UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

SPECIALIZAREA INFORMATICĂ

Lucrare de licență

DETECȚIA DE ANOMALII ÎN SETURI DE DATE CU DISTRIBUȚII VARIABILE FOLOSIND DEEP LEARNING

Absolvent
Alex-Mihai Serafim

Coordonator științific
Prof. Cristian Rusu

București, iunie 2023

Rezumat

Detecția de anomalii într-un set de date are ca scop recunoașterea unor puncte de interes, care trezesc suspiciuni legate de natura lor și care pot prezenta un pericol pentru un sistem. Are implicații importante în domenii precum analizarea fraudelor bancare, prevenirea atacurilor asupra sistemelor informatice, cât și în aplicații ce țin de vederea artificială. Majoritatea metodelor existente sunt bazate pe algoritmi probabilistici, modele liniare sau clustering. Ca un model de detecție a anomaliilor să fie considerat performant, este necesar ca acesta să versatil și fiabil, adică să poată fi aplicat unei varietăți de seturi de date și să se descurce în scenarii în care distribuția datelor se schimbă.

În această lucrare urmăresc să studiez performanța algoritmilor de detecție a anomaliilor în contexte de variabilitate a distribuției, cu un accent pus pe o metodă nouă în domeniu. Voi folosi atât metode supervizate cât și metode nesupervizate pentru a cuantifica efectul etichetelor în cadrul antrenării prin măsurarea diferențelor de acuratețe. Modelele vor fi testate în două tipuri de situații: (i) folosind seturi de date sintetice; aici voi propune doi algoritmi de generare a datelor care să simuleze o variabilitate a distribuțiilor și (ii) folosind seturi de date naturale bazate pe trafic de rețea și imagini.

Am folosit metode de reducere a dimensionalității datelor pentru a vizualiza variabilitatea seturilor de date și pentru a confirma că metodele de generare concepute de mine sunt valide. De asemenea, am căutat să compar modele cât mai variate, iar metoda nouă testată folosește *deep learning*.

Concluzia de bază în urma experimentelor a fost că, deși variabilitatea datelor este un fenomen complex, greu de modelat și de măsurat, prin adresarea sa și prin construirea modelelor de detecție a anomaliilor în jurul acestei probleme, ea poate fi mitigată. În acest context, metodele supervizate au obținut performanțe semnificativ mai bune, însă în detecția de anomalii avem rareori acces la etichetele exemplelor de antrenare.

Abstract

Anomaly detection in a dataset has the purpose of identifying points of interest, which raise suspicions about their provenance and which could pose a danger to a system. Anomaly detection has meaningful implications in areas such as bank fraud analysis, preventing attacks in computer systems, as well as in applications regarding computer vision. Most of the existing methods rely on probabilistic algorithms, linear models or clustering. Efficient anomaly detection models have to be both versatile and reliable, meaning they can work on a multitude of datasets and they can handle distribution shift scenarios.

In this work I aim to assess the performance of algorithms conceived for anomaly detection in distributionally shifted environments, with an emphasis on a novel approach. I will use both supervised and unsupervised methods to quantify the effect of labels at train time by measuring the differences in accuracy. The models will be tested in two scenarios: (i) using synthetic datasets, where I will propose two data generation algorithms to simulate distribution shift and (ii) using natural datasets based on network traffic and images.

I have used dimensionality reduction algorithms to visualize this shift and to confirm that the generation methods proposed hereby are valid. Also, I aimed to compare a variety of methods and the novel approach described uses deep learning.

The main conclusion of the experiments was that, while distribution shift is a complex, hard to model and hard to measure phenomenon, by addressing it and conceiving anomaly detection models around this problem, it can be mitigated. In this context, supervised methods have reported significantly better performance, but in anomaly detection we seldom have access to training labels.

Cuprins

1 Introducere **6**

1.1 Context 6

1.2 Scopul lucrării 7

1.3 Contribuții personale și obiective 7

1.4 Structura lucrării 8

2 Preliminarii **9**

2.1 *Distribution shift* 9

2.2 Metode nesupervizate de detecție a anomaliilor 11

2.3 Metode supervizate de detecție a anomaliilor 12

2.4 PyOD 12

2.5 t-SNE 12

2.6 ACR-DSVDD 13

2.6.1 DeepSVDD 13

2.6.2 *Adaptive Centered Representations* 13

2.6.3 *Batch Normalization* 15

3 Seturi de date sintetice **16**

3.1 Seturi de date generate aleator 16

3.1.1 Metoda de generare 16

3.1.2 Rezultate 18

3.2 Seturi de date generate prin rotația matricei de covarianță 19

3.2.1 Matricea de rotație 19

3.2.2 Metoda de generare 20

3.2.3 Rezultate 23

3.2.4 Ablație 23

3.2.5 Concluzie 23

4 *Anoshift [6]* - *distribution shift* într-un set de date natural **25**

4.1 Motivație 25

4.2 Descrierea setului de date 25

4.2.1 Proveniența datelor 25
4.2.2 Procesarea datelor 26
4.2.3 Vizualizarea shift-ului 27
4.3 Metode folosite și rezultate 28
4.3.1 Alegerea hiperparametrilor 30
4.4 Concluziile experimentului 30

5 *Distribution shift* în imagini **32**
5.1 CIFAR-100 32
5.2 Metode folosite și rezultate 32
5.3 Efectul inițializării parametrilor 34

6 Concluzii **36**

Bibliografie **38**

Capitolul 1

Introducere

1.1 Context

În contextul analizei datelor, detecția de anomalii (AD - *anomaly detection*) constă în identificarea instanțelor care deviază de la comportamentul standard al datelor și care pot fi diferențiate de celelalte instanțe din setul de date. Definiția unei anomalii și criteriile de identificare ale acesteia diferă în funcție de problema pe care încercăm să o rezolvăm, însă în general considerăm anomalie orice exemplu care nu respectă distribuția întregului set de date, caz în care anomalia poate fi numită și un *outlier* al distribuției (*outlier* și *anomaly* sunt frecvent folosite alternativ pentru a se referi la același concept în literatura de specialitate).

Există numeroase domenii pentru care algoritmii de detecție a anomaliilor prezintă interes, printre care:

- identificarea tranzacțiilor frauduloase în domeniul bancar sau a tentativelor de evaziune fiscală [13]
- prevenirea atacurilor și intruziunilor prin analizarea traficului de rețea [1]
- identificarea defectelor de fabricație, în combinație cu vederea artificială [22]
- în medicină și sănătate publică prin analizarea datelor pacienților. [7]

Marea majoritate a algoritmilor de detecție concepuți de-a lungul timpului sunt fie bazați pe metode probabiliste, modele liniare, metode de clustering sau metode bazate pe densitatea spațiului atributelor. Însă odată cu creșterea dimensionalității seturilor de date (mai multe seturi de date bazate pe imagini), a apărut nevoia de algoritmi care să aibă performanță bună în spații *sparse*; algoritmi bazați pe deep learning. Astfel, au fost concepute metode bazate pe AutoEncoders [3] sau pe generare adversarială [23] [32], care funcționează în regim nesupervizat.

Odată cu aceste metode au apărut și seturi de date *benchmark* din ce în ce în mai complexe. ADBench [10] este cea mai complexă colecție de astfel de *benchmark*-uri, cu 57 de seturi de date diferite și 30 de algoritmi implementați.

O problemă majoră în ingineria sistemelor de *machine learning* este o degradare a performanței atunci când un model este lansat în producție. De cele mai multe ori sursa acestui eșec este variabilitatea distribuțiilor datelor (*distribution shift*). Detecția de anomalii este în special susceptibilă acestui fenonem deoarece la antrenarea modelului se fac anumite asumpții legate de natura datelor care nu se adeveresc întotdeauna. Ceea ce la faza de antrenare este considerat comportament normal, poate reprezenta anomalie pe măsură ce datele suferă modificări. În cazuri extreme, modelele simpliste care nu sunt actualizate își pot inversa predicțiile și să aibă performanță mai slabă decât alegerea aleatoare. Problema a început să atragă mai multă atenție în ultimii ani: autorii din [16] au identificat și analizat *distribution shift* în 10 seturi de date diferite.

1.2 Scopul lucrării

Lucrarea are ca scop analizarea capacității de generalizare și identificare de *distribution shift* în detecția de anomalii prin măsurarea performanței algoritmilor consacrați în două contexte diferite: unul sintetic și unul natural. Va fi detaliată o metodă nouă bazată pe deep learning și proiectată special pentru *distribution shift*. Voi arăta astfel importanța abordării acestei probleme în construirea unui algoritm de detecție a anomaliilor.

1.3 Contribuții personale și obiective

Ca rezultat al cercetării acestei probleme, am setat două obiective care vor îndeplini scopul lucrării:

(i) Comparația metodelor supervizate și nesupervizate pentru problema distribution shift-ului. Este de interes să observăm comportamentul algoritmilor atunci când furnizăm informație suplimentară și dacă va crește astfel capacitatea de detecție.

(ii) Analizarea performanței unui algoritm nou de AD pe un dataset real, care prezintă *shift* și care nu poate fi detectat în mod regulat de metodele curente (cea mai bună acuratețe raportată este sub 50%)

Contribuția personală asupra acestui subiect constă în elaborarea a două metode de generare a unui dataset sintetic cu anomalii, dintre care una modelează variabilitatea datelor; în testarea unei colecții de metode de AD pe acest dataset; precum și analizarea și interpretarea rezultatelor și evidențierea unor concluzii relevante pentru acest domeniu. Codul sursă public al acestei lucrări se găsește la adresa aceasta.

1.4 Structura lucrării

Prezenta lucrare este formată din capitolele următoare:

1. Introducere — prezentarea generală a domeniului, cu importanța sa, metodele curente utilizate și noțiuni teoretice de bază
2. Preliminarii — definirea fenomenului de *distribution shift*, metode de detecție folosite și concepte auxiliare
3. Seturi de date sintetice — formalizarea a două metode de generare de seturi de date, împreună cu rezultatele obținute pe aceste seturi de date
4. Anoshift — descrierea unui set de date natural, discuție pe baza rezultatelor
5. *Distribution shift* în imagini — extinderea conceptului de *shift* pe imagini
6. Concluzii — un rezumat al observațiilor și posibile direcții de cercetare viitoare

Capitolul 2

Preliminarii

2.1 *Distribution shift*

În sensul general, o distribuție dictează comportamentul datelor, fiind o funcție care definește probabilitea ca un punct să ia o anumită valoare. Problema *distribution shift*-ului apare atunci când distribuția datelor de antrenare diferă față de distribuția datelor de test. Fenomenul este întâlnit în mai toate aplicațiile de *machine learning* și este în special important în contextul detecției de anomalii. Pot fi identificate trei tipuri principale de *distribution shift* (considerând distribuțiile de antrenare $P_{antrenare}(x, y)$, distribuțiile de test $P_{test}(x, y)$, $P(x)$ distribuția datelor și $P(y)$ distribuția etichetelor):

- *label shift* (figura 2.1) - variabilitate în ceea ce privește distribuția etichetelor între antrenare și test: $P_{antrenare}(x, y) = P_{test}(x, y)$ dar $P_{antrenare}(y) \neq P_{test}(y)$. Acest tip de *shift* este o problemă în cazurile în care avem prezumția că y implică x, spre exemplu în cazul clasificatorului bayesian naiv.

- *concept shift* - variabilitate conceptuală, în sensul în care definiția unei clase se poate schimba: $P_{antrenare}(x, y) \neq P_{test}(x, y)$. Acest tip de shift este greu de combătut și necesită o reevaluare a definițiilor claselor.

- *covariate shift* (figura 2.2) - variabilitate în distribuția atributelor între antrenare și test: $P_{antrenare}(x, y) = P_{test}(x, y)$, dar $P_{antrenare}(x) \neq P_{test}(x)$. Apare în acele probleme în care x implică y și este cel mai întâlnit tip de *shift*; este tipul de *shift* pe care îl voi studia în continuare.

Covariate shift este o problemă importantă deoarece reprezintă un caz particular de *Domain Adaptation* [29], adică proprietatea unui algoritm de a generaliza și de a se adapta unei distribuții de test diferită dar din același domeniu cu distribuțiile de antrenare; proprietate folositoare în contextul AD deoarece dorim să construim un model care să funcționeze pentru mai multe tipuri de input-uri sau pentru input-uri variabile în timp.

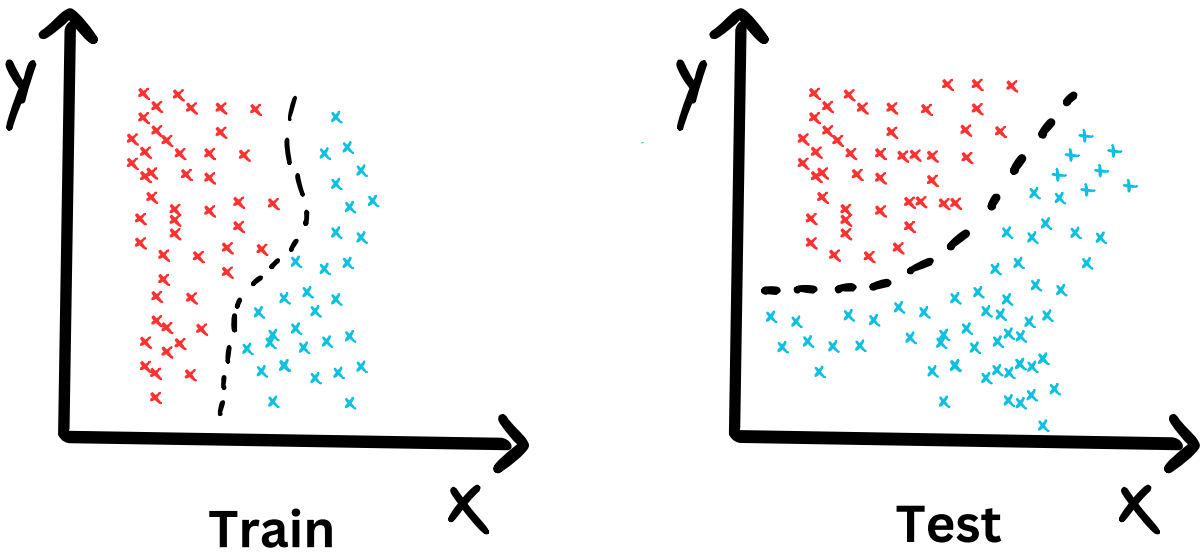


Figura 2.1: Ilustrarea calitativă a *label shift*-ului pentru o clasă de antrenare și una de test.

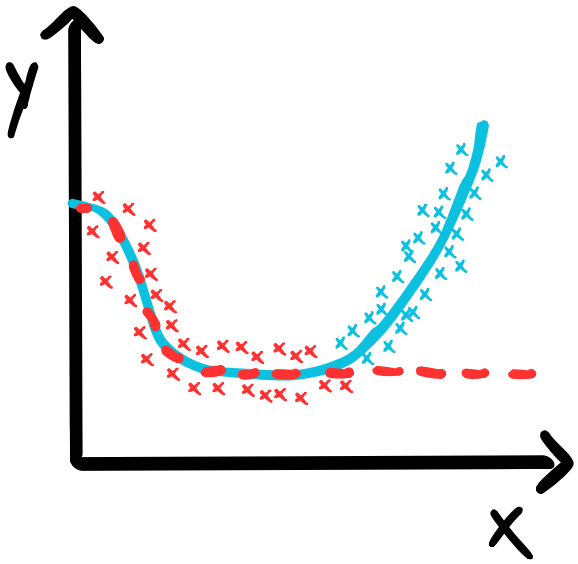


Figura 2.2: Ilustrarea calitativă a *covariate shift*-ului. Punctele roșii reprezintă datele la antrenare, iar cele albastre datele la test.

Din moment ce lucrăm cu date multidimensionale, *covariate shift*-ul poate fi greu de identificat și vizualizat, astfel că sunt necesare metode de reducere a dimensionalității. *Principal Component Analysis* [25] reduce dimensionalitatea și minimizează pierderea de informație, găsind dimensiunile care introduc cea mai multă variantă în setul de date și păstrând cele mai relevante dimensiuni. PCA este folosit în unele metode de AD ca prim pas de preprocesare a datelor. În continuare, conform cu literatura de specialitate, voi face referire la *covariate shift* drept *distribution shift*.

2.2 Metode nesupervizate de detecție a anomaliilor

Problema detecției de anomalii variază semnificativ în funcție de aplicație. Este dificil de stabilit un prag după care un exemplu să fie clasificat sau nu drept anomalie, iar strictețea acestui prag este considerată de la caz la caz, în funcție de cât de strict este dorită clasificarea. Detecția de anomalii este asemănătoare cu detecția de noise, diferența fiind că noise-ul nu reprezintă instanțe malițioase, ci pur și simplu instanțe nedorite. [4]

Obținerea unui set de date etichetat pentru a face detecție supervizată de anomalii este o sarcină grea și costisitoare. Din acest motiv, cercetarea a fost concentrată asupra detecției nesupervizate și majoritatea metodelor din prezent funcționează în această manieră. Câteva metode de interes au fost selectate pentru a fi comparate cu detecția supervizată:

- LODA[28]: ansamblu de histograme care aproximează funcția densitate de probabilitate pe câte o dimensiune. Poate procesa seturi de date mari foarte rapid.
- ECOD[20]: metodă probabilistă care aproximează distribuția setului de date și calculează probabilitatea ca un punct să se afle în distribuție, pentru fiecare dimensiune
- INNE[2]: spre deosebire de alte metode bazate pe izolare, care folosesc diviziuni paralele cu axele, INNE izolează fiecare exemplu x_i în hipersfera sa și folosește mărimea hipersferei ca măsură a anomaliei; anomaliile se vor afla în zone mai puțin dense și vor avea o hipersferă mai mare
- OC-SVM[33]: SVM optimizat pentru clasificarea unei singure clase; se găsește cel mai mic hiperplan care să separe toate exemplele normale
- DeepSVDD[31]: găsește cea mai mică hipersferă care separă toate exemplele normale; bazat pe deep learning și stă la baza metodei descrise în continuarea lucrării.
- ACR-DSVDD [19]

2.3 Metode supervizate de detecție a anomaliilor

Detecția supervizată de anomalii este rareori folosită în practică din cauza absenței etichetelor și din cauza tendinței algoritmilor supervizați de a nu identifica tipurile noi de anomalii. Totuși, este relevant să observăm importanța etichetelor și a informațiilor suplimentare oferite de acestea în timpul antrenării. Algoritmii supervizați folosiți sunt următorii:

- XGBOD[37]: folosește detectori nesupervizați pentru a extrage reprezentări mai semnificative ale datelor cu care construiește un nou spațiu al atributelor; este aplicat apoi un clasificator asupra atributelor augmentate
- CD [5]: metodă bazată pe distanța lui Cook și cuantifică impactul unui singur punct asupra unui model de regresie

2.4 PyOD

PyOD [38] este o bibliotecă Python dedicată detecției și generării de outlieri prin distribuții multivariate. Conține 40 de algoritmi de AD (majoritatea unsupervised), atât algoritmi vechi (Cook's Distance [5], 1977), cât și algoritmi state-of-the-art (ECOD[20], 2022). Permite de asemenea generarea de outlieri pornind de la distribuții multivariate, însă am ales să construiesc propriile metode de generare pentru a avea mai mult control asupra parametrilor. PyOD permite prototiparea ușoară, în doar câteva linii de cod. Toți algoritmii primesc datele sub aceeași structură, deci modificările necesare pentru a rula un model nou sunt minime. Fiecare model permite modificarea hiperparametrilor specifici.

2.5 t-SNE

t-SNE — *t-distributed stochastic neighbor embedding* [24] — este o metodă de reducere a dimensionalității care permite vizualizarea punctelor din spații mari în spații de 2 sau 3 dimensiuni. Este un algoritm iterativ care urmărește să producă o reprezentare cât mai apropiată în spațiul simplu la fiecare iterație, urmând 2 pași:

- pentru un punct se calculează distanțele față de celelalte (de cele mai multe ori folosind distanța euclidiană), iar folosind această distanță se așează punctele pe o *t-distribuție* care are punctul de interes în centru; se calculează scorul de similaritate dintre puncte în funcție de așezarea lor pe distribuție
- se minimizează distanța dintre distribuția originală ($p(x)$) și cea proiectată ($q(x)$)

cu ajutorul *Kullback-Leibler divergence*:

$$D_{KL}(p(x)||q(x)) = \sum_{x \in X} p(x) ln \frac{p(x)}{q(x)}$$

Voi folosi t-SNE pe parcursul acestei lucrări pentru a ilustra *distribution shift*-ul între mai multe seturi de date.

2.6 ACR-DSVDD

2.6.1 DeepSVDD

SVDD (Support Vector Data Description) [35] este o metodă kernel asemănătoare cu SVM, însă care folosește o hipersferă în loc de un hiperplan ca să separe datele. Pentru ambii este nevoie totuși de *feature engineering* înainte de antrenare, iar în cel mai rău caz complexitatea crește pătratic cu numărul de exemple la antrenare [36].

Obiectivul metodei DeepSVDD este să găsească cea mai mică hipersferă care curpinde toate datele de antrenament învățând o transformare din spațiul de input al datelor într-un spațiu de output mai mic din punct de vedere al dimensionalității. Transformările exemplelor normale se vor situa în interiorul hipersferei, în timp ce exemplele anormale se vor situa în exterior, scorul de anomalie fiind definit drept distanța de la punct la centrul sferei. Pentru a implementa DeepSVDD-ul, este folosit un Autoencoder pentru a reduce dimensiunea input-ului și rețele convoluționale bazate pe arhitectura LeNet [18] pentru a reprezenta hipersfera.

2.6.2 *Adaptive Centered Representations*

Zero-shot learning este o paradigmă de *machine learning* care abordează problema clasificării unor clase care nu au fost observate în timpul antrenării. Conceptul este folositor în special pentru *Domain Adaptation*, astfel că implicit poate fi extins să combată *distribution shift* din moment ce scopul său principal este să generalizeze cât mai eficient. Modelele de *zero-shot learning* învață reprezentări ale claselor folosite în timpul antrenării și se bazează pe reprezentări asemănătoare ale datelor pentru a putea clasifica exemplele noi. Majoritatea metodelor de *zero-shot learning* din prezent se bazează pe *foundation models* — modele antrenate pe o cantitate foarte mare de date, care sunt de cele mai multe ori limitate la clasele pe care au fost antrenate (precum modelul preantrenat CLIP [30])—, iar metoda propusă în [19] este o alternativă *lightweight*, care nu necesită foarte multe date și care este bazată pe DeepSVDD.

Setup-ul de antrenare prezentat în [19] constă în utilizarea unor submulțimi distincte din setul de date, submulțimi separate fie conceptual în funcție de clasa pe care o reprezin-

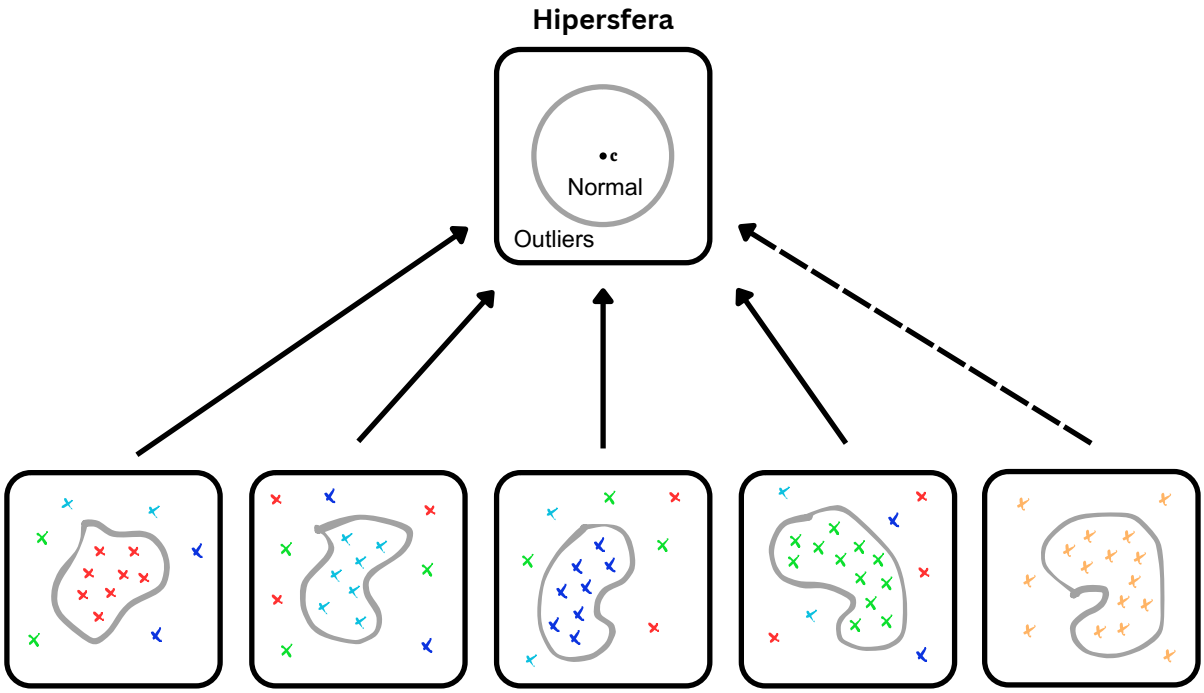


Figura 2.3: Modul de funcționare al ACR-DSVDD. Săgețile pline sugerează că setul a fost folosit la antrenarea modelului, iar săgeata punctată reprezintă inferența pentru un anumit set de test. Seturile de antrenare sunt contaminate cu exemple din celelate clase; seturile de test conțin doar exemple din clasa proprie.

tă, fie temporal în cazul în care dorim să testăm capacitatea de detectare de *distribution shift* a modelului. Modelul va converge apoi spre o soluție care rezolvă simultan toate subset-urile de antrenare. Această metodă de antrenare este denumită de autori *Adaptive Centered Representations (ACR)*, iar setup-ul de antrenare care folosește DeepSVDD va fi denumit în continuare ACR-DSVDD. Figura 2.3 ilustrează acest setup.

Capacitatea de generalizare a ACR-DSVDD-ului este îmbunătățită în continuare prin contaminarea fiecărui subset de antrenare D cu exemple anormale, care nu fac parte din distribuția subset-ului (și care fac parte din $\overline{D}$). Astfel, pentru ACR-DSVDD se definește următoarea funcție de *loss* pentru fiecare exemplu:

$$L(x) = \begin{cases} d(x, S), & \text{pentru } x \in D \\ 1/d(x, S), & \text{pentru } x \in \overline{D} \end{cases}$$

unde $d(x, S)$ este distanța euclidiană dintre punctul x și centrul hipersferei DeepSVDD-ului S, pe fiecare dimensiune.

ACR-DSVDD poate fi considerat așadar un algoritm *self-supervised*, în sensul în care folosește conceptul de contaminare a batch-urilor de antrenament îmbunătățind astfel capacitatea sa de adaptare, dar nu primește informații despre structura anomaliilor reale. Detecția anomaliilor se face la nivel de batch. Din moment ce algoritmul functionează

într-o manieră zero-shot, este nevoie de un batch întreg pentru a infera natura distribuției și pentru a putea pe baza acestei distribuții să prezică outlierii.

2.6.3 *Batch Normalization*

Este de asemenea important stratul de *Batch Normalization*[14] din implementarea DeepSVDD-ului; acesta normalizează reprezentările intermediare ale rețelei convoluționale și plasează exemplele normale cât mai aproape de centrul hipersferei.

Batch Normalization este o metodă de normalizare la nivel de batch a output-urilor straturilor intermediare dintr-o rețea neurală și care urmărește să combată *internal covariate shift*: modificarea distribuției de output de la un strat la altul. Din acest motiv, mecanismul de *Batch Normalization* poate acționa ca un detector de anomalii rudimentar, ducând exemplele din afara distribuției mai departe de medie.

Pentru un batch $B = \{x_i | i \in [1, m]\}$ de mărime m, *Batch Normalization* funcționează în felul următor:

- se calculează media și varianța

$$\mu = \frac{1}{m} \sum_{i=1}^{m} x_i$$

$$\sigma^2 = \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu)^2$$

- se calculează $x_i^{'}$ normalizând fiecare exemplu din batch (c este o constantă pentru stabilitate numerică, are valoare mică, de obicei $1e-3$; este adăugată pentru a ține sub control valoarea lui $x_i^{'}$ în cazul în care varianța tinde spre 0)

$$x_i^{'} = \frac{x_i - \mu}{\sqrt{\sigma^2 + c}}$$

- se evaluează activarea stratului de Batch Normalization y_i în funcție de $x_i^{'}$

$$y_i = \gamma x_i + \beta$$

Aici γ și β sunt parametri ai modelului și vor fi actualizați la backpropagation, iar calculele se efectuează pentru fiecare dimensiune din *feature space*.

În mod obișnuit, *Batch Normalization* este dezactivat la faza de inferență a modelului; pentru *ACR-DSVDD* în schimb este esențială folosirea lui la testare pentru a putea utiliza reprezentările învățate în cadrul antrenării cu distribuții înrudite cu cele de test.

Capitolul 3

Seturi de date sintetice

În ambele metode ce urmează, datele au fost generate folosind o distribuție normală (gaussiană). Forma generală a unei astfel de distribuții este $\mathcal{N}(\mu, \sigma^2)$, unde μ este media punctelor, iar σ^2 este varianța, o valoare care indică dispersia punctelor.

3.1 Seturi de date generate aleator

3.1.1 Metoda de generare

Ca primă metodă de comparare a modelelor prezentate mai sus, am generat un dataset sintetic aleator care să simuleze un *shift* în distribuția datelor de antrenare versus datele de test. Algoritmul de generare funcționează conform următorilor pașii:

- se definește o distribuție normală multivariată pornind de la un vector de medii σ ales aleator și de la o matrice de covarianță aleatoare
- se extrag N puncte din această distribuție
- se generează mai multe clustere care deplasează punctele cu valori în $[-\sigma_m, +\sigma_m]$, unde σ_m este media vectorului σ
- se generează anomalii alegând un procent din cele N puncte și trecându-le într-unul dintre clustere

Se pot modifica parametri precum procentul de *outlieri* la antrenare sau la test, numărul de clase diferite la antrenare și la test, cât și dimensiunea spațiului datelor. *Shift*-ul dintre distribuții poate fi vizualizat prin intermediul unei reprezentări *t-SNE* care mapează puncte dintr-un spațiu mai mare într-un spațiu 2D, păstrând în același timp distanțele dintre clase și implicit permite ilustrarea *shift*-ului. În figura 3.2 am folosit t-SNE pentru a ilustra *shift*-ul în 2 seturi de date generate aleator. Din moment ce covariațele

sunt generate aleator, distribuțiile nu sunt înrudite și sunt clar separabile. Această metodă definește așadar un *shift* extrem, în care distribuțiile nu au legătură unele cu altele. Neajunsul acesta este adresat în secțiunea următoare.

Aceeași figură afișează o distribuție într-un spațiu mare, generată împreună cu *outlierii* săi. Aceștia urmează un comportament specific, se află spre marginea distribuției, însă sunt infiltrați și printre exemplele normale. Figura 3.1 ilustrează același comportament, de această dată pentru puncte generate în două dimensiuni.

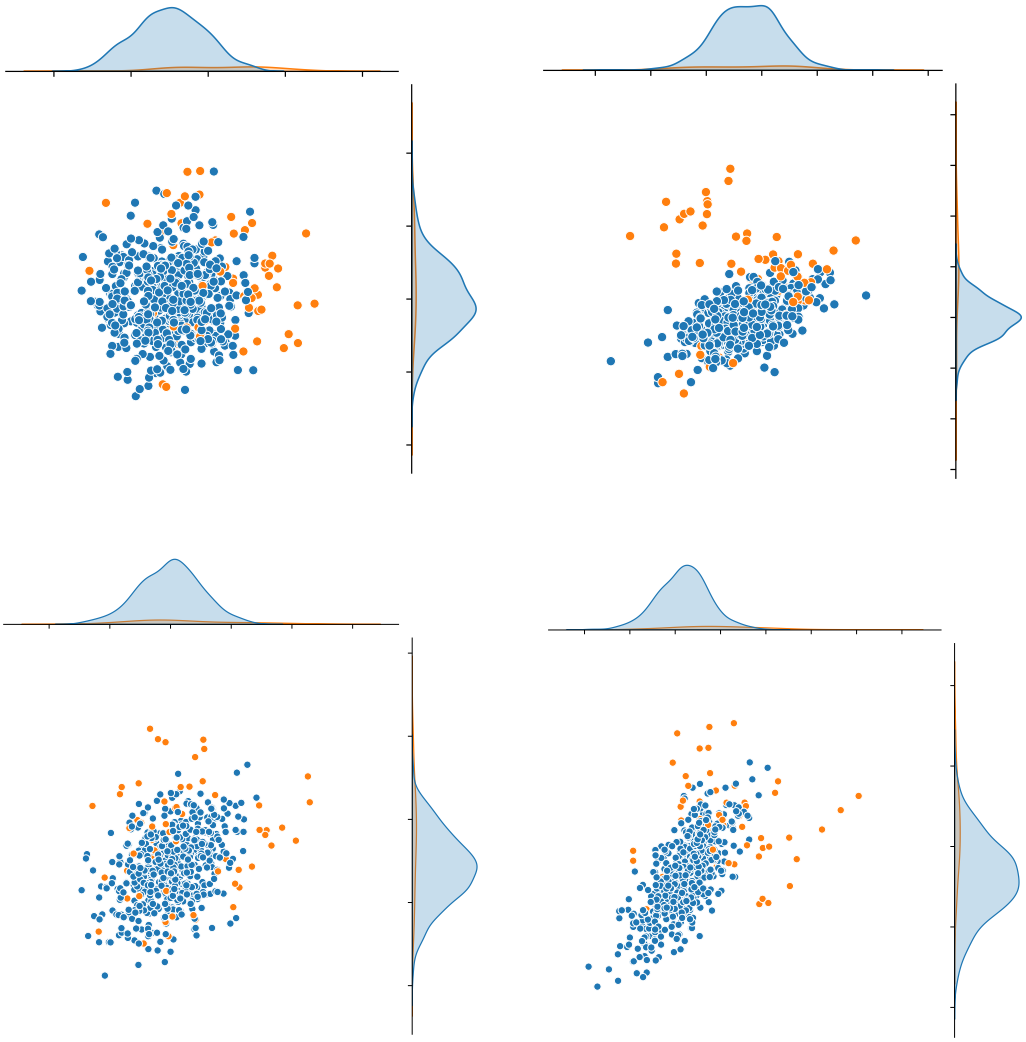
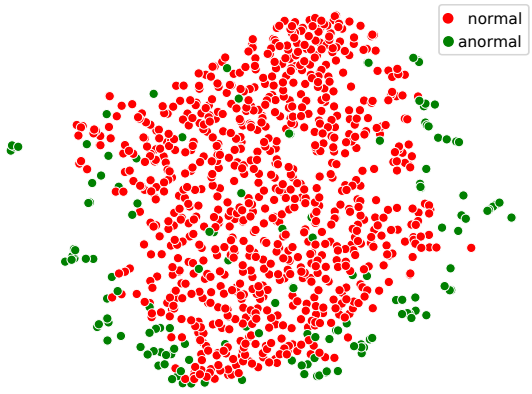
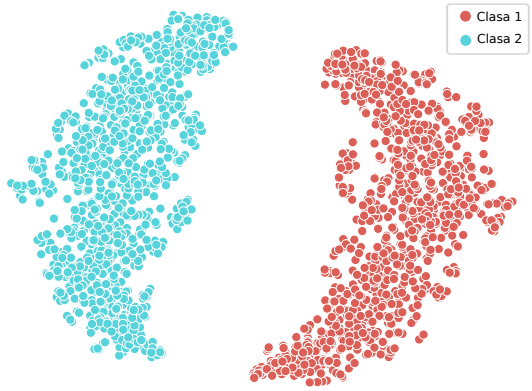


Figura 3.1: Patru distribuții în două dimensiuni generate aleator, afișate împreună cu outlierii corespunzători.

Am rulat experimente într-un spațiu mare, cu 256 de feature-uri; dataset-ul de antrenare a fost împărțit în 100 de clase cu distribuții diferite a câte 3000 de exemple fiecare și la testare am folosit 20 de clase a câte 2000 de exemple fiecare. Fiecare clasă de antrenare a fost modificată să conțină 50% *outlieri*, amplasați în 5 clustere diferite, iar la testare au fost făcute experimente pentru 1%, 10% și 20% *outlieri*.



(a) Toate punctele fac parte din aceeași distribuție, punctele roșii sunt normale, iar cele verzi anormale.



(b) Punctele fac parte din două distribuții distincte; punctele roșii corespund clasei 1, iar punctele albastre clasei 2.

Figura 3.2: Distribuții în spațiul mare generate aleator.

3.1.2 Rezultate

Model	1%	10%	20%
XGBOD	79.0 ±4.36	**82.3 ±2.92**	**82.1 ±1.70**
CD	**98.4 ±1.89**	73.4 ±8.70	63.9 ±10.6
ACR-DSVDD	54.2 ±1.09	51.4 ±1.26	50.2 ±0.90
LODA	55.9 ±5.64	55.5 ±3.20	55.7 ±2.75
ECOD	53.2 ±6.57	53.3 ±1.99	53.7 ±1.51
INNE	74.2 ±3.60	73.9 ±2.30	74.6 ±1.84
HBOS	55.3 ±3.10	53.2 ±2.27	53.0 ±1.65
OC-SVM	**83.0 ±3.85**	**81.4 ±1.93**	**80.8 ±1.86**
DeepSVDD	39.7 ±6.59	38.6 ±2.88	43.7 ±2.84

Tabela 3.1: Valorile AUC pentru metodele folosite (media și deviația standard pentru 3 teste separate) pentru procente diferite de outlieri la testare.

Arhitectura folosită pentru implementarea ACR-DSVDD este compusă din 4 straturi convoluționale cu *kernel* 3×3, a câte 64 de neuroni fiecare. După oricare strat convoluțional a urmat unul de *Batch Normalization*. Au fost folosite 64 de batch-uri per iterație, fiecare batch având 40 de exemple. Optimizatorul ales a fost Adam cu *learning rate* de $1e-4$.

Tabelul 3.1 conține rezultatele pentru experimentul cu seturile de date generate aleator. Pentru a evita problema găsirii unui prag pentru a considera un exemplu ca fiind outlier, rezultatele sunt exprimate sub forma scorului AUC (*Area Under Curve*) care este o măsură a capacității de discriminare a unui model.

Majoritatea metodelor nesupervizate prezintă rezultate cu puțin peste 50%, ceea ce înseamnă că se descurcă mai bine decât alegerea aleatoare însă nu au capacitate sem-

nificativă de detectare a anomaliilor. Acest fapt se datorează diferențelor mari dintre distribuțiile de antrenare și test: în timpul antrenării nu pot fi învățate *pattern*-uri folositoare la testare.

Se deosebesc totuși două metode: OC-SVM și INNE, cu o medie de 81.7% respectiv 72.3% pe cele trei experimente. OC-SVM are însă de suferit dacă nu creștem numărul de date de antrenament pe măsură ce creștem numărul de atribute, spațiul devenind prea *sparse*. Odată ce trecem de 300 de atribute, scorul OC-SVM nu trece cu mult de 50%.

DeepSVDD-ul are cele mai slabe rezultate, iar informația suplimentară de care se folosește ACR-DSVDD-ul crește performanța cu doar câteva procente peste 50%. Metodele supervizate se comportă în mod așteptat, însă CD suferă de instabilitate pe măsură ce creștem procentul de anomalii la test.

Concluzia acestui experiment este că simularea *distribution shift*-ului în mod complet aleator nu permite unui model să învețe reprezentări relevante și care pot fi folosite în timpul testării. În secțiunea următoare voi descrie o metodă care să construiască date cu un *distribution shift* mai structurat.

3.2 Seturi de date generate prin rotația matricei de covarianță

3.2.1 Matricea de rotație

Distribution shift este un fenomen complex și greu de modelat în mod sintetic. A doua metodă de generare a seturilor de date se bazează pe rotația matricelor de covarianță care descriu distribuțiile în jurul unor axe, producând astfel distribuții înrudite.

O matrice de rotație ne permite să efectuăm o rotație asupra unei alte matrice într-un spațiu euclidian, pe o anumită axă. Dându-se un unghi θ în radiani, este definită matricea următoare care efectuează, prin înmulțire cu o matrice dată, o rotație într-un sistem de coordonate cartezian:

$$R = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

Mai departe, matricea R poate fi extinsă pentru a efectua o rotație unei matrici de dimensiune $N \times N$:

$$R_x = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & cos\theta & -sin\theta \\ 0 & 0 & \cdots & sin\theta & cos\theta \end{pmatrix}$$

Asemănător se definesc matrici și pentru celelate două rotații:

$$R_y = \begin{pmatrix} \cos\theta & 0 & \cdots & \sin\theta \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \\ -\sin\theta & 0 & \cdots & \cos\theta \end{pmatrix}$$

$$R_z = \begin{pmatrix} \cos\theta & -\sin\theta & \cdots & 0 \\ \sin\theta & \cos\theta & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

3.2.2 Metoda de generare

Funcția de generare primește ca parametrii matricea de covarianță inițială, gradul total de rotație și numărul total de matrici noi de generat. Pașii de generare sunt următorii:

1. se definesc matricile de rotație R_x, R_y, R_z în funcție de dimensiunea spațiului
2. se calculează unghiul de rotație pentru fiecare matrice nouă împărțind unghiul total la numărul de matrici de generat
3. se descompune matricea de covarianță în valorile și vectorii proprii corespunzători
4. se înmulțește vectorul propriu cu matricea de rotație
5. se construiește matricea nouă de covarianță înmulțind rezultatul de la pasul 4 cu matricea identitate scalată cu valoarea proprie și se înmulțește la final cu transpusa pentru a asigura simetria matricei
6. pașii 3-5 se repetă pentru R_x, R_y, R_z

Secvența de cod următoare exemplifică o astfel de metodă de generare, pentru o singură axă. Este folosită biblioteca *numpy.linalg* [27] pentru toate operații pe matrice.

```
# Convert angle to radians
radians = np.radians(degrees)
# Compute step for each iteration
step = radians / iterations
rotated_mat = []
# Do the eigendecomposition of the original matrix
eigenvalues, eigenvectors = np.linalg.eigh(cov)
```

```
for i in range(iterations):
    rotation = step * (i + 1)
    # Rotation matrix
    rotation_mat = np.eye(cov.shape[0])
    rotation_mat[0, 0] = np.cos(rotation)
    rotation_mat[0, 1] = -np.sin(rotation)
    rotation_mat[1, 0] = np.sin(rotation)
    rotation_mat[1, 1] = np.cos(rotation)
    # Rotate eigenvectors by rotation matrix
    eig_rotated = eigenvectors @ rotation_mat
    # Rebuild the covariance matrix
    cov_rotated = eig_rotated @ np.diag(eigenvalues) @ eig_rotated.T
```

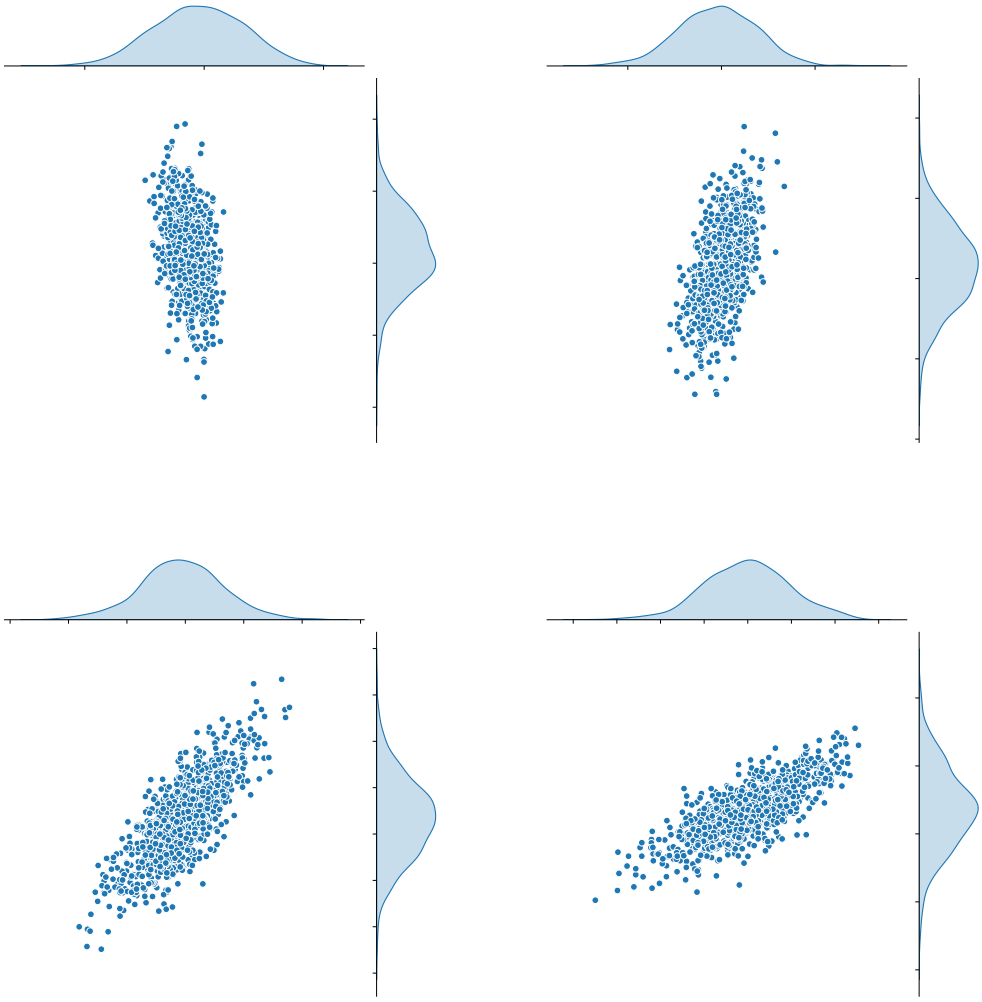


Figura 3.3: Patru distribuții în două dimensiuni generate cu o rotație de 75°.

Setup-ul experimental a fost asemănător cu cel folosit la seturile de date aleatoare. Am generat 20 de clase de antrenament cu câte 3000 de exemple fiecare și 10 clase de

test cu 2000 de exemple fiecare. Unghiul total al rotației a fost de 180°, clasele de testare fiind generate în continuarea celor de antrenament.

Spre deosebire de metoda precedentă, spațiul datelor a fost mai mic, cu 16 atribute. Din moment ce acum variem doar 3 feature-uri, față de a varia toate feature-urile în mod aleator, spațiul trebuie să fie mai restrâns pentru ca cele 3 atribute să introducă un *shift* perceptibil. Figura 3.3 ilustrează *shift*-ul în două dimensiuni

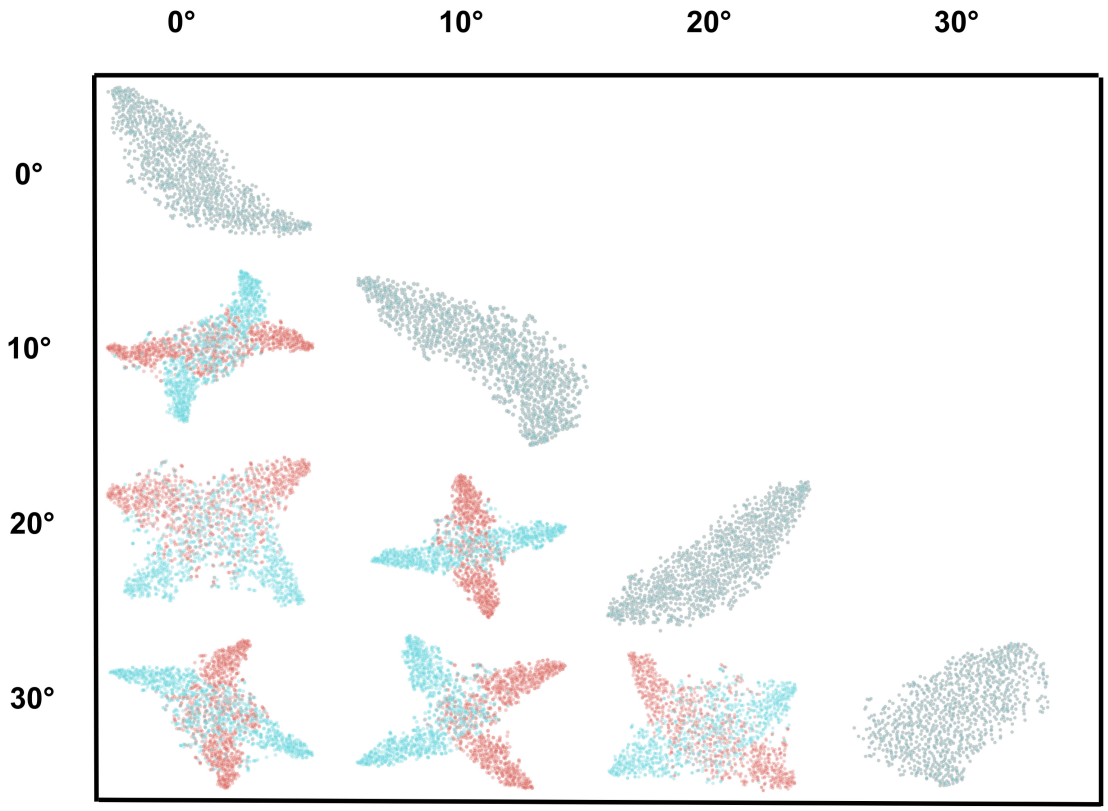


Figura 3.4: Distribuții generate folosind metoda rotației, cu un unghi total de 30°, comparate două câte două folosind t-SNE. Au fost folosite 1500 de exemple din fiecare clasă. Rezultatele sunt simetrice cu diagonala. Clasa principală este reprezentată de punctele roșii.

Shift-ul în mai multe dimensiuni este ilustrat în 3.4. Reprezentările sunt ordonate într-o matrice, unde clasa de bază este clasa de pe coloană, iar punctele corespunzătoare clasei de bază sunt marcate cu roșu; pe diagonală se află distribuția afișată independent. Pe măsură ce gradul de rotație crește, datele devin din ce în ce mai distincte și *shift*-ul devine mai accentuat. Comparând cu reprezentările metodei precedente, putem conclude că acum avem parte de un *shift* mai structurat.

3.2.3 Rezultate

Tabelul 3.2 conține rezultatele pentru experimentul cu seturile de date generate prin rotația matricei de covarianță. Aproape toate modelele au un scor semnificativ mai bun decât alegerea aleatoare. În acest setup, modelele supervizate au un avantaj clar față de cele nesupervizate. Cea mai bună performanță este obținută de XGBOD (media de 97.2% pe toate experimentele); CD nu este la fel de robust când crește procentul de anomalii.

Metodele nesupervizate își îmbunătățesc performanța în general, aproape toate reușind să aibă rezultate de încredere pentru detecția de outlieri. DeepSVDD eșuează la procente de anomalii mai mari, însă ACR-DSVDD generalizează foarte bine pe acest tip de shift structurat. Acest fapt se explică prin asumpția inițială pe care o face paradigma ACR: distribuții care au legătura între ele.

Model	1%	10%	20%
XGBOD	98.9 ±0.78	**96.8 ±0.92**	**96.0 ±0.78**
CD	**99.4 ±0.59**	96.5 ±0.42	85.7 ±0.78
ACR-DSVDD	**89.8 ±4.60**	**83.7 ±1.33**	**83.5 ±4.18**
LODA	68.0 ±2.40	63.7 ±0.95	72.8 ±1.13
ECOD	68.3 ±3.50	68.2 ±3.39	66.9 ±2.49
INNE	74.8 ±3.16	70.6 ±3.14	**74.0 ±2.03**
HBOS	67.7 ±5.10	67.0 ±3.65	64.9 ±2.78
OC-SVM	**76.3 ±4.49**	**74.8 ±4.07**	72.1 ±3.63
DeepSVDD	61.0 ±5.33	37.4 ±1.17	39.4 ±1.11

Tabela 3.2: Valorile AUC pentru metodele folosite (media și deviația standard pentru 3 teste separate) pentru procente diferite de outlieri la testare.

3.2.4 Ablație

Pentru a testa validitatea acestui mod de generare, am rulat un subset din modele și pe un setup în care rotația este nulă. Prin eliminarea componentei de rotație, generăm practic aceeași matrice de covarianță. Astfel, eliminăm shift-ul pentru seturi de antrenament; acest problemă de detecție a anomaliilor este mai ușoară, așadar ne-am aștepta să obținem acurateț mai mare. Din tabelul 4.3 observăm că performanța este mai bună, astfel că metoda descrisă introduce un shift însemnat.

3.2.5 Concluzie

Metoda propusă de generare modelează un shift în datele de antrenare, însă o abordare mai complexă ar însemna modificarea mai multor dimensiuni din spațiul atributelor. Pentru aceasta, ar putea fi folosit un algoritm NRMG(N-dimensional Rotation Matrix

Model	10%
ACR-DSVDD	90.3 ±2.37
LODA	67.9 ±0.98
ECOD	65.2 ±3.04
INNE	89.9 ±1.40
OC-SVM	90.0 ±1.23

Tabela 3.3: Valorile AUC pentru metodele folosite (media și deviația standard pentru 5 teste separate), cu 10% outlieri la testare și rotație 0°.

Generation Algorithm) [26], care generează i rotație în mai multe dimensiuni; sau alte tipuri de matrice de transformare.

Atât modelele supervizate, cât și cele nesupervizate au reușit să detecteze în mod regulat anomaliile din seturile de date. Cele supervizate au atins, în medie peste 95% acuratețe, însă din cauza lipsei etichetelor nu sunt folosite în pratică. Există diferențe în funcție de metodele de generare a datelor, a doua metodă având un *shift* mai organizat modelele au reușit să generalizeze mai eficient. INNE a fost cea mai stabilă metodă, având rezultate asemănătoare în ambele experimente, iar ACR-DSVDD a avut parte de o îmbunătățire foarte mare. Este interesant și rezultatul XGBOD-ului pe primul set de date; faptul că a avut acuratețe mai mică decât o metodă nesupervizată. Asta demonstrează inabilitatea metodelor supervizate de a prezice anomalii noi și ne spune că informația adițională nu a fost un factor pozitiv pentru acest model. Așadar, este importantă analiza calitativă a setului de date și identificarea *shift*-ului înainte de alegerea unui model.

Capitolul 4

Anoshift [6] - distribution shift într-un set de date natural

4.1 Motivație

Datele sintetice generate până acum au modelat într-o manieră simplistă *distribution shift*-ul. Este dificil să modelăm realist acest concept fără a utiliza date naturale, care urmează tendințe complexe și suferă transformări imprevizibile. De asemenea, detecția anomaliilor în trafic de rețea este una dintre cele mai des întâlnite utilizări ale algoritmilor de AD și reprezintă o direcție de cercetare importantă.

Sunt puține seturi de date care îndeplinesc condițiile necesare pentru a putea fi studiate din perspectiva shift-ului, în principal din cauza dificultății colectării datelor: trebuie culese de-a lungul unei perioade mari de timp. Kyoto 2006+ [34] îndeplinește această condiție și de aceea a fost ales să fie studiat în [6]. În continuare voi oferi o descriere a acestui dataset, voi detalia experimente efectuate pe acesta și voi testa variații în arhitectura ACR-DSVDD.

4.2 Descrierea setului de date

4.2.1 Proveniența datelor

Anoshift este construit pe baza setului de date Kyoto 2006+ [34], care conține log-uri de trafic de rețea obținute de o colecție de *honeypots* din cadrul Universității din Kyoto. Un *honeypot* este un dispozitiv amplasat într-o rețea, făcut să reproducă comportamentul unui dispozitiv obișnuit și al cărui scop este să atragă atacuri asupra sa pentru a permite analizarea acestor atacuri. Sunt create intenționat vulnerabilități pe astfel de dispozitive pentru a încuraja atacatorii să profite de ele; un *honeypot* nu interacționează cu alte dispozitive din rețeaua sa, așa că majoritatea traficului este de natură malițioasă. Dataseturile colectate din *honeypots* prezintă un procent semnificativ mai mare de anomalii decât

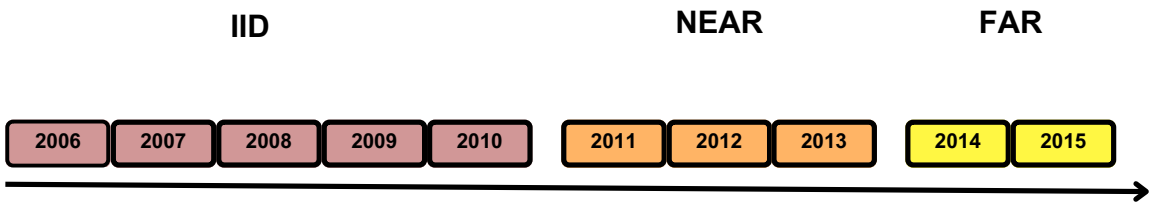


Figura 4.1: Divizarea propusă de autori a setului de date după ani, pentru a testa modelul pe un *distribution shift* puternic.

alte tipuri de dataset-uri de AD.

În [34] sunt folosite mai multe tipuri de dispozitive, atât mașini Windows, cât și Unix sau Linux, în 5 rețele diferite din cadrul Universității. Atacurile au fost monitorizate între noiembrie 2006 și decembrie 2015, [34] fiind dataset-ul cel mai amplu din acest punct de vedere. Pe măsură ce protocoalele de protecție ale rețelelor se îmbunătățesc, sunt concepute atacuri noi într-un mod adversarial. Aceasta este sursa shift-ului în seturi de date care privesc traficul de rețea și care este prevalent în perioada de 10 ani din Kyoto 2006+.

Kyoto 2006+ a fost ales să stea la baza dataset-ului Anoshift datorită perioadei mari de colectare a datelor, cât și datorită faptului că celelalte dataset-uri de trafic de rețea sunt triviale (s-au obținut rezultate peste 90%). Un punct important al lucrării este divizarea datelor cronologic, în funcție de anul în care au fost colectate. Astfel, se poate testa un anumit model pe distribuții mai mult sau mai puțin diferite de cele de antrenament. Figura 4.1 ilustrează acest split. Modelele sunt antrenate pe IID (*independent and identically distributed*) și testate separat pe IID, NEAR și FAR. Ne așteptăm ca un model să aibă performanță mai slabă pe un set cât mai îndepărtat de cel de train.

4.2.2 Procesarea datelor

Kyoto 2006+ conține inițial 14 atribute de bază, precum durata conexiunii, tipul de serviciu folosit, numărul de bytes trimiși și recepționați și numărul de conexiuni încercate. Sunt disponibile și atribute adiționale, în principal clase binare care indică dacă o anumită conexiune a fost raportată ca fiind malițioasă de către unul dintre tool-urile de detecție a atacurulor instalate pe *honeypots*.

Anoshift folosește toate cele 24 de atribute și propune transformarea a 3 dintre feature-uri adiționale în feature-uri categorice prin *binning*. În preprocesarea datelor pentru ACR-DSVDD, am folosit reprezentare *one-hot* pentru feature-urile categorice, obținând astfel un *feature space* final cu dimensiunea de 551.

4.2.3 Vizualizarea shift-ului

O vizualizare a datelor folosind t-SNE permite ilustrarea clară a shift-ului setului de date de-a lungul timpului (figura 4.2). Reprezentările sunt ordonate într-o matrice, unde clasa de bază este clasa de pe coloană, iar punctele corespunzătoare clasei de bază sunt marcate cu roșu; pe diagonală se află distribuția afișată independent. Cum t-SNE ia în calcul distanța dintre puncte, existența unei alte distribuții în plot modifică forma reprezentărilor. Am ales să afișez date din 2006, 2009, 2012 și 2015, cuprinzând astfel clase din toate cele trei split-uri. Observăm un shift gradual între 2006 și 2015; chiar dacă 2012 și 2015 sunt apropiate temporal, există un shift accentuat ceea ce face setul de date în special dificil pe FAR. Comparând aceste reprezentări cu cele ale datelor generate sintetic, observăm un shift mai gradual, însă mai puțin predictibil. Spre deosebire de datele generate aleator, distribuțiile nu sunt ușor separabile.

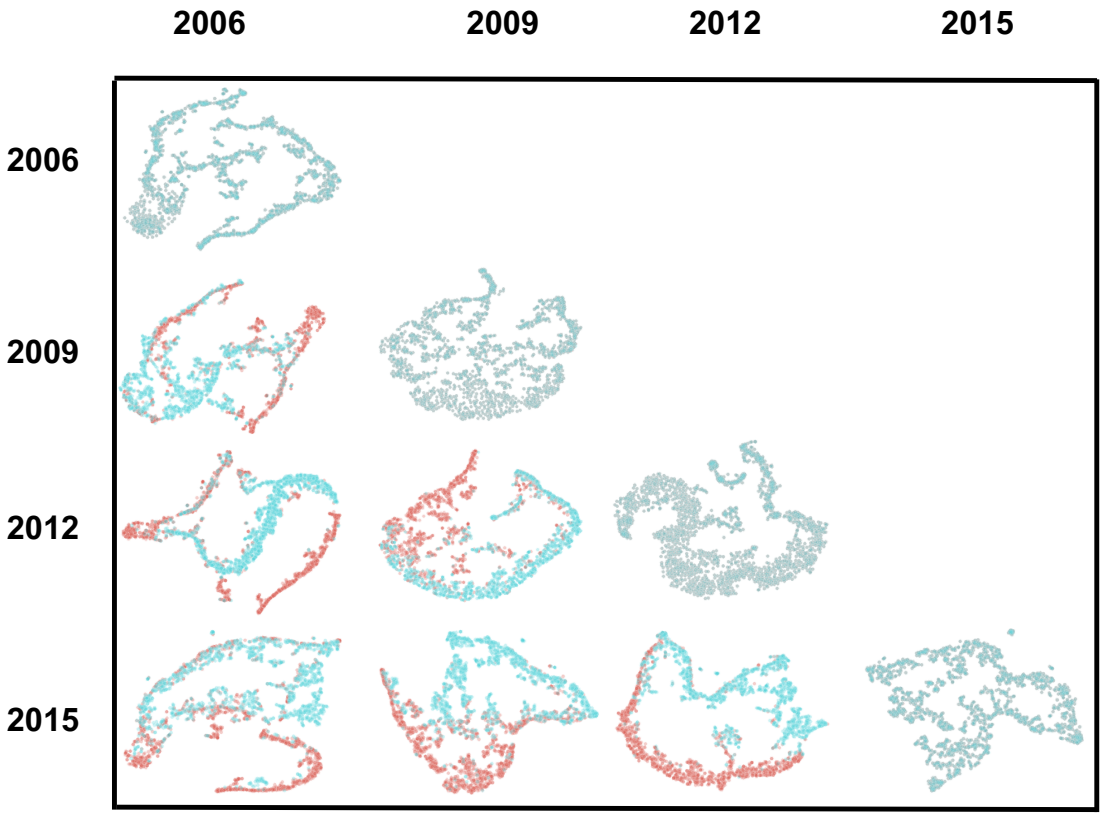


Figura 4.2: Date divizate cronologic, comparate două câte două folosind t-SNE. Au fost folosite 1500 de exemple din fiecare clasă. Rezultatele sunt simetrice cu diagonala. Clasa principală este reprezentată de punctele roșii.

4.3 Metode folosite și rezultate

Am rulat experimente pentru a testa ACR-DSVDD pe cele 3 tipuri de split. În prezent, ACR-DSVDD obține cele mai bune rezultate pe split-ul FAR. A fost folosit setup-ul sugerat în [6]: am antrenat și validat pe date din IID și am testat pe IID, NEAR și FAR. Pentru a respecta paradigma ACR, la train au fost folosite doar exemple normale și am contaminat fiecare batch cu exemple normale din alte batch-uri. Arhitectura modelului folosit (alegerea hiperparametrilor va fi detaliată în secțiunea următoare) constă în 4 *hidden layers* a câte 128 de neuroni, fiecare *hidden layer* fiind urmat de un strat de Batch Normalization. Centrul hipersferei (ultimul strat, *fully connected*) are dimensiune de 32 de neuroni. Este folosit optimizer-ul *Adam* cu un learning rate de $1e-4$; fiecare iterație trece prin 32 de batch-uri, iar batch-urile folosite conțin 512 exemple. Am antrenat timp de 1000 de iterații.

Sunt raportate rezultate și pentru alte metode de AD. [19] folosește OC-SVM, DeepSVDD și Isolation Forest [21]. Autorii din [6] folosesc ca metodă principală de benchmark un model BERT, în configurație de *Masked Language Model*. Mai precis sunt ascunse porțiuni aleatoare din input iar modelul învață să completeze secvențele lipsă. Scorul anomaliei este dat de probabilitatea modelului de a selecta inputul lipsă corect.

Autorii Anoshift propun o tehnică de *knowledge distillation* ca metodă generală de îmbunătățire a performanței. Knowledge distillation constă în transferul de cunoștințe de la o rețea complexă, numită *teacher network*, la o rețea simplă, numită *student network*. Într-un astfel de setup, *teacher network* este antrenată parțial pe un subset de date, iar *student network* este antrenată în continuare cu o funcție de loss care ia în calcul performanța pe setul de date și distanța distribuțiilor activărilor celor două rețele, calculate cu KL Divergence. Această tehnică a îmbunătățit performanța modelului BERT cu aproximativ 3%. În experimentele efectuate, nu am constatat o îmbunătățire a acurateții ACR-DSVDD-ului cu *knowledge distillation*, însă poate fi o direcție de urmat pentru alte modele de deep learning.

Model	1%	5%	10%	20%
ACR-DSVDD	**62.2 ±3.2**	**62.7 ±2.8**	**59.9 ±1.3**	**59.1 ±2.9**
OC-SVM	49.6 ±0.2	49.6 ±0.2	49.5 ±0.1	49.5 ±0.1
DeepSVDD	34.6 ±0.3	34.7 ±0.2	34.7 ±0.1	34.7 ±0.1
IForest	25.8 ±0.4	26.1 ±0.1	26.0 ±0.1	26.0 ±0.1
BERT	28.6 ±0.3	28.7 ±0.1	28.7 ±0.1	28.7 ±0.1

Tabela 4.1: Valorile AUC pentru metodele folosite (media și deviația standard pentru 3 teste separate) pentru procente diferite de outlieri la testare. Rezultate pentru split-ul FAR.

Majoritatea modelelor nu ating acurateție ridicată pe split-ul FAR, cu rezultate mai

Model	1%	5%	10%	20%
ACR-DSVDD	**74.3 ±2.3**	**74.0 ±3.1**	**72.4 ±3.7**	**70.7 ±2.0**
OC-SVM	62.6 ±0.1	62.6 ±0.2	62.7 ±0.1	62.6 ±0.1
DeepSVDD	62.3 ±0.2	62.5 ±0.1	62.5 ±0.1	62.5 ±0.1
IForest	54.6 ±0.2	54.7 ±0.1	54.6 ±0.1	54.7 ±0.1
BERT	**64.6 ±0.2**	**64.6 ±0.1**	**64.7 ±0.2**	**64.7 ±0.1**

Tabela 4.2: Valorile AUC pentru metodele folosite (media și deviația standard pentru 3 teste separate) pentru procente diferite de outlieri la testare. Rezultate pentru split-ul NEAR.

Model	2006	2010	2011	2012	2013	2014	2015
ACR-DSVDD	86.3	80.5	81.8	75.2	72.6	66.9	60.5

Tabela 4.3: AUC detaliat pentru un experiment cu 5% anomalii.

slabe decât alegerea aleatoare. Remarcabil, BERT nu învață o reprezentare care să îi permită să extrapoleze pe date noi; alternativa sa de deep learning ACR-DSVDD este o opțiune viabilă pentru acest split, cu un scor care scade însă pe măsură ce creștem procentul de anomalii. De asemenea, suferă de o instabilitate numerică având o deviație standard semnificativă. Acest lucru se datorează probabil antrenării limitate.

Split-ul NEAR are parte de rezultate mai satisfăcătoare; toate modelele au câteva procente bune peste 50%. Modelele kernel, OC-SVM și DeepSVDD, și BERT au procente apropiate, iar ACR-DSVDD suferă de aceeași instabilitate și se observă același fenomen al scăderii performanței odată cu creșterea procentului de anomalii.

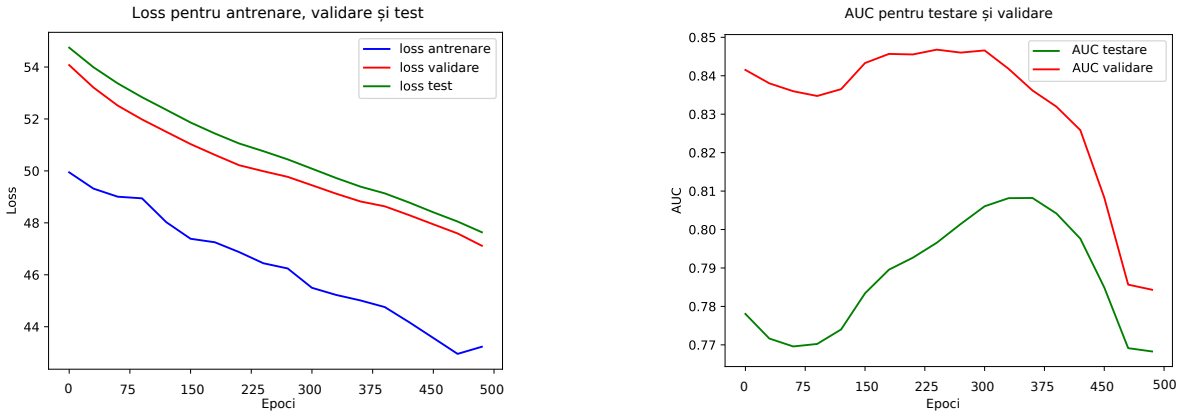


Figura 4.3: AUC și Loss în timpul antrenării, pentru un experiment cu 5% anomalii.

ACR-DSVDD este predispus la overfit timpuriu. Figura 4.3 arată că în timp ce loss-ul scade atât pe train cât și pe validation, scorul AUC se degradează cu câteva procente, așa că rețeaua nu mai învață reprezentări care să detecteze anomaliile de la testare, ci se

concentrează pe exemplele cu care sunt contaminate batch-urile. Din acest motiv, este folositor un mecanism de *early stopping*.

4.3.1 Alegerea hiperparametrilor

Modelul este, în general, robust și are capacitate de generalizare chiar și cu o rețea mai puțin densă. Am ales ca punct de plecare concluziile căutării de hiperparametri ale autorilor din [19], astfel că nu am fost nevoit să efectuez multe experimente în acest sens. Am ales să testez performanța modelului variind mărimea batch-urilor și structura rețelei neurale.

Batch size	IID	NEAR	FAR
512	84.2	**74.0**	**62.7**
256	**84.6**	73.7	59.5
128	**84.6**	73.6	61.5

Tabela 4.4: AUC(%) pentru diferite batch size-uri (5% anomalii).

Setup	IID	NEAR	FAR
128 neuroni, 4 straturi	**84.2**	74.0	**62.7**
64 neuroni, 4 straturi	82.5	**75.7**	60.8
64 neuroni, 2 straturi	78.2	72.6	56.1
32 neuroni, 2 straturi	77.6	69.1	50.2
32 neuroni, 4 straturi	80.6	71.9	52.2

Tabela 4.5: AUC(%) pentru diferite arhitecturi ale rețelei neurale (5% anomalii).

Mărimea batch-ului nu influențează acuratețea pentru IID, însă are un efect pentru datele îndepărtate. În mod previzibil, o rețea mai simplă are performanță semnificativ mai slabă, însă și setup cu 32 neuroni și 2 straturi ascunse reușește să detecteze shift-ul, cel puțin pentru NEAR. Numărul de straturi ascunse are cel mai mare efect asupra performanței generale. Am considerat acuratețea pe FAR ca fiind rezultatul cel mai însemnat și pe baza acestuia am ales arhitectura pentru experimente.

4.4 Concluziile experimentului

Anoshift scoate în evidență natura complexă a *distribution shift*-ului. Pentru testele apropiate temporal cu datele de antrenare, majoritatea modelelor obțin acurateță bună, însă avantajul modelelor care au mecanisme de mitigare a *shift*-ului iese la iveală pe datele îndepărtate. În absența shift-ului, adică în split-ul IID, DeepSVDD-ul are rezultate satisfăcătoare, obținând o acurateță de 92.6%, însă scade sub 50% pe FAR. Adăugând

DeepSVDD-ului o tehnică care îmbunătățește generalizarea (ACR), devine modelul cu cea mai ridicată acuratețe. Fiind un set de date recent conceput, niciun alt model nu a raportat acuratețe mai bună decât ACR-DSVDD pe Anoshift.

Capitolul 5

Distribution shift în imagini

Până acum, am discutat doar despre shift în dataset-uri tabelare. Poate fi prevalent însă și în imagini, unde detecția de anomalii poate avea aplicații folositoare precum identificarea defectelor de fabricație sau automatizarea procesului de interpretare a imaginilor medicale. Shift-ul în imagini poate avea mai multe cauze, precum metode diferite de colectare a datelor între train și test (antrenăm modelul pe imagini generate de un dispozitiv medical și dorim să îl folosim pe un alt dispozitiv) sau schimbări conceptuale ale datelor.

5.1 CIFAR-100

CIFAR-100 [17] este un set de date care conține 100 de clase diferite de imagini de dimensiune 32×32, însumând 60000 de exemple. Nu este un set de date complex, cea mai bună metodă obținând acuratețe de 96.2% [15], însă poate fi interesat de studiat din perspectiva shift-ului. CIFAR-100-C [12] este un dataset derivat, în care imaginile au fost modificate prin aplicarea unui filtru de noise; datele din CIFAR-100-C pot fi considerate shifted față de dataset-ul original. Pot fi aplicate filtre precum Gaussian Noise, Shot Noise sau Motion Blur.

În continuare voi folosi imaginile distorsionate prin adăugarea de Gaussian Noise. Este cel mai întâlnit tip de noise și apare în imagini în mod natural în condiții de luminozitate scăzută. Filtrul de noise este creat prin extragerea de valori dintr-o distribuție gaussiană pentru fiecare pixel. Figura 5.1 ilustrează o astfel de transformare. Toate cele 3 canale ale imaginilor au fost folosite și au fost în prealabil normalizate cu statistici precalculate pe fiecare canal.

5.2 Metode folosite și rezultate

Pentru a rula experimentele, am folosit un setup asemănător ca până acum. La antrenare au fost folosite exclusiv exemple din CIFAR-100; fiecare iterație a trecut prin 64 de

Figura 5.1: Rezultatul aplicării unui filtru de Gaussian Noise.

batch-uri. Un batch era format din 60 de exemple; clasa normală a fost selectată aleator și anomaliile au fost alese din clasele rămase. S-a folosit un split antrenare-validare de 80 %. Datele de test au fost alese din CIFAR-100-C (imaginile contaminate cu noise) și batch-urile au fost construite în același mod ca în faza de train.

Arhitectura rețelei pe care am folosit-o pentru ACR-DSVDD constă în 4 *hidden layers* cu câte 128 de neuroni și straturi de Batch Normalization după fiecare *hidden layer*. Pentru a accelera procesul de învățare a rețelei, *weight-urile* sunt inițializate după o distribuție Xavier și *bias-urile* sunt inițializate cu 0. Secțiunea următoare detaliază acest procedeu și efectele sale asupra antrenării.

CLIP este un *foundation model* deep, antrenat pe un număr foarte mare de perechi text-imagine și poate fi folosit pentru detecția de anomalii prin probabilitatea sa de a potrivi o imagine cu un anumit label. ResNet152 [11] este o rețea reziduală preantrenată cu o arhitectură foarte complexă —152 de straturi. Din această rețea este eliminat ultimul strat, este adăugat un strat fully connected și un strat de Batch Normalization. CLIP și această variantă modificată de ResNet sunt folosite pentru a compara performanța ACR-DSVDD.

Model	1%	5%	10%	20%
ACR-DSVDD	**87.7 ±2.6**	**84.3 ±3.2**	**83.9 ±2.9**	**82.7 ±2.5**
CLIP	82.2 ±1.1	82.6 ±0.9	82.3 ±0.9	**82.6 ±0.1**
ResNet152	75.6 ±2.3	73.2±1.3	73.2±0.8	69.9 ±0.6

Tabela 5.1: AUC(%), detecția de anomalii pe setul de date CIFAR-100-C.

Conform tabelului de mai sus, ACR-DSVDD raportează cea mai bună performanță pentru acest experiment, în special comparat cu ResNet. Totuși, este variant față de procentul de anomalii și scade din performanță până la nivelul modelului CLIP pe setul cu 20% anomalii. De menționat este că CLIP nu este fiabil pe imagini artificiale, precum imaginile medicale. Combinat cu faptul că este un model greu de antrenat și proprietar, utilizările sale sunt limitate.

5.3 Efectul inițializării parametrilor

Inițializarea parametrilor modelului are un efect semnificativ asupra stabilității și poate preveni problema gradienților care dispar sau care cresc nemărginit. Majoritatea modelelor care se ocupă cu *meta-learning* se folosesc de inițializarea parametrilor pentru a crește performanța de generalizare, așa cum a fost propus prima dată în [8]. O astfel de metodă de inițializare se folosește de distribuția Xavier [9]. Distribuția Xavier are rolul de a crea o distanță cât mai mică între distribuțiile inițiale ale activărilor și este doar o distribuție uniformă mărginită inferior și superior în funcție de numărul de neuroni din stratul actual și stratul precedent.

Figura 5.2 scoate în evidență comportamentul aceluiași model ACR-DSVDD, testat cu și fără inițializarea parametrilor. Loss-ul pornește de la o valoare mai mică în cazul inițializării, iar loss-ul pentru validare și test se apropie mai mult de loss-ul pentru antrenare. Acuratețea suferă de asemenea modificări: inițializarea adaugă aproximativ 2 procente și are parte de o variantă mai mică pe parcursul antrenării.

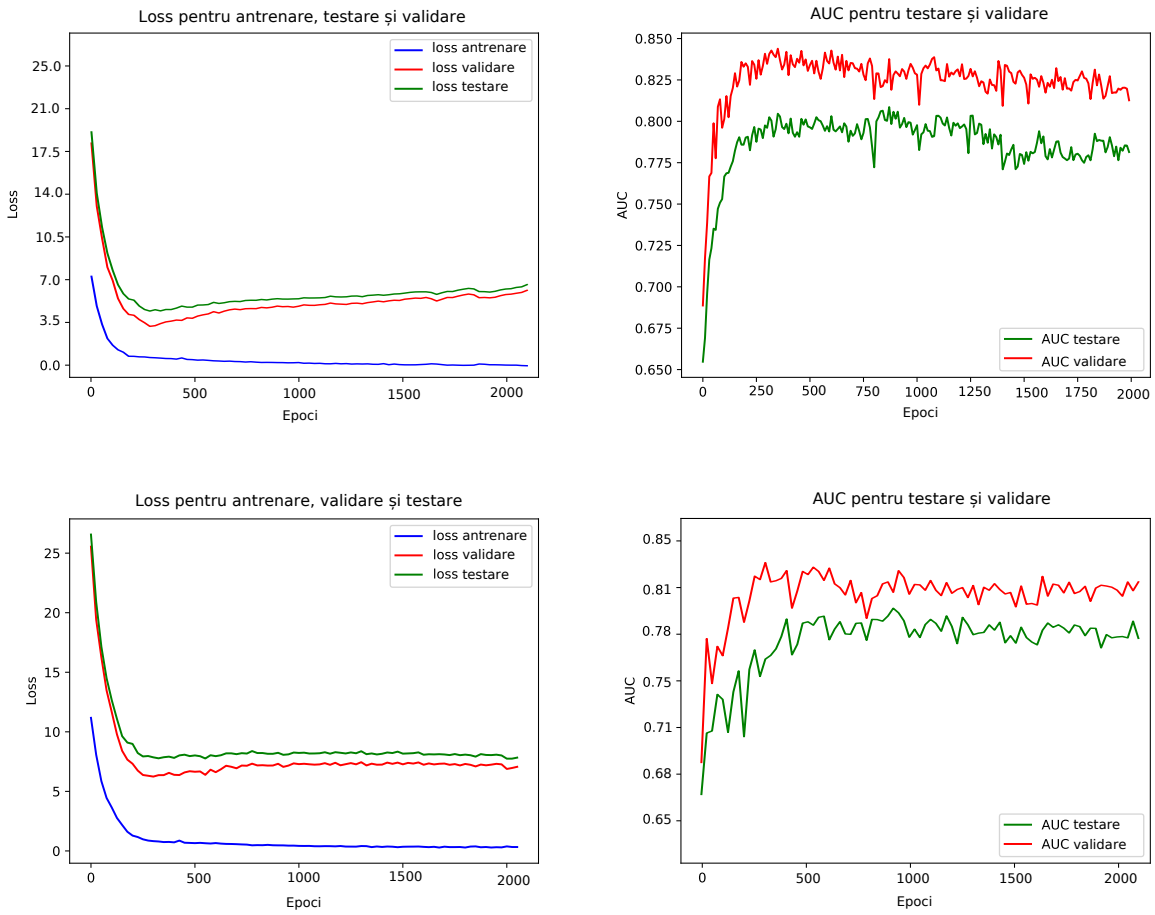


Figura 5.2: Graficele pentru Loss și AUC pentru un experiment cu 5% anomalii. Graficele superioare au fost generate cu inițializarea parametrilor, cele inferioare nu.

Capitolul 6

Concluzii

Lucrarea a studiat în detaliu comportamentul metodelor de detecție a anomaliilor în seturi de date care suferă de *distribution shift*. Am expus fundamente teoretice și am efectuat experimente în diverse scenarii, iar în urma experimentelor efectuate, putem corobora observații importante legate de *distribution shift* și de procedeele de mitigare ale acestuia.

În primul rând, este un fenomen complex și dificil de modelat în mod sintetic, însă am conceput două metode de generare a unui set de date cu distribuții variabile:

- prin generarea aleatoare de matrice de covarianță — această metodă este un caz extrem de *shift*. Generarea aleatoare înseamnă că un model nu poate extrage niciun fel de informație despre natura distribuțiilor, singura constantă între distribuții fiind doar clusterele în care sunt plasați *outlierii*. În acest context, metodele supervizate au avut un avantaj marginal față de cele nesupervizate.

- prin rotația matricei de covarianță — un caz realist de *shift*, în care un model poate să învețe natura distribuțiilor și direcția de deplasare a acestora. Conform ilustrațiilor *t-SNE*, *shift*-ul este generat cu succes, însă seturile de date sunt limitate la spații mici, din moment ce rotația matricei se petrece doar pe 3 dimensiuni. Pentru a putea extrapola pe spații mai mari, se pot folosi algoritmi de generare a rotațiilor în mai multe dimensiuni, ceea ce poate fi studiat mai departe. Și în acest caz învățarea supervizată a produs scoruri mai bune, însă și variantele nesupervizate au reușit să prezică cu acuratețe acceptabilă. Aici se remarcă performanța ACR-DSVDD-ului și capacitatea sa de a învăța tendința *shift*-ului.

Rezultatele experimentelor pe seturile de date sintetice au demonstrat că informația suplimentară de care beneficiază metodele supervizate are un efect pozitiv asupra capacității de generalizare.

În al doilea rând, mulți algoritmi de AD eșuează în condiții de distribuții variabile. Anoshift nu are un spațiu al atributelor foarte larg, însă perioada mare de colectare a

datelor face ca acestea să sufere modificări semnificative. Experimentele pe acest set de date arată că în condiții de *shift* natural, metodele rudimentare scad drastic în acuratețe, însă pot fi îmbunătățite. Am observat în acest experiment cum DeepSVDD-ul simplu, ajutat de mecanismul ACR, își dublează acuratețea și devine cel mai bun model pe acest set de date. ACR este gândit ca o metodă care poate fi aplicată oricărui model de bază, în cazul în care setul de date cu care lucrăm suferă de *shift*. Este relevant să testăm ce model de bază obține acuratețea cea mai bună cu paradigma ACR, așa că poate reprezenta o direcție viitoare de cercetare.

În ultimul rând, am considerat important să confirm aplicabilitatea ACR-DSVDD-ului și pe alte tipuri de date, mai precis pe imagini din CIFAR-100. În acest caz, distribuțiile variază mai subtil, așa că această metodă nu aduce îmbunătățiri semnificative față de alte modele testate.

Bibliografie

[1] Mohiuddin Ahmed, Abdun Naser Mahmood și Jiankun Hu, „A survey of network anomaly detection techniques", în *Journal of Network and Computer Applications* 60 (2016), pp. 19–31, ISSN: 1084-8045, DOI: https://doi.org/10.1016/j.jnca.2015.11.016, URL: https://www.sciencedirect.com/science/article/pii/S1084804515002891.

[2] Tharindu R. Bandaragoda, Kai Ming Ting, David Albrecht, Fei Tony Liu, Ye Zhu și Jonathan R. Wells, „Isolation-based anomaly detection using nearest-neighbor ensembles", în *Computational Intelligence* 34.4 (), pp. 968–998, DOI: https://doi.org/10.1111/coin.12156.

[3] Christopher P. Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins și Alexander Lerchner, *Understanding disentangling in β-VAE*, 2018, arXiv: 1804.03599 [stat.ML].

[4] Varun Chandola, Arindam Banerjee și Vipin Kumar, „Anomaly Detection: A Survey", în *ACM Comput. Surv.* 41 (Iul. 2009), DOI: 10.1145/1541880.1541882.

[5] R. Dennis Cook, „Detection of Influential Observation in Linear Regression", în *Technometrics* 19.1 (1977), pp. 15–18, ISSN: 00401706, URL: http://www.jstor.org/stable/1268249 (accesat în 11.6.2023).

[6] Marius Dragoi, Elena Burceanu, Emanuela Haller, Andrei Manolache și Florin Brad, *AnoShift: A Distribution Shift Benchmark for Unsupervised Anomaly Detection*, 2023, arXiv: 2206.15476 [cs.LG].

[7] Tharindu Fernando, Harshala Gammulle, Simon Denman, Sridha Sridharan și Clinton Fookes, *Deep Learning for Medical Anomaly Detection – A Survey*, 2021, arXiv: 2012.02364 [cs.LG].

[8] Chelsea Finn, Pieter Abbeel și Sergey Levine, *Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks*, 2017, arXiv: 1703.03400 [cs.LG].

[9] Xavier Glorot și Y. Bengio, „Understanding the difficulty of training deep feedforward neural networks", în *Journal of Machine Learning Research - Proceedings Track* 9 (Ian. 2010), pp. 249–256.

[10] Songqiao Han, Xiyang Hu, Hailiang Huang, Mingqi Jiang și Yue Zhao, *ADBench: Anomaly Detection Benchmark*, 2022, arXiv: 2206.09426 [cs.LG].

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren și Jian Sun, *Deep Residual Learning for Image Recognition*, 2015, arXiv: 1512.03385 [cs.CV].

[12] Dan Hendrycks și Thomas G. Dietterich, *Benchmarking Neural Network Robustness to Common Corruptions and Surface Variations*, 2019, arXiv: 1807.01697 [cs.LG].

[13] Waleed Hilal, S. Andrew Gadsden și John Yawney, „Financial Fraud: A Review of Anomaly Detection Techniques and Recent Advances", în *Expert Systems with Applications* 193 (2022), p. 116429, ISSN: 0957-4174, DOI: https://doi.org/10.1016/j.eswa.2021.116429, URL: https://www.sciencedirect.com/science/article/pii/S0957417421017164.

[14] Sergey Ioffe și Christian Szegedy, *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*, 2015, arXiv: 1502.03167 [cs.LG].

[15] H M Dipu Kabir, *Reduction of Class Activation Uncertainty with Background Information*, 2023, arXiv: 2305.03238 [cs.CV].

[16] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton Earnshaw, Imran Haque, Sara M Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn și Percy Liang, „WILDS: A Benchmark of in-the-Wild Distribution Shifts", în *Proceedings of the 38th International Conference on Machine Learning*, ed. de Marina Meila și Tong Zhang, vol. 139, Proceedings of Machine Learning Research, PMLR, 18–24 Jul 2021, pp. 5637–5664, URL: https://proceedings.mlr.press/v139/koh21a.html.

[17] Alex Krizhevsky, „Learning Multiple Layers of Features from Tiny Images", în 2009.

[18] Yann LeCun, Léon Bottou, Yoshua Bengio și Patrick Haffner, „Gradient-Based Learning Applied to Document Recognition", în *Proceedings of the IEEE*, vol. 86, 11, 1998, pp. 2278–2324, URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.7665.

[19] Aodong Li, Chen Qiu, Marius Kloft, Padhraic Smyth, Maja Rudolph și Stephan Mandt, *Zero-Shot Batch-Level Anomaly Detection*, 2023, arXiv: 2302.07849 [cs.LG].

[20] Zheng Li, Yue Zhao, Xiyang Hu, Nicola Botta, Cezar Ionescu și George Chen, „ECOD: Unsupervised Outlier Detection Using Empirical Cumulative Distribution Functions", în *IEEE Transactions on Knowledge and Data Engineering* (2022), pp. 1–1, DOI: 10.1109/tkde.2022.3159580.

[21] Fei Tony Liu, Kai Ting și Zhi-Hua Zhou, „Isolation-Based Anomaly Detection”, în *ACM Transactions on Knowledge Discovery From Data - TKDD* 6 (Mar. 2012), pp. 1–39, DOI: 10.1145/2133360.2133363.

[22] Jiaqi Liu, Guoyang Xie, Jingbao Wang, Shangnian Li, Chengjie Wang, Feng Zheng și Yaochu Jin, *Deep Industrial Image Anomaly Detection: A Survey*, 2023, arXiv: 2301.11514 [cs.CV].

[23] Yezheng Liu, Zhe Li, Chong Zhou, Yuanchun Jiang, Jianshan Sun, Meng Wang și Xiangnan He, *Generative Adversarial Active Learning for Unsupervised Outlier Detection*, 2019, arXiv: 1809.10816 [cs.LG].

[24] Laurens van der Maaten și Geoffrey Hinton, „Viualizing data using t-SNE”, în *Journal of Machine Learning Research* 9 (Nov. 2008), pp. 2579–2605.

[25] Andrzej Maćkiewicz și Waldemar Ratajczak, „Principal components analysis (PCA)”, în *Computers Geosciences* 19.3 (1993), pp. 303–342, ISSN: 0098-3004, DOI: https://doi.org/10.1016/0098-3004(93)90090-R, URL: https://www.sciencedirect.com/science/article/pii/009830049390090R.

[26] *N-dimensional Rotation Matrix Generation Algorithm*, http://article.sapub.org/10.5923.j.ajcam.20170702.04.html, Accesat: 15-06-2023.

[27] *NumPy LinAlg Library*, https://numpy.org/doc/stable/reference/routines.linalg.html, Accesat: 15-06-2023.

[28] Tomás Pevný, „Loda: Lightweight on-line detector of anomalies”, în *Machine Learning* 102 (Iul. 2015), DOI: 10.1007/s10994-015-5521-0.

[29] Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer și Neil D. Lawrence, *Dataset Shift in Machine Learning*, The MIT Press, 2009, cap. 2, ISBN: 0262170051.

[30] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger și Ilya Sutskever, *Learning Transferable Visual Models From Natural Language Supervision*, 2021, arXiv: 2103.00020 [cs.CV].

[31] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller și Marius Kloft, „Deep One-Class Classification”, în *Proceedings of the 35th International Conference on Machine Learning*, ed. de Jennifer Dy și Andreas Krause, vol. 80, Proceedings of Machine Learning Research, PMLR, Oct. 2018, pp. 4393–4402.

[32] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Ursula Schmidt-Erfurth și Georg Langs, *Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery*, 2017, arXiv: 1703.05921 [cs.CV].

[33] Bernhard Schölkopf, John Platt, John Shawe-Taylor, Alexander Smola și Robert Williamson, „Estimating Support of a High-Dimensional Distribution”, în *Neural Computation* 13 (Iul. 2001), pp. 1443–1471, DOI: 10.1162/089976601750264965.

[34] Jungsuk Song, H. Takakura, Yasuo Okabe, Masashi Eto, Daisuke Inoue și Koji Nakao, „Statistical analysis of honeypot data and building of Kyoto 2006+ dataset for NIDS evaluation”, în Apr. 2011, pp. 29–36, DOI: 10.1145/1978672.1978676.

[35] David M.J. Tax și Robert P.W. Duin, „Support Vector Data Description”, în *Machine Learning* 54.1 (Ian. 2004), pp. 45–66, DOI: 10.1023/B:MACH.0000008084.60811.49, URL: https://doi.org/10.1023/B:MACH.0000008084.60811.49.

[36] Sreekanth Vempati, Andrea Vedaldi, Andrew Zisserman și C. V. Jawahar, „Generalized RBF Feature Maps for Efficient Detection”, în *British Machine Vision Conference*, 2010.

[37] Yue Zhao și Maciej K. Hryniewicki, „XGBOD: Improving Supervised Outlier Detection with Unsupervised Representation Learning”, în *2018 International Joint Conference on Neural Networks (IJCNN)*, IEEE, Iul. 2018, DOI: 10.1109/ijcnn.2018.8489605.

[38] Yue Zhao, Zain Nasrullah și Zheng Li, „PyOD: A Python Toolbox for Scalable Outlier Detection”, în *Journal of Machine Learning Research* 20.96 (2019), pp. 1–7, URL: http://jmlr.org/papers/v20/19-011.html.