterații produce componente principale care au varianțe din ce în ce mai mici. În cazul în care informația potențialelor de acțiune se poate găsi în varianță, folosind PCA se pot alege primele 2 sau 3 componente principale ca dimensiunile unui nou spațiu. Astfel, se poate asigura o dimensionalitate redusă a setului de date cu o pierdere minimă de informație.

Prin alegerea unui prag minim de varianță pentru componente principale rezultate se poate reduce pierderea de varianță, dar există cazuri în care numărul de componente principale să nu fie cu mult mai mic decât numărul de dimensiuni al setului original de date.

Datorită faptului că diferitele dimensiuni pot avea valori diferite a deviației standard, cele cu valori ridicate pot avea o pondere mai mare în calculul componentelor principale. Pentru a evita această dificultate, setul de date trebuie normalizat înainte de a fi aplicat PCA, mai ales în cazurile în care dimensiunile folosesc unități diferite de măsură.

3.4.2. Metode bazate pe derivate

O metodă alternativă PCA-ului este folosirea derivatelor de gradul 1 și 2. Această metodă are avantajul de a putea diferenția descărcați electrice similare de la neuroni diferiți. Prin derivare se scot în evidență frecvențele înalte și se ignoră zgomotul, astfel se evidențiază diferențele dintre potențialele de acțiune. [Yang2009] [Paraskevopoulou2013]

Această metodă a avut rezultate mai bune decât PCA în implementările prezentate în resurse. Metoda poate fi folosită pentru a reduce dimensionalitatea setului de date sau pentru Extragerea Trăsăturilor.

3.5. Metode bazate pe procesarea de semnal

3.5.1. Magnitudine și Fază

Un semnal complex, format dintr-o parte reală și una imaginară, pot fi calculate două caracteristici, și anume magnitudinea și fază date de formulele (3.1), respectiv (3.2). Forma de magnitudine și fază poate fi extrasă din multiple forme ale unui număr complex: rectangulară, trigonometrică, polară, etc. [Lyons2008]

Forma de magnitudine și fază oferă posibilitatea de a vizualiza numerele complexe într-un sistem de axe ca vectori: orizontal - real și vertical - imaginar. Lungimea vectorului fiind echivalent cu magnitudinea, iar unghiul față de axa orizontală fiind fază.

$$M = \sqrt{x^2 + y^2} \quad (3.1)$$

$$\varphi = \frac{y}{x} \quad (3.2)$$

3.5.2. Short-Time Fourier Transform

Short-Time Fourier Transform (STFT) este o variație a Fourier Transform-ului (FT). Această metodă este folosită pentru a determina frecvența sinusoidală și fază unui semnal care se schimbă în timp. Avantajul oferit de STFT, care nu era oferit de FT, este faptul că se ia în considerare și timpul în care frecvențele sunt găsite, nu doar frecvențele.

Acest avantaj al STFT-ului apare datorită faptului că semnalul este analizat folosind o fereastră (sliding window). Parametrii cu cel mai mare impact al STFT-ului sunt [Scipy2020]:

- Fereastra (Window) – fereastra prin care se face analizarea semnalului, astfel schimbarea semnalului $x[n]$, la momentul n , se face prin multiplicarea cu $w[n]$, valoarea ferestrei

- Numărul de cadru (Frame number) – notat ca și „l”, este timpul de indexare, se folosește pentru iterarea prin semnal și obținerea segmentelor de timp dorite, această segmentare este cea care oferă precizia în timp și frecvența nevoieă
- Hop-Size – reprezintă numărul de puncte peste care se sare de la un segment de timp la următorul

Rezultatul STFT-ului, spre deosebire de FT care produce un singur spectru Fourier, este o secvență de spectre. Fiecare segment de timp produce propriul spectru, cu valori diferite deoarece analizează părți diferite de semnal, dar fiecare segment de timp are aceeași lungime.

Forma ferestrei este importantă în producerea rezultatului dorit. Pentru semnal, o forma buna este similară cu funcția Gaussiana, pentru spectru trebuie luate în considerare două caracteristici: lobul principal (*main lobe*-ul), definit ca numărul de coșuri dintre 2 trecheri prin zero și lobul cu partea cea mai înaltă (*highest-side lobe*-ul), distanța dintre main lobe și highest-side lobe. Pentru fiecare din aceste caracteristici sunt importante dimensiunile de lățime, respectiv înălțime. În mod natural, fiecare fereastră de analiză trebuie aleasă în funcție de date și de scopul folosirii, luând în considerare compromisurile necesare pentru a găsi rezultatele dorite.

Variații a transformării Fourier împreună cu exemplificările acestora pot fi găsite în lucrarea de licență a Andreea-Maria Gui. [Gui2020]

3.5.3. Superlet Transform

Superlet Transform reușește să rezolve problema Wavelet Transform-ului de a separa grupuri suprapuse. Această problemă apare datorită Principiului de Incertitudine Heisenberg (*Heisenberg Uncertainty Principle*), acest principiu afirma faptul că un semnal nu poate fi măsurat cu precizie mare din punct de vedere a timpului și a frecvenței într-o singura măsurătoare.

Ideea acestei metode este de a combina două cazuri pentru a crește precizia și în frecvență și în timp. Un wavelet mai îngust rezultă într-o precizie mai mare a timpului și mai mică a frecvenței, acest fapt este datorat numărului de cicluri care influențează precizia în timp. Dacă numărul de cicluri este ridicat atunci va crește precizia frecvenței, dar va scădea precizia în timp. [Moca2019]

Prin folosirea mai multor wavelet-uri, fiecare cu propriul număr de cicluri. Acest set de wavelet-uri va conține wavelet-uri cu un număr mic de cicluri pentru precizia în timp și wavelet-uri cu un număr mare de cicluri pentru precizia frecvenței. Numărul de ordin a superlet-ului reprezintă numărul de wavelet-uri pe care îl folosește.

Superlet Transform-ul este definit prin numărul de ordin care determină câte wavelet transform-uri conține, fiecare wavelet cu propriul număr de cicluri. Formula pentru această metodă este (3.3).

$$SL_{f,o} = \{\psi_{f,c} | c = c_1, c_2, \dots, c_o\} \quad (3.3)$$

Numărul de cicluri dintr-un wavelet în superlet poate fi calculat în funcție de numărul de cicluri din primul wavelet prin însumare sau înmulțire. Rezultatul superlet-ului poate fi exprimat ca și media rezultatelor aplicării Continuous Wavelet Transform-

ului de multiple ori cu numărul de cicluri a fiecărui wavelet din setul Superlet Transform-ului.

Variații a algoritmilor de tip wavelet împreună cu exemplificările acestora pot fi găsite în lucrarea de licență a Diana-Georgeta Lazea. [Lazea2020]

3.5.4. Hilbert Transform

Hilbert Transform (HT) este o metodă de analizare a frecvenței și a timpului. Componentele negative ale frecvenței sunt translatate cu $+90^\circ$, iar cele pozitive cu -90° . Spre deosebire de STFT și SLT, HT nu necesită ajustarea parametrilor.

Un semnal analitic reprezintă un semnal cu valori complexe cu componentele negative ale frecvenței egale cu 0. Fiind valori complexe, semnalul analitic conține o parte reală și una imaginară, acestea reprezintă semnalul original și respectiv rezultatul Hilbert Transform-ului.

Această transformare păstrează caracteristica principală a unei descărcări electrice, și anume amplitudinea. Semnalul rezultat având valori mai ridicate decât cel original și este pozitiv în toate punctele semnalului.

HT este o transformare liniară care transformă o funcție $u(t)$ cu valori reale, în funcția $H(u)(t)$ cu valori complexe, definit prin formula (3.4).

$$H(u)(t) = \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{u(x)}{t-x} dx$$

În Gruparea Potențialelor, unde semnalele dintr-o formă de undă sunt valori pozitive, semnalul analitic necesar poate fi obținut prin aplicarea Discrete Fourier Transform asupra formei de undă. Astfel, obținând un semnal analitic acesta poate fi convertit în forma de magnitudine și fază care pot fi folosite ca posibile caracteristici ulterioare.

Multiple variante ale acestui algoritm sunt prezentate și exemplificate de către Alexandru Hristache în lucrarea sa de licență. [Hristache2020]

3.5.5. Fast Fourier Transform

Fast Fourier Transform (FFT) este o variantă mai rapidă de a calcula Discrete Fourier Transform (DFT). Complexitatea DFT-ului este de $O(N^2)$, iar a FFT-ului de $O(N \log N)$. FFT descompune DFT în $\log N$ etape, fiecare constând în $N/2$ calcule. [Heckbert1995]

FFT aduce îmbunătățirea vitezei de procesare doar pentru semnale care conțin semnale de lungimi care sunt puteri ale numărului 2. Există mai multe metode de preprocesare a semnalului înainte de a aplica FFT:

- Potențialul original (lungime 79) - nu aduce îmbunătățire la viteza
- Completare cu zero (*Zero-padded*) (lungime 128) - aceasta preprocesare face magnitudinea semnalului să fie mai netedă [Serra], netezimea este datorată faptului că magnitudinea va avea o lungime mai mare cu valori interpolate, acest fapt se poate vizualiza în figura 3.3.

- Translatarea completării cu zero (*Zero-phase windowing*) (lungime 128) - similar cu zero-padded, dar se face o translatare, astfel încât partea semnalului care conține 0-uri să fie la mijlocul semnalului, creează o simetrie a magnitudinii și aduce netezime fazei, dar devine asimetrică [Serra]

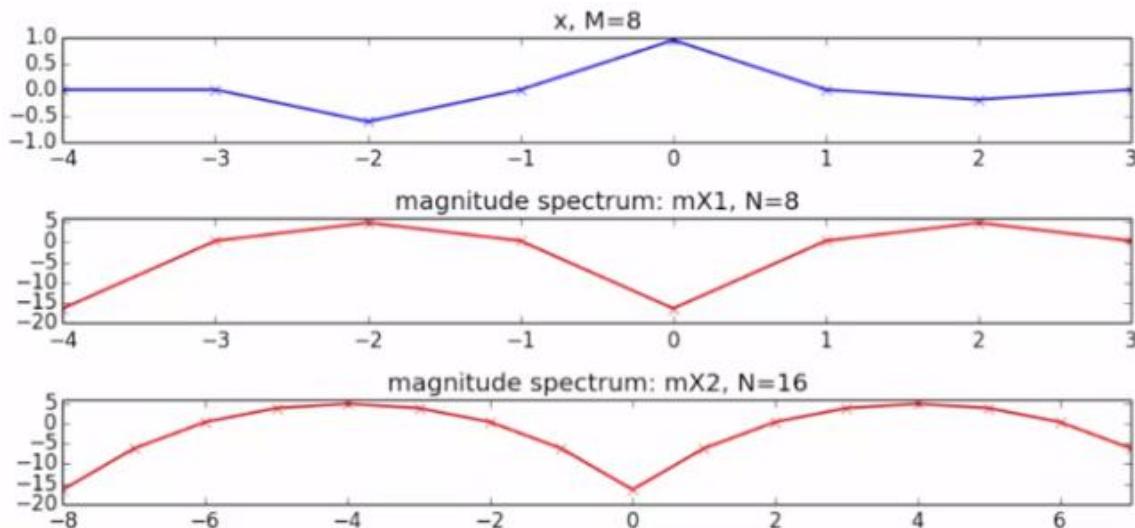


Figure 3.3 – Primul grafic afișează semnalul original, al doilea este magnitudinea rezultată din FFT, iar ultimul grafic reprezintă magnitudinea pentru semnalul zero-padded [Serra]

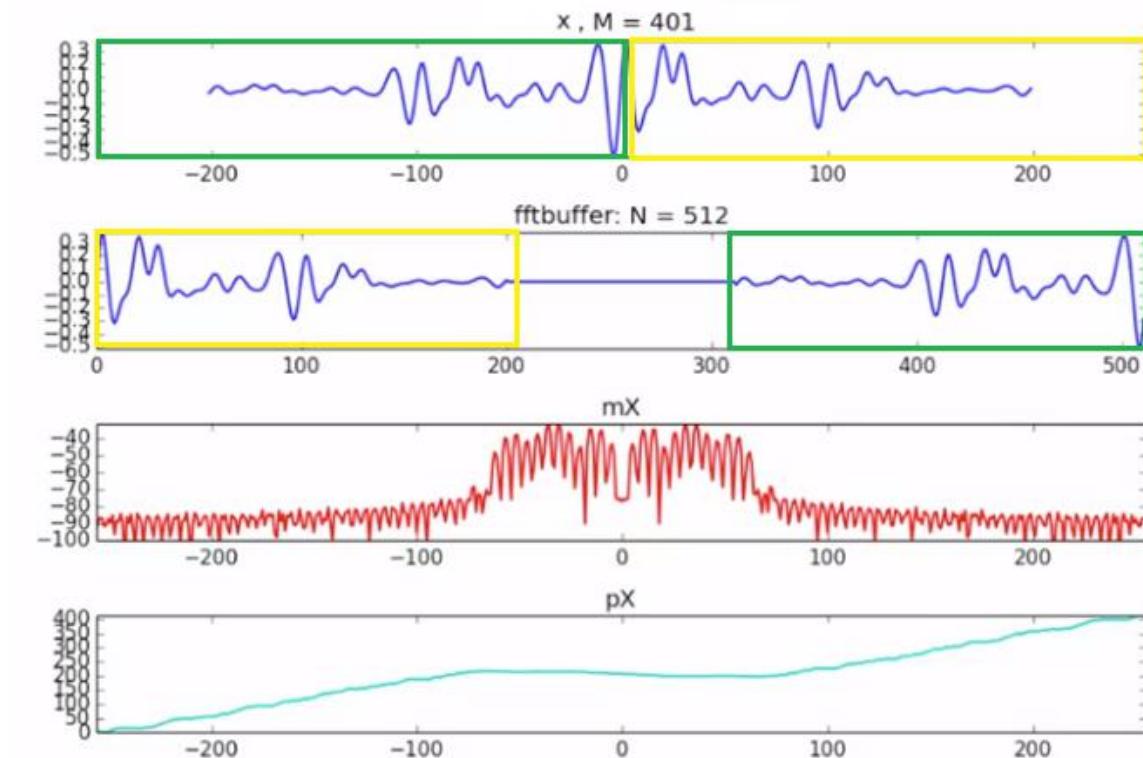


Figure 3.4 – Primul grafic afișează semnalul original, al doilea este semnalul pe care a fost aplicat zero-phasing, ultimele două grafice sunt magnitudinea și respectiv faza rezultată prin aplicarea FFT-ului asupra semnalului zero-phased [Serra]

Un alt tip de preprocesare folosit pentru semnale înainte de aplicarea FFT-ului este aplicarea de window-uri. Există multe tipuri de window-uri, în figura 3.5 sunt afișate window-urile Blackman și Gaussian. Aceste window-uri ajută la minimizarea efectelor FFT-ului asupra părților de semnal care sunt discontinue, acestea apar ca părți de frecvență mare a rezultatului. Aceste window-uri sunt aplicate prin înmulțirea element cu element cu semnalul, prin urmare lungimea window-ului trebuie să fie aceeași cu cea a semnalului.

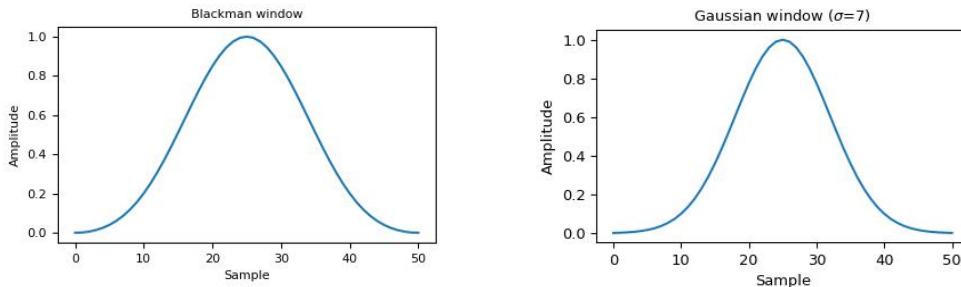


Figure 3.5 – Window-ul Blackman (stanga) și Gaussian (dreapta)

Aceste metode au fost aplicate ca pași de procesare pentru datele de intrare ale autocodificatoarelor în lucrarea de disertatie a [Coporîie2021].

3.5.6. Multitaper: Discrete Prolate Spherical Surfaces (DPSS)

Metoda multitapering-ului depășește unele din limitările ale analizei Fourier conventionale. Multitapering-ul creează mai multe estimări independente. Aceste estimări pot fi folosite ca window-uri și în același fel sunt înmulțite element cu element cu semnalul. Datorită faptului că aceste estimări sunt ortogonale în perechi unele cu altele, semnalele windowed oferă estimări independente ale spectrului. Astfel se amplifică părți diferite ale semnalului pentru fiecare estimare și oferă mai multă informație independentă pentru fiecare semnal în parte. Taper-ul folosit în acest proiect este denumit Discrete Prolate Spheroidal Sequences sau secvențe Slepian.

Această metodă a fost aplicată ca pas de procesare pentru datele de intrare ale autocodificatoarelor în lucrarea de disertatie a [Coporîie2021].

3.6. Metode bazate pe învățare

3.6.1. Autocodificatoare

Autocodificatorul este un tip de rețea neuronală (*neural network*) capabil să învețe automat din date nesupervizate. Această metodă este concepută pentru a produce un date de ieșire (*output*) cât mai similar cu inputul, în general cu același număr de dimensiuni. Autocodificatorul descoperă, prin învățare nesupervizată, structura internă a datelor prin calcularea varianței din input. Reprezentarea creată, fiind informația relevantă din date, poate fi folosită pentru a clasifica datele. [Pinaya2020]

Autocodificatorul poate fi generalizat cu structura $n / p / n$, unde $0 < p < n$, după modelul din figura 3.1. Astfel Autocodificatorul are 3 părți interconectate [Baldi2012]:

- Codificator
- Cod latent
- Decodor

Codificatorul este o rețea neuronală care primește inputul inițial și generează codul. Acest cod este folosit ca input pentru decodor care va genera un output similar cu inputul din codificator. În general, codificatorul și decodorul au straturi oglindite.

Codul generat între codificator și decodor poate fi folosit ca un nou set de caracteristici.

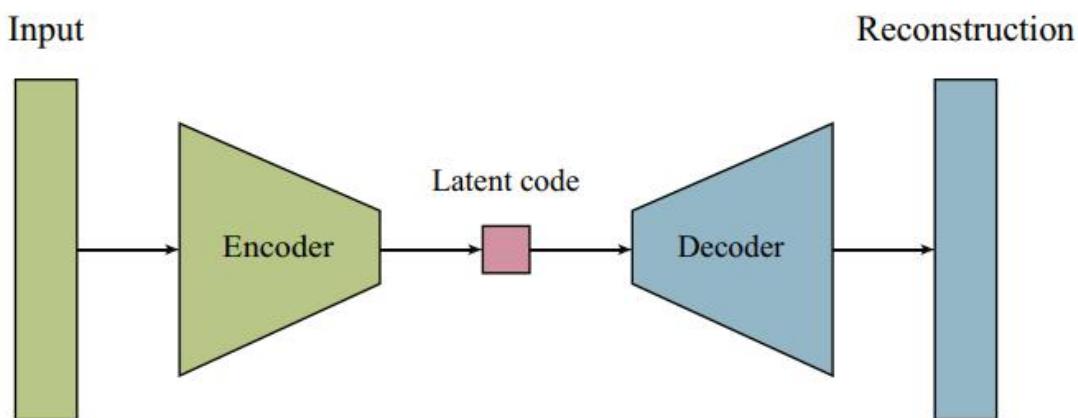


Figure 3.1 - Structura Autocodificatorului de tip $n / p / n$ [Pinaya2020]

Autocodificatorul permite folosirea unei arhitecturi personalizate cu un număr ales de straturi (*layer*). Crescând complexitatea arhitecturii, autocodificatorul va putea să învețe coduri mai complexe. De asemenea, codificatorul și decodorul pot fi rețele neuronale conectate în totalitate sau în cazul în care inputurile sunt imagini poate să conțină straturi convoluționale. [Pinaya2020]

Scopul unui Autocodificator este de a introduce inputul în output, pentru a își realiza scopul se folosește de un loss function, care în general este suma pătratelor erorii (mean squared error) între input și output. Folosind aceasta funcție, autocodificatorul este penalizat cand outputul este diferit de input. [Pinaya2020]

În literatura se găsesc mai multe tipuri de Autocodificator [Pinaya2020]:

- “Subcomplet” (*Undercomplete*) Autocodificator

Pentru a obține informație dintr-un Autocodificator cea mai simplă metodă este de a constrânge codul să fie mai mic decât input-ul. Un astfel de Autocodificator se numește „undercomplete” codul având mai puțini neuroni decât inputul. În cazul în care inputul are toate caracteristicile independente devine o problema pentru Autocodificator, dar dacă există o structură subiacentă, aceasta poate fi descoperită.

Ideea acestui tip de Autocodificator este proiectarea inputului într-un spațiu cu dimensionalitatea redusă. Folosind o funcție liniară de activare, rezultatul autocodificatorului este similar cu PCA. Dar, cum o rețea neuronală este capabilă să învețe relații neliniare, această metodă poate fi considerată o generalizare a PCA-ului.

- Autocodificator Dezgomotificator (*Denoising*)

Pentru antrenarea acestui tip de Autocodificator, inputul este corupt parțial prin adăugarea zgomotului sau prin folosirea unei măști. Modelul este antrenat pentru a reproduce inputul nemodificat. Astfel, diferența dintre metoda anterioară și aceasta se găsește în antrenare și nu în arhitectura.

Pentru a reproduce inputul, autocodificatorul este nevoit să descopere și să învețe relațiile dintre variabilele inputului și să deducă variabilele modificate sau care lipsesc. Pentru inputuri complexe, cum ar fi imaginile, modelul se va baza pe informația găsită dintr-o combinatie de inputuri.

- Autocodificator Risipit (*Sparse*)

Acest tip de Autocodificator poate să învețe o reprezentare utilă chiar și cu un număr mare de neuroni pentru cod. Pentru a realiza asta, modelul este limitat în numărul de neuroni activi, un neuron este activat doar dacă outputul este aproape de 1. Această metodă are rezultate bune chiar și cu un cod mare pentru că autocodificatorul este obligat să reprezinte inputul ca o combinație de puțini neuroni.

Există mai multe abordări pentru a implementa acest tip: L1-norm, KL-divergence.

- Autocodificator Adversarial

Un dezavantaj al metodelor anterioare este faptul că poziționarea reprezentărilor câteodată duce la spații goale, după cum se poate observa în figura 3.2. Autocodificatoarele adversariale sunt o combinație între un Autocodificator general și noțiunea de învățare adversarială.

În literatură, rețelele generativ adversariale sunt 2 modele care concurează între ele pentru a obține rezultate cât mai bune. Unul dintre modele primește input-uri și generează date sintetice care emulează caracteristicile inputului, denumit generator. Iar celălalt model, denumit discriminator, primește inputuri originale și cele generate sintetic și decide cărui grup aparține un input.

Aceste 2 rețele învăță simultan, singura metodă de a obține rezultate mai bune fiind interacțiunea dintre ele. Bazându-se pe feedbackul

discriminatorului, generatorul își ajustează rețeaua pentru a produce outputuri mai dificile pentru discriminator.

Autocodificatoarele aduersariale folosesc învățarea adversarială pentru a aduce distribuția inputului codat cât mai aproape de o distribuție predefinită prin adăugarea unei rețele discriminatoare.

Rețeaua discriminatoare primește valori din distribuția dorită și codul generat din învățare, acestea trebuie să aibă aceleași dimensiuni. În timpul învățării discriminatorul clasifică, decide de care din aceste 2 tipuri aparține inputul primit. Prin adaugarea discriminatorului, partea de codificator trebuie să genereze un cod care decodorul îl poate folosi pentru reconstruire și să genereze cod destul de asemănător cu cel din distribuție pentru a păcăli discriminatorul.

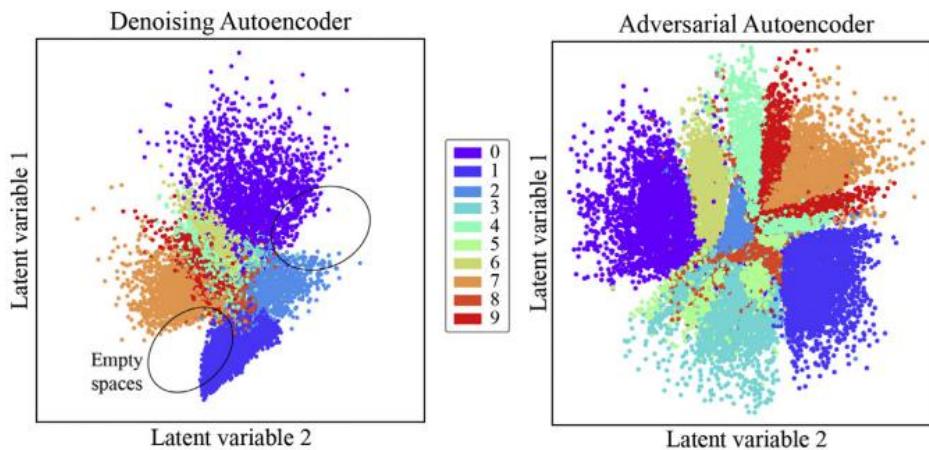


Figure 3.2 – Datorită faptului ca un denoising autocodificator nu primește nicio constrângere, spațiul obținut are goluri, acest lucru nu se întâmplă pentru autocodificatoarele aduersariale [Pinaya2020]

3.6.2. Ensemble

Învățarea de ansamblu (ensemble) este combinarea mai multor modele pentru rezolvarea unei probleme. Cea mai comună aplicare a acestei metode, este combinarea rezultatelor mai multor arbori de decizie (denumit “forrest”). În general, decizia este luată de majoritatea arborilor.

Combinarea mai multor modele poate fi benefică când mai multe modele oferă rezultate nesatisfăcătoare. Pentru sistemele bazate pe votare, este recomandat un număr impar de modele. Printre aceste metode de asamblare sunt următoarele:

- Algebrice - cele mai simple, se aplică o operație matematică asupra tuturor rezultatelor obținute de modele pentru un sample, exemple de operații: sumă, produs, minim, maxim, mediană, sumă ponderată
- Bagging (bootstrap aggregating) - modelele sunt antrenate pe subseturi ale setului inițial. Rezultatele sunt obținute prin mediere sau prin votare. Este recomandat pentru modele complexe deoarece reduce varianta și poate evita overfitting-ul.

- AdaBoost (adaptive boosting) - învățare iterativă pentru a învăța din greșelile modelelor slabe. Este recomandat pentru modele slabe, nu este la fel de predispus către overfitting ca alți algoritmi.
- Stacked Generalization - învățarea este făcută de două nivele de modele, primul nivel învăță folosind datele inițiale. Setul de date este împărțit în $N+1$ subseturi (unde N este numărul de modele din primul nivel). Al doilea învăță din outputurile primului nivel și din al $N+1$ -ulea subset. Motivul celor 2 nivele este ca al doilea să învețe dacă primul nivel nu a reușit.

3.6.3. Long Short-Term Memory (LSTM)

LSTM este o arhitectură de rețea neuronală recurrentă (RNN) care se folosește de învățarea bazată pe gradient. Problema rețelelor recurente, care este rezolvată de LSTM, este lipsa unei memorii de lungă durată. Dacă o secvență este destul de lungă, informația din pașii incipienți este pierdută. Astfel gradientul scade în timp, iar când devine destul de mic nu mai contribuie la învățare. [Hochreiter1997]

Aceasta problemă apare în general la primele straturi din rețea care nu mai primesc actualizări din cauza unui gradient prea mic, din acest motiv primele straturi se opresc din învățare.

Soluția adusă de LSTM se bazează pe porți, un mecanism intern care reglează fluxul de informație. Aceste gate-uri învăță care date să fie păstrate, astfel poate să transmită informația importantă mai departe.

LSTM se folosesc de celule de memorie pentru a transmite informații prin lanțul de secvență. Astfel, chiar și informația din primii pași poate să ajungă în pașii mai târzii. Aceste gate-uri învăță care informații să le păstreze și care să le uite.

Datorită faptului ca rețelele neuronale recurente au o stare internă sunt potrivite pentru date cu serii de timp. Cu toate că LSTM rezolvă problema RNN-urilor, nici acestea nu oferă rezultatele cele mai bune pentru problemele de serii de timp. O arhitectură LSTM nu reușește să reprezinte caracteristicile complexe ale datelor, în special dacă datele din seria de timp sunt pe un interval lung și sunt neliniare. [Sagheer2019]

Capitolul 4. Prezentarea proiectului

4.1. Seturi de date

Un prim set de date folosit a fost sugerat de Transylvanian Institute of Neuroscience (TINS), acest set de date originând din Department of Engineering, University of Leicester, UK. Procesul prin care aceste date sintetice au fost create este descris în articolul [Pedreira2012]. Pentru crearea acestui set s-au folosit înregistrări din neocortexul unei maimuțe, s-au folosit 594 de forme diferite de descărcări electrice, împreună cu etichetarea corespunzătoare. Folosind aceste forme de descărcări electrice au fost create 95 de simulări. Fiecare simulare este un set de date de sine stătător. Simulările au număr de grupuri care variază între 2 și 20, de asemenea fiecare conține și un grup de zgromot.

Articolul menționat prezintă de asemenea diferenți algoritmi de grupare și rezultatele aplicării acestora asupra setului de date. Sunt prezentate limitările algoritmilor de grupare, din 20 de unități diferite s-au putut detecta în caz maxim 10 dintre acestea. Observând acest fapt, proiectul s-a îndreptat înspre pasul de Extragerea Trăsăturilor cu scopul de a extrage caracteristicile potrivite pentru a putea grupa cu o acuratețe mai mare.

Setul de date a fost generat bazându-se pe un set real înregistrat „in vivo” pe o maimuță. Forma de undă are 316 puncte și a fost esantionată la 96 KHz, aceasta este rata de eșantionare originală. Această frecvență a fost ulterior redusă la 24 KHz, astfel folosim 79 de puncte pentru a descrie un potențial de acțiune. [Pedreira2012]

Grupul de zgromot a fost adăugat pentru a crește complexitatea grupării. Un astfel de grup de zgromot conține 20 de forme diferite de potențiale de acțiune. Fiecare din neuronii care au generat aceste forme diferite este estimat la o distanță de 50-140 μm de electrod. Amplitudinea pentru aceste potențiale de acțiune a fost fixată la 0.5. Grupul e noise simulează un neuron cu o rata de descărcare de 5 Hz, fiecare din cele 20 de forme fiind descărcate de un neuron cu o rata de descărcare în medie de 0.25 Hz urmărind o distribuție Poisson. [Pedreira2012]

Restul grupurilor simulează potențialele generate de un neuron estimat la o distanță de 0-50 μm de electrod. Rata de descărcare are o medie între 0.1 și 2 Hz, urmărind o distribuție Poisson. Aceste medii diferite între neuroni generează fenomenul de grupuri imbalansate. Amplitudinile urmăresc o distribuție normală și au fost scalate la valori între 0.9 și 2 pentru a emula date reale. Simulările nu conțin potențiale suprapuse, fiecare potențial având o distanță de minim 0.3 ms de următorul. Adăugarea potențialelor suprapuse ar complica în plus problema. [Pedreira2012]

Fiind un set de date generat, ele conțin și etichete pentru fiecare potențial de acțiune. Acest fapt permite folosirea metricilor externe pentru evaluarea performanțelor.

4.1.1. Structura setului de date

Acest set de date conține 95 de simulări, fiecare din acestea fiind un set independent de date. Fiecare simulare conține un grup de zgromot și un număr variabil de grupuri de semnale de neuroni între 2 și 20. Fiecărei varietăți de număr a grupurilor îi corespund 5 simulări. Astfel pentru 2 grupuri avem 5 simulări, pentru 3 grupuri 5 simulări și în mod analog restul simulărilor.

Fiind un set de date etichetat, pentru simplitate, grupul de zgomot are ca etichetă 0 în toate aceste simulări, pentru graficele următoare acestea vor apărea cu culoarea albă.

Fiecare simulare are un număr aleator de potențiale de acțiune. Un potențial de acțiune reprezintă o schimbare de potențial a unui neuron. Mai multe potențiale de acțiune ale aceluiași neuron reprezintă un grup. Informații despre conținuturile fiecărei simulări pot fi găsite în tabelul 1 din anexă.

4.1.2. Caracteristicile setului de date

Setul de date prezintă provocările specifice datelor neuronale, grupurile conțin segmente suprapuse și sunt nebalansate. Grupurile nebalansate se datorează faptului că neuronii au rate diferite de descărcare, ceea ce rezultă într-un număr diferit de potențiale de acțiune la finalul înregistrării. Suprapunerea apare în cazul în care descărcările electrice ale doi sau mulți neuroni sunt similare și/sau dacă metoda de Extragerea Trăsăturilor folosită nu reușește să separe caracteristicile formei de undă care disting potențialele de acțiune.

Setul de date a fost generat, ceea ce înseamnă că nu este o înregistrare a activității creierului ci au fost create folosind formulele matematice și distribuții statistice. Fiind un set de date sintetic, fiecare descărcare conține și etichetarea reală. Aceste etichete ne oferă răspunsul corect al Grupării Potențialelor, apartenența fiecărui potențial de acțiune la neuronul care l-a produs. Etichetarea reală ne oferă posibilitatea de a folosi metode externe de validare a Grupurilor, acest tip de metode au fost prezentate în Capitolul 3.3. Datele neuronale, în general, nu conțin etichetarea reală, pentru a procura etichetele fiecare neuron ar avea nevoie de un electrod introdus intracelular. Acest fapt nu permite folosirea metodelor de validare externe. Știind acuratețea metodelor folosite pe un set de date cu etichetare reală este util deoarece putem extrapola comportamentul metodelor pe date reale.

4.1.3. Formatul setului de date

Datele sunt împărțite în două tipuri de fișiere de format .mat. Prin urmare, este nevoie de un pas de parsare pentru a accesa datele unei singure simulări. Setul de date conține 95 de fișiere .mat care reprezintă fiecare simulare și un fisier .mat pentru etichetare reală.

Fișierele sunt încărcate ca dicționare, pentru cele 95 de fișiere care conțin semnalele unei simulări, acestea pot fi accesate folosind cheia "data". Semnalul este un vector unidimensional. Fișierul de etichetare reală denumit "ground_truth.mat" conține începutul fiecărui potențial de acțiune și clasa aferentă acestuia. Pentru a accesa începutul se folosește cheia "spike_first_sample", respectiv "spike_classes" pentru clasa potențialului de acțiune. Aceste două chei conțin matrici, astfel trebuie indexate folosind numărul simulării pentru a accesa datele unei simulări.

Știind indexul de început al potențialelor de acțiune al unei simulări și lungimea unui potențial, 79, semnalul din fișierul corespunzător simulării este parcurs. Un potențial este extras ca un vector de mărime 79 începând cu indexul de start și următoarele 78.

4.2. Preprocesarea potențialelor de acțiune

Datorită modului în care setul de date a fost creat a fost nevoie de un pas de preprocesare înainte aplicării metodele de extragere a trăsăturilor. În cazul setului de date prezentat, un potențial de acțiune este format din 79 de puncte. Acestea trebuie selectate astfel încât varful descărcării electrice, punctul maxim al amplitudinii, să fie la același index pentru toate potențialele de acțiune.

Inițial, potențialele de acțiune primesc un punct de start, acest punct împreună cu următoarele 78 formând potențialul de acțiune. Pentru a alinia potențialele de acțiune, punctul de start trebuie să fie translatat astfel încât vârful să ajungă la index prestabilit.

Metoda incipientă aleasă pentru preprocesare este calcularea mediei vârfurilor tuturor potențialelor și translatarea fiecărui punct de start cu diferența dintre aceasta medie și varful acelui potențial de acțiune. Aceasta metoda poate fi vizualizată în figura 4.1 este reprezentat de următoarea formulă:

$$\text{new_start}_{\text{spike}} = \text{old_start}_{\text{spike}} - (\text{index}_{\text{align}} - \text{peak}_{\text{spike}})$$

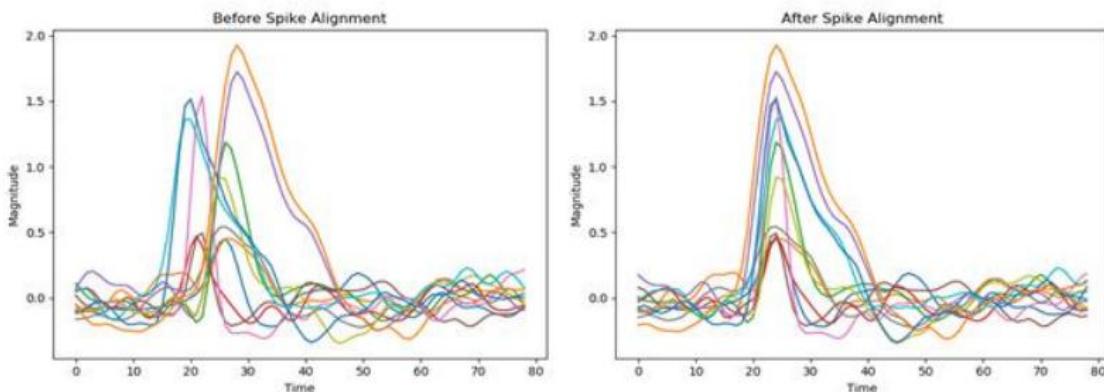


Figure 4.1 - Alinierea potențialelor de acțiune cu media vârfurilor

Termenul “*index*” reprezintă indicele la care vor fi aliniate toate punctele prin translatare. Pentru cazul de aliniere la medie, acesta este chiar media indicilor vârfurilor. Alte posibilități sunt de a da un index prestabilit pentru a avea o parte dorită a tuturor potențialelor la același index. Un exemplu ar fi alinierea amplitudinii la mijlocul semnalului.

Termenul “*peak*” reprezintă indicele care va fi aliniat la “*index*”. Denumirea a fost aleasă în acest mod deoarece în general se vrea alinierea la indicele unui vârf. Pentru alinierea la medie, acest termen ia valoarea indicelui amplitudinii.

Alinierea potențialelor de acțiune aduce un rezultat mai bun al Extragerii Trăsăturilor prin reducerea riscului de overclustering datorită formei nonstandard a grupurilor rezultate. De asemenea, lipsa alinierii poate duce la deformarea unor grupuri cu o distribuție gaussiană. [Quiroga2007]

4.3. Extragerea Trăsăturilor

Pasul de Extragere a Trăsăturilor este un proces de reducere a dimensionalității și este necesar deoarece majoritatea algoritmilor de grupare nu pot finaliza într-un timp acceptabil. De asemenea, în cazul datelor neuronale nu se găsește etichetarea reală, ceea ce înseamnă ca este nevoie și de un pas de vizualizare a rezultatului de către un expert pentru a valida rezultatul. De aceea, este de preferat ca datele să fie aduse într-un spațiu cu 2 sau 3 dimensiuni.

4.3.1. Principal Component Analysis

PCA a fost prezentat din punct de vedere teoretic în Capitolul 3 la Secțiunea de Metode de Extragere a Trăsăturilor. PCA folosit direct asupra punctelor din semnal nu va produce separabilitatea dorită, după cum se poate observa în figura 4.2.

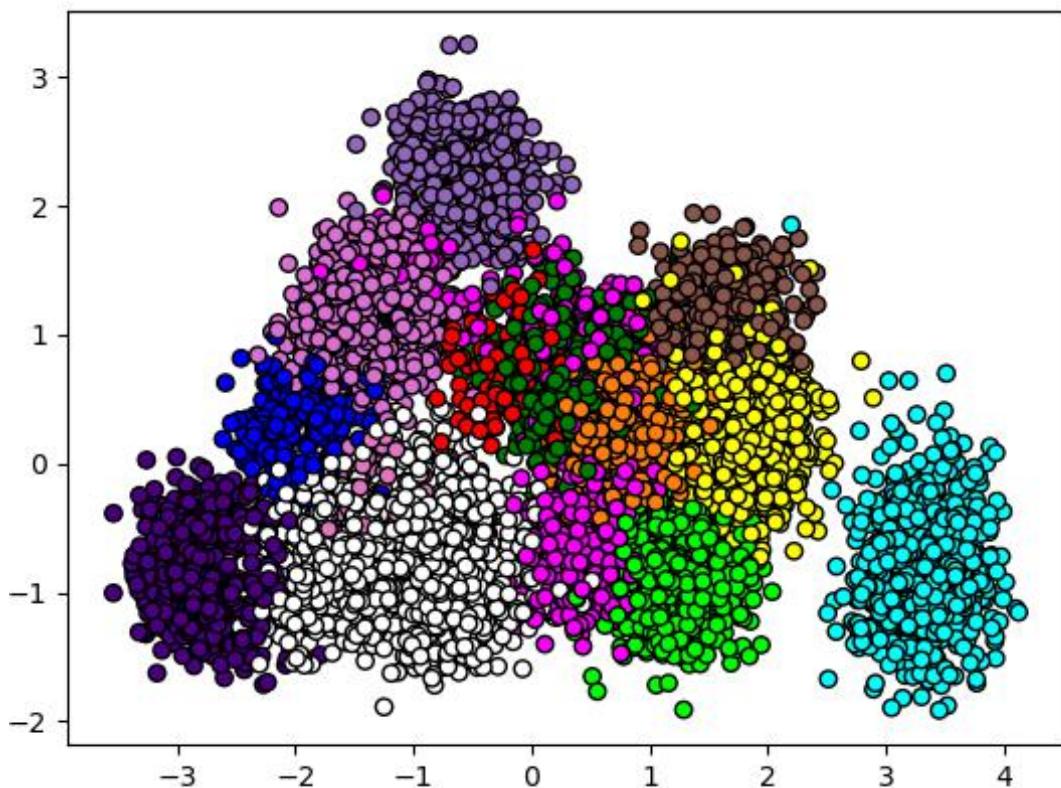


Figure 4.2 – Grupurile simulării 28 (14 la număr) cu etichetarea reală rezultate din PCA în 2 dimensiuni

4.3.2. Extragerea Trăsăturilor folosind derivatele de grad 1 și 2

Această metodă se bazează pe presupunerea că dacă distanța între neuronul X și electrod este mai mică decât neuronul Y și electrod, cu un semnal aliniat și o formă similară, amplitudinea descărcării electrice a neuronului X va fi mai mare. Astfel, se poate realiza separabilitate între potențiale de acțiune similare pentru pasul de Grupare.

Folosind derivatele se pot găsi punctele de extremitate (minimul și maximul) pentru fiecare, aceste valori furnizează informații despre forma potențialului de acțiune și pot fi folosite ca caracteristici pentru Extragerea Trăsăturilor. Vizualizarea metodei este exemplificată în figura 4.3. [Paraskevopoulou2013]

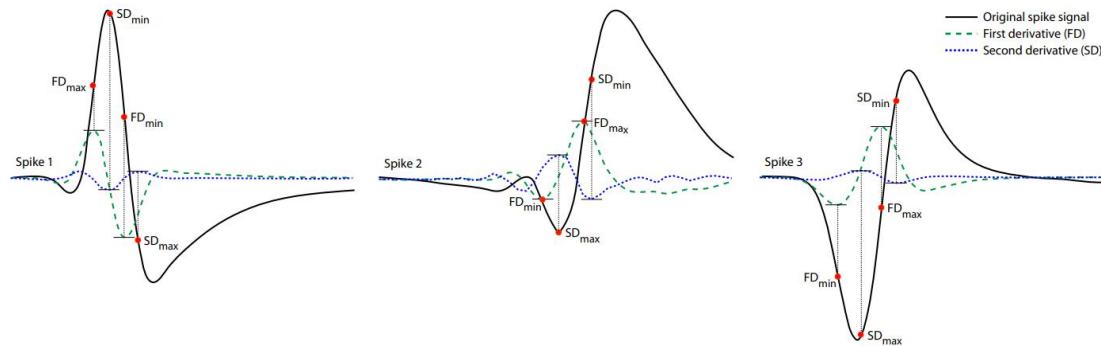


Figure 4.3 – Forme tipice a unei descărcări electrice, împreună cu prima (FD) și a doua derivată (SD), adnotate pe forma potențialului de acțiune sunt extremele (minimul și maximul) derivatelor [Paraskevopoulou2013]

Această metodă prezintă 4 posibile caracteristici, care pot fi încorporate în multiple combinații pentru a ajunge la un număr acceptabil de dimensiuni. Aceste combinații sunt prezentate în tabelul 4.1.

Numărul metodei	Numărul de dimensiuni	Caracteristicile
1	3	FD _{min} ; FD _{max} ; SD _{min}
2	3	FD _{min} ; FD _{max} ; SD _{max}
3	3	FD _{min} ; SD _{min} ; SD _{max}
4	3	FD _{max} ; SD _{min} ; SD _{max}
5	2	FD _{max} -FD _{min} ; SD _{max} -SD _{min}
6	2	(FD _{max} +FD _{min})/2; (SD _{max} +SD _{min})/2
7	4	FD _{min} ; FD _{max} ; SD _{min} ; SD _{max}

Tabel 4.1 – Combinăriile posibile ale celor 4 caracteristici oferite de metoda derivatelor de gradul 1 și 2 [Paraskevopoulou2013]

Formula folosită pentru a găsi derivata semnalului este următoarea:

$$derivata[i] = \frac{semnal[i] - semnal[i - 1]}{i - (i - 1)}$$

Pentru a calcula derivata semnalului, se va aplica aceasta formula pentru fiecare indice “i” a semnalului începând cu 1. A doua derivată se calculează analog prin aplicarea formulei asupra primei derive. Pentru păstrarea lungimii semnalului se poate ataşa un 0 la finalul semnalului după fiecare derivare.

Prin calcularea derivatei unui semnal, se obține panta tangentei la fiecare punct din semnal. Aceasta metoda accentuează schimbările de frecvențe din semnal și reduce zgomotul, iar din punctele rezultate din calcularea primei derive se pot obține caracteristici noi.

Metoda 6 a fost sugerată de către autori ca fiind cea mai bună variantă, cu toate acestea nu a avut rezultate mai bune decât celelalte și chiar mai slabe decât PCA, după cum se poate observa în figura 4.4.

Variante alternative și combinații diferite au fost încercate de Diana-Georgeta Lazea în lucrarea ei de licență. [Lazea2020]

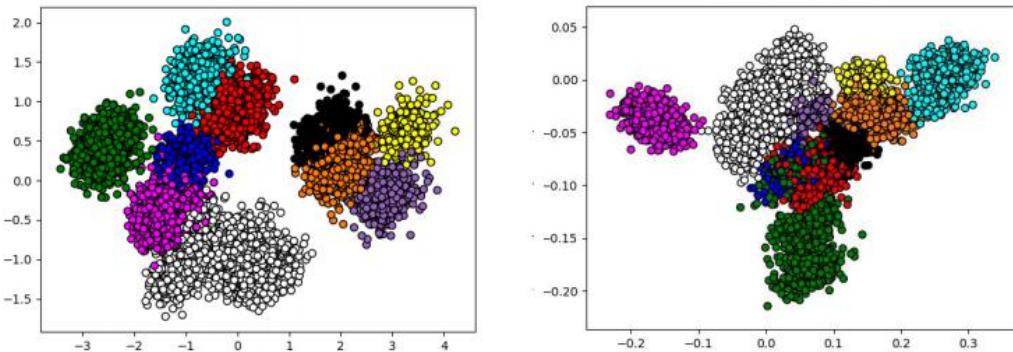


Figure 4.4 – Stânga prezintă rezultatul aplicării PCA-ului asupra simulării 15 (9 grupuri), iar dreapta aplicarea metodei 6 a derivatelor asupra aceleiași simulări

4.4. Pipeline

Folosirea unui pipeline pentru Extragerea Trăsăturilor a intervenit după ce s-a observat că algoritmii Short Time Fourier Transform, Superlet Transform și Hilbert Transform au reușit să separe diferite forme ale descărcarilor electrice. Prin vizualizarea graficelor se poate observa că fiecare din aceste metode oferă separabilitate pe diferențe grupuri când sunt aplicate pe aceeași simulare.

Astfel, prin introducerea acestor metode într-un pipeline se pot sepa mai multe grupuri decât ar putea sepa oricare metodă singură datorită faptului că fiecare din aceste metode sepa grupuri cu forme diferite de potențiale de acțiune, puține forme fiind separate de două dintre aceste metode. Aceste afirmații sunt confirmate de figurile 4.6, 4.7, 4.8.

Din acest motiv, s-a propus un pipeline format prin combinarea celor 3 metode. Astfel, un set de date va trece prin mai multe metode de Extragere a Trăsăturilor. Formele diferite ale descărcarilor electrice și evaluarea empirică a rezultatelor ne sugerează ca un pipeline ar fi mai util decât un sistem de votare. Fluxul (*flow*) pipeline-ului este reprezentat în figura 4.5.

Pentru determinarea grupurilor care sunt separate de restul s-a folosit coeficientul Silhouette, care determină cât de bine este definit un grup, aceasta metrică a fost descrisă în capitolul 3. Astfel, s-a introdus un prag pentru scorul obținut de coeficientului Silhouette, care dacă este depășit reprezintă faptul că grupul este destul de separat pentru a fi identificat de un algoritm de grupare.

Dacă unul sau mai multe grupuri sunt identificate ca fiind separate de o metodă, acestea sunt izolate. Iar restul punctelor sunt trecute printr-o altă metodă de Extragerea Trăsăturilor a pipeline-ului. Dacă după o trecere prin toate metodele încă au rămas puncte neseparate, se face încă o iterație a punctelor rămase prin pipeline. Punctul de oprire este

separarea tuturor punctelor în grupuri bine definite sau incapacitatea de a mai găsi grupuri separate într-o iterație a pipeline-ului de către toate metodele.

Dezvoltarea acestei metode a fost făcută în colaborare cu Andreea Maria Gui, Diana-Georgeta Lazea și Alexandru Hristache.

Pentru a testa corectitudinea rezultatelor găsite, un set de date a fost generat folosind forme de potențiale de acțiune din diferite simulări. Acest nou set de date conține 9 grupuri, cu forme de descărcari electrice alese astfel încât să fie separabile de o singură metodă, 3 forme pentru fiecare metoda de Extragerea Trăsăturilor din pipeline. Această verificare poate fi vizualizată în figura 4.9. [Gui2020] [Hristache2020] [Lazea2020]

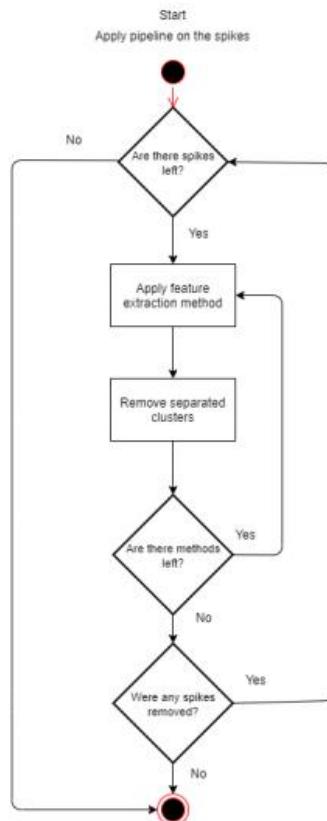


Figure 4.5 – Flow-ul pipeline-ului folosit pentru Extragerea Trăsăturilor

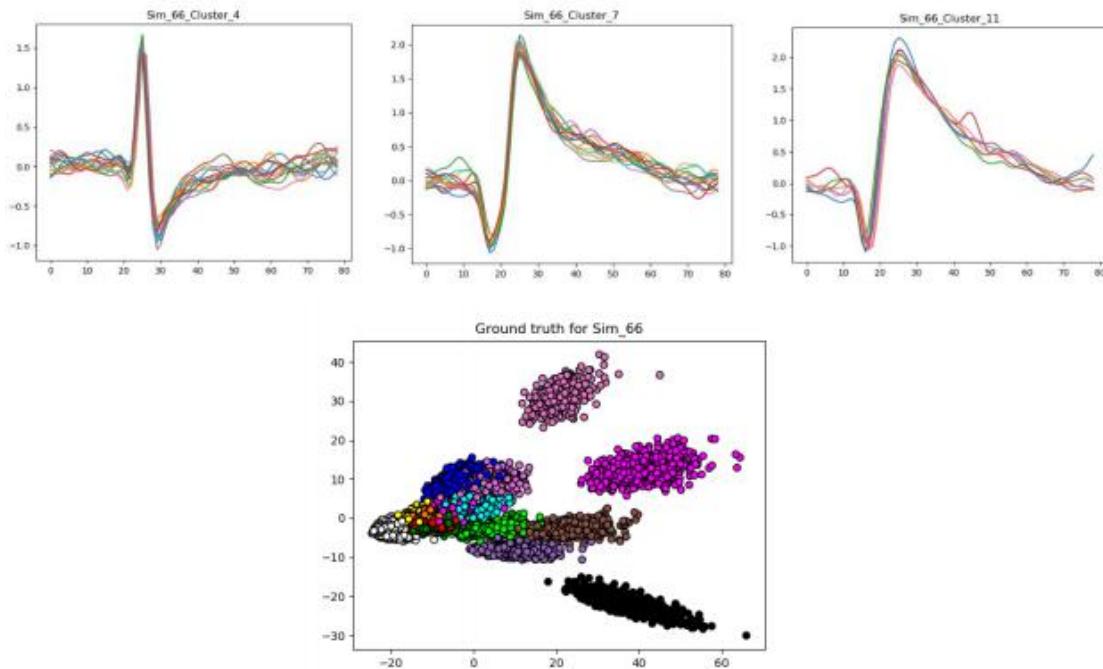


Figure 4.6 – Forme de potențiale de acțiune care sunt separate de Superlet Transform, jos se pot observa grupurile separate (negru, magenta și roz) de acest algoritm [Lazea2020]

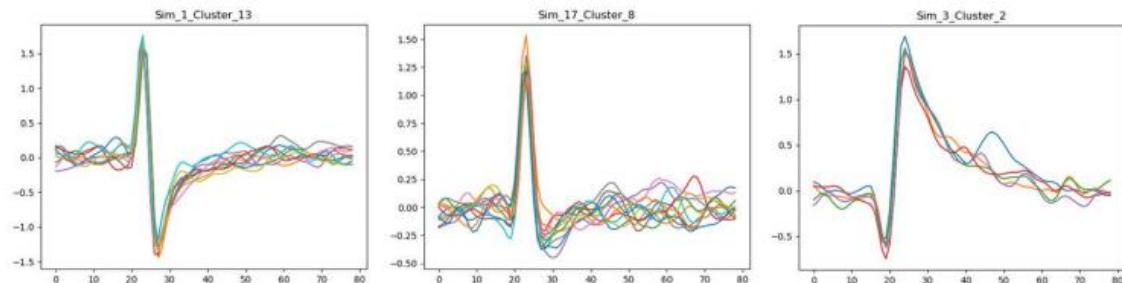


Figure 4.7 – Forme de potențiale de acțiune care sunt separate de Hilbert Transform [Hristache2020]

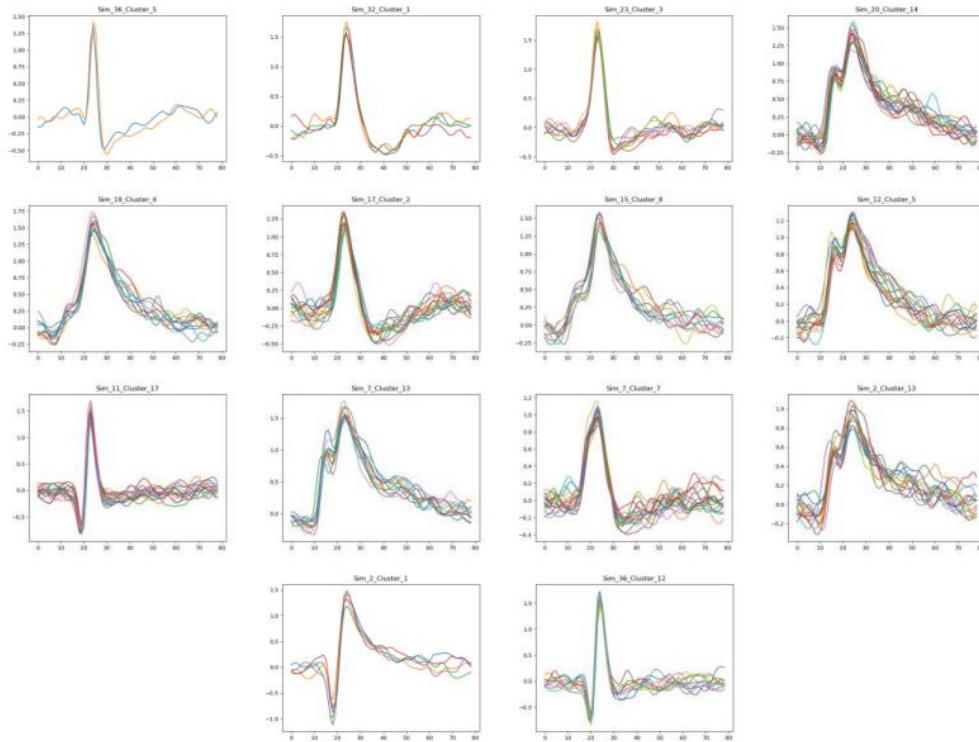


Figure 4.8 - Formele de potențiale de acțiune care sunt separate de Short Time Fourier Transform din primele 30 de simulari [Gui2020]

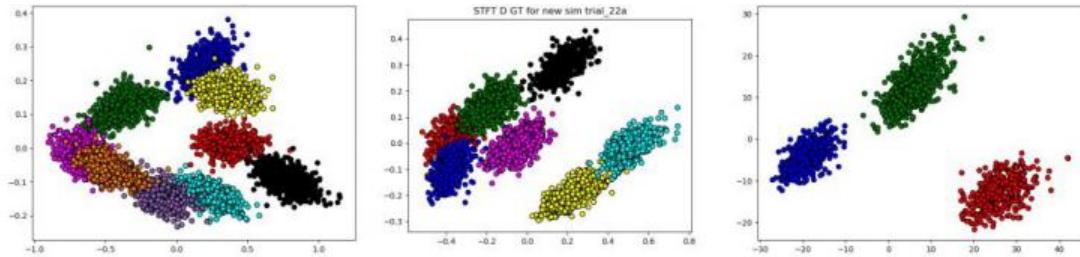


Figure 4.9 - Etichetele folosite pentru vizualizare sunt cele oferite de etichetarea reală, în stânga este întregul set de date generat, prin aplicarea Hilbert transform-ului se înlătura grupurile roșu și negru rezultând în graficul din mijloc cu culori actualizate, Short Time Fourier Transform înlătura grupurile negru, galben și albastru deschis (cyan), culorile fiind actualizate din nou, iar Superlet Transform cele rămase [Gui2020]

Pasul de Extragerea Trăsăturilor poate fi urmat încă o reducere a dimensionalității. Reducerea dimensionalitatii ne asigură că se ajunge la un număr acceptabil de dimensiuni pentru pasul de grupare. Este nevoie de acest pas doar în cazul în care numărul de trăsături rezultat din Extragerea Trăsăturilor este prea mare. Astfel, reducerea dimensionalității este făcută folosind PCA și metoda derivatelor.

4.5. Autocodificator

4.5.1. Modelul arhitecturii

Modelul arhitecturii este determinat de dimensionalitatea setului de date și de dimensionalitatea dorită pentru noul set de caracteristici. Știind setul de date și nevoia de a reduce cât mai mult dimensionalitatea datorită complexității algoritmilor de grupare și nevoie de vizualizare, pentru primul model input-ul și output-ul vor avea 79 de dimensiuni iar codul 2. Reconstruirea unui potențial de acțiune este afișată în figura 4.2.

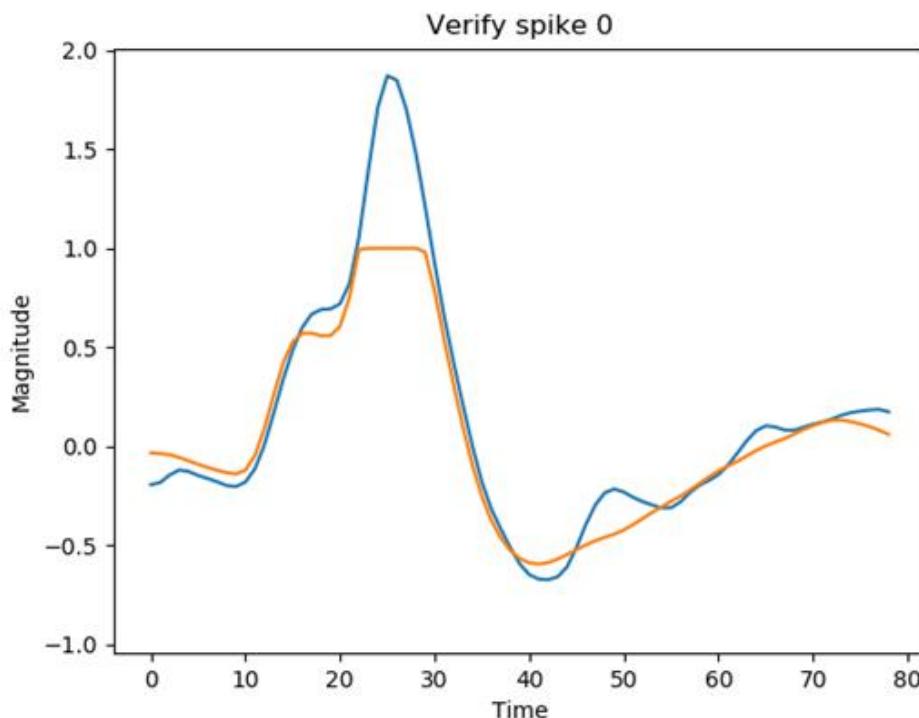


Figure 4.2 – Verificarea rezultatului Autocodificatorului pe un potențial de acțiune, linia albastră este potențialul original iar cea portocalie este reconstrucția Autocodificatorului

Se poate observa inabilitatea modelului de a reproduce fidel potențialul de acțiune. Cu toate acestea, un astfel de model antrenat pe o simulare oferă destulă separabilitate pentru a fi grupat, după cum se poate observa în figura 4.3.

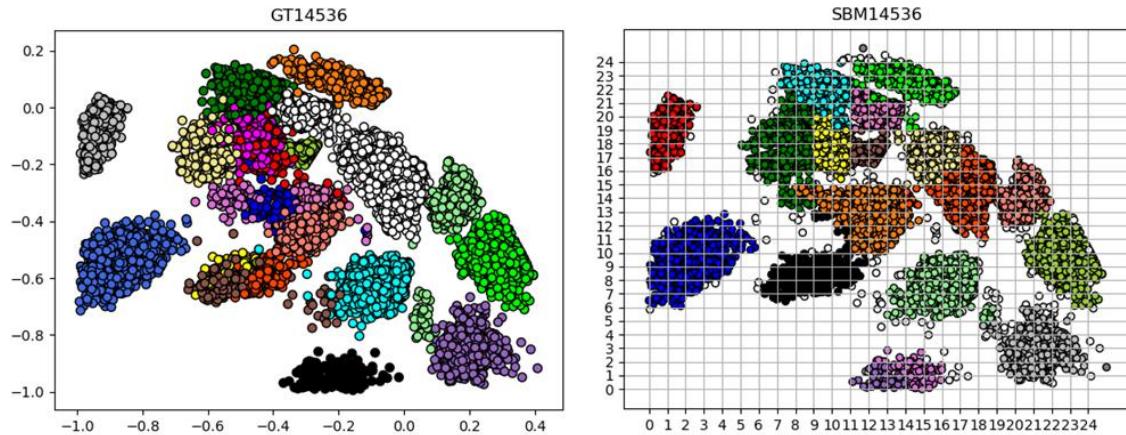


Figure 4.3 – În stânga este rezultatul (cu etichetare reală) Autocodificatorului pentru simularea 79 antrenat pe simularea 79, iar în dreapta este aplicarea algoritmului SBM de grupare pe setul de date cu caracteristicile rezultate din Autocodificator

O astfel de antrenare nu este tipică rețelelor neuronale, în antrenare și precizare datele introduse sunt diferite. Astfel în precizare modelul primește date nemaivăzute. Această separare nu am considerat că este relevantă pentru cazul autocodificatoarelor, deoarece scopul acestora nu este de învăță legatura dintre date și etichete ci de a reduce datele de intrare fără considerent pentru etichete.

Următorul pas pentru a observa comportamentul modelului este de a introduce date noi, similar cu testarea clasică pentru rețele neuronale. Am ales 2 simulări cu un număr mic de grupuri și anume simularea 4 cu 3 grupuri și simularea 8 cu 5 grupuri. Rezultatul modelului antrenat pe simularea 79 cand primește aceste simulări este arătat în figura 4.4.

Pentru datele noi, Autocodificatorul reușește să ofere separabilitate pentru unele dintre grupuri dar separă puncte din același grup sau pune împreună puncte din grupuri diferite, din acest motiv am ales refacerea antrenării cu mai multe simulări.

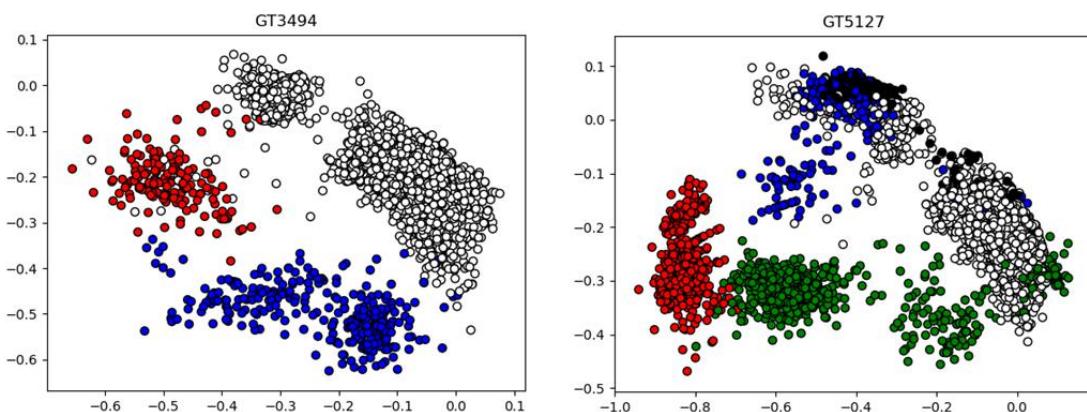


Figure 4.4 – În stânga este rezultatul (cu etichetare reală) Autocodificatorului (antrenat pe simularea 79) pentru simularea 4, rezultatul (cu etichetare reală) Autocodificatorului (antrenat pe simularea 79) pentru simularea 8

De asemenea, ca Autocodificatorul să poată reproduce potențialul de acțiune are nevoie de o funcție de activare care rezultă și în valori negative. Din acest motiv am ales tanh pentru straturile de cod și reconstrucție, în defavoarea sigmoid sau ReLU. Iar pentru restul straturilor ascunse a fost folosită funcția ReLU. Comparația dintre sigmoid vs tanh poate fi vizualizată în figura 4.5. Datorită nevoii de similaritate a output-ului cu input-ul am ales Mean Squared Error (MSE) ca funcția de loss.

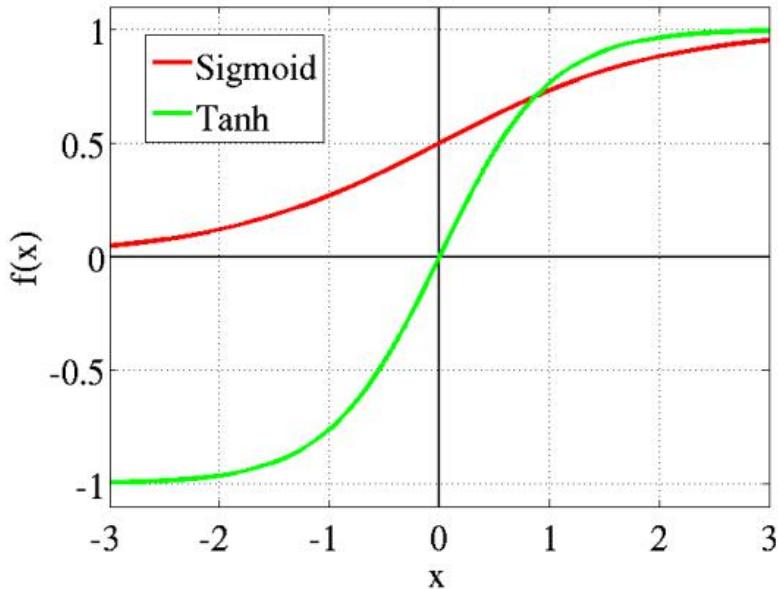


Figure 4.5 – Vizualizarea funcțiilor de activare sigmoid și tanh

Pentru a evita suprapotrivirea (*overfitting*), la stratul codului am adăugat regularizarea de tip L1 de $10e-7$. Modelul folosește optimizarea Adam cu o rata de învățare de 0.0001. Antrenarea este făcută pentru 100 de epoci, folosind întregul set de date.

4.5.2. Antrenarea Autocodificatorului

Pornind de la ipoteza prezenței zgomotelor în date, și că acestea au fost generate astfel încât amplitudinea să fie mai mică decât amplitudinea altor potențiale de acțiune, după cum se poate observa în figura 4.6, am aplicat filtrarea zgomotului ca pas de

preprocesare.

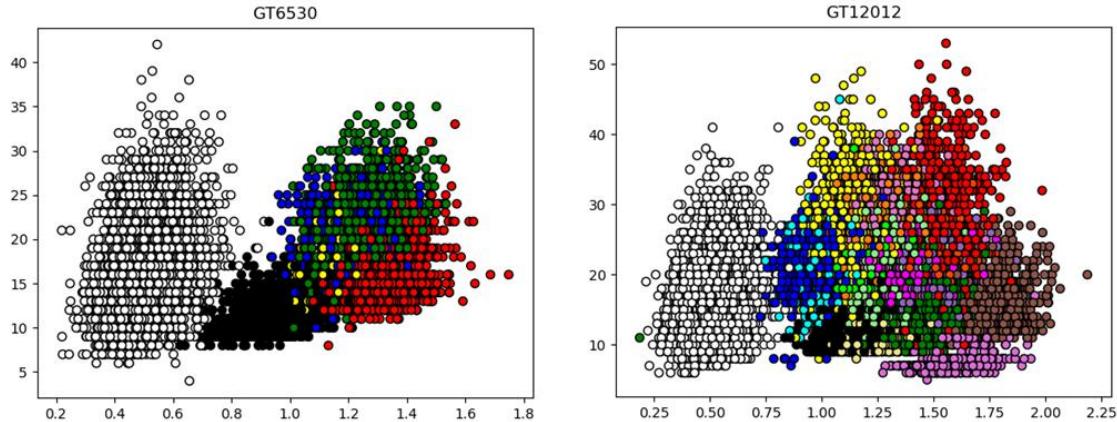


Figure 4.6 – Zgomotul (alb) are amplitudinea mult redusă față de alte grupuri, în general până în valoarea 0.75, pe axa X este amplitudinea (cel mai punct dintr-un potențial de acțiune), iar pe axa Y distanța dintre cele mai apropiate puncte de minim de amplitudine

Astfel antrenarea modelului este făcută folosind 50% din potențialele simulărilor 4, 8 și 79 cu zgomotul eliminat. Acest tip de antrenarserver_tasks_waiting_timee a adus rezultate variate, unele simulări au avut rezultate foarte bune, multe grupuri separate iar altele suprapuse sau grupuri din etichetarea reală separate când nu ar fi trebuit.

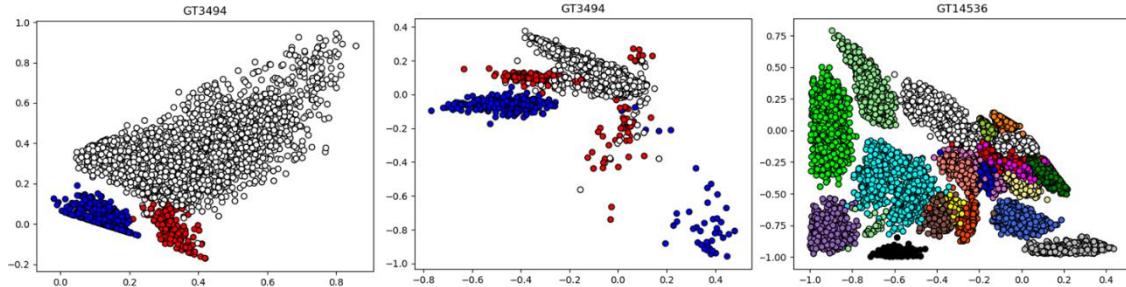


Figure 4.7 – În stânga, este rezultatul modelului pe toate punctele din simularea 4, oferă separabilitate moderată cu toate ca există suprapunere, în mijloc rezultatul pentru simularea 8 nu oferă destulă separabilitate pentru grupuri încât un algoritm de grupare să fie aplicat, în dreapta rezultatul pentru simularea 79, oferă separabilitate pentru multe din grupurile simulării

Dacă ne uităm la capabilitatea modelului de a reproduce un potențial de acțiune, în figura 4.8, putem observa că un astfel de model are un blocaj deoarece încearcă să codeze informația în prea puține dimensiuni, astfel nefiind capabil să construiască o reproducere precisă a potențialului de acțiune. Acest fapt poate fi observat și în figura 4.2.

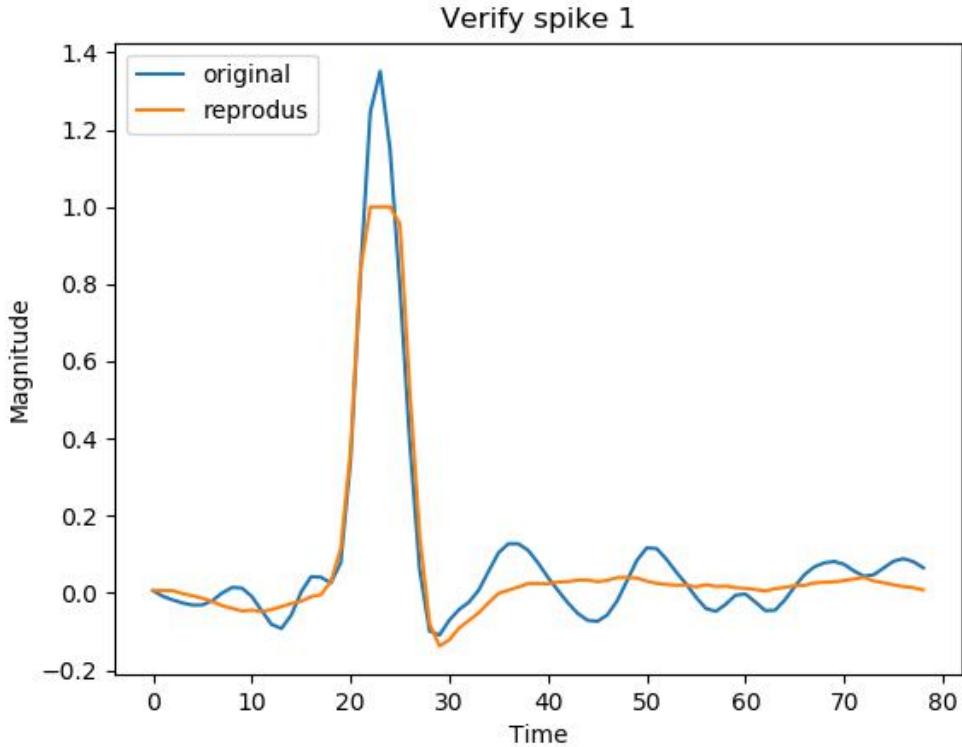


Figure 4.8 - Verificarea rezultatului autocodificatorului pe un potențial de acțiune, linia albastră este potențialul de acțiune original iar cea portocalie este reconstrucția autocodificatorului

4.5.3. Aplicarea PCA

Datorită problemei prezentate anterior, am ales să extind codul la diferite valori pentru a vedea dacă rezultatele sunt îmbunătățite. Astfel, s-au construit 5 modele cu coduri de 10, 20, 30, 40, 50 fiecare antrenat cu același set de date. Pentru codurile rezultate am aplicat algoritmul PCA pentru a reduce dimensionalitatea setului de date la 2 pentru a fi vizualizabil.

Cum este de așteptat, mărirea codului aduce și o reproducere mai bună a potențialului de acțiune, după cum se poate observa în figura 4.9. Cu toate acestea, reproducerea vârfului maxim (a amplitudinii) prin observații empirice, este determinată de abruptitatea fazei de creștere (rising phase) și nu de mărimea codului.

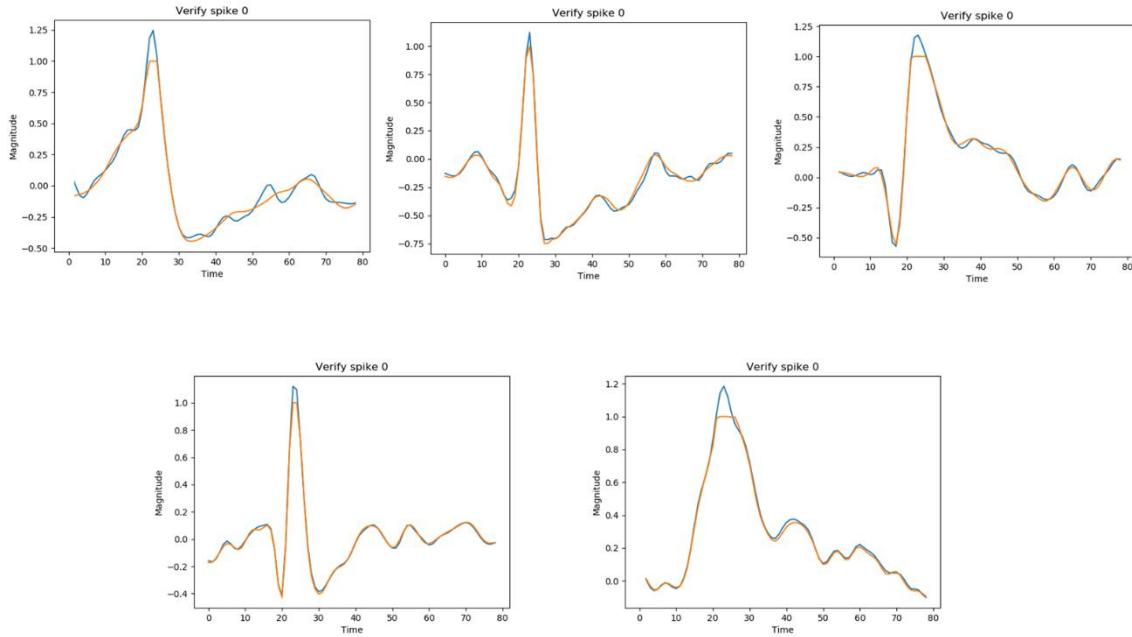


Figure 4.9 – De la stânga la dreapta și de sus în jos sunt cele 5 reproduceri ale unor potențiale cu codurile de 10, 20, 30, 40, 50

De asemenea, vizualizand rezultatele în figura 4.10 putem observa cele mai bune rezultate sunt oferite de un cod de marime 20. Codurile mai mici aduc o împărțire a grupurilor, iar codurile mai mari o alungire a grupurilor care duce la suprapunerea lor.

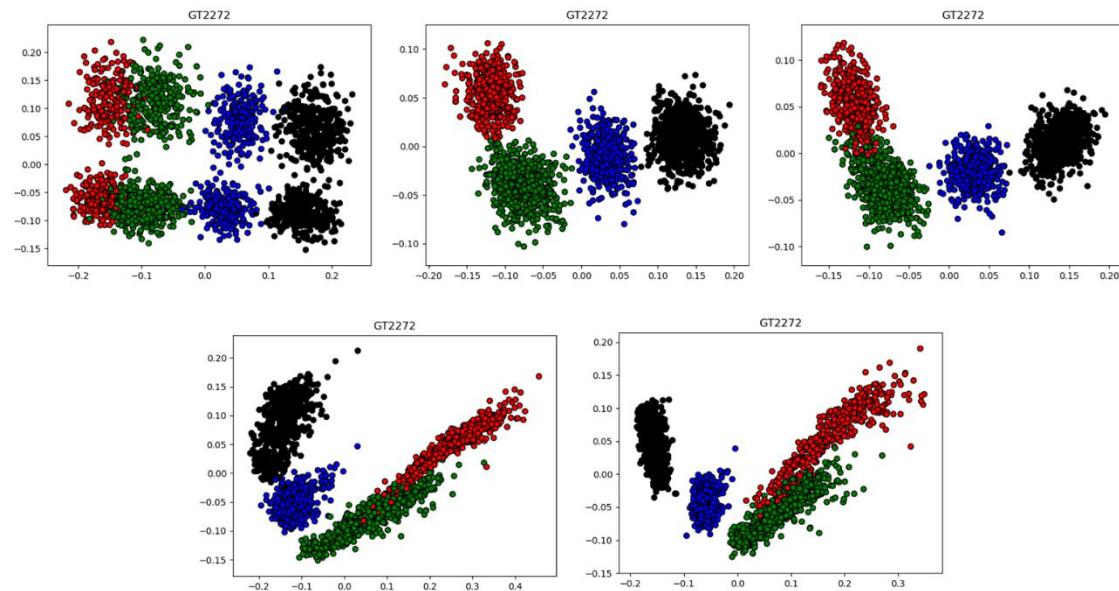


Figure 4.10 – Vizualizarea rezultatelor folosind modelele cu codurile de 10,20,30,40,50 de la stânga la dreapta și de sus în jos pentru simularea 4

4.5.4. Pre-antrenare

Pasul de preantrenare inițializează ponderilor și abaterilor statistice ale straturilor Autocodificatorului. Fără acest pas, acestea sunt initializate aleator, din acest motiv preantrenarea ar putea îmbunătăți rezultatele prin creșterea șansei de a evita un minim local și de a găsi minimul global al gradient descent-ului.

Tipul de preantrenare implementat este greedy. Se face o antrenare stratificată pentru subseturi de straturi. [Sagheer2019]

Folosind inputul de 79 se creează un model parțial cu arhitectura 79-70-79, acest model este antrenat urmând ca valorile ponderilor și abaterilor statistice să fie salvate ca valori în preantrenare iar rezultatele codului (70) să fie salvate pentru a fi folosite ca input pentru următorul „strat”. Astfel următorul strat avea modelul va 70-60-70, și se vor repeta salvările anterioare. În mod analog se repetă până se ajunge la modelul parțial care are codul echivalent cu codul dorit. În acest mod se generează valorile inițiale ale ponderilor și abaterilor statistice pentru model.

Ca în subcapitolul anterior, creșterea mărimei codului aduce o reproducere mai bună a potențialelor de acțiune, de asemenea se poate observa o deplasare a rezultatelor. Reproducerea vârfului maxim nu este îmbunătățită de preantrenare, după cum se poate observa în figura 4.11.

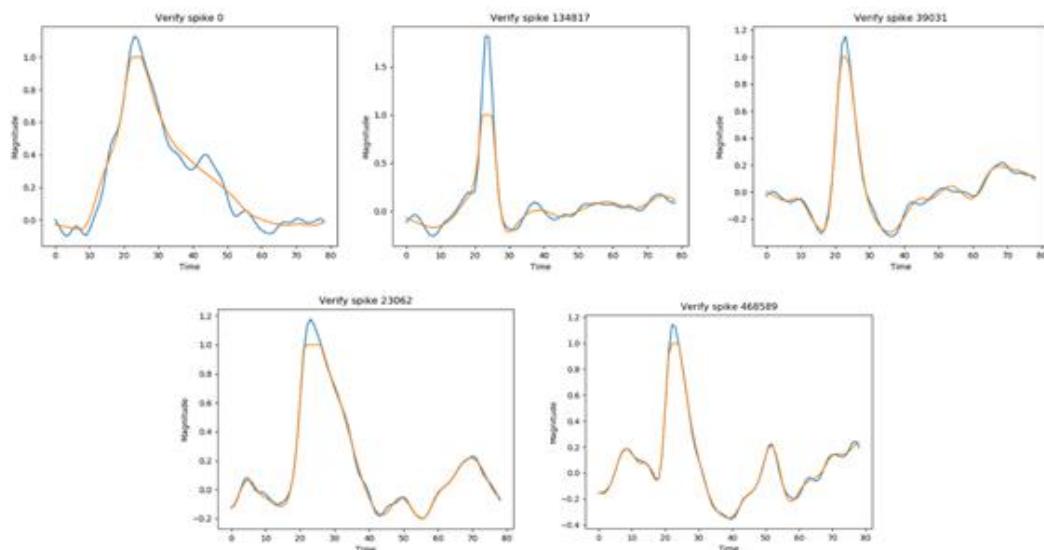


Figure 4.11 – De la stânga la dreapta și de sus în jos sunt cele 5 reproduceri ale unor potențiale de acțiune pentru modelele cu codurile de 10, 20, 30, 40, 50

Această deplasare se poate observa și mai clar în vizualizarea rezultatelor, figura 4.12. În cazul anterior alungirea începea la un cod de 30, în acest caz abia la 50, iar împărțirea se face până la un cod de 20, pe când înainte doar la codul de 10 se întampla.

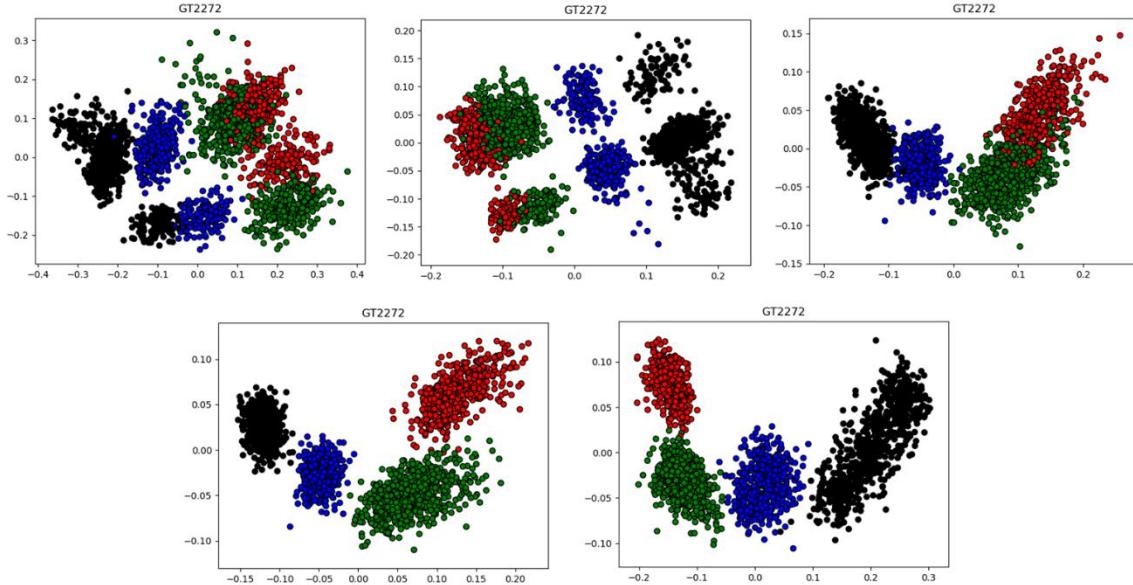


Figure 4.12 - Vizualizarea rezultatelor folosind modelele preantrenate cu codurile de 10,20,30,40,50 de la stânga la dreapta și de sus în jos pentru simularea 4

4.5.5. Expansiune

Rezultatele oferite de Autocodificatoare până în acest punct al proiectului nu au fost satisfăcătoare. Am luat în considerare posibilitatea ca datorită faptului ca încercăm să reducem codul la un număr mai mic decât inițial să se creeze un “bottleneck”. Din acest motiv am făcut o creștere constantă a mărimii straturilor Autocodificatorului. Astfel arhitectura a devenit [80, 90, 100, 110] și un cod de 120. Autocodificatorul va extinde semnalul și astfel va amplifica informația disponibilă pentru PCA. Rezultatele codului au fost extrase și PCA a fost aplicat pentru noul spațiu cu 120 de dimensiuni. Reproducerea potențialelor de acest model este afișată în figura 4.13.

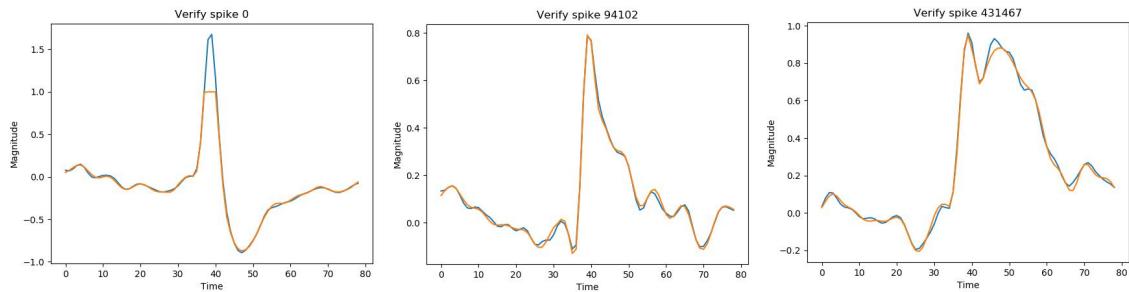


Figure 4.13 - Reproducerea potențialelor de acțiune de către Autocodificatorul cu expansiune (cu un cod de 120)

Rezultatele obținute de Autocodificatorul expandat pot fi vizualizate în figura 4.14. Se poate observa că nu aduce separabilitate adițională față de PCA, în unele situații chiar reduce separabilitatea unor grupuri.

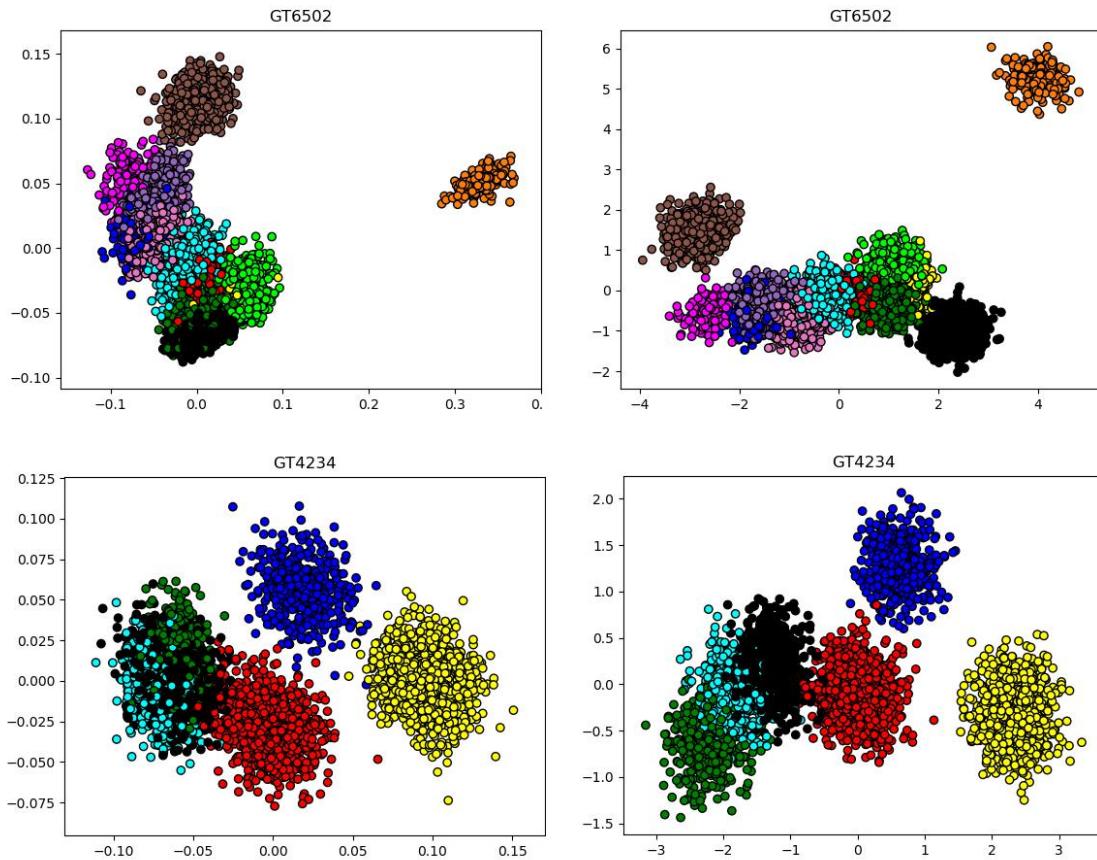


Figure 4.14 - Vizualizarea rezultatelor folosind modelul expandat cu codul 120 (în stânga) în comparație cu rezultatele PCA-ului (dreapta) pentru simulările 3 (sus) și 22 (jos)

4.6. SBM îmbunătățit

În implementarea algoritmului SBM [Ardelean2019], datorită structurii matriciale folosite creșterea dimensionalitatii duce la o creștere exponențială a memoriei necesare pentru a reține n-array-ul construit.

O soluție la aceasta problema descrisă în [Ardelean2019] este de a înlocui structura inițială cu o structură bazată pe grafuri. Problema se poate descrie mai clar în următorul fel: Fiind un set de date cu 5000 de puncte și 4 dimensiuni, la un PN=25 va crea 390,625 de „chunk”-uri, fiind doar 5000 de puncte de împărțit în 390,000 de „chunk”-uri, majoritatea dintre acestea vor avea 0 asociat ca și număr. Folosind structura de graf, nu mai este nevoie ca aceste chunk-uri goale să fie reținute. În figura 4.13, simularea 4 (care conține 5127 de puncte) este exemplificată cu ambele structuri pentru PN=5, chiar și pentru 2 dimensiuni se poate observa o îmbunătățire, de la 25 de „chunk”-uri reținute la 22 (3 dintre ele fiind cu valoarea 0).

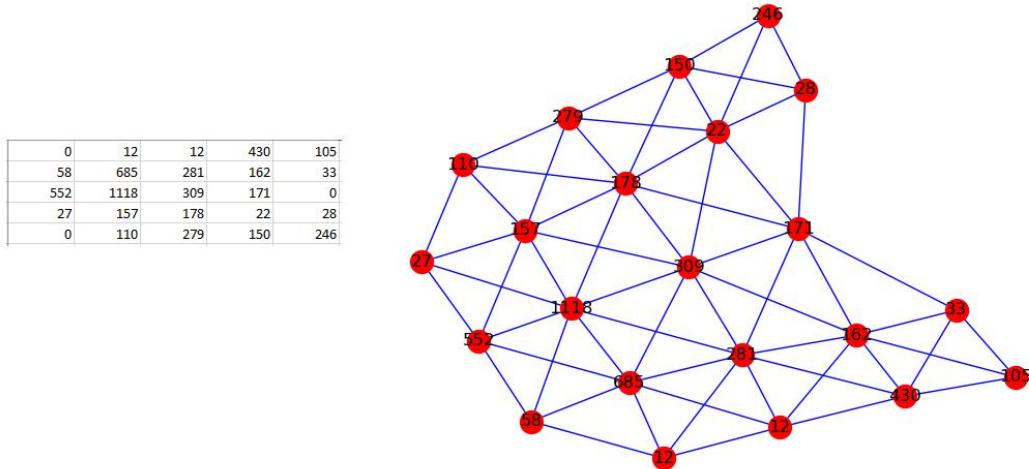


Figure 4.13 – Vizualizarea structurii folosite în cele 2 implementări ale algoritmului (pentru un set de date cu 2 dimensiuni), în stânga apare structura matriceală cu $PN \times PN$ valori (în acest caz $PN=5$, adică 25 de celule), iar în dreapta se pot observa 22 de noduri în graf

Această nouă implementare bazată pe structura de graf, menține rezultatele obținute înapoi cu o îmbunătățire din punct de vedere a complexității spațiului. În figura 4.14, se poate observa faptul că rezultatele obținute de ambele implementări sunt identice.

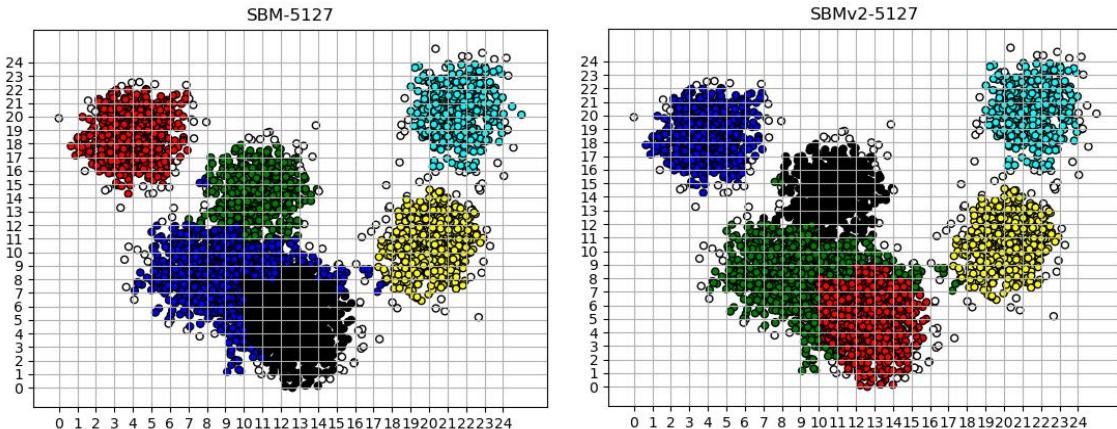


Figure 4.14 – Vizualizarea rezultatelor celor 2 implementări ale algoritmului, în stânga este implementarea din [Ardelean2019], iar în dreapta este implementarea bazată pe grafuri, se poate observa ca punctele obținute ca un grup în stânga sunt de asemenea un grup în dreapta (diferențele de culori sunt irelevante deoarece sunt bazate pe o mapare de tip număr-culoare)

4.7. Autocodificator LSTM

După cum a fost menționat în 3.4.8, LSTM-urile sunt un tip de arhitectură de rețea neuronală recurrentă, acestea sunt diferite de rețelele neuronale feedforward prin conexiunile de feedback. Sunt capabile să proceseze secvențe de date, cum ar fi

înregistrări de vorbire sau video. Astfel, sunt potrivite pentru procesarea potențialelor acestea fiind secvențe de date.

Arhitectura precedentă de Autocodificator, a fost modificată din straturi dense de neuroni în straturi de LSTM pentru părțile dintre input și cod și dintre cod și output. Straturile de cod și output au rămas straturi dense. Arhitectura are 2 straturi pentru codificator și 2 pentru decodor cu 64, 32 și respectiv 32, 64 de neuroni pe fiecare strat. Mărimea codului a fost aleasă empiric 20, știind rezultatele obținute la Autocodificatoare.

Ştiind ca LSTM-urile se bazează și pe inputul curent și pe cele anterioare am ales să încerc 2 preprocesari. Prima preprocesare împarte potențialul de acțiune în 4 porțiuni egale de 20 de puncte fără suprapunere (potențialul de acțiune conține 79 de puncte și a fost adăugat un 0 la final pentru uniformitate).

A doua preprocesare suprapune părți de potențial de acțiune, se împart în 8 porțiuni egale de 10 cu 10 puncte de suprapunere între 2 porțiuni adiacente (0->20, 10->30, 20->40 ...).

Luând în considerare informațiile despre LSTM și cele 2 preprocesări, am considerat ca varianta care conține suprapunere va obține rezultate mai bune. După cum se poate observa din 4.15 și 4.16 varianta cu suprapunere reușește să reproducă mult mai consistent potențialele de acțiune, aproape perfect în unele cazuri și doar o reducere a amplitudinii în altele.

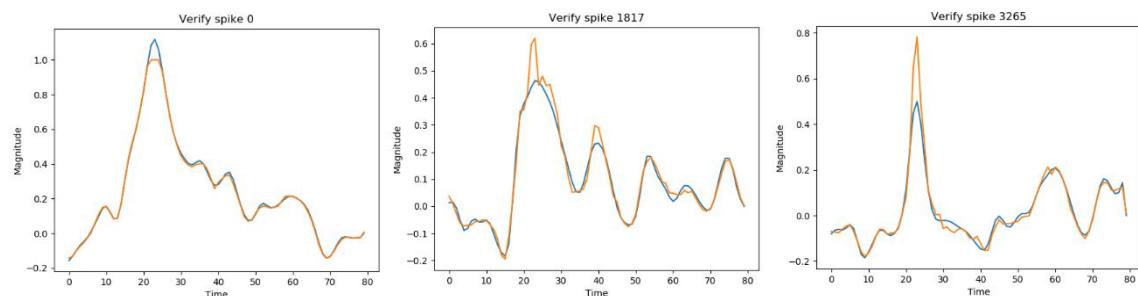


Figure 4.15 – Verificarea reproducerii potențialului de acțiune de către LSTM Autocodificator antrenat pe date fără suprapunere

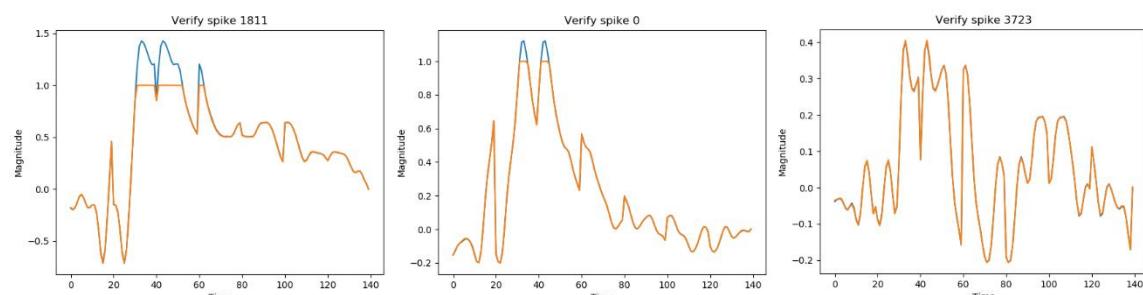


Figure 4.16 - Verificarea reproducerii potențialului de acțiune de către LSTM Autocodificator antrenat pe date cu suprapunere (am lăsat suprapunerea pentru a verifica în totalitate reproducerea, deoarece outputul este format tot din 8 porțiuni)

Următorul pas este aplicarea PCA-ului pe codul rezultat (de mărime 20) pentru a vizualiza grupurile. Rezultatele pot fi văzute în 4.17 și 4.18 și este clar ca varianta fără suprapunere este capabilă să separe mult mai bine. Concluzia empirică este faptul că în varianta cu suprapunere apare „overfitting”.

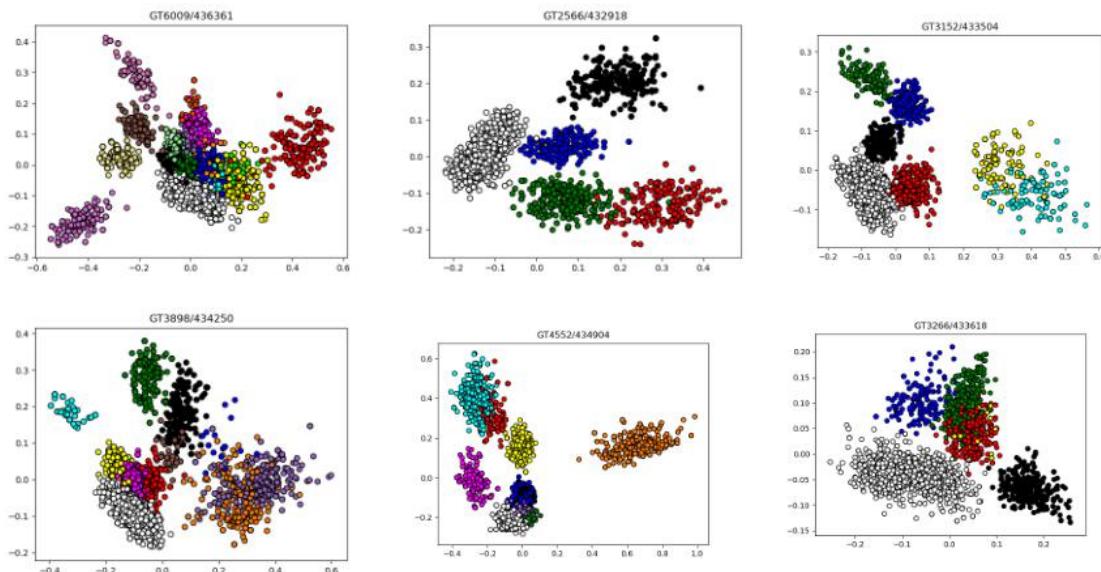


Figure 4.17 – Vizualizarea grupurilor rezultate din Autocodificatorul LSTM (după aplicarea PCA) fără suprapunere – pentru simulările 1,4,24,13,17,26 (de la stânga la dreapta și sus în jos)

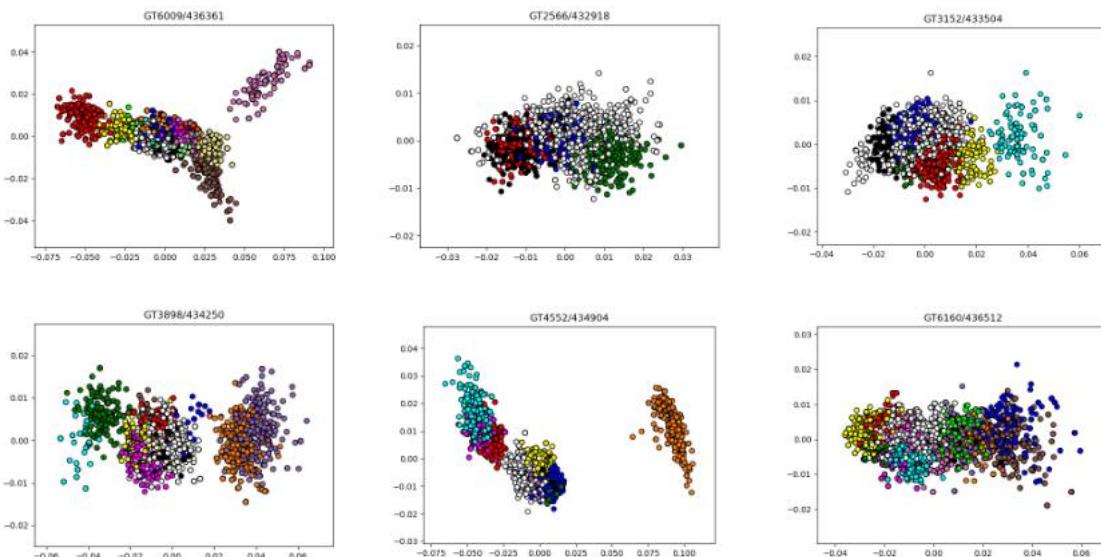


Figure 4.18 - Vizualizarea grupurilor rezultate din Autocodificatorul LSTM (după aplicarea PCA) cu suprapunere – pentru simulările 1,4,24,13,17,26 (de la stânga la dreapta și sus în jos)

Capitolul 5. Rezultate teoretice și Interpretări

5.1. Rezultate inițiale

La începutul proiectului, am încercat să folosim metodele de grupare prezentate în capitolul 3 verificând rezultatele folosind validările din proiectul de licență. [Ardelean2019] Astfel, folosind metricile ARI și AMI în cele două setări de validare (ALL și NNP) am verificat rezultatele celor 3 algoritmi: K-Means, DBSCAN și SBM.

Nevoia celor 2 setări ALL și NNP a apărut datorită faptului că rezultatele algoritmilor DBSCAN și SBM conțin și zgomot. Știind că datele folosite pentru validare conțin etichetare reală, am vrut să obținem și acuratețea punctelor grupate deoarece în etichetare reală nu există zgomot și acest lucru reduce acuratețea. Motivatia și explicația setărilor sunt detaliate în exemplificarea algoritmului SBM. [Ardelean2019] Rezultatele pot fi examineate în tabela 5.1.

Tabel 5.1 – Acuratețea medie a celor 3 algoritmi pentru cele 95 de simulări ale setului de date pentru cele 2 setări

Algoritm/Metrică	ARI-ALL	AMI-ALL	ARI-NNP	AMI-NNP
K-Means	0.51	0.688	0.578	0.724
DBSCAN	0.206	0.253	0.726	0.73
SBM	0.601	0.666	0.614	0.691

Rezultatele obținute scot în evidență câteva observații:

- Există o diferență mare între acuratețea obținută de ARI și AMI pentru K-Means, acest fapt ne indică că K-Means nu este cea mai bună metodă de grupare pentru aceste date
- Pentru DBSCAN, există o diferență mare între cele 2 setări ALL și NNP, ceea ce ne indică că există foarte multe puncte clasificate ca și zgomot, se datorează dificultăților de a alege parametrii adecvăți datorită faptului că grupurile sunt imbalansate și suprapuse
- SBM, fiind un algoritm creat pentru date neuronale și provocările acestora sunt cele mai stabilă rezultate [Ardelean2019]

Datorită faptului că K-Means și DBSCAN nu pot fi folosite pentru a valida corect rezultatele, s-a continuat folosirea doar a algoritmului SBM cu diferite metode de Extragere a Trăsăturilor și/sau Reducere a Dimensionalității.

Acuratețea pentru cele 3 metode de Extragere a Trăsăturilor pot fi examineate în tabela 5.2. Se poate observa că extragerea a 3-a componentă principale a PCA-ului nu crește acuratețea rezultatelor ci chiar o scade. Variantele a celei de-a 3-a metoda de Extragere Trăsăturilor din tabel sunt detaliată în lucrarea de licență a Diana-Georgeta Lazea [Lazea2020]. Rezultatele obținute folosind metoda derivatelor fiind mai bune în medie ca PCA 2D cu aproximativ 19% pentru ARI și 1% pentru AMI.

Tabel 5.2 – Acuratețea medie a algoritmului SBM pentru cele 95 de simulări ale setului de date pentru cele 2 seturi de date

Metoda de Extragere a Trăsăturilor / Metrica	ARI-ALL	AMI-ALL	ARI-NNP	AMI-NNP
PCA 2D	0.601	0.666	0.614	0.691
PCA 3D	0.386	0.552	0.534	0.664
Metoda derivatelor	0.712	0.693	0.725	0.713

5.2. Evaluarea performanței pipeliine-ului

Rezultatele prezentate au fost măsurate pe setul de date prezentat în capitolul 4.1 [Pedreira2012] Pentru a simplifica urmărirea au fost alese simulările cu un număr de grupuri între 2 și 6. Rezultatele Extragerii Trăsăturilor prin pipeline sunt prezentate în tabela 5.3, împreună cu specificarea unei separări reușite, observații și numărul de pași până la oprire.

După cum se poate observa din tabelul 2 din anexă, există posibilitatea în care grupurile sunt separate dar pragul ales este prea mare, această inconveniență apare datorită valorii mai mici a coeficientul Silhouette dacă grupul este împrăștiat sau are forma convexă. Chiar și în cazurile în care nu ajunge la separabilitate totală, pipeline-ul reușește să separe o parte din grupuri.

Iterarea pipeline-ului de Extrageră Trăsăturilor pe simularea 22 este prezentată în figura 5.1. Pentru această simulare pipeline-ul nu reușește să separe în totalitate. Pragul folosit pentru coeficientul Silhouette este de 0.65 cu metrica Mahalanobis pentru distanța la STFT și HT, iar metrica Euclidiană pentru Superlet Transform.

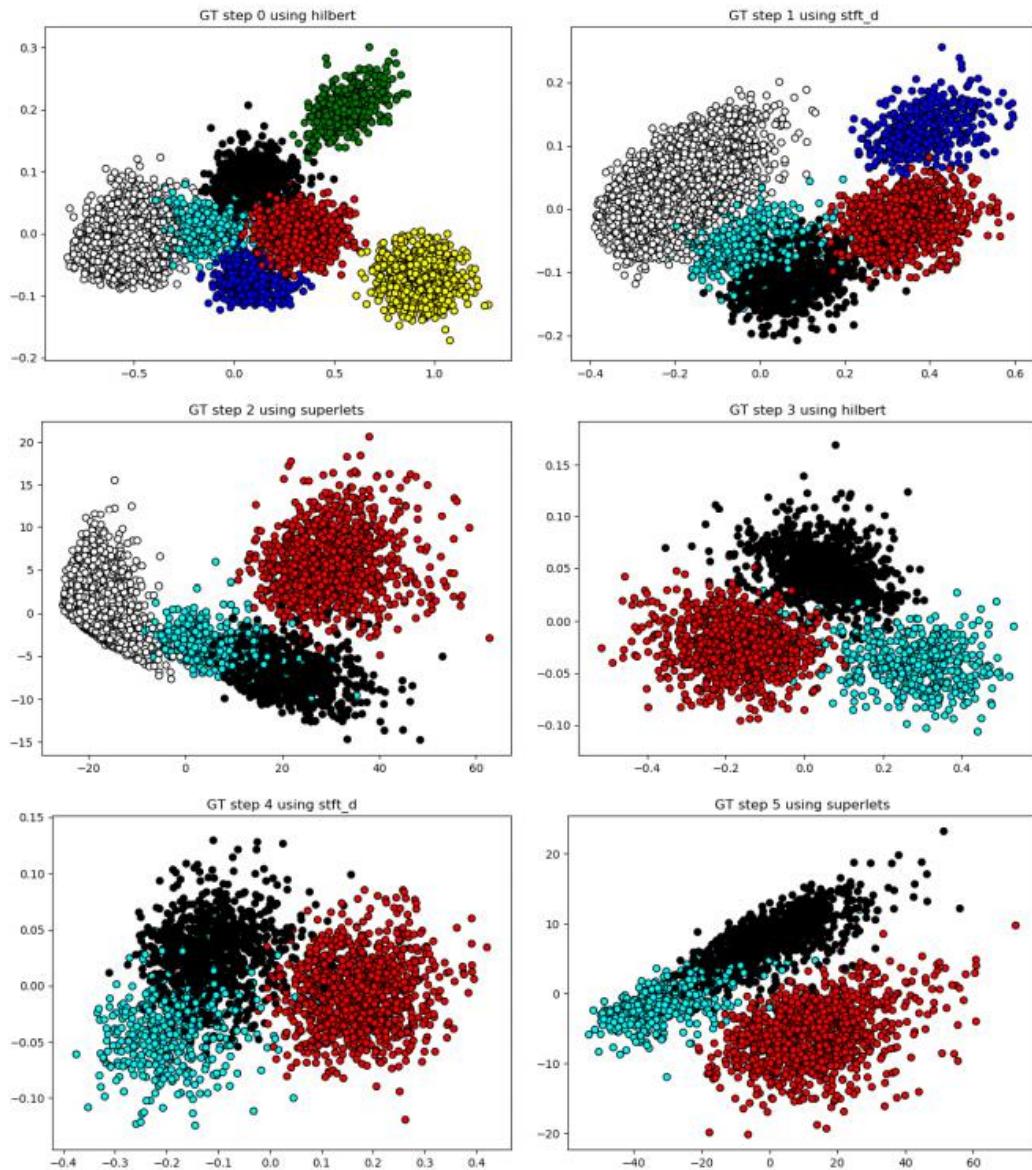


Figure 5.1 – Vizualizarea flow-ului pipeline-ului pe simularea 22

După cum se poate observa, 3 din cele 6 grupuri și zgromotul au fost separate de către pipeline. Modificările în simulare la fiecare pas pot fi urmărite și în tabela 5.4. În primul pas, HT reușește să izoleze 2 dintre grupuri (galben și verde) care au coeficientul peste prag. În continuare, se elimină grupul albastru de către STFT în pasul 2 și zgromotul de către SLT în pasul 3. În următorii 3 pași, se aplică cele 3 metode fără a reuși să izoleze încă un alt grup, în acest punct pipeline-ul se oprește. Se poate observa că chiar și pe simulările fără separabilitate totală, pipeline-ul are rezultate superioare aplicării metodelor individuale.

Table 5.4 – Exemplificarea pașilor pipeline-ului

Pasul	Metoda	Grup	Coeficientul Silhouette	Izolarea
-------	--------	------	-------------------------	----------

1	HT	0 (zgomot)	0.379	
		1	0.433	
		2	0.578	
		3	0.686	Da
		4	0.581	
		5	0.664	Da
		6	0.393	
2	STFT	0 (zgomot)	0.366	
		1	0.525	
		2	0.658	Da
		4	0.309	
		6	0.251	
3	SLT	0 (zgomot)	0.731	Da
		1	0.396	
		4	0.319	
		6	0.437	
4	HT	1	0.527	
		4	0.535	
		6	0.555	
5	STFT	1	0.450	
		4	0.372	
		6	0.363	
6	SLT	1	0.296	
		4	0.245	
		6	0.516	

5.3. Evaluarea performanțelor autocodificatorului

Alegerea parametrilor specificați în capitolul 4 a fost făcută prin evaluarea rezultatelor autocodificatorului pentru mai multe variante pentru fiecare parametru.

Numărul de epoci pentru antrenare a fost ales 100, cu toate ca un număr mai mare ar fi putut aduce rezultate mai bune, datorită duratei de timp necesare pentru antrenare. Comparația dintre valorile alese pentru numărul de epoci poate fi vizualizată în tabela 5.5.

Rata de învățare a fost aleasă ca 0.0001 în mod analog prin comparația a mai multor valori. Această comparație între valorile ratei de învățare poate fi vizualizată în tabela 5.6.

Impactul alinierii asupra PCA-ului și a autocodificatorului poate fi vizualizat în tabela 5.7.

Table 5.5 – Comparație a scorurilor pentru diferite variante ale numărului de epoci (Delta reprezintă diferența dintre scorul autocodificatorului și scorul PCA-ului)

	Delta	Delta	Delta

Număr de epoci	50	100	500
ARI	-0.018	-0.0210	0.0042
AMI	-0.011	-0.0042	0.0045
Homogeneity	0.0021	0.0560	0.0244
Completeness	-0.0133	-0.0326	-0.012
V-measure	-0.010	-0.0038	0.0045
CHS	1810.51	-2.86	-802.5
DBS	0.6360	0.2085	-0.111
SS	0.0535	0.03413	0.0726

Table 5.6 – Comparație a scorurilor pentru diferite variante ale ratei de învățare
(Delta reprezintă diferența dintre scorul autocodificatorului și scorul PCA-ului)

	Delta	Delta	Delta	Delta
Rata de învățare	0.01	0.001	0.0001	0.00001
ARI	-0.347	-0.014	0.022	-0.06
AMI	-0.353	0	0.0027	-0.048
Homogeneity	-0.351	0.058	-0.009	-0.077
Completeness	0.039	-0.046	0.03	-0.008
V-measure	-0.353	0.0005	0.002	-0.004
CHS	142345	355.22	9461.8	15076
DBS	8.608	0.11	-0.019	0.497
SS	-0.2	0.05	0.113	-0.022

Table 5.7 – Comparație a scorurilor pentru diferite variante de aliniere între scorul autocodificatorului și scorul PCA-ului

	Autocodificator	PCA	Autocodificator	PCA	Autocodificator	PCA

Aliniere	La medie	La medie	Nu	Nu	La maxim global	La maxim global
ARI	0.623	0.601	0.579	0.554	0.673	0.597
AMI	0.729	0.727	0.769	0.753	0.761	0.726
Homogeneity	0.719	0.728	0.811	0.810	0.761	0.726
Completeness	0.782	0.757	0.760	0.725	0.786	0.752
V-measure	0.730	0.728	0.770	0.753	0.762	0.727
CHS	20865.36	11403.56	19439.72	12017.59	18601.29	11394.32
DBS	1.315	1.334	0.831	1.003	1.64	1.333
SS	0.421	0.308	0.431	0.340	0.426	0.308

După cum a fost menționat în capitolul 4.5.4, Autocodificatorul preantrenat pare să translateze rezultatele de la mărimea codului de 20 la 50. Acest lucru se poate observa și în Tabelele 5.8 și 5.9, cele mai multe simulari pentru Autocodificator au rezultatul cel mai bun pentru codul de mărime 20, iar pentru Autocodificatorul preantrenat codul de mărime 50.

Table 5.8 – Rezultatele Autocodificatoarelor cu 5 mărimi ale codului urmat de aplicarea PCA-ului pentru a reduce spațiul în 2 dimensiuni, celule conțin {numărul de simulari pentru care codul respectiv avea cel mai bun rezultat / scorul mediu pentru aceste simulații}

Metoda de validare / Code size	10	20	30	40	50
ARI	10 / 0.39	38 / 0.56	23 / 0.52	4 / 0.41	18 / 0.5
AMI	8 / 0.49	43 / 0.62	21 / 0.6	4 / 0.51	17 / 0.58

Table 5.9 - Rezultatele Autocodificatoarelor preantrenate cu 5 mărimi ale codului urmat de aplicarea PCA-ului pentru a reduce spațiul în 2 dimensiuni, celule conțin {numărul de simulari pentru care codul respectiv avea cel mai bun rezultat / scorul mediu pentru aceste simulații}

Metoda de validare /	10	20	30	40	50

Code size					
ARI	6 / 0.41	10 / 0.42	14 / 0.5	27 / 0.52	36 / 0.54
AMI	7 / 0.52	8 / 0.51	16 / 0.58	24 / 0.6	38 / 0.62

De asemenea, a fost luată în considerare adăugarea Autocodificatorului ca un pas în pipeline. Următorul pas a fost validarea rezultatelor folosind Coeficientul Silhouette pentru fiecare punct și s-a făcut o medie pentru fiecare grup. Acest pas a arătat separabilitatea fiecărui grup și s-a putut observa că grupurile separate de Autocodificator, sunt deja separabile de STFT, hilbert sau PCA. Astfel, am considerat că nu merită adăugarea algoritmului în pipeline.

Observând că fiecare cod oferă cele mai bune rezultate pentru anumite simulari, am făcut validarea rezultatelor pentru toate simularile folosind toate codurile. Media rezultatelor afișată în tabela 5.10 este calculată folosind validarea pentru fiecare simulare cu codul care oferă cel mai bun rezultat. Știind că grupul de zgromot este singurul generat multi-unit, s-a făcut antrenarea în 2 moduri “ALL” conține puncte din toate grupurile iar NNP din toate mai puțin grupul de zgromot.

Table 5.10 - Rezultatele Autocodificatorului în comparație cu PCA, folosind 2 moduri, antrenare cu toate punctele din seturile de date și fără punctele de zgromot, media afișată este făcută prin alegerea celor mai bune rezultate

Algoritm	PCA	PCA	Autocodificator	Autocodificator
Mode	ALL	NNP	ALL	NNP
ARI	0.601	0.621	0.62	0.62
AMI	0.726	0.733	0.71	0.72
Homogeneity	0.728	0.743	0.68	0.71
Completeness	0.751	0.757	0.77	0.75
V-measure	0.727	0.734	0.71	0.72
CHS	11389	14677	11497	9892
DBS	1.335	1.283	1.79	1.79
SS	0.307	0.418	0.29	0.37

Validarea pentru varianta Autocodificatorului care face expansiunea codului înainte de aplicarea PCA-ului a fost făcută similar. Au fost calculate scorurile pentru fiecare simulare în cele 2 moduri menționate anterior și s-a făcut media rezultatelor.

Table 5.11 - Rezultatele Autocodificatorului expandat (cod 120) în comparație cu PCA, folosind 2 moduri, antrenare cu toate punctele din seturile de date și fără punctele de zgromot

Algorithm	PCA	PCA	Expansion	Expansion
Mode	ALL	NNP	ALL	NNP
ARI	0.601	0.621	0.488	0.537
AMI	0.726	0.733	0.665	0.689
Homogeneity	0.728	0.743	0.651	0.681
Completeness	0.751	0.757	0.715	0.745
V-measure	0.727	0.734	0.666	0.690
CHS	11389	14677	9624	11051
DBS	1.335	1.283	1.563	1.700
SS	0.307	0.418	0.235	0.351

5.4. Evaluarea performanței autocodificatoarelor LSTM

Pentru a verifica aplicabilitatea LSTM-urilor, se va aplica PCA pe intervalul 20-40 al potențialelor de acțiune și se va compara cu rezultatul codului rezultat din LSTM (de mărime 20) pentru intervalul 20-40.

După cum a fost menționat în subcapitolul 4.7, LSTM-ul primește un potențial de acțiune sub formă de mai multe input-uri și va rezulta în același număr de coduri. De asemenea s-a observat ca se obține rezultate mai bune fără suprapunere.

Putem observa empiric că LSTM-urile nu par să aducă informație utilă în preprocesarea potențialelor de acțiune din figura 5.2. Locația grupurilor este aceeași și nu apar grupuri adiționale separate de LSTM care nu sunt separate de PCA.

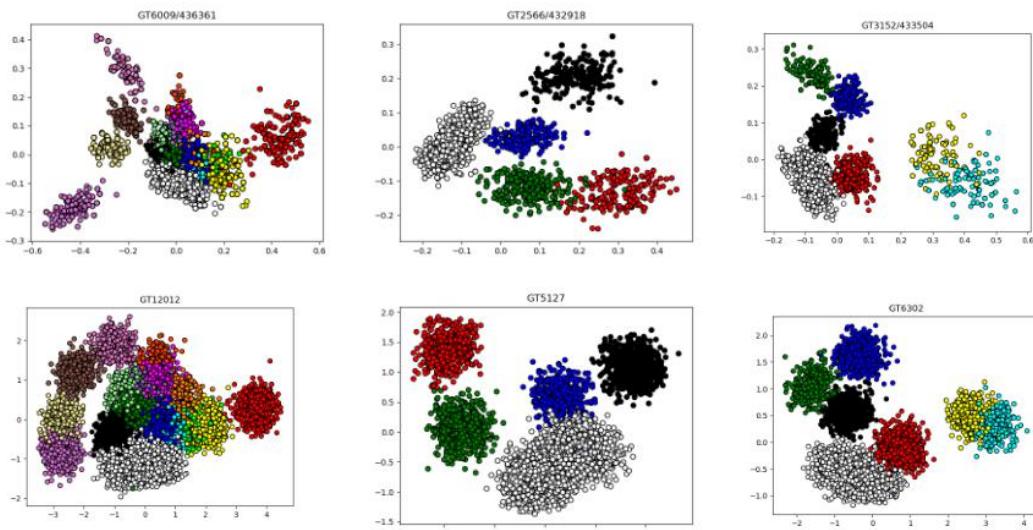


Figure 5.2 - Pe prima linie este rezultatul LSTM-ului cu 4 inputuri de mărime 20, după aplicarea PCA pentru vizualizare a codului care contine amplitudinea (interval 20-40) pentru simularile 1,4,24 iar pe a doua linie este extragerea intervalului fără preprocesare și aplicarea directă a PCA-ului

Pentru a verifica corectitudinea observației făcute mai sus, am modificat potențialul de acțiune în două moduri și am re-aplicat comparația dintre LSTM cu PCA și doar PCA.

Primul mod este aplicarea min-max scaling-ului, aceasta scalare a fost aplicată pe toate potențialele folosite la învățare pentru a asigura ca punctele de minim și maxim global vor fi toate la aceeași amplitudine, astfel renunțând la informația conținută de amplitudine.

Al doilea mod este aplicarea Z-score-ului, care va amplifica minimul și maximul global al potențialelor astfel scoțând amplitudinea în evidență. O vizualizare a rezultatelor pe un potențial de acțiune a acestor metode poate fi vizualizată în figura 5.3.

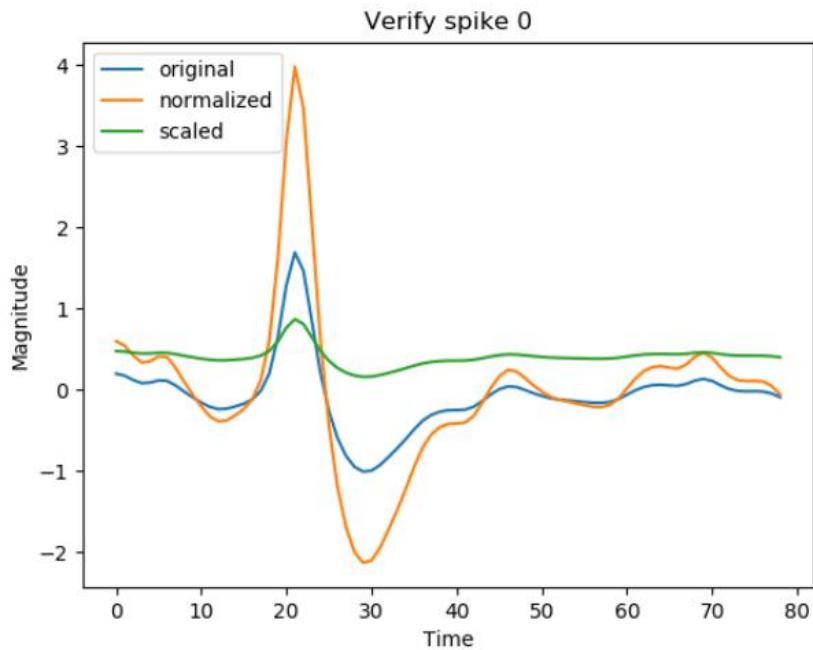


Figure 5.3 - Linia albastră reprezintă potențialul de acțiune original, linia portocalie este potențialul de acțiune după aplicarea Z-score, iar linia verde după min-max scaling

Următorul pas a fost aplicarea LSTM-ului și a PCA-ului pentru cele două noi seturi de potențiale de acțiune și compararea rezultatelor. Astfel, dacă rezultatele sunt echivalente și metodele sunt echivalente iar LSTM-ul nu aduce extragerea de caracteristici necesară. Rezultatele LSTM-ului și PCA-ului pentru primul mod se pot vizualiza în figura 5.4, iar al doilea mod în 5.5.

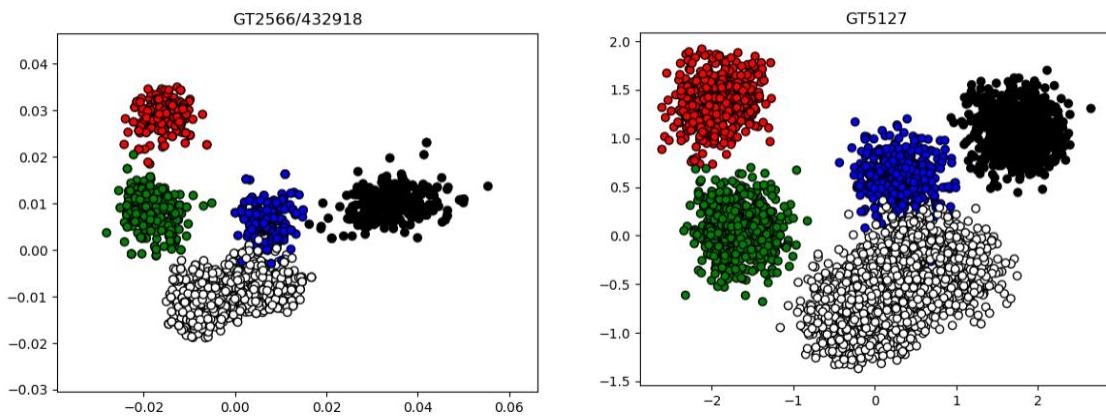


Figure 5.4 - În stânga rezultatul LSTM-ului pentru potențialele de acțiune cu min-max scaling iar în dreapta a PCA-ului

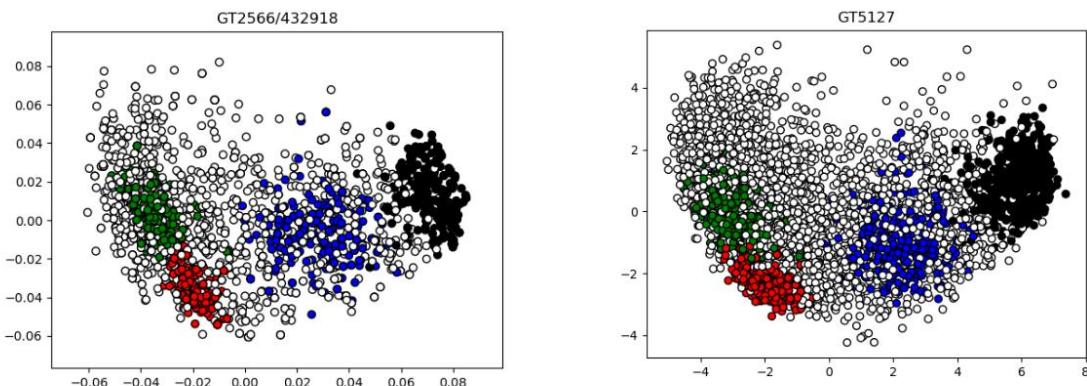


Figure 5.5 - În stânga rezultatul LSTM-ului pentru potențialele de acțiune cu Z-score iar în dreapta a PCA-ului

Ultimul pas a fost validarea rezultatelor folosind metricile prezentate în capitolul 3. Antrenarea modelului a fost făcută folosind 50% (selectat aleator) din fiecare grup din fiecare simulare. Scorurile au fost calculate pentru rezultatele obținute pentru punctele rămase (tot 50%) din fiecare simulare pentru PCA și LSTM. Media scorurilor poate fi vizualizată în tabela 5.12.

Table 5.12 - Rezultatele Autocodificatorului cu LSTM în comparație cu PCA, folosind 2 moduri, antrenare cu punctele din toate grupurile din seturile de date și fără punctele de zgromot, media afișată este făcută prin alegerea celor mai bune rezultate

Algorithm	PCA	PCA	LSTM AE	LSTM AE
Mode	ALL	NNP	ALL	NNP
ARI	0.439	0.531	0.537	0.446
AMI	0.657	0.685	0.672	0.629
Homogeneity	0.727	0.766	0.727	0.705
Completeness	0.626	0.648	0.657	0.605
V-measure	0.660	0.688	0.676	0.634
CHS	4997	7432	4738	4982
DBS	1.400	1.227	1.736	1.493
SS	0.269	0.420	0.302	0.341

5.5. Evaluarea SBM-ului îmbunătățit

Îmbunătățirea descrisă în capitolul anterior se referă la numărul de chunk-uri, acestea sunt folosite în toți pașii următori ai algoritmului. În varianta originală a algoritmului, numărul de chunk-uri era determinat de formula PN^D , unde PN este partitioning number-ul iar D numărul de dimensiuni. În aceasta varianta numărul de chunk-uri nu mai este determinat de această formulă ci de dispersia punctelor în spațiul de caracteristici. Aceste rezultate sunt afișate în tabela 5.13.

Table 5.13 – Numărul de chunk-uri create de versiunea originală și de cea cu graf, setul de date folosit este simularea 4, care conține 5127 de puncte, se poate observa că varianta pe graf nu trece de această limită și că numărul de chunk-uri este redus semnificativ pentru versiunea bazată pe grafuri

Numărul de dimensiuni	SBM original	SBM graf
2	625	343
3	15,625	1659
4	390,625	4072
5	9,765,625	4981
6	244,140,625	5111

Această îmbunătățire oferă posibilitatea de a rula algoritmul pentru mai multe dimensiuni fără a avea probleme cu memoria. În versiunea originală, dacă erau introduse peste 10 dimensiuni, ar fi fost nevoie de un sistem cu mai multă memorie RAM decât cele utilizate în general. Cu această îmbunătățire, numărul de chunk-uri nu poate deveni mai mare decât numărul de puncte din setul de date, astfel asigură faptul că un număr ridicat de dimensiuni nu necesită memorie adițională. Această îmbunătățire afectează și timpul de procesare, rezultatele sunt afișate în tabela 5.14.

Table 5.14 – Timpul de procesare pentru versiunea originală și de cea cu graf, setul de date folosit este simularea 4, care conține 5127 de puncte

Dimensions/ Time(s)	SBM original	SBM graph
2	0.076	0.146
3	0.709	1.253
4	0.767	1.449

Capitolul 5

5	6.558	1.131
6	190.845	2.710

Capitolul 6. Concluzii

6.1. Obiectivul

Gruparea Potențialelor este un pas important pentru înțelegerea funcționării creierului la nivel celular. Datorită evoluției metodelor de înregistrare, Gruparea Potențialelor poate să aducă mai multă informație. Gruparea este o etapă importantă pentru a distinge automat grupurile de potențiale de acțiune între ele și de a le atribui unui neuron. Această etapă este realizată datorită etapei anterioare acesteia și anume Extragerarea Trăsăturilor. Prin Extragerarea Trăsăturilor se conturează formele grupurilor, fără a avea rezultate satisfăcătoare în acest pas, Gruparea devine irealizabilă.

6.2. Contribuții

Contribuțiiile personale aduse în acest proiect pot fi împărțite în următorul fel: contribuții pentru proiect și organizarea echipei și contribuții în domeniul Grupării Potențialelor:

- Dezvoltarea extragerii și procesării potențialelor de acțiune într-o formă utilizabilă ușor pentru proiect
- Support adus pentru diferitele dificultăți întâlnite de echipă și organizarea muncii colaborative pe Git
- Studiul pe bază a tuturor metodelor pentru a oferi suport
- Organizarea refactorizărilor necesare pentru a ușura utilizarea proiectului
- Dezvoltarea și validarea metodelor menționate în capituloanele anterioare
 - Autocodificatoare simple
 - Autocodificatoare preantrenate
 - Autocodificatoare expandate
 - Autocodificatoare LSTM
 - Îmbunătățirea SBM-ului

6.3. Rezultate Obținute

Validarea rezultatelor a fost făcută în comparație cu PCA, cea mai uzual folosită metodă de Extragerarea Trăsăturilor și Reducerea dimensionalității. Din metodele prezentate, Autocodificatoare care folosesc LSTM-uri au oferit pentru media scorurilor a celor 95 de simulări un rezultat de 6 din 8 scoruri mai mari decât cele oferite de PCA pe aceleași date. Autocodificatoare simple cu diferențe alinierii pentru media scorurilor a celor 95 de simulări a rezultat în 6 până la 8 scoruri mai mari decât cele oferite de PCA.

Îmbunătățirea SBM-ului a adus o micșorare a timpului de procesare și a memoriei necesare pentru seturi de date cu mai multe dimensiuni.

6.4. Dezvoltări Ulterioare

Un prim pas ar fi aplicarea metodelor descrise pe mai multe seturi de date reale și de a analiza performanțele algoritmilor în comparație cu cele existente în literatură. Acest

pas ar ajuta la demonstrarea corectitudinii metodelor cu toate ca validarea rezultatelor ar fi mai complicată pentru date reale înregistrate extracelular.

Până în acest punct al proiectului au fost explorați pașii de Extragerea Trăsăturilor și Grupare ai Grupării Potențialelor. O posibilă direcție de continuare ar fi investigarea pasului de Detectare a Potențialelor sau chiar cel de Filtrare. De asemenea, pașii de Extragerea Trăsăturilor și Grupare nu au fost epuizați, sunt multe metode care pot fi încercate, combinate sau chiar create.

În această lucrare SBM a fost îmbunătățit pentru seturi de date cu dimensionalitate ridicată, dar complexitatea pe numărul de dimensiuni a rămas exponențială. Aceasta complexitate ar putea fi îmbunătățită.

O problema întâlnita des în partea de grupare este overclustering-ul. Ar trebui studiate și probate alte metode de Grupare sau metode de unire a grupurilor.

Bibliografie

- [Ardelean2019] E.-R. Ardelean, A. Stanciu, M. Dînsoreanu, R. Potolea, and C. Lemnaru, “*Space Breakdown Method A new approach for density-based clustering*” in IEEE 15th International Conference on Intelligent Computer Communication and Processing, 2019, doi: 10.1109/ICCP48234.2019.8959795.
- [Baldi2012] P. Baldi “*Autoencoders, Unsupervised Learning, and Deep Architectures*”, Department of Computer Science, University of California, Irvine (2012)
- [Bear2007] M. F. Bear, B. W. Connors, and M. A. Paradiso, “*Neuroscience - Exploring the Brain*” in Lippincott Williams & Wilkins, Third Edit., 2007, pp. 23–98.
- [Coporîie2021] A. Coporîie, “*Metode de preprocesare spectrală a datelor pentru extragerea trăsăturilor folosind autocodificatioare*”, Dissertation Thesis, 2021
- [Deborah2010] L. Jegatha Deborah, R. Baskaran, & A. Kannan, “*A Survey on Internal Validity Measure for Cluster Validation*” Int. J. Comput. Sci. Eng. Surv., vol. 1, no. 2, pp. 85–102, 2010, doi: 10.5121/ijcses.2010.1207.
- [Ester1996] M. Ester, H. P. Kriegel, J. Sander & X. Xu, “*A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*”, Institute for Computer Science, University of Munich, 1996
- [Gui2020] A.-M. Gui, “*Spike Sorting Approaches using Fourier Transform-based Features*”, License Thesis, 2020
- [Heckbert1995] P. Heckbert “*Fourier Transforms and the Fast Fourier Transform (FFT) Algorithm*”, Notes 3, Computer Graphics 2, 15-463, 1995
- [Hristache2020] A. Hristache, “*Spike Sorting Approaches using Signal Processing Methods*”, License Thesis, 2020
- [Hochreiter1997] S. Hochreiter & J. Schmidhuber, “*Long Short-Term Memory*”, Neural Computation 9 (8):1735-1780, 1997
- [Lazea2020] D.-G. Lazea, “*Spike Sorting Approaches using Wavelet and Derivative-based Features*”, License Thesis, 2020
- [Lyons2008] R. Lyons, “*Quadrature Signals : Complex , But Not Complicated*” Dspguru.Com, 2008
- [MacQueen1967] J. MacQueen, “*Some methods for classification and analysis of multivariate observations*”, Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, 281--297, University of California Press, Berkeley, Calif., 1967
- [Mishra2017] S. Mishra, U. Sarkar, S. Taraphder, S. Datta, D. Swain, R. Saikhom, S. Panda & M. Laishram, “*Principal Component Analysis*”. International Journal of Livestock Research. 1. 10.5455/ijlr.20170415115235, 2017
- [Moca2019] V. V. Moca, A. Nagy-Dăbâcan, H. Bârzan, and R. C. Mureşan, “*Superlets: time frequency super-resolution using wavelet sets,*” bioRxiv, p. 583732, 2019, doi: 10.1101/583732.
- [Paraskevopoulou2013] S. E. Paraskevopoulou, D. Y. Barsakcioglu, M. R. Saberi, A. Eftekhar, and T. G. Constandinou, “*Feature extraction using first and second derivative extrema (FSDE) for real-time and hardware-efficient spike sorting*” J. Neurosci. Methods, vol. 215, no. 1, pp. 29–37, 2013, doi: 10.1016/j.jneumeth.2013.01.012.

- [Pedregosa2011] F. Pedregosa et al., “*Scikit-learn: Machine Learning in Python*” J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011
- [Pedreira2012] C. Pedreira, J. Martinez, M. J. Ison, & R. Quian Quiroga, “*How many neurons can we see with current spike sorting algorithms?*” J. Neurosci. Methods, vol. 211, no. 1, pp. 58–65, 2012, doi: 10.1016/j.jneumeth.2012.07.010
- [Pinaya2020] L. Pinaya, W. H., Vieira, S., Garcia-Dias, R., & Mechelli, A. “*Autoencoders. Machine Learning*”, 193–208. doi:10.1016/b978-0-12-815739-8.00011-0, 2020
- [Quiroga2007] R. Quiroga, “*Spike sorting*” Scholarpedia, 2007, doi: 10.4249/scholarpedia.3583.
- [Rendon2011] E. Rendón, I. Abundez, A. Arizmendi & E. M. Quiroz. “*Internal versus External cluster validation indexes*”, International Journal of Computers and Communications Issue 1, Volume 5, 2011
- [Rey2015] H. G. Rey, C. Pedreira, & R. Quian Quiroga, “*Past, present and future of spike sorting techniques*” Brain Res. Bull., vol. 119, pp. 106–117, doi: 10.1016/j.brainresbull.2015.04.007, 2015
- [Sagheer2019] A. Sagheer, M. Kotb. “*Unsupervised Pre-training of a Deep LSTM-based Stacked Autoencoder for Multivariate Time Series Forecasting Problems*”. Sci Rep 9, 19038. <https://doi.org/10.1038/s41598-019-55320-6>, 2019
- [Santos2009] J. M. Santos & M. Embrechts, “*On the use of the adjusted rand index as a metric for evaluating supervised classification*” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 5769 LNCS, no. PART 2, pp. 175–184, , doi: 10.1007/978-3-642-04277-5_18, (2009)
- [Scipy2020] SciPy, “*Short Time Fourier Transform*”, <https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.stft.html>, 2020
- [Serra] X. Serra, “*Audio Signal Processing for Music Applications*”, <https://www.coursera.org/lecture/audio-signal-processing>.
- [Vinh2009] N. X. Vinh, J. Epps, & J. Bailey, “*Information theoretic measures for clusterings comparison: Is a correction for chance necessary?*” Proc. 26th Int. Conf. Mach. Learn. ICML 2009, pp. 1073–1080, 2009.
- [Yang2009] Z. Yang, Q. Zhao, & W. Liu, “*Improving spike separation using waveform derivatives*” J. Neural Eng., vol. 6, no. 4, doi: 10.1088/1741- 2560/6/4/046006, 2009

Anexa 1 – Informații Relevantе

Glosar

Termenul în engleză	Traducerea folosită
Spike Sorting	Gruparea potențialelor
Spike	Potențial de acțiune
Overshoot	Depășire superioară
Undershoot	Depășire inferioară
Rising Phase	Faza de creștere
Falling phase	Faza de descreștere
Feature Extraction	Extragerea Trăsăturilor
Dimensionality reduction	Reducerea Dimensionalității
Clustering	Gruparea
Cluster	Grup
Autoencoder	Autocodificator
Encoder	Codificator
Decoder	Decodor
Layer	Strat
Weight	Pondere
Bias	Abatere statistică
Pretrained	Preantrenat
Noise	Zgomot
Label	Etichetă
Ground truth	Etichetarea reală
Shift	Translatare
Bin/Chunk	Coș

Dataset	Set de date
Sparse	Risipit

Tabel 1. Detalii despre simulările setului de date

Numărul simulării	Număr de grupuri	Număr de potențiale
1	16	12012
2	19	12784
3	12	9277
4	4	5127
5	15	11758
6	11	10186
7	17	13999
8	2	3494
9	19	15653
10	20	15149
11	20	14982
12	20	13488
13	10	7792
14	3	4507
15	9	9683
16	8	7556
17	9	9098
18	7	7004
19	18	13807
20	14	11186
21	4	4293
22	6	7101
23	17	13374
24	6	6302
26	13	12313
28	14	12081
29	3	4177
30	5	5210
31	11	8224

Anexa 1

32	10	9078
33	4	4169
34	8	6851
35	12	9481
36	18	12377
37	10	9954
38	13	11148
39	2	4719
40	7	7360
41	15	13347
42	9	6746
43	16	10871
45	8	6666
46	2	4202
47	15	10977
48	7	6823
49	11	9496
50	14	11211
51	18	11746
52	12	9255
53	3	4490
54	5	6923
55	19	14117
56	17	12784
57	6	6301
58	7	7860
59	2	4176
60	12	10158
61	17	13280
62	6	6649
63	13	7634
64	4	4890
65	17	13523
66	14	12572
67	13	11377
68	9	7170
69	15	12051
70	11	9395

Anexa 1

71	19	13459
72	16	11183
73	9	8436
74	20	13065
75	19	13908
76	3	4791
77	5	7084
78	10	8518
79	20	14536
80	10	7757
81	8	7937
82	14	10430
83	4	5206
84	6	5809
85	18	14512
86	18	13847
87	5	6671
88	8	6414
89	3	4573
90	12	10565
91	16	10821
92	11	8184
93	7	6113
94	2	4990
95	15	12462

Table 2 – Rezultatele separării prin pipeline pe simulari

Numărul simularii	Numărul de grupuri	Numărul de pași	Separare	Observații
8	2	6	Da	Separabile de la pasul 3 dar pragul este prea mare
39	2	6	Nu	Zgomotul a fost separat
46	2	1	Da	
59	2	6	Da	Separabile de la pasul 4 dar pragul este prea mare
94	2	1	Da	
14	3	9	Da	Separabile de la pasul 6 dar pragul este prea mare

29	3	6	Da	
53	3	6	Da	
76	3	6	Nu	Zgomotul și 1 grup au fost separate
89	3	6	Nu	Zgomotul și 1 grup au fost separate
4	4	3	Da	
21	4	2	Da	
33	4	6	Nu	1 grup a fost separat
64	4	1	Da	
83	4	1	Da	
30	5	9	Da	Separabile de la pasul 6 dar pragul este prea mare
54	5	6	Nu	Zgomotul și 2 grupuri au fost separate
77	5	6	Nu	Zgomotul și 1 grup au fost separate
87	5	9	Nu	Zgomotul și 2 grupuri au fost separate
22	6	6	Nu	Zgomotul și 3 grupuri au fost separate
24	6	6	Nu	Zgomotul și 2 grupuri au fost separate
57	6	6	Nu	Zgomotul și 2 grupuri au fost separate
62	6	9	Nu	2 grupuri au fost separate
84	6	6	Nu	Zgomotul și 2 grupuri au fost separate

Anexa 2

Această anexă conține prima iterație a articolului "Addressing Challenges in Density-based Clustering via Feature Extraction using Autoencoders" care a fost trimisă pentru review la jurnalul Mathematics.

Article

Addressing Challenges in Density-based Clustering via Feature Extraction using Autoencoders

Eugen-Richard Ardelean¹, Andreea Coporiie², Mihaela Dinsoreanu³, Rodica Potolea⁴, Camelia Lemnaru⁵ and Raul Cristian Muresan⁶

¹ Department of Computer Science, Technical University of Cluj-Napoca; ardeleaneugenrichard@gmail.com

² Department of Computer Science, Technical University of Cluj-Napoca; coporiie.andreea@gmail.com

³ Department of Computer Science, Technical University of Cluj-Napoca; Mihaela.Dinsoreanu@cs.utcluj.ro

⁴ Department of Computer Science, Technical University of Cluj-Napoca; Rodica.Potolea@cs.utcluj.ro

⁵ Department of Computer Science, Technical University of Cluj-Napoca; Camelia.Llemnaru@cs.utcluj.ro

⁶ Experimental and Theoretical Neuroscience Lab, Transylvanian Institute of Neuroscience; raul.muresan@gmail.com

* Correspondence: ardeleaneugenrichard@gmail.com; Tel.: 0770 106 978

Abstract: Spike sorting is a process aiming to cluster spikes captured from a recorded signal, from the brain, based on the firing neurons. The process is prone to errors that will affect the results in any of its many steps. Overlapping clusters is such a difficulty. They can be introduced by the phenomenon of electrode drift during recording or due to the inability of the chosen feature extraction method to find characteristics that separate said clusters. Our goal is to investigate how autoencoders could benefit the process of feature extraction in order to boost the performance of clustering algorithms. The validation of the results was done on a series of synthetic datasets. The results obtained from validation are comparable on average with the well-known algorithm, PCA, with which it was compared.

Citation: Lastname, F.; Lastname, F.; Lastname, F. Title. *Publications* **2021**, *9*, x. <https://doi.org/10.3390/xxxxx>

Academic Editor: Firstname
Lastname

Received: date

Accepted: date

Published: date

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Spikes represent the activity of neurons, and each neuron has a unique spike shape determined by the morphology of its dendrite tree. The shape of a spike can be affected by the distance and orientation of a neuron in relation to the recording electrode. [1] Extracellular recordings, done outside of a neuron, register the changes of the potential of the membranes of multiple neurons. Due to the recording of multiple such changes, the provenance of a spike remains unknown. Spike sorting tackles this problem by attempting to assign each spike to the neuron that fired it, by grouping spikes that have similar characteristics together.

Imbalanced clusters appear due to the different roles that neurons have, excitatory and inhibitory. These roles have different firing rates which generate a different number of spikes in a period of time, therefore the clusters become imbalanced. [2] The shape of a spike may be different whilst firing in quick succession from the shape of an isolated

discharge. The distance and orientation of a neuron with respect to the electrode can also modify its shape. [3] This difference can be amplified by the phenomenon called electrode drift. These are several challenges that make this problem difficult.

Our aim is to improve the performance of the clustering algorithms by providing separated clusters. A secondary goal is to find features that group together spikes in the conventional definition of a cluster that will simplify the task of clustering. Due to the challenges presented, we justify that it is sufficient to separate clusters that are unable to be distinguished by other methods.

The approach relies on experimenting with different variants of pre-processing and architectures and comparing the results with a state-of-the-art feature extraction method.

The rest of the paper is organized as follows: section 2 recounts the problems that we are dealing with and provide a description of the proposed approach. In section 3 we present the evaluation methods and the quantitative and qualitative analysis of the results. Section IV discusses common methods of feature extraction that are used, one of which is considered state-of-the-art and is used as point of reference. Section V discusses the limits of the approach and the conclusions we have reached.

2. Materials and Methods

2.1. Problem characterization

A spike is a signal, and a part of a longer signal, the recording. Being a signal, it is defined as a vector of size D. At the same time, a spike can also be viewed as a point in a D dimensional space. We start from the assumption that the number of characteristics that describe a spike is potentially infinite. This elevated dimensional space imposes high complexities for the clustering algorithms. Therefore, our interest lies in reducing the size of the spike while also keeping it distinguishable from the spikes of other neurons. This process is called Feature Extraction.

Because there is not a universal feature extraction method for all datasets, our goal is to study the features obtained through autoencoders in comparison with those of PCA. As stated in the introduction, there are many errors that may be introduced during the process of spike sorting that will result in overlapping clusters. In some cases, PCA will output clusters that have overlap. Therefore, we are looking into other methods that may separate the overlapping clusters.

2.2. Solution overview

Autoencoders are neural networks able to learn features from unlabeled data. The first part of an autoencoder, the encoder maps the input into a latent code, while the second part, the decoder, attempts to make a reconstruction of the input as output. Similar to other feature extraction methods, autoencoders discover a pattern of variation in the data that retains enough information to restore the input.[8] Figure 1 in the Appendix shows the difference between the original spike and its reconstruction.

The latent code contains relevant information about the original input and by reducing the size of the code with respect to that of the input, it mirrors a feature extraction method.

Being neural networks, the encoder and decoder allow for custom architectures to be created. Increasing the architecture complexity will result in more complex latent codes.

2.3. Detailed algorithm

2.3.1. General information about the models

The models that will be presented below have multiple hidden layers and the encoder and decoder have mirrored, symmetric architectures. The encoder and decoder layers have ReLU activation functions, while the code and output layers have the tanh activation function. In addition, the code layer contains L1 regularization of 10e-7 to prevent overfitting. The model uses Adam optimization with a learning rate of 0.0001 and the Mean Squared Error (MSE) as a loss function.

For the code and output layers, the tanh activation function has been chosen instead of the ReLU function because it allows for negative values, which is needed in the spike reconstruction. The hidden layers use the ReLU activation function. The input and output being the spike and its reconstruction, both signals, we have deemed that MSE is the most suited loss function.

The number of epochs for the training has been tested for values of 50, 100 and 500. The value of 100 has been chosen through validation and because of the high number of model trials and the required time for training. The evaluation of the performance of the 3 values for the number of epochs can be viewed in table I in subsection 3.4.1 of the Validation section.

This section presents multiple approaches and models that can be simplified to the diagram presented in figure 1.

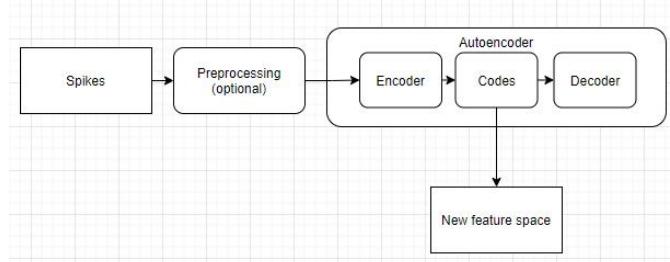


Figure 1. General model for the approach

2.3.2. Autoencoder with a code size of 2 (AE-2)

The first architecture contained eight hidden layers, for both the encoder and the decoder, in decreasing order with a code size of 2.

2.3.3. Autoencoders with different code sizes (AE-10, AE-20, AE-30, AE-40, AE-50)

These models differ from the AE-2 model by the architecture of the model. The number of hidden layers lowers as the code size increases, though successive hidden layers keep their characteristic of decreasing sizes. The code sizes chosen are 10, 20, 30, 40 and 50.

2.3.4. Pretrained autoencoders (PTAE-10, PTAE-20, PTAE-30, PTAE-40, PTAE-50)

In literature, pretraining appears as a possible way of improving results. The chosen approach for pretraining is in a greedy layer-wise manner. [10] The idea behind pretraining is to initialize the weights and biases of the model before training in order to avoid reaching local minimums because of the random initialization. Therefore, pretraining is made before the actual training of the model with the inputs.

The pretraining was done in the following manner:

- Training a layer and saving the weights and biases for that layer - for example, the first layer contains 70 neurons with an input and output of 79, therefore it would result in $79 \times 70 + 70 \times 79$ weights and 70 biases
- Saving the codes resulted - for the previous example, codes were of size 70
- For the pretraining of the next layer, the steps are repeated, but the input consists of the codes saved.
- These steps are repeated until the desired code size is reached as the layer
- The final step is to use all the weights and biases saved as the initial values for the training

The PTAE models have the same architecture as the AE models, the only difference is the additional step of pretraining.

The analysis of the results of the autoencoder models showed a similarity to the results of PCA. We inferred that the autoencoders react to variance as PCA does, therefore a switch to the frequency domain has been made in order to investigate characteristics that are invisible in the time domain, such as the distribution of the signal's energy.

2.3.5. FFT Autoencoders (FFT-AE)

To transform the inputs into the frequency domain, we have used the Fast Fourier Transform (FFT). FFT was applied for each of the spikes of a simulation, resulting in a real and imaginary part.

We have chosen the following criteria and evaluated all possible outputs:

- Input variations:
 - Original spike
 - Zero padded spike
 - Zero phased spike
 - Reduced spike
- Input type
 - Real
 - Imaginary
 - Magnitude
- Alignment
 - Align to average
 - No alignment

Using the product rule, 4^*3^*2 , we have trained 24 models in order to take all combinations into consideration.

The order of applying of the various choices is the following: the spikes are aligned to the desired index, they are modified with one of the presented variations. After these steps are finished, FFT is applied resulting in input types presented.

The original spike version applies no modification before applying FFT. The zero padded version adds to the end of the spikes zero until the nearest power of 2 is reached. The zero phased version, after adding the zeros at the end, shifts the spike until the zeros are in the middle. The reduced spike version deletes the last values from a spike until the nearest power of 2 is reached.

The real and imaginary parts are obtained as outputs of the FFT, the magnitude is calculated as the root of the sum of the squared real and squared imaginary parts.

As observed in the evaluation of the previous models (table 3 in subsection M4 of the Validation section), the code 20 offers the best results, and it was chosen as the code size for these 24 models.

Through empirical observation of the simulations and the performance evaluation presented in tables 12-15, we have seen that the magnitude offered the most consistent values for the metrics.

2.3.6. Windowed FFT Autoencoders (W-FFT-AE)

Consequently, we have chosen to add the utilization of windows in the preprocessing made before the FFT. These windows allow the amplification of certain parts of the spike depending on the alignment chosen. As before, we have evaluated multiple settings:

- Window type
 - Gaussian
 - Blackman
- Input type
 - Real
 - Imaginary
 - Magnitude
 - Phase
 - Power and phase concatenated
- Alignment
 - Align to average
 - No alignment
 - Align to global maximum
 - Align to global minimum

Using the product rule, $2^6 \times 4$, we have trained 48 models in order to take all combinations into consideration.

The order of applying of the various choices is the following: the spikes are aligned to the desired index, the window is applied, followed by a padding to the nearest power of 2 (presented in 3.4.5.). After these steps are finished, FFT is applied, resulting in input types presented.

Windows are applied as an element-wise multiplication with each of the spikes of the input data, thus they are required to have the same length as the spike (79).

The phase is calculated as the arctangent of the division of imaginary and real part. The power is the sum of the squared real and squared imaginary part.

Through empirical observation of the simulations and the performance evaluation presented in tables 16-23 from the appendix, the best results were offered by the magnitude of spikes that were aligned to the global maximum. The window type did offer different results but no conclusive best.

2.3.7. Ensemble Autoencoders (SAE)

Considering that the windows have had different but similar results by comparison, we have chosen to use ensemble methods to combine their results.

The chosen ensemble methods are the following:

- Algebraic
 - Mean
 - Sum

- Product
 - Max
 - Min
- Stacked Generalization

We have deemed that the median and weighted sum were unsuitable because of the fact that we only had two results to combine. Through similar logic, we have eliminated Bagging. And we have chosen not to use AdaBoost because the results from autoencoders will never agree, being numbers.

The algebraic methods just apply the mentioned rules for the two results of the two windows.

For Stacked generalization, 14 autoencoders have been created as the first tier. And another autoencoder that represents the second tier. The second tier autoencoder will receive as inputs the outputs of the autoencoders of the first tier.

Using several validation metrics, presented in table VI in subsection 3.4.7 of the Validation section, we have reached the conclusion that none of the algebraic methods improve the results of applying only one window, whether Blackman or Gaussian.

The ensemble stacking was only able to separate up to 2 clusters per simulation. The results of the ensemble stacking can be viewed in table VII of subsection 3.4.7 of the Validation section.

2.3.8. Slepian FFT Autoencoders (S-FFT-AE)

Slepian sequences have been applied in order to separate each spike into multiple pieces, each piece having a certain part of the original spike amplified. We have chosen to use 5 windows generated by the Slepian sequences with a half bandwidth of 4.0. The windows have the same size as the spikes, 79. Equivalent to the applying of the Gaussian or Blackman windows, an element-wise multiplication between the window and the spike is enacted.

Before the applying of the window, the spike has been aligned to the global maximum, followed by the applying of one of the Slepian sequences. After these steps, FFT is applied and the magnitude as calculated, which is used as the input to the autoencoder.

2.3.9. Cascaded Autoencoders (CAE)

The cascaded model was developed based on the Slepian model from subsection 2.3.8, thus the same input type was used. The signals generated by the Slepian sequences were applied as windows on the magnitude with an alignment to the global maximum.

A number of autoencoders have been generated, in this case 5, the same as the number of Slepian sequences generated. A sixth autoencoder, describes the cascaded model, as it receives the codes of the 5 previously mentioned autoencoders.

The signals generated by the Slepian sequences are shown in figure 4 in the Appendix. A detailed diagram of the model can be found in figure 5 in the Appendix.

2.3.10. Expanded Code Autoencoders (EAE)

EAE is an autoencoder that increases the number of neurons in each layer up to a code of 120, then applies PCA in order to reduce the feature space to 2 dimensions. In this method, we have tried to amplify the information contained in each spike.

2.3.11. LSTM Autoencoders (LAE)

This model contains LSTM layers instead of the dense layers used until this point. In our trials, choosing a code size of 2 for LSTM proved to be ineffective, because the model was unable to learn, all points resulted from the code being set at 0-0. Therefore, we have chosen the code size of 20, that offered the overall best results for the AE set of autoencoders. The LSTM model had a different architecture, using hidden layers that were powers of 2 between the input size and the code size.

Having the odd number of 79 as the length of the spike, in order to ease the split required for LSTMs we round 79 to the nearest multiple of 10. Thus, we have padded a zero at the end of each spike for LSTMs.

LSTMs require the splitting of the inputs into multiple parts, we have empirically chosen the two following ways:

- 4 inputs, no overlap (example: input 1 = 0-20, input 2= 20-40, input 3 = 40-60, input 4 = 60-80 of each spike)
- 8 inputs, half overlap (example: input 1 = 0-20, input 2 = 10-30, input 4 = 20-40 and so on)

3. Results

3.1. Description of datasets/simulations

Validation was made by comparing the different variants of autoencoders with PCA. The dataset chosen contains 95 simulations and it originates from Department of Engineering, University of Leicester UK. Each simulation is a dataset in and of itself. The creation of these simulations was based on the recordings from the neocortex of a monkey. The dataset contains 594 different spike shapes. [9] The article that presents the simulations also reviews different clustering algorithms and their results. Out of 20 different units, these algorithms were able to detect 10 in the best case.

The dataset was generated based on a real dataset recorded “in vivo”. The waveform contains 316 points originally sampled at 96 KHz, afterwards this frequency was reduced to 24KHz, therefore 79 points describe a spike for this dataset. Being a synthetic dataset, each of these spikes has a label, which allows for external metrics to be used.

As mentioned before, the dataset contains 95 simulations, each being a dataset in and of itself. Each simulation contains a multi-unit cluster, which is the noise and an additional number of clusters that varies between 2 and 20. Each unique number of clusters has 5 simulations.

The generated multi-unit cluster was added in order to increase the complexity of clustering for the tested algorithms. The simulated multi-unit contains 20 spike shapes. Each of the 20 neurons firing, being between 50-140 μ m from the electrode. The amplitude of the spikes was fixed to 0.5, with a firing rate of 5Hz, with each neuron having a firing rate mean of 0.25Hz following a Poisson distribution.

The rest of the clusters are single-unit and considered to be between 0 and 50 μ m from the electrode. The firing rate follows a Poisson distribution with a mean between 0.1 and 2Hz. The amplitudes follow a normal distribution and have been scaled to values between 0.9 and 2 to simulate real data. No overlapping spikes are present in the data, each spike having at least 0.3ms between them.

The dataset is contained in 2 types of files of .mat format. Thus, it requires a step of parsing to access each simulation in particular. There are 95 .mat files that represent the simulations and a .mat file for the ground truth.

The files were loaded as dictionaries, each simulation file contains the recording at the “data” key. The recording is found in the file as a one-dimensional vector. The ground truth file contains the start of each spike in a recording and their classes. To access the start of the spikes the “spike_first_sample” key has to be used, while “spike_classes” key accesses the labels. Each of these two requires the simulation number as index to access the starting indexes and the labels of a simulation.

Knowing the starting points for each spike in a simulation (from the ground truth file) and the length of a spike (79), the data of the .mat file of the simulation is traversed. Each spike is extracted as the start index and the following 78 points from the whole signal.

Figure 2 and 3 in the Appendix show two simulations from the dataset presented. Figure 2 shows simulation 4, that contains 4 clusters and the noise cluster (white), while figure 3, shows simulation 2 that contains 19 clusters and the noise cluster (white). The spikes presented in the figures have been preprocessed using PCA.

The alignment was made using the following formula:

$$\text{new_start}_{\text{spike}} = \text{old_start}_{\text{spike}} - (\text{index}_{\text{align}} - \text{peak}_{\text{spike}}), \quad (1)$$

The *index* in (1) represents the point to which all spikes will be shifted. For the alignment to the average, it is the average of the indexes of the amplitudes of all spikes. For the alignment to the global maximum and the global minimum the index to which all spikes are shifted has been chosen to be 39 (the middle of the signal). This was done to improve the amplifications offered by windows.

The *peak* in (1) represents the index at which the desired peak is found. For the average alignment and the alignment to global maximum, it is the index of the global maximum, while for the alignment to the global minimum it is equal to the index of the global minimum.

The alignment of the spikes brings standardization to the input data that will prevent the risk of overclustering and the deformation of clusters due to different parts of the spikes being used.

Alignment is applied as the first step of preprocessing.

3.2. Performance metrics

For the validation of the results, 8 metrics were used: Adjusted Rand Index (ARI), Adjusted Mutual Information (AMI), Homogeneity (Hom), Completeness (Com), V-Measure (VM), Calinski-Harabasz Score (CHS), Davies-Bouldin Score (DBS) and Silhouette Score (SS).

ARI (3) is an adjustment of the Rand Index (RI) metric in order to handle chances. ARI is an external clustering metric, therefore it requires a ground truth for the dataset. RI (2) makes comparisons between pairs of points to determine if it is an agreement, when the 2 points are in the same cluster for both the predicted and the true labels, or a disagreement, when they belong to different clusters.

$$\text{RI} = \frac{\text{agreements}}{\text{agreements} + \text{disagreements}} \quad (2)$$

$$\text{ARI} = \frac{\text{RI}-\text{ExpectedRI}}{\text{MaxRI}-\text{ExpectedRI}} \quad (3)$$

AMI (5), as ARI, is an adjustment of the Mutual Information (MI) metric with the use of entropy (H). MI (4) is calculated between 2 clusters U and V, where N is the size of the dataset and $|X|$ is the number of points in subset X.

$$\text{MI}(U, V) = \sum_{i=0}^{|U|} \sum_{j=0}^{|V|} \frac{|U_i \cap V_j|}{N} \log \frac{N|U_i \cap V_j|}{|U_i||V_j|} \quad (4)$$

$$\text{AMI} = \frac{\text{MI}(U, V) - E(\text{MI}(U, V))}{\text{average}(H(U), H(V)) - E(\text{MI}(U, V))} \quad (5)$$

V-Measure (6) is the harmonic mean of Homogeneity and Completeness. A cluster is considered to be homogeneous when all the points of that cluster are part of the same class. By switching the predicted and true labels completeness is obtained. Completeness is achieved when all the points of a class are part of the same cluster.

$$\text{V-Measure} = \frac{(1+\beta)*\text{Hom}*\text{Com}}{\beta*\text{Hom}+\text{Com}} \quad (6)$$

All the metrics presented until this point, are external metrics and require a ground truth to compare with the predicted labels. Further, all these metrics have bounded scores, ARI and AMI have [-1, 1], whilst Homogeneity, Completeness and V-Measure have [0, 1] with higher values being the more desirable.

DBS (8) finds the mean similarity between clusters, where similarity is defined by the distance between clusters and their sizes. The minimum value of this index is 0. The closer the result is to 0, the better separation exists between clusters.

$$R_{ij} = \frac{s_i - s_j}{d_{ij}} \quad (7)$$

$$DB = \frac{1}{k} \sum_{i=1}^k \max R_{ij} \quad (8)$$

CHS (9) also known as Variance Ratio Criterion, calculates the ratio between the intracluster and intercluster dispersion. The dispersion is defined as the sum of squared distances. For this metric, a higher value means a better result.

$$CHS = \frac{\text{tr}(Bk)}{\text{tr}(Wk)} \frac{n-k}{k-1} \quad (9)$$

SS (10) is calculated by measuring the mean distance between a point and the rest of the points of that cluster and the mean distance between the point and all the points of the nearest cluster. The score is bound between [-1, 1] where -1 represents an incorrect clustering, 0 overlapping clusters and 1 a dense clustering. SS aims for the standard concept of a cluster, dense and well separated therefore these will give a higher score.

$$SS = \frac{b-a}{\max(a,b)} \quad (10)$$

These last three metrics are internal and therefore do not require a ground truth to be used. However, because the dataset used contains the ground truth, these metrics were used with the ground truth not the clustering results to indicate the separability of clusters offered by the feature extraction method.

3.3. Settings

In the computation of the previously mentioned metrics, for the external metrics we have used the labels obtained from applying SBM [11] with a partitioning number of 25 and a centroid threshold of 5 and the labels provided by the dataset, the ground truth. For the internal metrics, the features resulted and the ground truth were used as inputs, in order to find how much separation was created by the feature extraction method with respect to the ground truth.

Through empirical observation, we have chosen to use 2 settings for the validation. In the dataset presented, the noise is generated as a multi-unit cluster, thus the settings are:

- ALL, the training and testing was made using all the points from a simulation / multiple simulations
- NNP (no-noise-points), the training and testing was made using all points but those of the noise cluster (in the ground truth noise has the label 0).

3.4. Measurements and Interpretation

3.4.1. General information about the models

Noise removal can modify the results of the evaluation. Noise spikes can be cut with an amplitude (highest peak) threshold of 0.8-1.0 depending on the simulation. By introducing noise removal, the learning time of the models is reduced, the performance of PCA is increased and the performance of the models presented varies.

Table 1. Comparison of different Number of Epochs on the results of the AutoEncoder for code size of 2 (Delta presents the difference between the autoencoder and pca)

	Delta	Delta	Delta
Epoch	50	100	500
ARI	-0.018	-0.0210	0.0042
AMI	-0.011	-0.0042	0.0045
Homogeneity	0.0021	0.0560	0.0244
Completeness	-0.0133	-0.0326	-0.012
V-Measure	-0.010	-0.0038	0.0045
CHS	1810.51	-2.86	-802.5
DBS	0.6360	0.2085	-0.111

SS	0.0535	0.3413	0.0723
----	--------	--------	--------

Table 2. Comparison of different Learning Rates on the results of the AutoEncoder for code size of 2 (Delta presents the difference between the autoencoder and pca) for 100 epochs

	Delta	Delta	Delta	Delta
Learning Rates	0.01	0.001	0.001	0.0001
ARI	-0.347	-0.014	0.022	-0.06
AMI	-0.353	0	0.0027	-0.048
Homogeneity	-0.351	0.058	-0.009	-0.077
Completeness	0.039	-0.046	0.03	-0.008
V-Measure	-0.353	0.0005	0.002	-0.004
CHS	142345	355.22	9461.8	15076
DBS	80608	0.11	-0.019	0.497
SS	-0.2	0.05	0.113	-0.022

Table 3.

	AE	PCA	AE	PCA
Alignment	To Average		No	
ARI	0.623	0.601	0.579	0.554
AMI	0.729	0.727	0.769	0.753
Homogeneity	0.719	0.728	0.811	0.810
Completeness	0.782	0.757	0.760	0.725
V-Measure	0.730	0.728	0.770	0.753
CHS	20865.365	11403.564	19439.729	12017.598
DBS	10315	1.334	0.831	1.003
SS	0.421	0.308	0.431	0.340

3.4.2. Autoencoder with a code size of 2 (AE-2)

Training the model on a simulation had a limited ability to reproduce spikes, only being able to reproduce abrupt peaks. Nevertheless, it was able to offer good results (in comparison with PCA) for the simulations it was trained on for simulations that have a high number of clusters 5-10+. It offered lower results for simulations with a low number of clusters (2-6). The values can be viewed in Table 4.

Even though the simulations were created in a similar manner with a lot of characteristics in common. A model trained on a simulation was rarely able to correctly separate clusters of a different simulation.

Taking into consideration the fact that the autoencoder was not able to reproduce spikes, we concluded that a code of 2 was a bottleneck for the architecture. Therefore, we have chosen to increase the code size and use PCA to reduce the resulting code further down to 2 dimensions.

Table 4. Comparison of different cluster sizes on the results of AE-2 and PCA (The values are the difference between the score of AE-2 and PCA)

Cluster Size	1 to 5	5 to 10	10 to 15	15 to 20
ARI	-0.2206	0.0165	0.0161	0.0338
AMI	-0.1466	0.0105	0.0372	0.0120
Homogeneity	-0.0846	0.0405	0.0811	0.0332
Completeness	-0.106	-0.008	-0.013	0.0123
V-Measure	-0.1453	0.105	0.003	0.0123
CHS	1113.05	169.363	1204.5	443.65
DBS	1.454	0.176	0.420	0.779
SS	-0.154	0.0505	0.0905	0.1208

3.4.3. Autoencoders with different code sizes (AE-10, AE-20, AE-30, AE-40, AE-50)

The AE models were created in parallel in order to compare the results of different code sizes between themselves and with PCA. The training was done on all simulations by randomly selecting 80% of a simulation (without noise points). The code sizes selected are 10, 20, 30, 40, 50. The models were trained for 100 epochs.

As expected, each increase in code size offered better reconstruction of the spike. However, none of the code sizes can reproduce the amplitude correctly for most spikes. Empirical observation on reconstructions points towards the steepness of the rising phase of a spike.

These models had an easier time with simulations that have a low number of clusters. For a very low number (2-3) of clusters in the simulation the best results are offered by a code size of 50. The rest by a lower code size of 20-30. Higher code sizes tend to elongate clusters for simulations with a higher number of clusters. However, all

code sizes had difficulties with simulations that have a higher number of clusters. Very few clusters are separated and in some cases the clusters separated are overlapped between themselves.

The average of the evaluations of the performance on all simulations can be viewed in table 5.

3.4.4. Pretrained autoencoders (PTAE-10, PTAE-20, PTAE-30, PTAE-40, PTAE-50)

The PTAE models have the same parameters as the previously mentioned 5 models. Training was done using 80% randomly selected of each simulation for 100 epochs.

The pretraining had no visible effect on the reproduction of the spikes, as before, the best reconstruction resulted from higher codes. And none of the code sizes was able to accurately restore the amplitude for most spikes.

Through empirical observations, the only difference obtained through pretraining is that lower codes (10-20) split clusters in two, best results are offered by codes of 30-40 and a code size of 50 elongates clusters.

We have compared the results of the autoencoder for each code size with those of PCA and none of the code sizes alone offers better results, on average, to that of PCA. The same comparison has been made with the 5 pretrained models of the same code sizes. The results of the validation metrics can be found in Table 5.

Table 5. Results of the AutoEncoder for each code size in comparison with the PreTrained Model and PCA (average on all simulations).

Cluster Size	Code size	AE	AE PT	PCA
ARI	10	0.394	0.414	
ARI	20	0.559	0.421	
ARI	30	0.523	0.499	0.621
ARI	40	0.413	0.527	
ARI	50	0.506	0.550	
AMI	10	0.563	0.573	
AMI	20	0.692	0.579	
AMI	30	0.679	0.659	0.733
AMI	40	0.599	0.659	
AMI	50	0.666	0.695	

The next validation was done using the results of all models with different code sizes, by choosing the best results out of all autoencoder models for each simulation. In this validation we are comparing the results of these autoencoders with those of PCA for

both ALL and NNP settings. The results presented are the average for all simulations and can be viewed in Table VI. It can be observed that the score of PCA is increased (for all metrics) when the noise cluster is removed, while that of the autoencoder models does not change significantly.

Table 6. Comparison between AutoEncoder (average of best performance for each simulation) and PCA results for both settings (average on all simulations)

	PCA	AE	PCA	AE
Setting	ALL	ALL	NPP	NPP
ARI	0.60	0.62	0.62	0.62
AMI	0.72	0.72	0.73	0.72
Homogeneity	0.72	0.72	0.74	0.71
Completeness	0.75	0.75	0.75	0.75
V-Measure	0.72	0.72	0.73	0.72
CHS	11389	9892	14677	9800
DBS	1.33	1.77	1.28	1.79
SS	0.30	0.37	0.41	0.37

3.4.5. FFT Autoencoders (FFT-AE)

The results of the evaluation of the 24 models can be found in tables XIII through XVI in the appendix. The results present the average of the performance on all simulations.

It can be observed that the magnitude offered the most consistent values for the metrics. Even though the real part seemed to have better scores in some variations. However, the clusters were less separated than those of the previous models.

3.4.6. Windowed FFT Autoencoders (W-FFT-AE)

The results of the evaluation of the 48 models can be found in tables 22 through 24 in the appendix. The results present the average of the performance on all simulations.

It can be seen that the best results were offered by the magnitude of spikes that were aligned to the global maximum. The window type did offer slightly different results but no conclusive best.

3.4.7. Ensemble Autoencoders (SAE)

For Stacked generalization, we have split the dataset into 16 subsets. 14 of these subsets are used to train the first tier of models, 7 with the Gaussian window applied and 7 with the Blackman window. While 2 of them are kept for the second tier. As there are 2 windows, to 8 of these 16 subsets the Blackman window has been applied, and Gaussian to the rest.

The outputs (reconstructed spikes) of these 14 autoencoders, have been concatenated with the 2 subsets left for the second tier and were fed as input for the second tier autoencoder.

The ensemble methods have had visually underwhelming results and it was the same case for the results of the validation metrics. None of the algebraic methods was able to surpass the results of applying only one of the windows. The exact numbers are shown in Table 7.

Table 7. Evaluation of the Algebraic Ensemble Methods in Comparison with just the windows and PCA

	ARI	AMI	SS
Ensemble - Mean	0.757	0.599	0.206
Ensemble - Sum	0.457	0.599	0.206
Ensemble - Prod	0.266	0.430	0.053
Ensemble - Min	0.397	0.556	0.170
Ensemble - Max	0.446	0.582	0.187
Blackman	0.483	0.640	0.263
Gaussian	0.499	0.627	0.236
PCA - NNP	0.621	0.733	0.418

Table 8. Evaluation of the Stacking Model with all inputs of type Gaussian and Blackman

	ARI	AMI	SS
Blackman	0.360	0.507	0.145
Gaussian	0.334	0.47	0.116

3.4.8. Slepian FFT Autoencoder (S-FFT-AE)

The evaluation of the model that used Slepian was made on the magnitude of the FFT result with an alignment to the global maximum.

By applying the Slepian sequences, we have trained a model with 5 times more input data, that was able to separate for each simulation up to 2 clusters, in some cases these clusters being overlapped. Seeing that the results were underwhelming, we have increased the code size to 50 from the previous 20, with no improvement. The results of this model can be viewed in table 9.

Table 9. Evaluation of the Slepian trained model

Metric	Score

ARI	0.331
AMI	0.490
Homogeneity	0.533
Completeness	0.493
V-Measure	0.492
CHS	3523.39
DBS	3.145
SS	0.131

3.1.9. Cascaded Autoencoders (CAE)

Each window has been applied to all spikes, thus generating 5 datasets, one for each autoencoder. Each of these subsets was fed into an autoencoder that had a code size of 20. The 5 results of these autoencoders for each spike are concatenated into a vector of size 100 that is fed into a sixth autoencoder that will also generate a code of 20, resulting in a cascaded model.

None of the 5 autoencoders with separated windowed inputs was able to separate more than 3 clusters for a simulation. This was the same case for the cascaded one. The clusters that were separated from the rest, if more than one, were overlapped between themselves. Iterative use of autoencoders on the overlapped clusters did not yield separation.

The results of each autoencoder can be seen in table 10.

Table 10. Evaluation of the 5 Slepian Window Autoencoder and the Evaluation of the final cascaded model

Metric	Window1	Window2	Window3	Window4	Window5	Final
ARI	0.106	0.147	0.176	0.200	0.324	0.234
AMI	0.287	0.289	0.337	0.347	0.479	0.407
Homogeneity	0.235	0.310	0.346	0.375	0.488	0.398
Completeness	0.508	0.314	0.391	0.365	0.518	0.486
V-Measure	0.290	0.292	0.340	0.349	0.481	0.410
CHS	8920.92	1432.16	1830.23	1954.19	3610.31	2630.70
DBS	6.905	7.350	5.192	5.475	3.639	4.305

SS	0.026	-0.012	0.021	-0.00297	0.114	0.083
----	-------	--------	-------	----------	-------	-------

3.4.10. Expanded Code Autoencoders (EAE)

Noticing that the results offered by reducing the feature space were underwhelming, we have implemented the opposite.

This method offered the best reproduction of the spike, but nevertheless it was not able to reconstruct the amplitude for most spikes. The separation of clusters was similar to that of PCA for most simulations.

The results of the expanded model, although perceivably similar with those of PCA, have been shown through the validation metrics that are actually worse in Table 11.

Table 11. Evaluation of the Expanded model (average on all simulations) compared to PCA

	PCA	PCA	Expanded	Expanded
Setting	ALL	NPP	ALL	NPP
ARI	0.601	0.621	0.488	0.537
AMI	0.726	0.733	0.665	0.689
Homogeneity	0.728	0.743	0.651	0.681
Completeness	0.751	0.757	0.715	0.745
V-Measure	0.727	0.734	0.666	0.690
CHS	11389	14677	9624	11051
DBS	1.335	1.283	1.563	1.700
SS	0.307	0.418	0.235	0.351

3.4.11. LSTM Autoencoder (LAE)

The training of the LSTM model was done by randomly selecting 50% of each cluster of each simulation.

The results of the no overlap version were similar to those of PCA and in some simulations offered more separation. For the overlapped spikes version, it was able to separate up to 2 clusters in the best of cases.

The LSTM model has had in some cases better results than those of PCA, through visual observation the increase in scores seems to be offered by the shape of the noise cluster. The use of the 2 settings has confirmed this hypothesis, as the results of the LSTM model are lower than those of PCA for the NNP setting, these results can be viewed in Table 12.

Table 12. Comparison between LSTM and PCA results for both settings

	PCA	LSTM	PCA	LSTM
Setting	ALL	ALL	NPP	NPP
ARI	0.439	0.537	0.531	0.446
AMI	0.657	0.671	0.685	0.629
Homogeneity	0.727	0.727	0.766	0.705
Completeness	0.626	0.657	0.648	0.605
V-Measure	0.660	0.676	0.688	0.634
CHS	4997	4768	7432	4982
DBS	1.400	1.736	1.227	1.493
SS	0.269	0.302	0.420	0.341

4. Discussion

The most used algorithm in feature extraction in general, but also for spike sorting in particular is Principal Component Analysis (PCA) [4].

PCA[4] outputs new characteristics called Principal Components that are linear combinations of the input values. The reduction of the dimensionality of the feature space is made by solving a problem of eigenvalues and eigenvectors. PCA preserves the variance as much as possible while being able to reduce the number of features.

Independent Component Analysis (ICA) [5] is a linear unsupervised technique for dimensionality reduction. This method searches for independent components by the use of statistical properties.

Linear Discriminant Analysis (LDA) [6] is a supervised learning technique with the goal of increasing the inter-cluster distance and decreasing intra-cluster distance. For this method, it is assumed that the data has a Gaussian distribution.

T-distributed Stochastic Neighbor Embedding (t-SNE) [7] is non-linear dimensionality reduction method, it minimizes the divergence between input features and the reduced feature space by using pairwise probability similarities. The divergence of two distributions is calculated using KL divergence which is minimized by applying gradient descent.

5. Conclusions

The proposed method of Feature Extraction and its many variants had varied performances in comparison with PCA. The initial model had difficulties with separating clusters in simulations with a low number of clusters but was able to separate them for simulations with a high number of clusters (comparison for different cluster

sizes was presented in table 3). The models with increased code sizes that followed had backwards performance, managing to better separate in simulations with a lower number of clusters.

Transferring the input into the frequency domain has shown no improvement to the performance from any of the many variants we tested. (the performance evaluation can be viewed in tables 5-9 and 12-23).

Expanding the spikes in order to generate more information did not improve the performance of the model (performance on average presented in table 10).

Introducing LSTMs in the model did manage to improve the overall results, but the improvement has occurred because of its ability to gather the noise into a very tight cluster (other models will scatter the points and will generate different densities in one cluster or separate the noise into multiple clusters). The performance evaluation of the LSTM model is presented in table 11.

The research leading to these results has received funding from: NO Grants 2014–2021, under Project contract number 20/2020 (RO-NO-2019-0504), a grant from the Romanian National Authority for Scientific Research and Innovation, CNCS-UEFISCD (code PN-III-P2-2.1-PED-2019-0277), and a H2020 grant funded by the European Commission (grant agreement 952096, NEUROTWIN).

Supplementary Materials: The following are available online at www.mdpi.com/xxx/s1, Figure S1: title, Table S1: title, Video S1: title.

Author Contributions: For research articles with several authors, a short paragraph specifying their individual contributions must be provided. The following statements should be used “Conceptualization, X.X. and Y.Y.; methodology, X.X.; software, X.X.; validation, X.X., Y.Y. and Z.Z.; formal analysis, X.X.; investigation, X.X.; resources, X.X.; data curation, X.X.; writing—original draft preparation, X.X.; writing—review and editing, X.X.; visualization, X.X.; supervision, X.X.; project administration, X.X.; funding acquisition, Y.Y. All authors have read and agreed to the published version of the manuscript.” Please turn to the CRediT taxonomy for the term explanation. Authorship must be limited to those who have contributed substantially to the work reported.

Funding: Please add: “This research received no external funding” or “This research was funded by NAME OF FUNDER, grant number XXX” and “The APC was funded by XXX”. Check carefully that the details given are accurate and use the standard spelling of funding agency names at <https://search.crossref.org/funding>. Any errors may affect your future funding.

Data Availability Statement: In this section, please provide details regarding where data supporting reported results can be found, including links to publicly archived datasets analyzed or generated during the study. Please refer to suggested Data Availability Statements in section “MDPI Research Data Policies” at <https://www.mdpi.com/ethics>. You might choose to exclude this statement if the study did not report any data.

Acknowledgments: In this section, you can acknowledge any support given which is not covered by the author contribution or funding sections. This may include administrative and technical support, or donations in kind (e.g., materials used for experiments).

Conflicts of Interest: Declare conflicts of interest or state “The authors declare no conflict of interest.” Authors must identify and declare any personal circumstances or interest that may be perceived as inappropriately influencing the representation or interpretation of reported research results. Any role of the funders in the design of the study; in the collection, analyses or interpretation of data; in the writing of the manuscript, or in the decision to publish the results must be declared in this section. If there is no role, please state “The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results”.

Appendix A

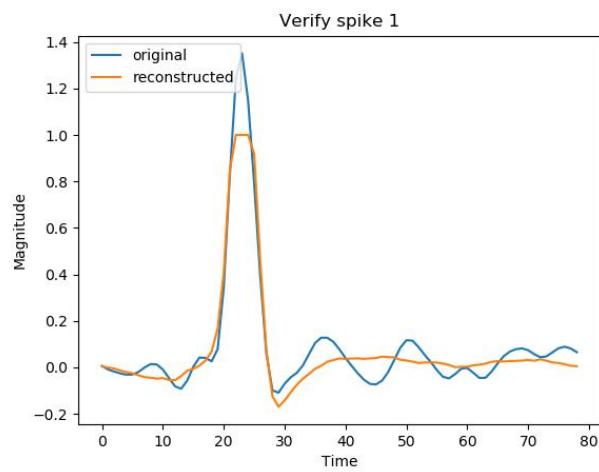


Figure A1. Original spike and the reproduction of it made by an autoencoder

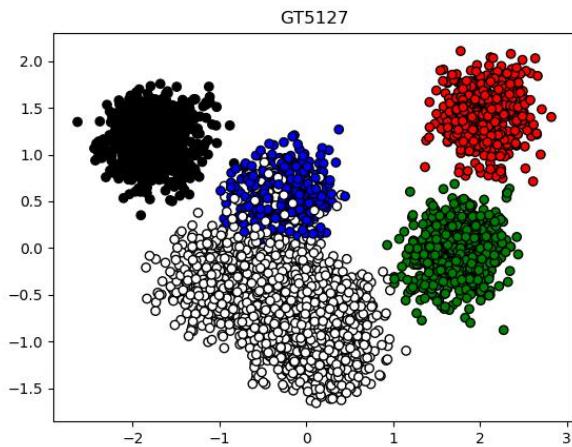


Figure A2. Figure 2 – Simulation 4

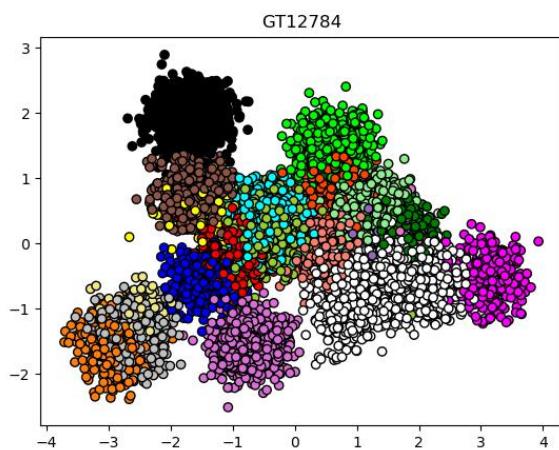


Figure A3. Figure 3 – Simulation 2

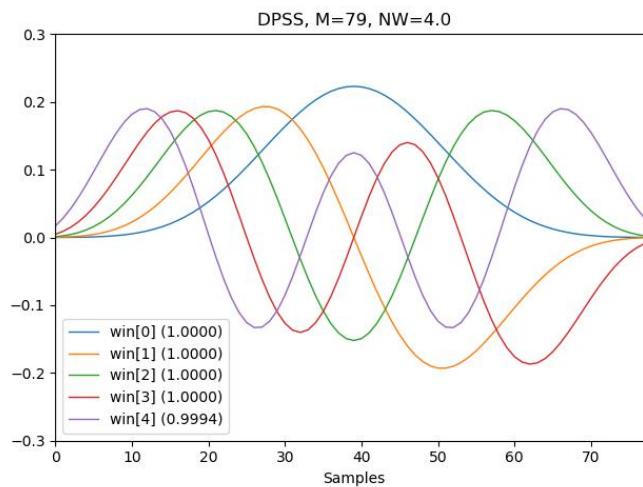


Figure A4. 5 slepiant sequences of length 79

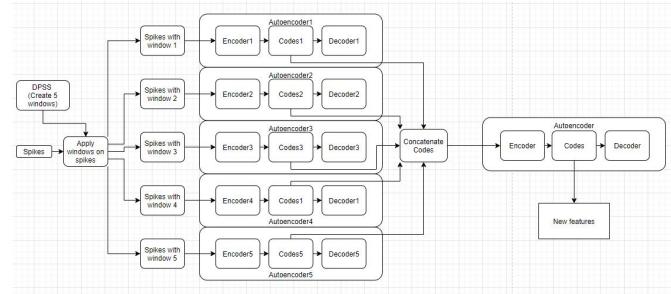


Figure A5. Detailed diagram of the cascaded model

Table A1. Evaluation of the FFT models with the original and padded input variants for all input types aligned to average

Input Variation	Original			Padded		
	Input type	Real	Imag	Magn	Real	Imag
Alignment				To Average		
ARI	0.341	0.206	0.309	0.319	0.238	0.277
AMI	0.499	0.355	0.475	0.477	0.39	0.459
Homogeneity	0.55	0.392	0.504	0.518	0.438	0.484
Completeness	0.494	0.363	0.493	0.483	0.382	0.486
V-Measure	0.501	0.358	0.477	0.48	0.393	0.461
CHS	4669.14	1622.77	4456.74	2746.89	1735.87	2885.78
DBS	2.817	4.275	3.614	2.862	4.536	3.200

SS	0.155	0.027	0.126	0.122	0.061	0.102
----	-------	-------	-------	-------	-------	-------

Table A2. Evaluation of the FFT models with the phased and reduced input variants for all input types aligned to average

Input Variation	Phased			Reduced			
	Input type	Real	Imag	Magn	Real	Imag	Magn
Alignment				To Average			
ARI	0.292	0.176	0.286	0.41	0.241	0.312	
AMI	0.436	0.337	0.455	0.563	0.402	0.457	
Homogeneity	0.469	0.343	0.49	0.602	0.428	0.489	
Completeness	0.443	0.389	0.466	0.57	0.433	0.472	
V-Measure	0.439	0.641	0.457	0.565	0.405	0.459	
2620.56	2085.98	336.26	5204.39	5204.39	2367.55	3112.28	
DBS	3.538	5.488	3.114	2.632	3.454	3.129	
SS	0.103	0.025	0.12	0.203	0.0807	0.199	

Table A3. Evaluation of the FFT models with the original and padded input variants for all input types with no alignment

Input Variation	Original			Padded			
	Input type	Real	Imag	Magn	Real	Imag	Magn
Alignment				No			
ARI	0.45	0.283	0.341	0.419	0.394	0.241	
AMI	0.611	0.44	0.506	0.588	0.553	0.42	
Homogeneity	0.651	0.504	0.505	0.616	0.595	0.461	
Completeness	0.617	0.424	0.569	0.602	0.561	0.428	

V-Measure	0.613	0.443	0.508	0.59	0.555	0.423
CHS	7564.51	2493.84	3785.36	5191.5	5792.53	2453.07
DBS	1.964	3.516	2.57	1.97	2.51	4.538
SS	0.274	0.094	0.167	0.232	0.198	0.076

Table A4. Evaluation of the FFT models with the phased and reduced input variants for all input types with no alignment

Input Variation	Phased			Reduced			
	Input type	Real	Imag	Magn	Real	Imag	Magn
Alignment				No			
ARI	0.355	0.176	0.294	0.431	0.34	0.332	
AMI	0.513	0.33	0.462	0.591	0.51	0.497	
Homogeneity	0.532	0.358	0.489	0.622	0.553	0.53	
Completeness	0.549	0.355	0.484	0.604	0.512	0.512	
V-Measure	0.515	0.333	0.464	0.593	0.512	0.499	
CHS	480.94	1545.36	3589.26	10781.77	5153.78	4560.14	
DBS	2.7680	5.096	3	2.616	2.679	3.111	
SS	0.160565	0.160535	0.123	0.231	0.151	0.15	

Table A5. Evaluation of the Windowed FFT models with the padded variants for all input types aligned to average using a Blackman window

Window Type	Blackman						
	Input type	Real	Imag	Magn	Power	Phase	Concat
Alignment				To Average			
ARI	0.35	0.39	0.318	0.336	0.034	0.392	
AMI	0.514	0.543	0.496	0.499	0.085	0.457	

Homogeneity	0.543	0.562	0.521	0.553	0.103	0.522
Completeness	0.532	0.57	0.53	0.493	0.09	0.437
V-Measure	0.516	0.545	0.498	0.502	0.09	0.46
CHS	4254.62	5302.16	4659.05	4080.42	172.13	3579
DBS	2.909	2.616	2.951	3.128	13.229	3.534
SS	0.148	0.163	0.143	0.115	0.11	0.085

Table A6. Evaluation of the Windowed FFT models with the padded variants for all input types with no alignment using a Blackman window

Window Type	Blackman							
	Input type	Real	Imag	Magn	Power	Phase	Concat	
Alignment				No				
ARI	0.439	0.484	0.454	0.326	0.036	0.314		
AMI	0.609	0.645	0.6	0.496	0.087	0.477		
Homogeneity	0.621	0.648	0.617	0.57	0.106	0.533		
Completeness	0.656	0.689	0.625	0.471	0.091	0.467		
V-Measure	0.611	0.647	0.602	0.499	0.091	0.48		
CHS	7405.84	12576.2	9928.61	3347.28	188.64	3400.97		
DBS	2.078	1.955	1.982	3.746	13.109	3.38		
SS	0.265	0.282	0.45	0.123	0.111	0.105		

Table A7. Evaluation of the Windowed FFT models with the padded variants for all input types with the global maximum alignment using a Blackman window

Window Type	Blackman						
	Input type	Real	Imag	Magn	Power	Phase	Concat

Alignment		To Global Maximum				
ARI	0.429	0.343	0.503	0.325	0.05	0.33
AMI	0.584	0.504	0.644	0.486	0.12	0.491
Homogeneity	0.633	0.548	0.657	0.549	0.149	0.547
Completeness	0.577	0.508	0.672	0.467	0.117	0.478
V-Measure	0.586	0.507	0.646	0.488	0.125	0.494
CHS	6512.11	2590.63	9261.91	3905.36	204.82	3272.52
DBS	2.058	3.533	1.772	3.257	12.413	3.28
SS	0.244	0.129	0.283	0.095	-0.107	0.115

Table A8. Evaluation of the Windowed FFT models with the padded variants for all input types with the global minimum alignment using a Blackman window

Window Type	Blackman					
	Input type	Real	Imag	Magn	Power	Phase
Alignment	To Global Minimum					
ARI	0.22	0.207	0.352	0.181	0.019	0.142
AMI	0.256	0.245	0.465	0.306	0.045	0.265
Homogeneity	0.381	0.358	0.506	0.379	0.058	0.297
Completeness	0.365	0.373	0.457	0.278	0.049	0.268
V-Measure	0.359	0.347	0.467	0.311	0.05	0.27
CHS	1329.76	1235.68	1229.72	680.03	51.727	542.04
DBS	6.244	6.654	6.542	7.453	33.488	8.62
SS	-0.0173	-0.037	0.073	-0.057	-0.129	-0.096

Table A9. Evaluation of the Windowed FFT models with the padded variants for all input types aligned to average using a Gaussian window

Window Type		Gaussian					
Input type		Real	Imag	Magn	Power	Phase	Concat
Alignment		To Average					
ARI		0.396	0.364	0.319	0.275	0.018	0.325
AMI		0.474	0.535	0.476	0.416	0.042	0.477
Homogeneity		0.477	0.555	0.513	0.496	0.047	0.516
Completeness		0.542	0.57	0.486	0.384	0.054	0.482
V-Measure		0.476	0.537	0.479	0.42	0.046	0.48
CHS		3411.77	6765.79	4224.14	2456.46	88.48	4116.97
DBS		3.703	2.969	3.381	4.934	19.033	3.787
SS		0.104	0.165	0.13	0.047	-0.106	0.115

Table A10. Evaluation of the Windowed FFT models with the padded variants for all input types with no alignment using a Gaussian window

Window Type		Gaussian					
Input type		Real	Imag	Magn	Power	Phase	Concat
Alignment		No					
ARI		0.45	0.411	0.342	0.345	0.02	0.351
AMI		0.62	0.596	0.51	0.509	0.052	0.525
Homogeneity		0.612	0.588	0.52	0.566	0.058	0.561
Completeness		0.68	0.652	0.549	0.497	0.0767	0.53
V-Measure		0.621	0.5979	0.512	0.512	0.056	0.527
CHS		17420.49	15104.6	8381.75	455.38	121.09	6101.48
DBS		2.547	3.181	2.906	3.23	14.998	3.271
SS		0.258	0.24	0.164	0.122	-0.116	0.153

Table A11. Evaluation of the Windowed FFT models with the padded variants for all input types with the global maximum alignment using a Gaussian window

Window Type		Gaussian					
Input type		Real	Imag	Magn	Power	Phase	Concat
Alignment		To Global Maximum					
ARI		0.481	0.386	0.474	0.383	0.0059	0.309
AMI		0.61	0.555	0.626	0.543	0.141	0.473
Homogeneity		0.627	0.564	0.64	0.608	0.172	0.536
Completeness		0.629	0.609	0.657	0.521	0.139	0.455
V-Measure		0.611	0.557	0.628	0.545	0.146	0.476
CHS		5690.51	6196.11	10023.819	5184.69	243.7	2498.55
DBS		2.097	2.439	1.928	3.044	10.673	2.784
SS		0.23	0.194	0.277	0.182	-0.102	0.122

Table A12. Evaluation of the Windowed FFT models with the padded variants for all input types with the global minimum alignment using a Gaussian window

Window Type		Gaussian					
Input type		Real	Imag	Magn	Power	Phase	Concat
Alignment		To Global Minimum					
ARI		0.288	0.268	0.198	0.221	0.0165	0.173
AMI		0.398	0.435	0.347	0.351	0.0476	0.296
Homogeneity		0.438	0.434	0.35	0.406	0.0547	0.336
Completeness		0.389	0.473	0.382	0.334	0.0608	0.289

V-Measure	0.401	0.437	0.35	0.355	0.052	0.301
CHS	888.41	1870.28	1811.89	1133.06	71.72	915.31
DBS	8.304	5.951	8.92	6.439	22.161	7.95
SS	0	0.025	0	-0.01	-0.12	-0.04

References

1. Bear, M. F.; Connors, B. W., Paradiso, M. A., "Neuroscience - Exploring the Brain" in Lippincott Williams&Wilkins, Third Edit., 2007; pp. 23–98.
2. Buzsaki, G. "Rhythms of the Brain", 2009
3. Lewicki , M., "A review of methods for spike sorting: the detection and classification of neural action potentials" in Network: Computation in Neural Systems, 1998
4. Mishra , S.; Sarkar, U.; Taraphder, S.; Datta, S.; Swain, D.; Saikhom , R.; Panda, S.; Laishram, M. "Principal Component Analysis". International Journal of Livestock Research. 1. 10.5455/ijlr.20170415115235, 2017
5. Hyvärinen, A. "Independent component analysis: recent advances" Phil. Trans. R. Soc. A.3712011053420110534, 2013
6. Tharwat, A.;Gaber, T.; Ibrahim, A.; Hassani, A.E. "Linear discriminant analysis: A detailed tutorial", Ai Communications 30(2):169-190, DOI: 10.3233/AIC-170729, 2017
7. Zhou, H.; Wang, F.; Tao, P., "t-Distributed Stochastic Neighbor Embedding Method with the Least Information Loss for Macromolecular Simulations" Journal of Chemical Theory and Computation, DOI: 10.1021/acs.jctc.8b00652, 2018
8. Pinaya , L.; Vieira W. H.; Garcia-Dias, S.; Mechelli, A. "Autoencoders. Machine Learning", 193–208. doi:10.1016/b978-0-12-815739-8.00011-0, 2020
9. Pedreira, C.; Martinez, J.; Ison, M. J.; Quiroga, R. "How many neurons can we see with current spike sorting algorithms?" J. Neurosci. Methods, vol. 211, no. 1, pp. 58–65, doi: 10.1016/j.jneumeth.2012.07.010, 2012
10. Sagheer , A.; Kotb, M. "Unsupervised Pre-training of a Deep LSTM-based Stacked Autoencoder for Multivariate Time Series Forecasting Problems". Sci Rep 9, 19038. <https://doi.org/10.1038/s41598-019-55320-6>, 2019
11. Ardelean, E.-R.; Stanciu, A.; Dinsoreanu, M.; Potolea, R.; Lemnaru, C. "Space Breakdown Method A new approach for density-based clustering" in IEEE 15th International Conference on Intelligent Computer Communication and Processing, 2019, doi: 10.1109/ICCP48234.2019.8959795.