НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

До захисту допущено
В. о. завідувача кафедри
_____ О.Л. Тимощук
«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системний аналіз і управління»
спеціальності 124 «Системний аналіз»
на тему: «Система розпізнавання кроків на основі вимірів віддалених сенсорів»

Виконав:
студент IV курсу, групи КА-64
Папаш Олексій Андійович _____

Керівник:
професор, д.т.н., професор кафедри ММСА
Данилов Валерій Якович _____

Консультант з економічного розділу:
доцент, к.е.н., доцент кафедри ТТПЕ
Шевчук Олена Анатоліївна _____

Консультант з нормоконтролю:
доцент, к.т.н., доцент кафедри ММСА
Коваленко Анатолій Єпіфанович _____

Рецензент:
доцент, к.т.н., с.н.с кафедри ММСА
Кисельов Генадій Дмитрович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 124 "Системний аналіз"

Освітньо-професійна програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ
В. о. завідувача кафедри
_____ О.Л. Тимощук
«___»_____20__ р.

ЗАВДАННЯ

на дипломну роботу студенту

Папашу Олексію Андрійовичу

1. Тема роботи «Система розпізнавання кроків на основі вимірів віддалених сенсорів», керівник роботи Данилов Валерій Якович, д.т.н., професор, затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом роботи _____

3. Вихідні дані до роботи

4. Зміст роботи

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Шевчук О.А., к.е.н., доцент		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Формулювання тематики (напряму) дослідження.	03.09.2019 – 30.09.2019	
2	Аналіз актуальності задач стосовно тематики дослідження	01.10.2019 – 30.10.2019	
3	Аналіз відомих результатів стосовно тематики дослідження	01.11.2019 – 30.11.2019	
4	Формулювання задач дослідження	01.12.2019 – 30.12.2019	
5	Уточнення теми дипломної роботи	25.02.2019	
6	Збір статичних даних, попередній аналіз даних	01.03.2020 – 30.03.2020	
7	Розробка програмного продукту для виконання обчислювальних експериментів	01.03.2020 – 30.04.2020	
8	Виконання обчислювальних експериментів, аналіз та оформлення результатів	01.05.2020 – 20.05.2020	
9	Оформлення пояснювальної записки у цілому	21.05.2020 – 31.05.2020	
10	Підготовка презентації для захисту	28.05.2020 – 01.06.2020	
11	Попередній захист дипломної роботи	01.06.2020 – 03.06.2020	
12	Захист дипломної роботи	15.06.2020 – 18.06.2020	

Студент Олексій ПАПАШ

Керівник Валерій ДАНИЛОВ

РЕФЕРАТ

Дипломна робота містить: 82 с., 40 рис., 7 табл., 2 дод. та 16 джерел.

СЕЙСМОАКУСТИКА, РОЗПІЗНАВАННЯ КРОКІВ, SIGNAL PROCESSING, STEP DETECTION, GEOPHONE

Об’єкт дослідження – виміри сейсмоакустичних хвиль, отримані з віддаленого приймального модуля.

Предмет дослідження – вейвлет перетворення та нейронні мережі.

Мета дипломної роботи – дослідити існуючі методи обробки сейсмоакустичних хвиль; знайти методи класифікації сигналів та реалізувати систему розпізнавання кроків на основі даних про них.

У роботі проведено дослідження сейсмоакустичних сигналів та методів їх класифікації. У ході виконання дипломної роботи було розроблено програмний продукт та отримано результати для кожного з методів, а також проведено їх порівняльний аналіз. Під час дослідження було виявлено, що використання нейронних мереж дозволяє отримати задовільні результати. За виконання подальших досліджень доцільно провести більше експериментів для отримання більшої навчальної вибірки, а також використати більш складні моделі.

ABSTRACT

Diploma work: 82 p., 40 fig., 7 tables, 2 appendixes, 16 sources.

SEISMOACOUSTICS, SIGNAL PROCESSING, STEP DETECTION, GEOPHONE

The object of research is seismic acoustic wave measurements obtained from a remote receiving module.

The subject of research - wavelet transform and neural networks.

The purpose of the thesis - to explore existing methods of processing seismic acoustic waves; find methods for signal classification and implement a step recognition system based on data about them.

The research of seismoacoustic signals and methods of their classification is carried out in the work. In the course of the thesis a software product was developed and the results for each of the methods were obtained, as well as their comparative analysis. During the study it was found that the use of neural networks allows to obtain satisfactory results. For further research, it is advisable to conduct more experiments to obtain a larger training set, as well as to use more complex models.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ВП – вейвлет перетворення.

ПЗ – програмне забезпечення.

СПР – система прийняття рішень.

WS – wavelet scattering

ВСТУП

На сьогоднішній день модель та поведінка сейсмоакустичні хвилі на поверхні земної кори мало досліджені порівняно із звуковими хвилями в повітрі, гідроакустичними хвилями, радіохвилями та глибинними сейсмічними хвилями. Малодослідженість сейсмоакустичних хвиль саме у верхніх шарах ґрунту підтверджується невеликою кількістю статей, пов’язаних з обробкою та аналізом таких хвиль [15-16]. Тому дослідження цих сигналів і знаходження методу розпізнавання кроків – це один із перших і вайжливих етапів до кращого розуміння і побудови моделі поширення даних хвиль.

Проте це дослідження має не тільки академічну цінність, але також і актуальне з практичної сторони, оскільки на основі нього можна будувати системи для охорони територій, наприклад, кордону. Також можливим застосуванням є побудова систем для відслідковування та дослідження руху тварин у заповідниках чи лісництвах.

РОЗДІЛ 1 ОГЛЯД СЕНСОРУ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.2 Огляд сенсору

У дослідженні використовуються дані, отримані від приймального модуля, розробленого у КПІ ім. Ігоря Сікорського. Основними його елементами для детектування сейсмоакустичних сигналів є геофони марки GS-20DX (вертикальний давач) та GS-20DXB (горизонтальний давач). Ці сенсори є приймачами коливальної швидкості, тобто приймачами звукових хвиль, які поширюються у верхніх шарах земної кори. Вони перетворюють рух ґрунту (зсуви) в напругу. Вимірами даного пристрою є градієнтні характеристики руху. Частота перетворення аналогового сигналу 800 Гц. На рис. 1.1 зображено схему розположення приймачів в приймальному модулі.

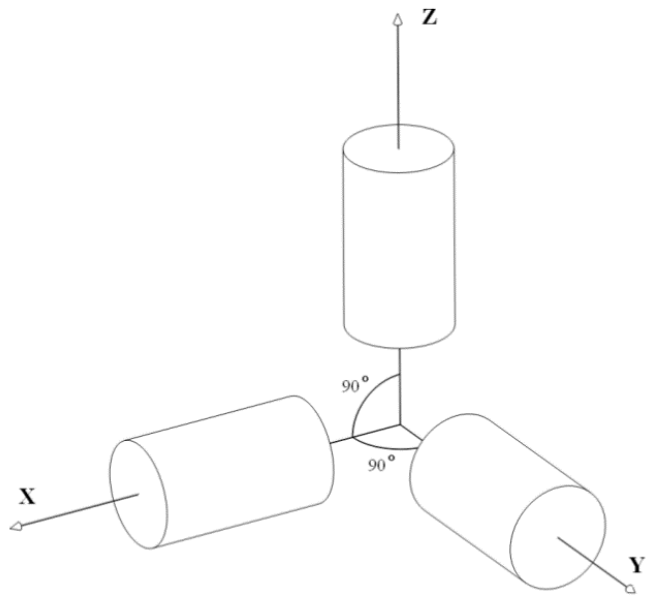


Рисунок 1.1

Вони мають косинусну діаграму спрямованості або «вісімку», тобто орієнтованих в 3-х ортогональних напрямках X, Y, Z.

1.3 Опис вимірів та отриманих даних експериментів

Дані з приймача отримуються у вигляді чотирьох каналів (це зумовлено конструкцією приладу), три із них відповідають складовій X, Y, Z, а один не несе корисної інформації, тому його можна ігнорувати. Тому перед початком аналізу усі файли експериментів були очищені від зайвого стовпчика даних. Було проведено 28 експериментів, під час яких приймальний прилад розміщували в ґрунті, а людина рухалася з різною швидкістю вздовж осі X або Y починаючи і закінчуючи ходьбу приблизно в 50 кроках від приладу. Тобто експерименти були наступного вигляду:

- людина крокує вздовж осі X;
- людина крокує вздовж осі Y;
- людина біжить вздовж осі X;
- і так далі.

Також додатково провівся запис шуму середовища. На рис. 2.1 зображено схему експериментів.

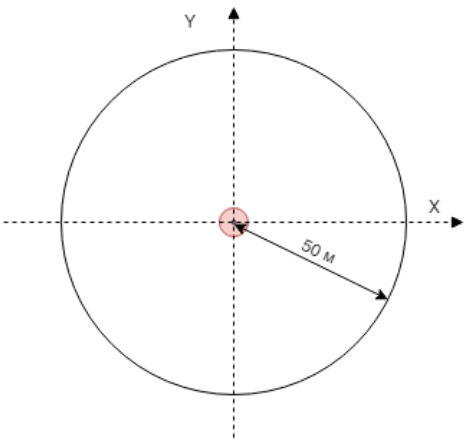


Рисунок 2.1

1.4 Постановка задачі дослідження

Для дослідження була запропонована постановка задачі, яка складається з наступних кроків:

1. Виконати огляд відомих методів дослідження та попередньої обробки сейсмоакустичних сигналів і сигналів в цілому.
2. Для задачі ДП сформулювати задачу класифікації та виконати огляд підходів для її вирішення.
3. Вибрати ефективні інструменти, за допомогою яких реалізувати попередні пункти.
4. Програмно реалізувати моделі для розв’язання задач попередньої обробки і класифікації та розробити систему прийняття рішень (СПР) на їх основі.
5. Використати створене ПЗ для обробки експериментальних даних та проаналізувати результати.

1.5 Висновки до розділу

У першому розділі наведено короткий опис приймального приладу, оглянуто приймачі сигналів. Розуміння конструкції дає змогу краще бачити картину необхідних кроків для дослідження. У розділі був описаний принцип проведених експериментів, дані яких досліджуються в даній роботі. Експерименти з яскраво вираженими кроками будуть проаналізовані в першу чергу. Також сформовано постановку задачі даного дипломного проекту.

РОЗДІЛ 2 ПОПЕРЕДНЯ ОБРОБКА ТА КЛАСИФІКАЦІЯ

2.1 Попередня обробка даних

Попередня обробка є важливим кроком у дослідженні, оскільки вона дає змогу виділити характерні ознаки для даних. Для дослідження вимірів сенсора було обрано вейвлет аналіз. Його суть полягає в тому, що проводиться вейвлет перетворення сигналу, потім результат візуалізується і далі відбувається пошук закономірностей або аномалій. Перевагою вейлевет-перетворення (ВП) перед перетворенням Фур'є є те, що ВП дозволяє прослідкувати за змінами чи тенденціями спектральних властивостей сигналу в часі і вказує на ті частоти (масштаби), що домінують у сигналі [1].

Згідно із [1] , вейвлет має бути неперервним, інтегровним, мати компактний носій, а також бути локалізованим як у часі, так і в частоті. Для того, щоб функцію можна було вважати достатньо добре локалізованою за виконання наступних умов:

$$\psi(t) < \frac{C}{(1+|t|)^{1+\varepsilon}}, \qquad \Psi(f) \leq \frac{C}{(1+|f|)^{1+\varepsilon}}, \qquad \varepsilon > 0$$

Також вейвлет має бути обмеженим

$$\|\psi(t)\| = \int_{-\infty}^{\infty} |\psi(t)|^2 dt < \infty$$

Середнє значення вейвлету повинно бути нульовим, тобто для нульового моменту має виконуватися:

$$\int_{-\infty}^{\infty} \psi(t)dt = 0$$

Також виникають ситуації, коли необхідно, щоб не тільки нульовий момент, але і всі перші n моментів дорівнювали нулю:

$$\int_{-\infty}^{\infty} t^n \psi(t)dt = 0$$

Вейлети n-го порядку дозволяють досліджувати структуру сигналу, що є більш високочастотною, та ігнорувати складові, що повільно змінюються.

Також важливою властивістю вейвлета є автомодельність або самоподібність. Усі вейвлети конкретного сімейства $\psi_{ab}(t)$ повинні мати ту ж форму, що і в материнського вейвлета, тобто однакове число осциляцій, незалежно від зсувів a і масштабування b (стиснення/розтягування).

Тобто в загальному випадку вейвлети $\psi_{ab}(t)$ конструюються із материнського вейвлета $\psi(t)$ шляхом часового масштабування a і зсуву по часу b:

$$\psi_{ab}(t) = \frac{1}{\sqrt{|a|}} \psi\left(\frac{t-b}{a}\right), \qquad a, b \in \mathbb{R}$$

Множник $\frac{1}{\sqrt{|a|}}$ забезпечує незалежність норми функції від масштабуючого числа a.

Одним із найперших і найпростіших вейвлетів є вейвлет Хаара. Він був запропонований Альфредом Хааром в 1909 році. Вейвлет Хаара використовують для стискання сигналів та компресії зображень. Він зображений на рис. 2.1, материнський вейвлет задається наступним чином:

$$\psi(t) = \begin{cases} 1, & 0 \le t < \frac{1}{2} \\ -1, & \frac{1}{2} \le t < 1 \\ 0, & t \notin [0,1) \end{cases}$$

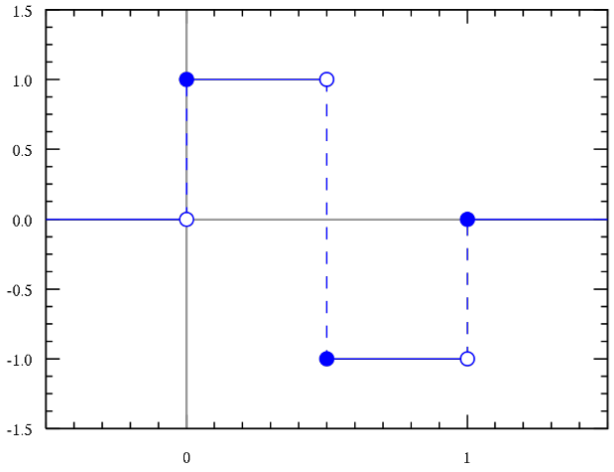


Рисунок 1.1

Вейвлет «Мексиканський капелюх» зображений на рис. 2.2 і має формулу:

$$\psi(t) = \frac{2}{\sqrt{3\sigma}\,\pi^{\frac{1}{4}}}\left(1 - \left(\frac{t}{\sigma}\right)^2\right) e^{-\frac{t^2}{2\sigma^2}}$$

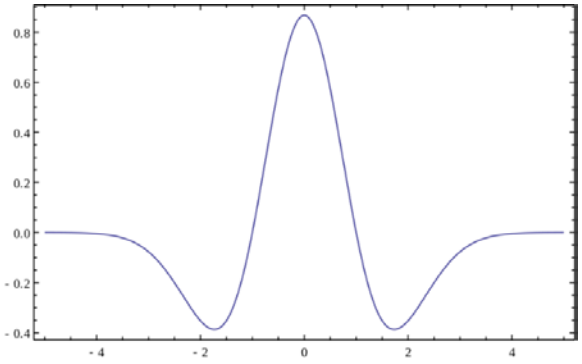


Рисунок 2.2

Проте найбільшу увагу було зосереджено на комплексному вейвлеті Морле і саме він буде далі використовуватися у роботі. Оскільки цей вейвлет був запропонований інженером-геофізиком Д. Морле, коли він зіштовхнувся із проблемою аналізу сигналів сейсмодатчиків, що мали високочатотну складову протягом короткого періода і низькочастотні складові протягом довгого періоду. Віконне перетворення Фур'є не давало змоги аналізувати обидві складові одночасно. Материнський вейвлет Морле зображено на рис. 2.3 і задається наступним чином:

$$\psi(t) = e^{-\frac{t^2}{2}} e^{i\omega t}$$

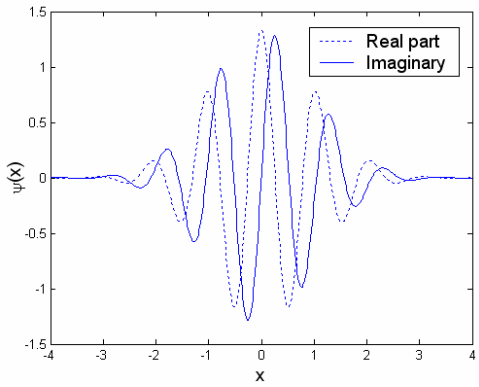


Рисунок 2.3

2.2 Задача класифікації

Для того щоб знати методи класифікації сигналів потрібно мати чітке уявлення задачі класифікації сигналів, що досліджуються у даному ДП. Тому сформулюємо її наступним чином: нехай маємо вектор ознак V , що може бути самим сигналом $S \in \mathbb{R}^n$ або наприклад його вейвлет перетворенням $\psi(S)$.

Суть задачі класифікації полягає у тому, що необхідно знайти функцію

$$f: V \to \{0,1\}$$

Було прийнято, що 1 означає, що на вхідних даних присутній крок, а 0 це відсутність кроку. Задачі такого типу можна вирішувати за допомогою нейронних мереж [2-3]. Тому для вирішення задачі ДП було обрано багатошаровий персептрон (рис. 2.4), проте з деякими модифікаціями, що будуть описані далі.

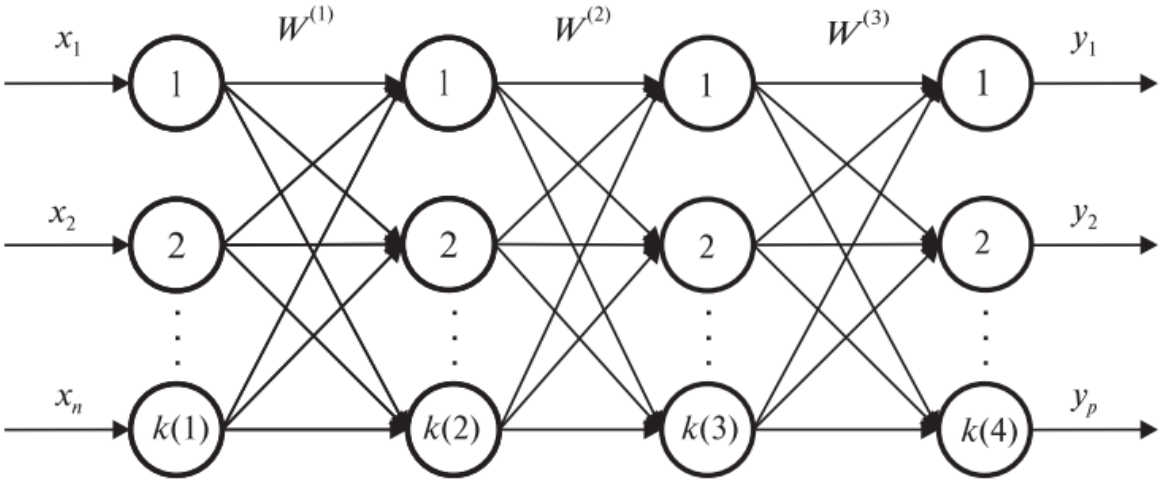


Рисунок 2.4

Архітектура багатошарового персептрона складається з багатьох шарів нейронних елементів. Першим шаром є вхідний слой на який подається вектор ознак, а останнім є вихідний шар, що дає кінцеві результати, закінчивши обробку інформації, яку він отримує від попередніх шарів. Шари, що знаходяться між вхідним і вихідним, називаються прихованими. Вони, так само як і вихідний шар, є оброблюючими. Вихід кожного нейрону попереднього шару з'єднаний з усіма входами нейронів наступного шару.

Нехай $W^{(i)}$- матриця вагових коефіцієнтів i-го шару. Тоді вихідне значення мережі Y для мережі з двома прихованими шарами можна визначити наступним чином [3]:

$$Y = F\big(F\big(F\big(XW^{(1)}\big)W^{(2)}\big)W^{(3)}\big),$$

де $X = (x_1, x_2, \dots, x_n)$ – вхідний сигнал,

F – оператор нелінійного перетворення.

Загальну кількість зв'язків багатошарового перспептрону можна підрахувати наступним чином [3]:

$$V = \sum^{p-1} n(i)n(i+1) + \sum_{i=1}^{p-1} n(i+1),$$

де p – це загальна кількість шарів, $n(i)$ – це кількість нейронів в i-му шарі.

Оператор F також називається функцією активації. Розглянемо декілька стандартних функцій активації. Найбільш поширеною функцією

активації попереднього століття була сигмоїдна функція (рис. 2.5). Вихідне значення j-го нейрону визначається наступним чином:

$$y_j = F(S_j) = \frac{1}{1 + e^{-S_j}}$$

$$F'(S_j) = y_j(1 - y_j)$$

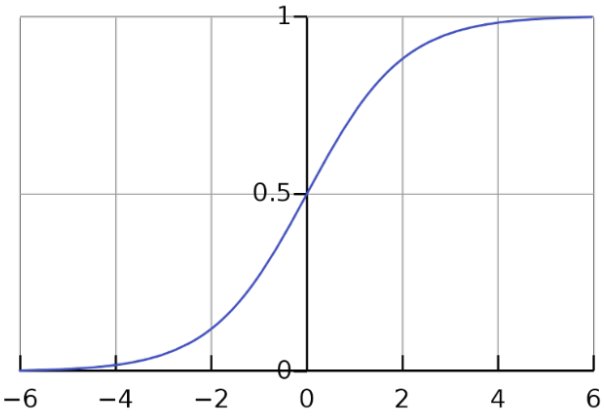


Рисунок 2.5

Також популярною функцією активації є гіперболічний тангенс (рис. 2.6):

$$y_j = th(S_j) = \frac{e^{S_j} - e^{-S_j}}{e^{S_j} + e^{-S_j}}$$

$$\mathrm{F}'(\mathrm{S_j}) = (1 - y_j^2)$$

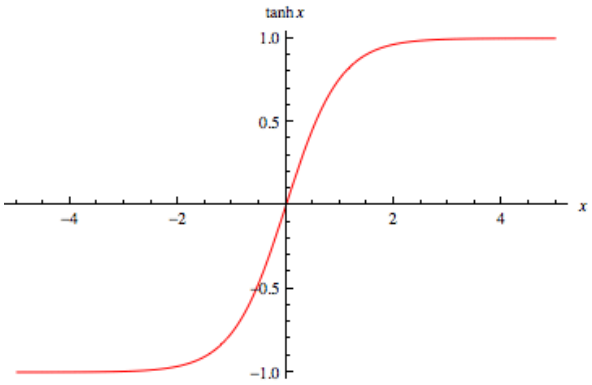


Рисунок 2.6

Проте найбільш цікавою функцією активації стала ректифікаційна функція активації ReLU (рис 2.7) у 2011 році, коли вийшла стаття [4], оскільки ReLU похідну дуже просто вичисляти. Функція має наступний вигляд:

$$y_j = F(S_j) = \begin{cases} S_j, & S_j > 0, \\ kS_j, & S_j \leq 0' \end{cases}$$

де $k = 0$ або якомусь маленькому значенню, наприклад $k = 0.001$

$$F'(S_j) = \begin{cases} 1, & S_j > 0 \\ k, & S_j \leq 0 \end{cases}$$

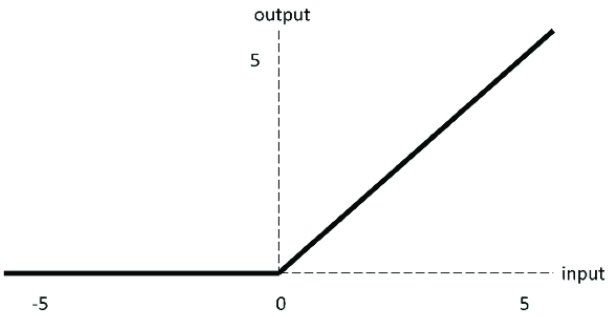


Рисунок 2.7

2.3 Навчання персептрона

Розглянемо процес навчання персептрона описаний у [2-3]. Нехай маємо навчальну вибірку (x_j, y_j), $j = 1 \dots N$. Під навчанням персептрона розуміємо знаходження таких вагових коєфіцієнтів у матрицях $W^{(i)}$, щоб мінімізувати сумарну квадратичну помилку нейронної мережі:

$$L = \frac{1}{2}\sum_{k=1}^{N}\sum_{j=1}^{p}\left(y_j^k - e_j^k\right)^2,$$

де y_j^k та e_j^k – це вихідне значення і значення з навчальної вибірки для k-го образу відповідно.

Вихідне значення j-го нейронного елемента для k-го образа дорівнює

$$y_j^k = F\left(S_j^k\right)$$

$$S_j^k = \sum_i \omega_{ij} y_i^k$$

Існує два підходи до навчання: послідовне (online) та групове (batch) навчання. У випадку послідовного навчання ваги зв’язків мережі модифікуються після кожного образу з начальної вибірки. Тоді для мінімізації сумарної квадратичної помилки мережі, ваги мережі повинні змінюватися наступним чином [3]:

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha \frac{\partial E}{\partial \omega_{ij}(t)}$$

Проте

$$\frac{\partial E}{\partial \omega_{ij}} = \left(y_j^k - e_j^k\right) F'\left(S_j\right) y_i$$

Тому в загальному правило зміни ваг буде виглядає так:

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha\left(y_j - e_j\right) F'\left(S_j\right) y_i$$

Отримане правило є правилом навчання багатошарових персептронів в загальному виді і називається узагальненим дельта-правилом.

У випадку групового навчання зміна ваг зв'язків нейронів відбувається після подачі підмножини або всієї множини навчальних образів на вхід нейронної мережі. Нехай подаємо $n \leq N$ образів на вхід мережі, тоді модифікація ваг задається наступним правилом:

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha \frac{\partial E(n)}{\partial \omega_{ij}(t)},$$

де $\omega_{ij}(t)$ – ваги мережі до подачі образів;

$E(n)$ – це сумарна квадратична помилка мережі для n образів.

$$E(n) = \frac{1}{2}\sum_{k=1}^{n}\sum_{j=1}^{p}\left(y_j^k - e_j^k\right)^2,$$

Проте

$$\frac{\partial E(n)}{\partial \omega_{ij}} = \sum_{k=1}^{n}\left(y_j^k - e_j^k\right)F'\left(S_j^k\right)y_i^k$$

Тому отримуємо так зване узагальнене дельта-правило для групового навчання [3]:

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha\sum_{k=1}^{n}\left(y_j^k - e_j^k\right)F'\left(S_j^k\right)y_i^k$$

Для мінімізації помилки E використовується алгоритм зворотного поширення помилки. Цей алгоритм був запропонований у 1986 році, проте до сих пір актуальний для навчання, не тільки багатошарових персептронів, але й нейронних мереж будь-якого типу. Нижче розписаний алгоритм зворотнього поширення помилки для послідовного навчання [2-3], який складається з наступних кроків:

1. Спочатку задається швидкість навчання (learning rate) α $(0 < \alpha < 1)$ і бажане значення середньоквадратичної помилки нейронної мережі E.

2. Вагові коефіцієнти ініціалізуються випадковими значеннями в достатньо вузькому діапазоні $[-0.01, -0.1]$

3. З навчальної вибірки на послідовно на вхід подаються вхідні образи $k = 1, N$ і для кожного такого образу виконуються наступні дії:

а) Відбувається прохід уперед (forward pass) по нейронній мережі, під час якого вираховуються виходи всіх нейронів мережі:

$$y_j = F\left(\sum_i \omega_{ij} y_i\right),$$

де ідекс j характеризує нейрони наступного шару відносно шару i.

б) Відбувається зворотній прохід (backward pass), тобто зворотнє поширення сигналу, в резульаті якого визначається похибка $\gamma_j, j = 1,2, ...,$ нейронів для всіх шарів мережі. Зокрема, для вхідного та вихідного шару відповідно дорівнюють:

$$\gamma_j = \sum_i \gamma_i F'(S_i)\omega_{ji}$$

$$\gamma_j = y_j - e_j$$

в) для кожного шару відбувається оновлення вагових коефіцієнтів за правилом:

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha\gamma_j F'\big(S_j\big)y_i$$

4. Після цього підраховується сумарна квадратична помилка мережі.

5. Якщо помилка задовільняє бажане значення, тоді алгоритм закінчується. Якщо ні відбувається перехід до пункту 3.

Проте під час використання алгоритму зворотнього поширення помилки, в основі якого є градієнтний метод, можуть виникнути наступні проблеми [2][3]:

- Повільна сходимість градієнтного метода з постійною швидкістю навчання
- Для класичного градієнтого методу потрібно підбирати швидкість навчання

Тому зазвичай в навчанні нейронних мереж не використовують класичний градієнтний метод, а його модифікації[3]. Під час практики було розглянуто декілька з цих модифікацій.

2.4 Оптимізація градієнтного спуску

Першим був градієнтний спуск з моментом [5]. Цей метод пришвидує градієнтний спуск, оскільки замість того щоб залежати від градієнта функції, він залежить від швидкості, яка виражається як:

$$v_{t+1} = \mu v_t - \alpha \nabla E(\omega_t)$$

А саме правило зміни ваг стає наступним:

$$\omega_{t+1} = \omega_t + v_{t+1}$$

Тут μ – це коефіцієнт моменту і знаходиться у відрізку [0,1]. Такий підхід дозволяє зменшити процедуру, оскільки швидкість збільшує зміщення в потрібному напрямку і зменшує кількість осциляцій. Нижче наведено (рис. 2.8) роботу класичного градієнтного спуску (a) та градієнтного спуску з моментом (b).

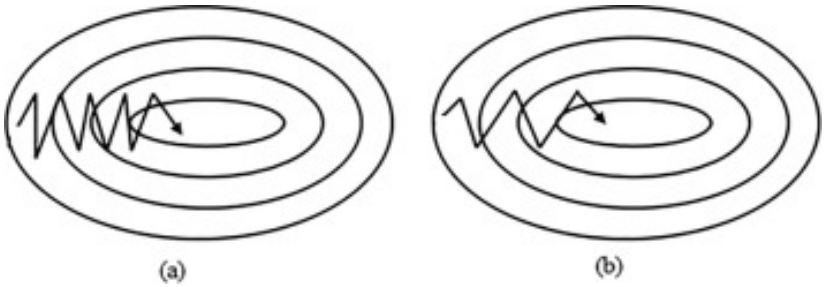


Рисунок 2.8

Наступним було розглянуто метод адаптивного градієнту або AdaGrad [6]. Особливістю цього методу є те що він звертає увагу на розміри оновлення параметрів. Нехай G_t – сума квадратів оновлень, а g_t – значення градієнту. Тоді AdaGrad можна записати наступним чином:

$$G_{t+1} = G_t + g_t^2$$

$$\omega_{t+1} = \omega_t - \frac{\alpha}{\sqrt{G_t + \varepsilon}} g_t$$

Тут $\varepsilon > 0$ вводиться для того, щоб уникнути ділення на 0.

Недоліком AgaGrad є те, що G_t не обмежена величина, а тому може збільшитися настільки, що будуть відбуватися маленькі оновлення вагів і алгоримт паралізує. Для вирішення цієї проблеми були придумані алгоритми RMSProp [7] та Adadelta [8]. В обох алгоритмах залишилася ідея менше оновлювати ваги, які занадто часто оновлюються, проте замість всієї суми оновлень, буде використовуватися усереднене значення квадрата градієнта. Для цього використовують експоненційне ковзне середнє, яке задане як

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2$$

Тому підставивши $E[g^2]$ замість G_t отримуємо

$$\omega_{t+1} = \omega_t - \frac{\alpha}{\sqrt{E[g^2]_t + \varepsilon}} g_t$$

Знаменник це корінь з середнього квадратів градієнтів, тому отриманий алгоритм так і називається RMSProp – root mean square propagation.

$$RMS[g]_t = \sqrt{E[g^2]_t + \varepsilon}$$

Алгоритм Adadelta відрізняється від RMSProp тим, що замість $RMS[g]_t$ використується $RMS[\Delta\omega]_t$:

$$\Delta\omega = -\frac{RMS[\Delta\omega]_{t-1}}{RMS[g]_t} g_t$$

$$\omega_{t+1} = \omega_t - \frac{RMS[\Delta\omega]_{t-1}}{RMS[g]_t} g_t$$

$$E[\omega^2]_t = \gamma E[\omega^2]_{t-1} + (1-\gamma)\omega_t^2$$

$$RMS[\Delta\omega]_t = \sqrt{E[\Delta\omega^2]_t + \varepsilon}$$

Можна помітити, що для коректної роботи алгоритму $RMS[\Delta\omega]_{-1}$ має бути ненульовим, а інашке всі наступні значення будуть дорівнювати нулю. Але це вже вирішено додаванням ε. Перевагою цих алгоритмів є той факт, що для них не потрібно підбирати швидкість навчання.

2.5 Модифікації моделі

Для навчання кращого класифікатору, замість чистого сигналу на вхід персептрону можна подавати результати вейвлет перетворення. Проте під час дослідження було знайдено статті [9-11], в яких описується підхід до отримання інваріативного до зсувів вигляду сигналів за допомогою вейвлет перетворень, який називається wavelet scattering (WS). На рис. 2.9, взятому з [11], зображена схема цього перетворення.

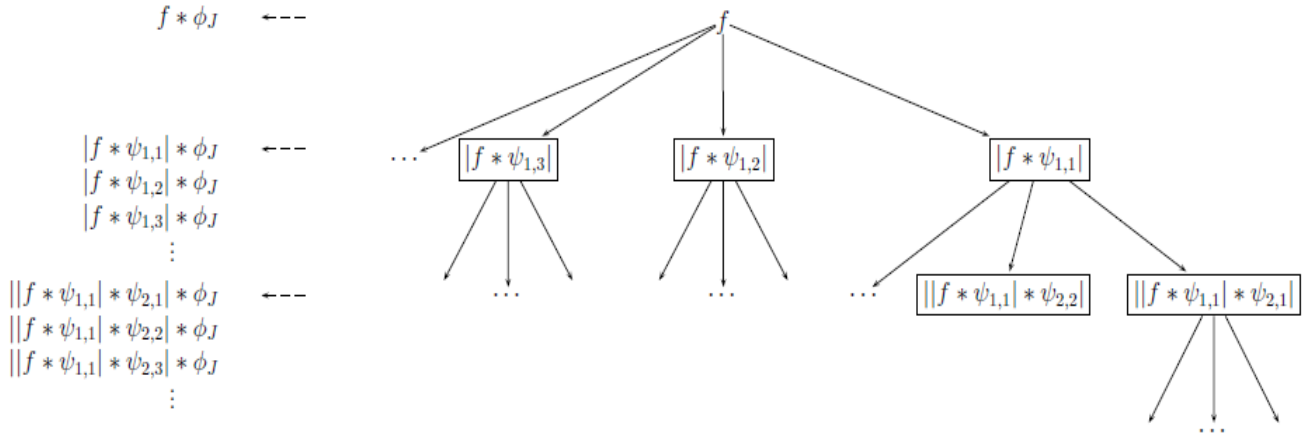


Рисунок 2.9

На схемі $\{\psi_{j,k}\}$ – це вейвлети, ϕ_J – функції масштабування, а вхідні дані це f. Це перетворення генерує ознаки в ітеративній процедурі. Спочатку, відбувається згортка даних із функцією масштабування $f * \phi_J$, результатом якої є коефіцієнти нульового порядку (що є простим усередненням сигналу).

Після цього відбувається наступне:

1. Отримується вейвлет перетворення вхідних даних.
2. Береться модуль результатів.
3. Результати усереднюються за допомогою фільтрів.

Так відбувається для кожного вузла.

Також для обробки і виокремлення часових характеристик використовують енкодери, в яких використовуються одновимірні згорткові шари та рекурентні шари, тобто шари зі зворотніми зв’язками [12-14].

2.6 Висновки до розділу

У даному розіділі було оглянуте вейвлет перетворення як засіб під час попередньої обробки сигналу, була сформульована задача класифікації і розглянута персептрон у ролі класифікатора. Також для передобробки сигналів запропоновано WS та шляхом додавання рекурентних (наприклад LSTM) та згорткових шарів до персептрона для виокремлення просторово-часових ознак. Також у розділі розглянуто процедуру навчання багатошарового перспетрона та її оптимізацію, що може допомогти під час реалізації і навчання нейронної мережі.

РОЗДІЛ 3 ВИБІР ІНСТРУМЕНТІВ ТА РЕАЛІЗАЦІЯ МОДЕЛЕЙ

3.1 Огляд інстументів

Серед інструментів, за допомогою яких можна реалізувати вище описані методи і моделі, розглядалися дві альтернативи – прикладний пакет для чисельних розрахунків MATLAB і мова програмування Python.

Проте після того як був проведений аналіз даних варіантів, то було обрано Python, оскільки Python має більше бібліотек для вирішення даної задачі, зокрема:

- pywavelets для вичислення вейвлет перетворень;
- pandas для зчитування даних із файлів різного формату;
- подальшого представлення їх у табличному форматі та роботи над цими таблицями;
- numpy для використання багатовимірних масивів та високорівневих математичних функцій;
- kymatio для використання wavelet scattering;
- sklearn для ефективної обробки та аналізу даних;
- matplotlib та scaleogram – для візуалізації графіків та результатів алгоритмів;
- tensorflow та keras для створення та навчання нейронних мереж.

Для розробки використовувалися редактор коду Visual Code та хмарний сервіс Google Colab, який дає змогу навчани нейромережі за допомогою потужностей відеокарт.

3.2 Попередня обробка

Для попередньої обробки було реалізовано модуль, що проводить вейвлет перетворення вхідних даних та візуалізує його. У модулі передбачена зміна материнського вейвлету, масштаб та можливість нормалізувати результати за Z каналом та за частотою.

Під нормуванням по Z складовій розуміється, нормування отриманих коефіцієнтів вейвлет перетворення за наступним принципом: для кожного каналу проводиться вейвлет перетворення і отримуються коефіцієнти, після цього береться максимальний за значенням коефіцієнт каналу Z, це значення встановлюється верхнею межею відображення графіків вейвлет перетворення всіх інших каналів. Під нормуванням за частотою розуміється, що коефіцієнти, які відповідають конкретній частоті нормалізуються до відрізку [0,1].

3.3 Класифікатор

Для навчальної вибірки було вручну розмічено і відібрано приблизно 300 кроків спосеред 4 експериментів. Оскільки довжина крока є невідомою, то відмічені були середини кроків, а в ПЗ реалізована генерація кроків на основі їх середин залежно від заданої довжини кроку, яка полягає у тому, що від середини відступається в обидві сторони половина заданої довжини.

Проте розмічені були тільки кроки, тому для того щоб збалансувати

навчальну вибірку даними про «не кроки», було вирішено створити такі записи використавши виміри з експерименту, під час якого записувався шум навколишнього середовища. Його виміри були розділені на фрагменти з вікном, що дорівнює довжині кроку та кроком руху вікна (64, 128, 256 або 400 відліки). А потім з цієї множини випадковим чином обрано 300 «не кроків». В цілому з навчальної вибірки було відібрано 5% у тестову вибірку.

Як було зазначено вище для реалізації та навчання нейромереж були використані бібліотеки tensorflow та keras.

Цей процес складався з наступих пунктів:

- Створювалася порожня послідовна модель;
- До моделі додавалися шари;
- Для моделі визначалися функція втрат, метрики та оптимізатор;
- Власне процес навчнаня.

Keras дозволяє створювати мережі пошарово, тобто структуру мережі можна створити послідовно додавши до моделі шари. Шари, що були використані були наступними:

- Input – це вхідний шар, параметром якого є розмірність вхідних даних;
- Dense – це повнозв’язний шар, тобто кожен нейрон якого з’єднаний з усіма нейронами попереднього шару;
- LSTM – рекурентнйи шар;
- Lambda – шар, що виконує над входами задану функцію;
- GlobalAveragePooling1D – рухає вікно по даних і знаходить середнє значення у цьому вікні;
- BatchNormalization – нормалізує вхідні дані;
- Conv1D – цей шар проводить одновимірну згортку.

Також з бібліотеки kymatio був взятий шар Scattering – шар, що виконує одновимірне перетворення WS. Особливістю цього шару є те, що розмірність його вхідних даних повинна бути степенем двійки.

Було реалізовано два типи мереж:

1) Багатошаровий персептрон із WS перетворенням після вхідного шару.

2) Багатошаровий перспетрон із lstm екодером.

Моделі першого типу, відрізнялися розмірністю вхідного шару. Модель другого типу була створена одна із довжиною кроку 256. Також Keras дозволяє візуалізувати схему мережі. Далі буде наведено детальніший опис створених типів моделей.

Перший тип мережі був побудований за рекомендаціями, описаними в [11]. На рис. 3.1 наведено схему мережі.

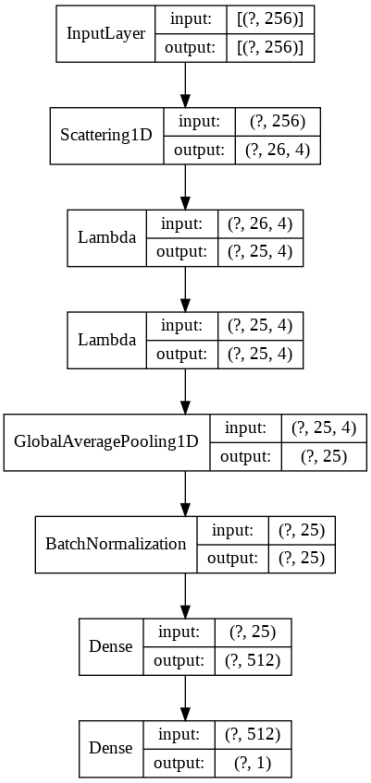


Рисунок 3.1

Перший Lambda шар видаляє коефіцієнти нульового порядку, оскільки вони не несуть корисної інформації, а другий шар, для того щоб збільшити значимість коефіцієнтів, бере логарифм від них. Далі йде шар GlobalAverage Pooling1D, який усереднює результати по часу, для того щоб досягнути інваріантності до зсувів. Після дані нормалізуються шаром BatchNormalization і потрапляють на вхід до повнозв'язного шару з функцією активації ReLU. Останній повнозв'язний шар є вихідним. Він має один вихід, тому що вкінці мережа повинна дати результат: чи є даний сигнал кроком чи ні. По суті відбувається бінарна класифікація, а для такого типу класифікації ідеально підходить сигмоїдальна функція активації.

Другий тип мережі (рис. 3.2) був створений на основі [12-13].

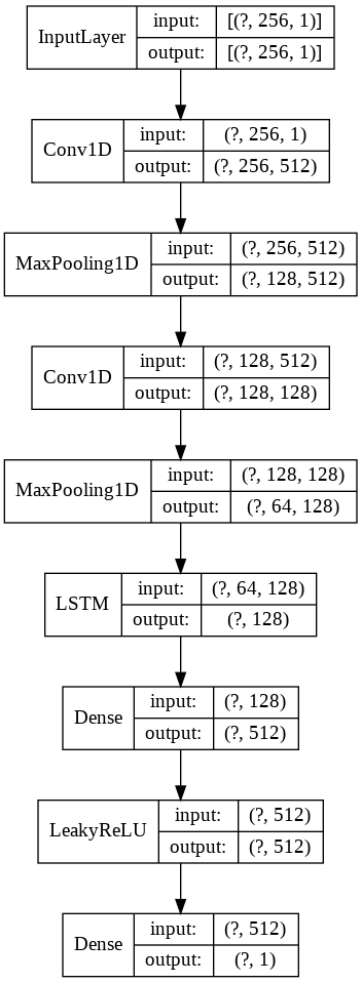


Рисунок 4.2

Зі схеми видно, що на початку відбуваються згортка і maxpooling послідовно двічі. Вони мають виділити структурні характеристики сигналу, а наступний LSTM шар повинен виокремити ще й часові. Повнозв'язні шари є аналогічними до шарів в попередній мережі, перший має фукнцію активації ReLU, а другий сигмоїду.

Оскільки класифікатор дає відповідь чи на сигналі крок чи ні, то фактично відбувається бінарна класифікація. Тому у ролі функції втрат було обрано не сумарну квадратичну помилку, а бінарну кросентропію (binary cross entropy):

$$L(y, \hat{y}) = -\frac{1}{n}\sum_{i=0}^{n}(y * \log(\hat{y}_i) + (1-y) * \log(1-\hat{y}_i)),$$

де $\hat{y}$ – це прогнозоване значення класу.

Бінарна кросентропія вимірює наскільки прознозоване значення класу далеко від справжнього значення (що дорівнює 0 або 1), а тоді усереднює помилки.

У ролі оптимізаторів навчання були обрані RMSprop та Adadelta. Для обох мереж RMSprop сходився до бажаних результатів краще за Adadelta. Навчання обох типів моделей було однаковим.

Навчання тривало 100 епох, але якщо точність перевищувала поріг в 90%, тоді навчання зупинялося. Мережа першого типу навчалася швидше за другу. Це пояснюється тим, що вона має меншу кількість параметрів, які потрібно навчати. Точність класифікації обох мереж в межах 85-95%.

3.4 Реалізація СПР

Оскільки моделі класифікують вхідні сигнали, але не враховують положення кроку в сигналі, то для коректного розпізнавання кроків у вхідному сигналі, необхідно реалізувати метод для локалізації кроків. Найбільш простим і логічним буде знаходження перетину близьких сигналів, що були розпізнані як крок. Цей підхід має спрацювати, оскільки крокує одна людина і кроки досить точно можна локалізувати. З використанням описаних вище методів та моделей можна побудувати систему для розпізнавання кроків. На рис. 5.3 зображено структуру СПР.

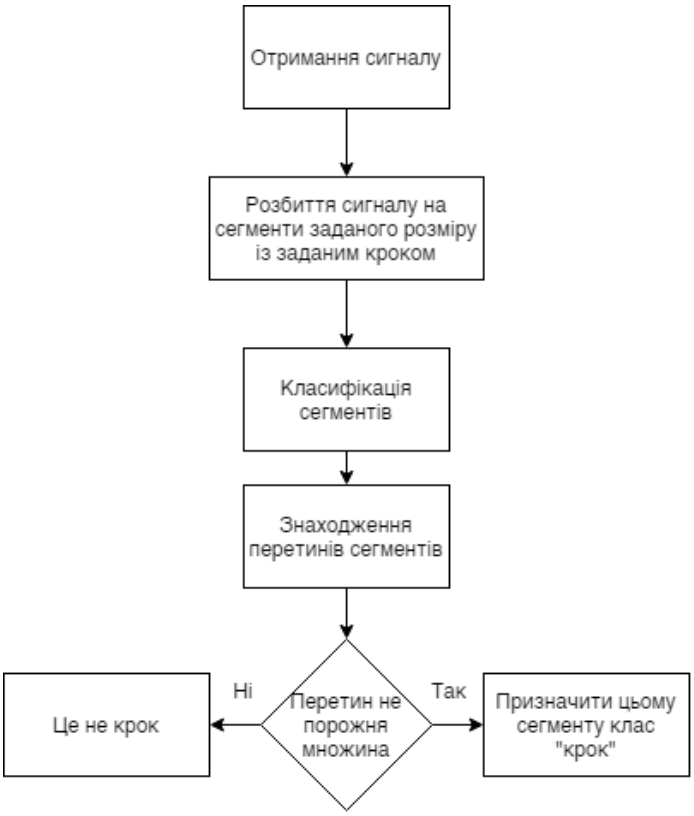


Рисунок 5.3

Як видно зі схеми, алгоритм СПР досить зрозумілий: на початку отримується сигнал (не обов’язково з 1 кроком), після цього сигнал розбивається на сегменти із заданою довжиною вікна і кроком зсуву. Отримані сегменти посилаються у класифікатор, який вказує чи є крок на даному сегменті чи ні. Відбираються всі сегменти із позитивним результатом та відбувається локалізація кроку за допомогою перетину. Після цього усі непорожні перетини позначаються як крок.

3.5 Висновки до розділу

У цьому розділі наведені аргументи щодо вибору Python у ролі інструмента для реалізації методів і моделей описаних у попередньому розділі і були описані бібліотеки, що полегшать розробку. Також була описана реалізація передобробки отриманих даних, розглянуто використані шари та розписана пошарова структура і спосіб навчання нейромереж. Всього навчено два типи мереж: 4 моделі першого типу для вхідної довжини кроку 128, 200, 256, 512, і одну мережу другого типу. Мережі вийшли досить точними – 85-95 %. Усі розробки було об’єднано в одну СПР, що дозволяє класифікувати і локалізувати кроки у сигналі.

РОЗДІЛ 4 ВИКОРИСТАННЯ МОДЕЛЕЙ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

4.1 Попередній аналіз даних

Сигнали записані під час одного з експериментів (рис. 4.1 і рис. 4.2) виглядають наступним чином:

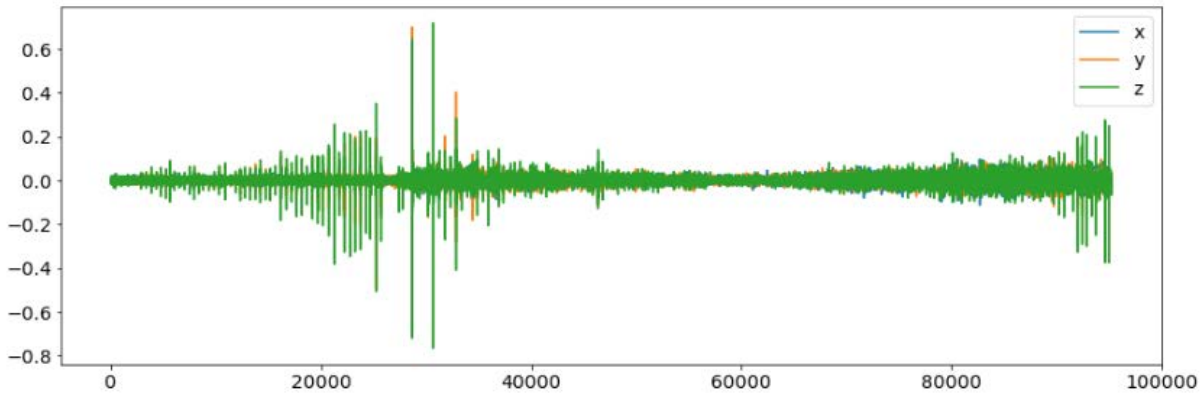


Рисунок 4.1 – Весь запис експерименту

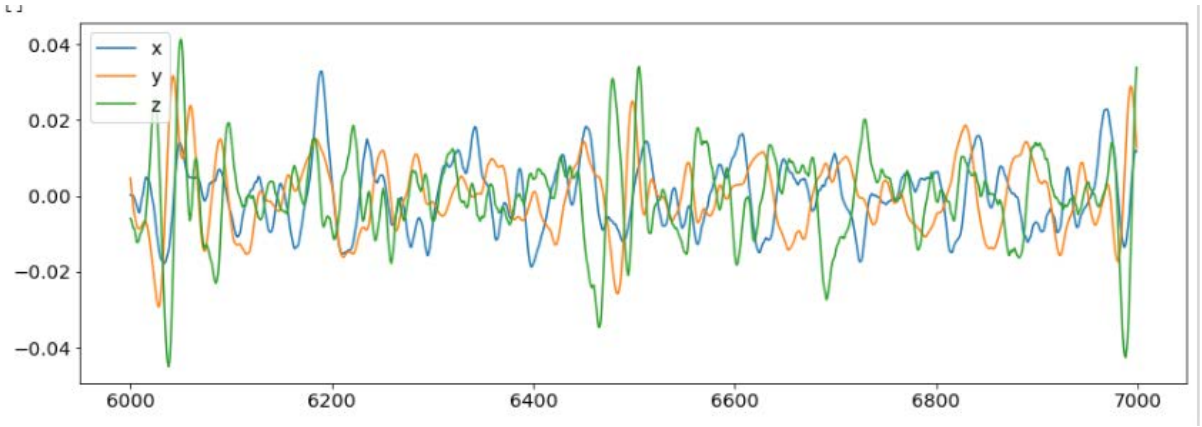


Рисунок 4.2 – Фрагмент запису з кроками

Неозброєним оком видно кроки, які вирізняються своєю амплітудою, тому перед використанням вейвлет перетворення було вирішено спробувати знаходити кроки за допомогою функцій, що знаходять вершини в сигналах.

Функції добре знаходили чітко виражені кроки недалеко від приладу. Проте після детальнішого аналізу було знайдено недоліки такого підходу:

1. Велика залежність від евристичних параметрів, наприклад, таких як мінімальна відстань між вершинами.
2. Навіть на файлі із шумом ці функції знаходили велику кількість помилкових кроків.

Нижче наведені результати роботи фукнції пошуку вершин сигналу для відрізку на якому чітко видно кроки (рис. 4.3) і на даних із шумом середовища (рис. 4.4), а вершини відмічені червоними хрестиками.

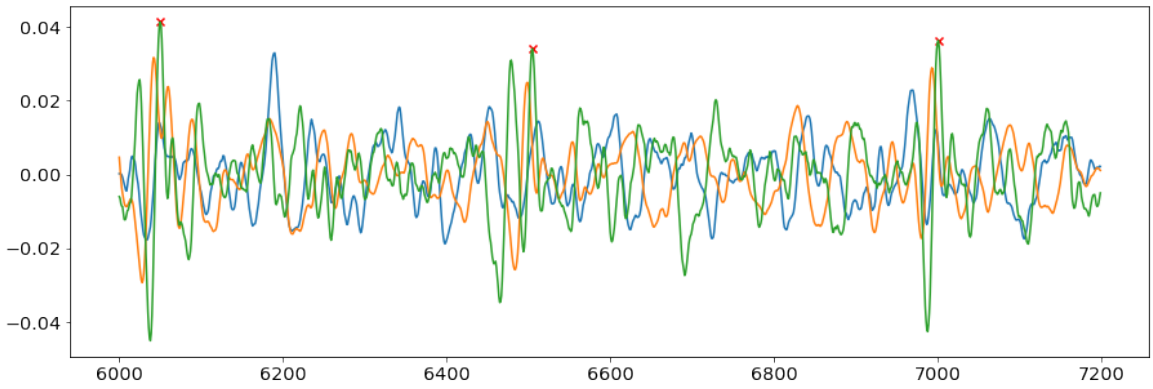


Рисунок 4.2

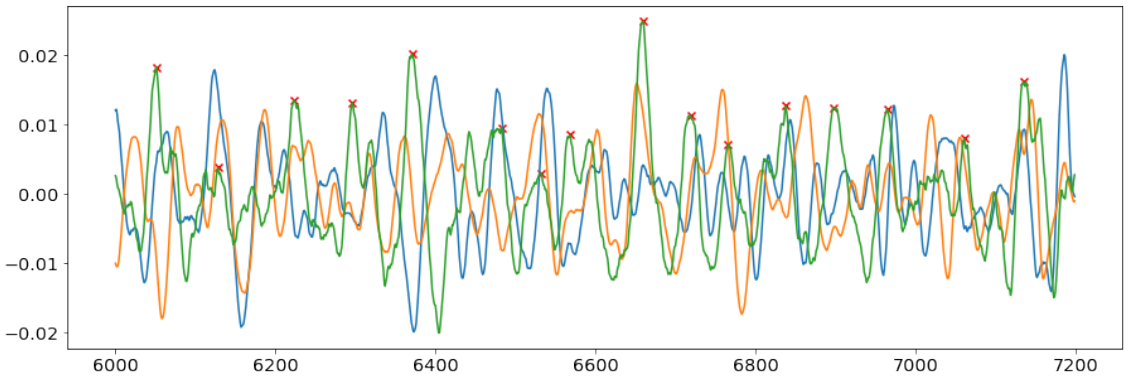


Рисунок 4.2

Результати показують недоліки цього підходу, тому цей варіант було відкинуто.

Наступним етапом був вейвлет-аналіз. Материнським вейвлетом для вейвлет перетворення було обрано вейвлет Морле. Для всіх кроків були візуалізовані коефіцієнти вейвлет перетворення з нормуванням коефіцієнтів по Z складовій та по частоті.

Нижче наведені графіки чітко виражених кроків недалеко від приймача. На рис. 4.3 візуалізовано усі 3 канали для кроку людини, що рухалася вздовж осі Y, а на рис. 4.4 – вздовж осі X відповідно.

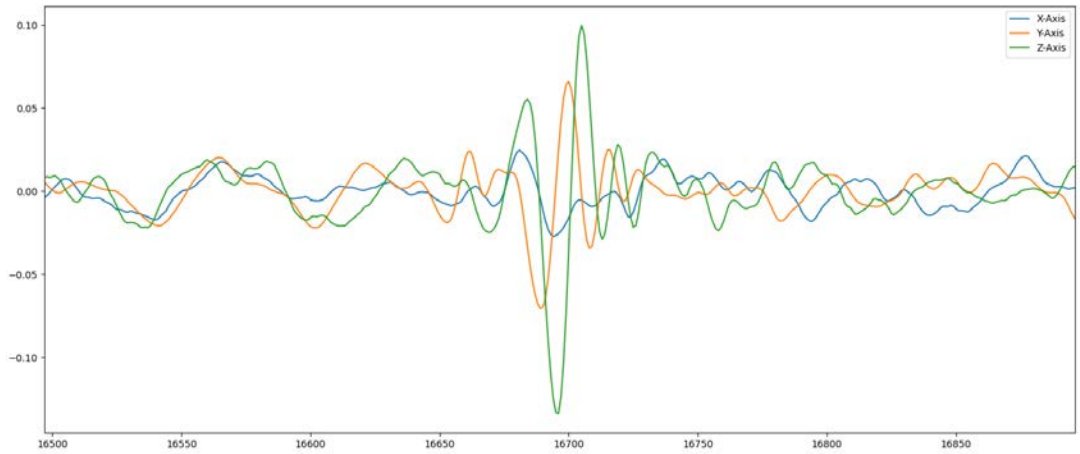


Рисунок 4.3

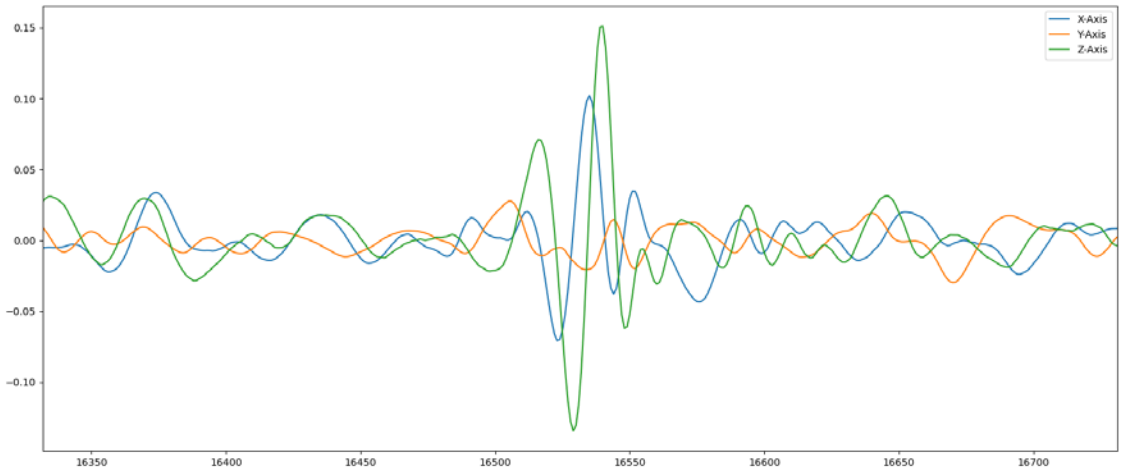


Рисунок 4.4

Було проведено ВП складових сигналу, на рис. 4.5 та рис. 4.5 зображені результати для випадку нормування по Z складовій для кроку людини.

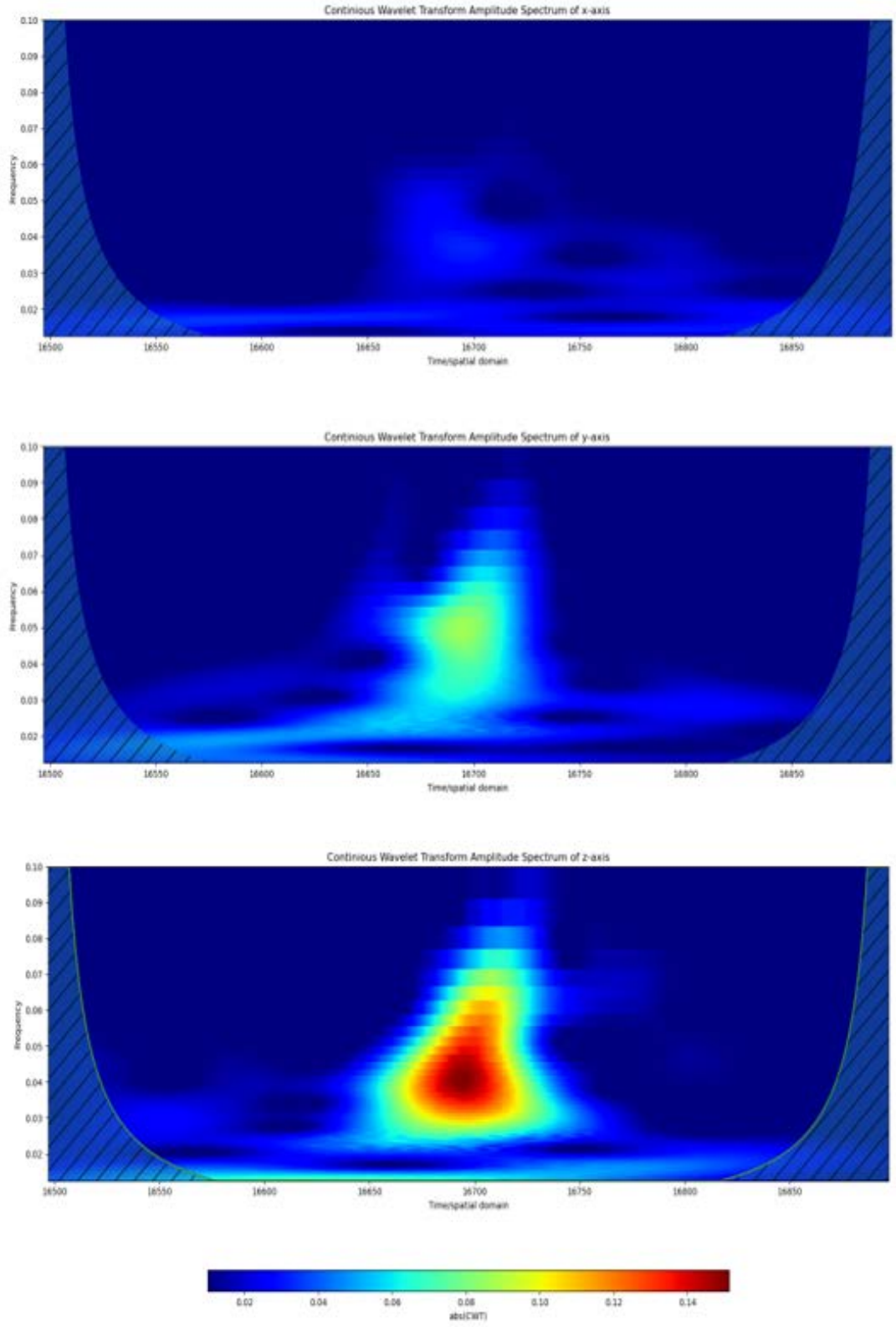


Рисунок 4.2 – ВП для кроку вздовж осі Y.

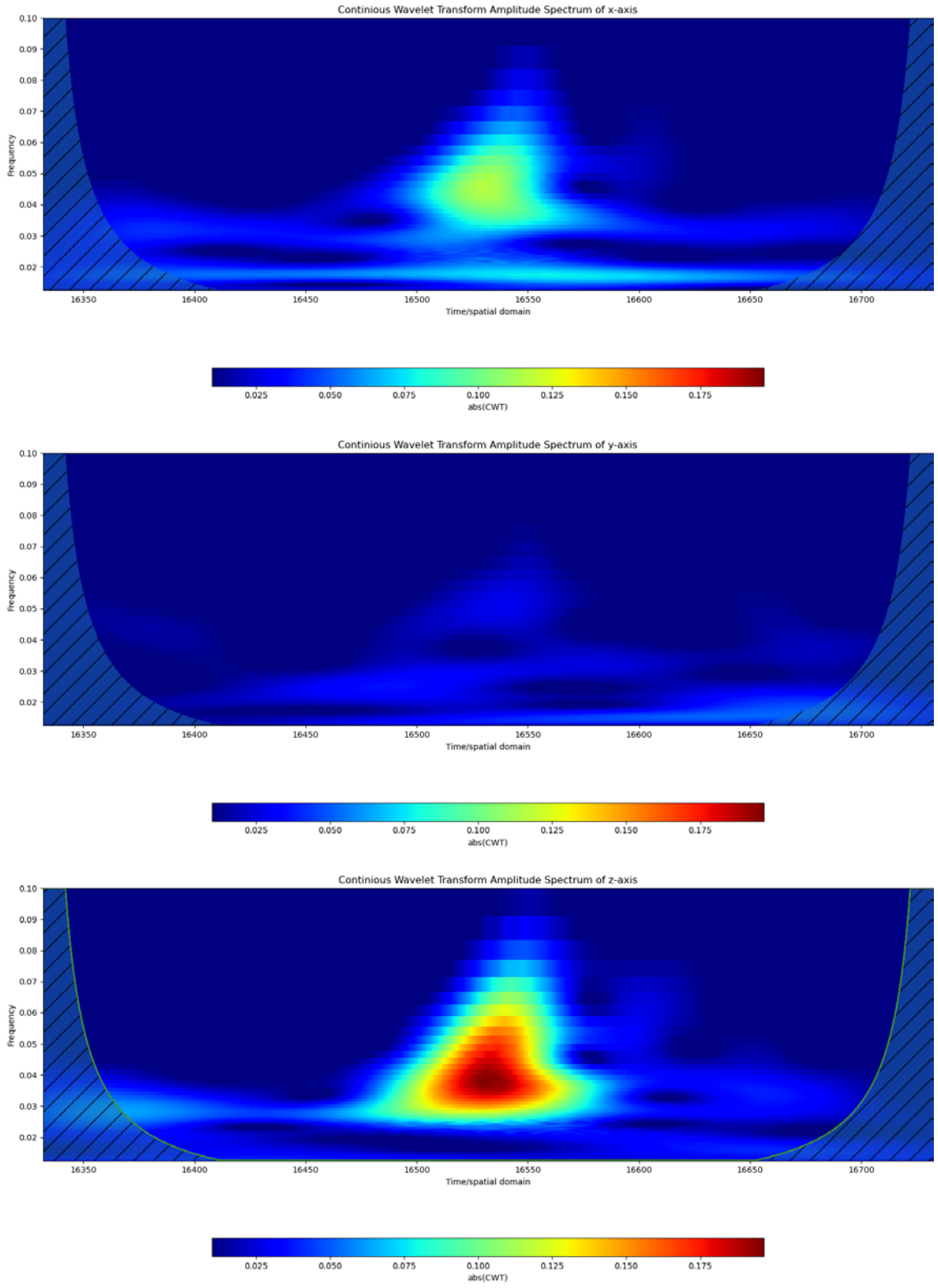


Рисунок 4.3 – ВП для кроку вздовж осі X.

З візуалізації ВП видно, що Z складова має найбільшу енергію. Також видно що енергія іншої складової, яка є перпендикулярною до напрямку руху мінімальна.

Аналогічно на рис. 4.6 та 4.7 зображені результати ВП з нормуванням по частоті.

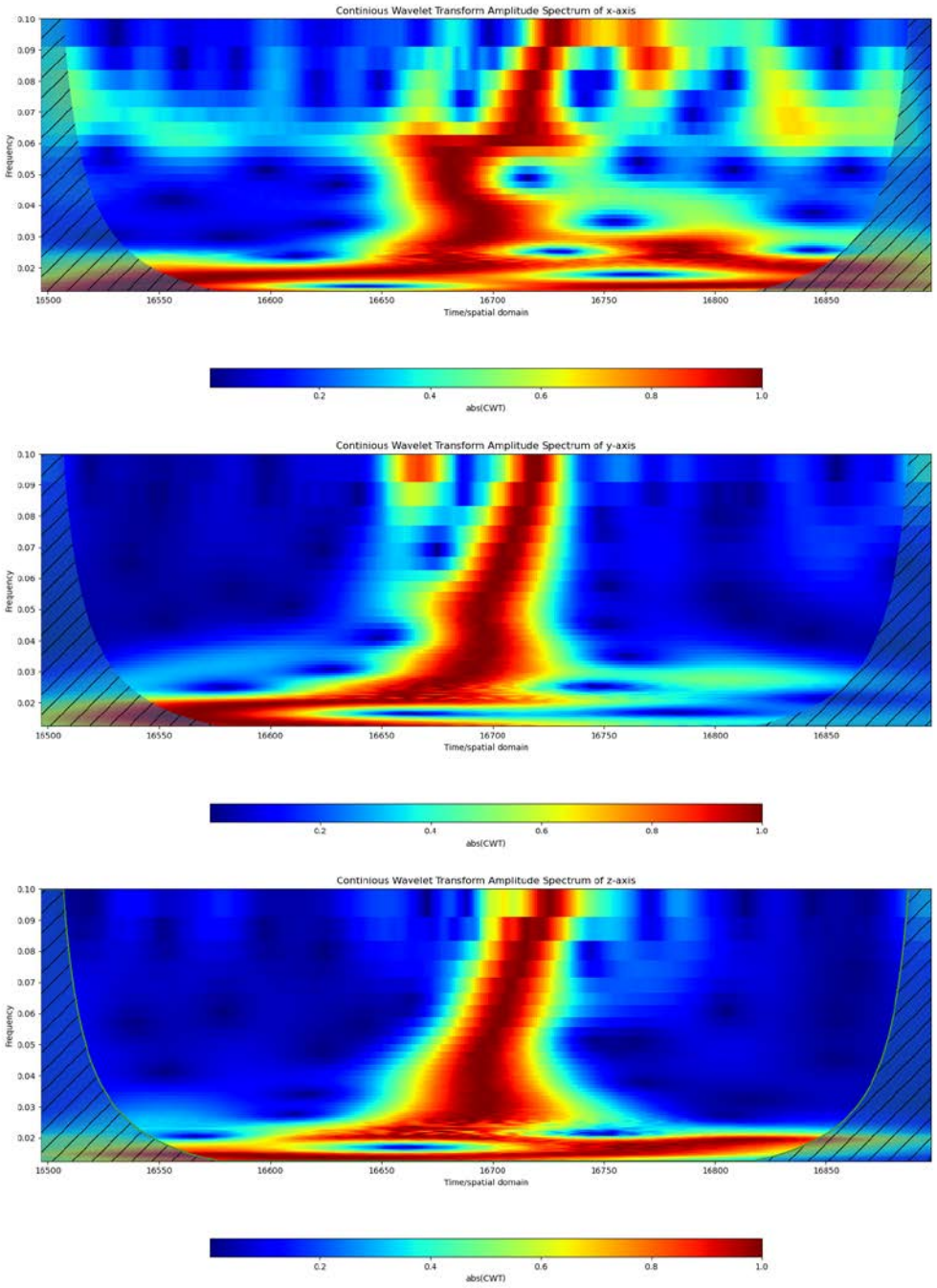


Рисунок 4.4 – ВП для кроку вздовж осі Y.

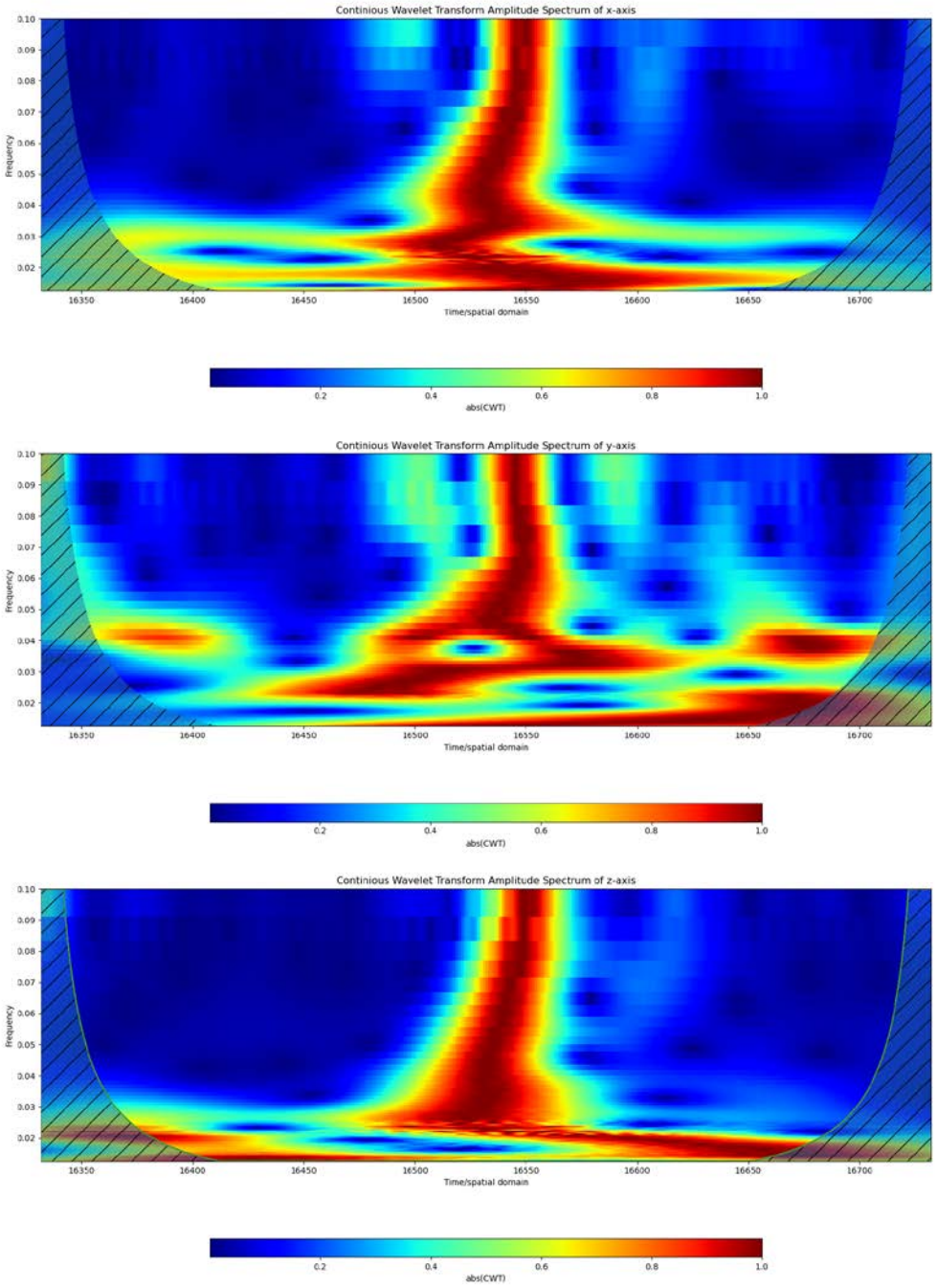


Рисунок 4.5 – ВП для кроку вздовж осі *X*

З результатів ВП із нормуванням по частоті помітний зсув у часі високих та низьких частот. Проте такий ефект спостерігається не в усіх випадках, а тому потрібно провести більше експериментів, щоб сказати, що це характерна ознака.

4.2 Проведення чисельних розрахунків та аналіз результатів

Відповідно до таблиці 4.1 були проведені досліди із моделями першого типу, під час яких вар’ювалися довжина кроку та крок зсуву.

Таблиця 4.1 – Таблиця з обраними довжинами кроку і зсувами.

Довжина кроку	Крок зсуву
128	64
200	64
200	128
256	64
256	128
512	64
512	128
512	256
512	400

Хоча точність навчених моделей була, практично кожна знаходила «хибні» кроки на файлі із шумом у великій кількості. На рис. 4.6 наведено результати роботи моделі з вікном 128 і кроком зсуву 64 на частині даних з експерименту, де чітко видно кроки.

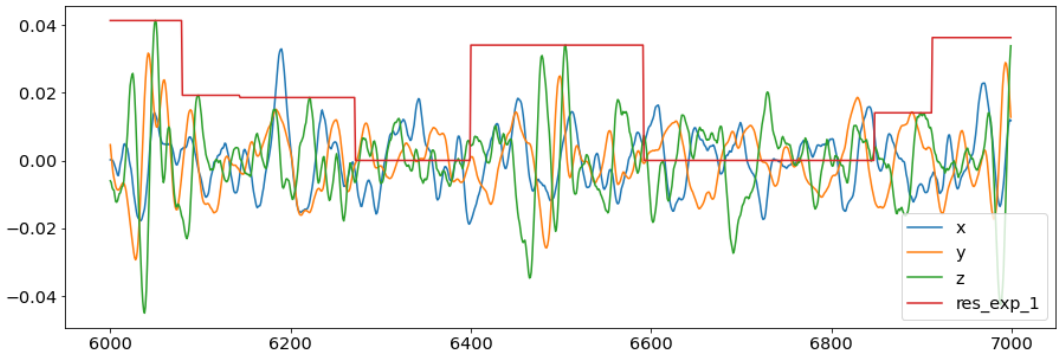


Рисунок 4.6

Зони, де черовна лінія не на рівні 0, розпізнаються мережою як кроки. Видно, що мережа більш-менш правильно їй знаходить. Проте якщо взяти файл із шумом отримуємо хибні результати класифікації. На рис. 4.7 та 4.8 видно, що модель все одно класифікує шум як кроки.

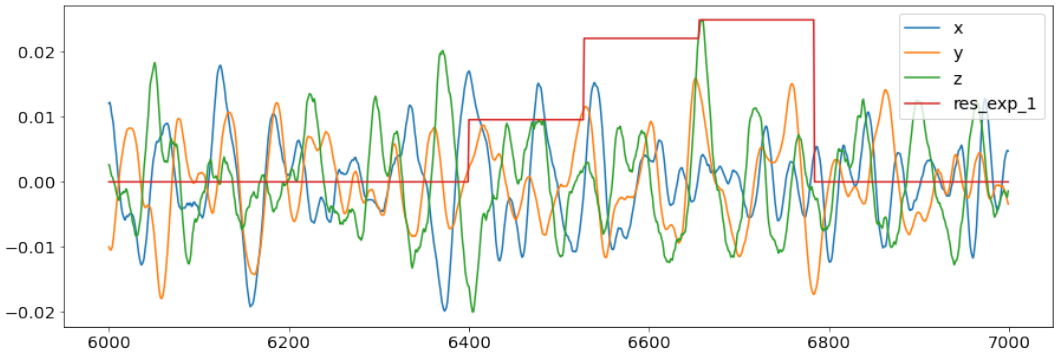


Рисунок 4.7

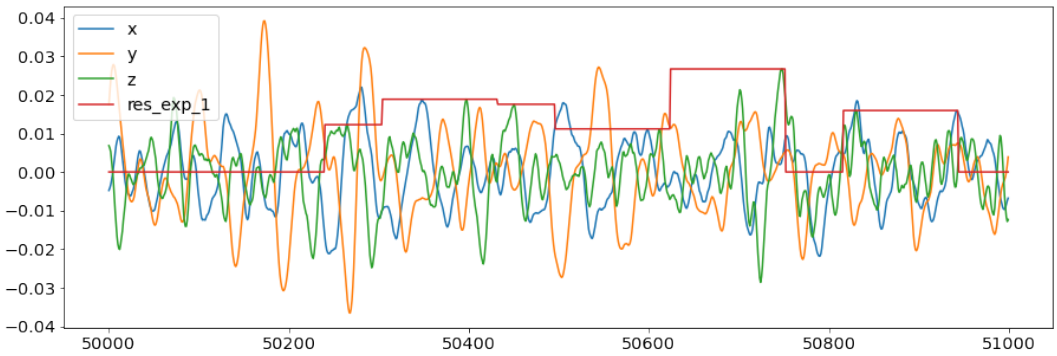


Рисунок 4.8

Тобто скоріш за все модель просто зосереджується на тому сигнали з кроками мають виражені вершини. І тому коли на вхід подається шум мережа шукає перепади амплітуд.

Для моделей з більшим вікном результати для чітких кроків аналогічні, це видно на рис. 4.9.

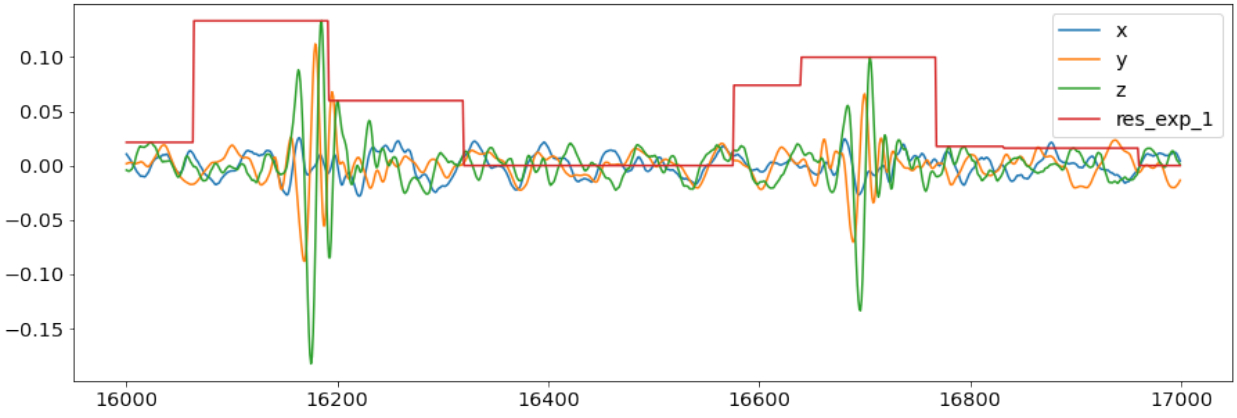


Рисунок 4.9

Проте розпізнаних кроків на даних із шумом середовища стало менше, що видно на рис. 4.10:

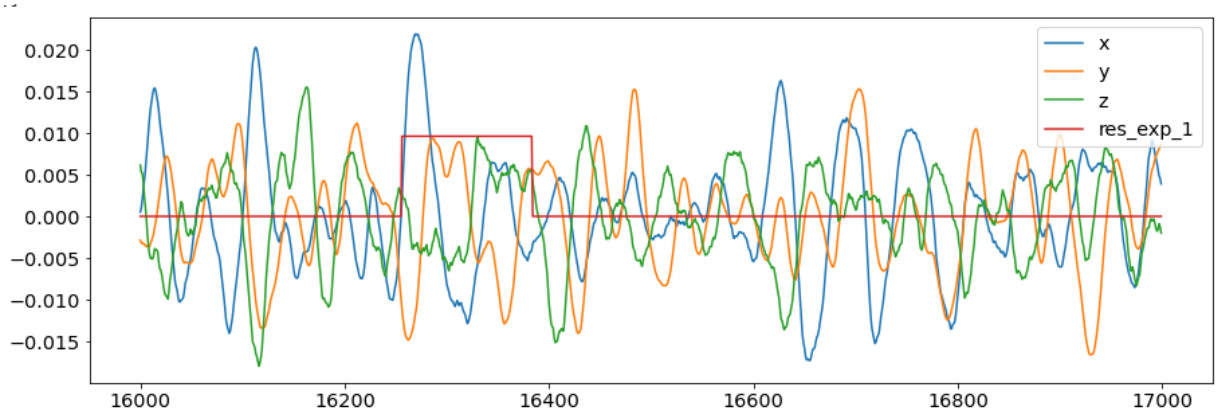


Рисунок 4.10

Також проведені експерименти різними кроками зсуву. Результати для малих зсувів (не більше 1/4 довжини кроку) не показали значних відмінностей від тих, що наведені вище.

Для моделі другого типу маємо наступні результати (рис. 4.11 і 4.12) на тих же даних.

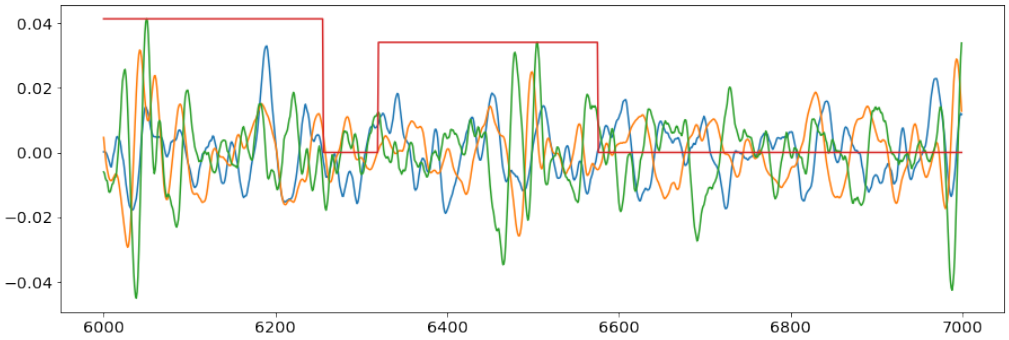


Рисунок 4.11

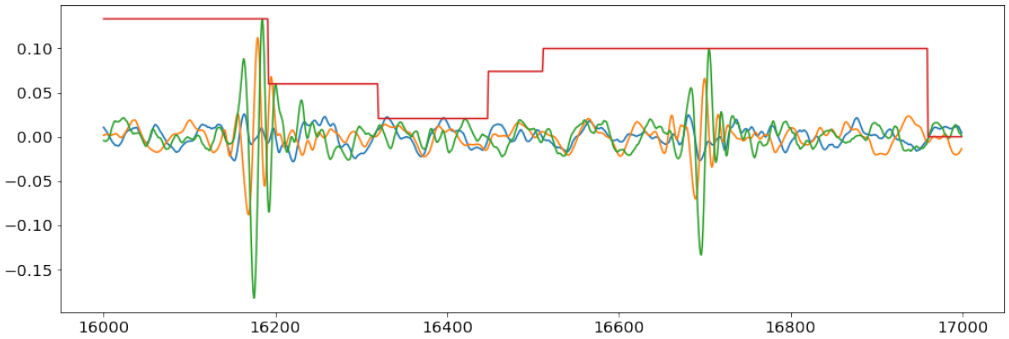


Рисунок 4.12

Видно, що результати практично однакові на частинах даних, де крок чітко видно. Проте відмінністю другої моделі, є те що вона знаходить значно менше кроків на даних з експерименту із шумом середовища, що підтверджують результати зображені на рис. 4.13 та 4.14.

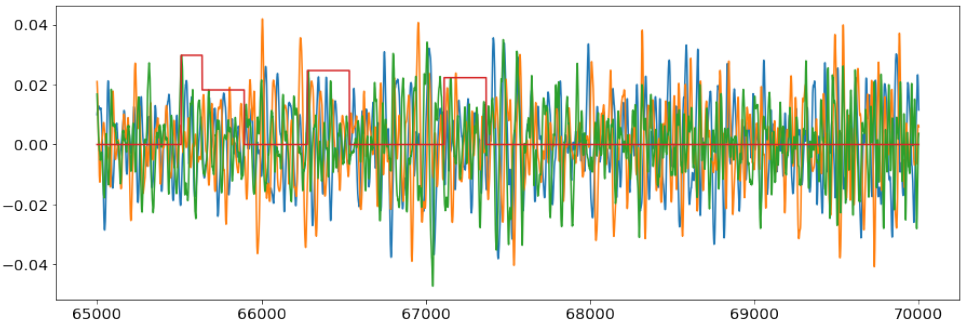


Рисунок 4.13

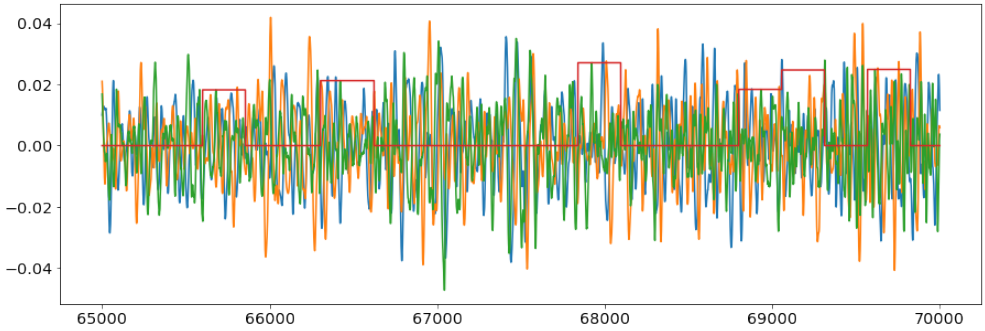


Рисунок 4.14

Якщо застосувати другу модель у СПР, то на всіх даних із шумом середовища серед приблизно 3000 сегментів хибно класифікуються тільки 3 сегменти. Результат роботи СПР на даних із шумом середовища зображено на рис.4.15

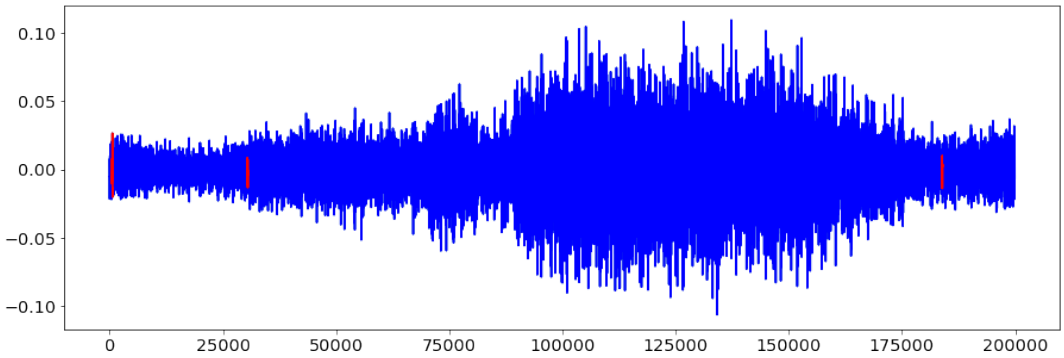


Рисунок 4.15

А на даних із кроками, процес розпізнавання кроків досить точний. Ничже на рис. 4.16 та 4.17 наведені результати роботи СПР.

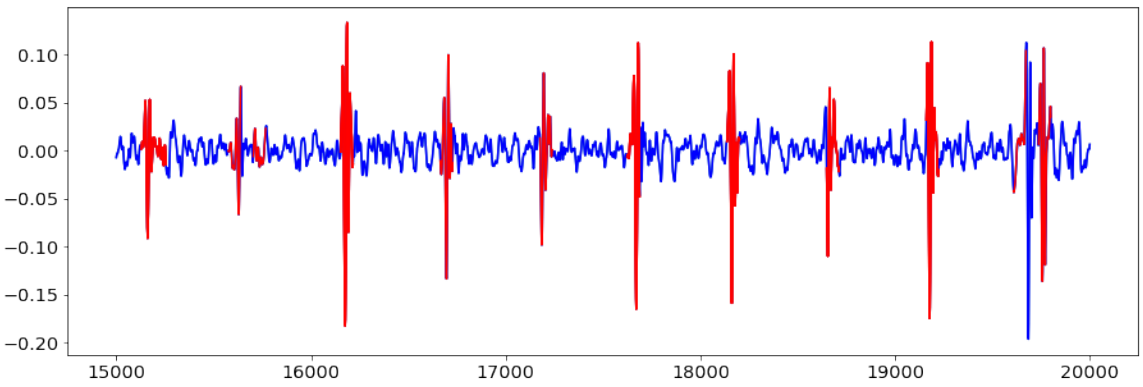


Рисунок 4.16

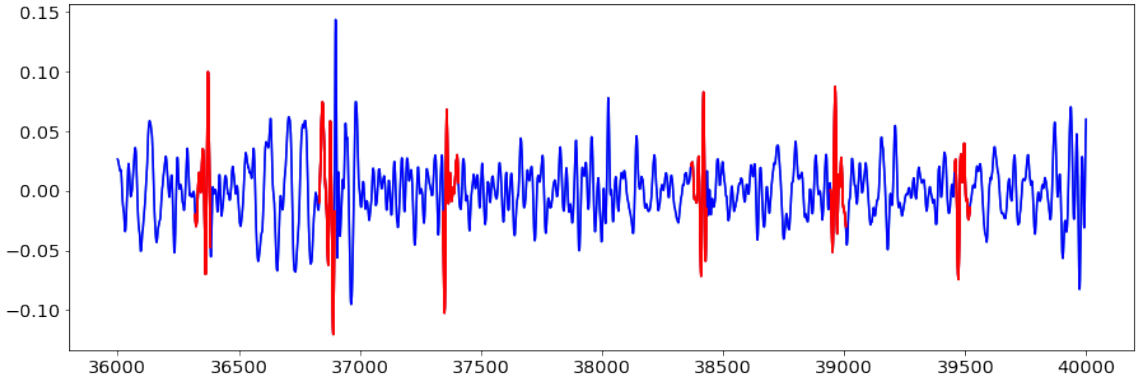


Рисунок 4.17

4.3 Висновки до розділу

Отже, з результатів вейвлет аналізу, прослідковується залежність між напрямком руху і частотно часовими харакатеристиками сигналу, що підтверджує згодадки про поводження хвилі у ґрунті. Також з результатів видно, що найбільше енергії має в собі Z складова, а тому для класифікації кроку є сенс використовувавти у навчанні саме її.

Також можна зробити висновок, що від кроку зсуву точність результатів не залежить. При збільшенні вікна отримується краща точність на даних із шумом, але на даних із кроками точність не зростає.

З результатів роботи другої моделі, було вирішено використати її у СПР. Точність роботи СПР з першого погляду задовільна, проте також з результатів її роботи видно проблему моделі (яку має також і перша) – це зосередження класифікатора на даних з амплідутними перепадами.

РОЗДІЛ 5 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОДУКТУ

5.1. Постановка задачі проектування

Спроектувати програмний продукт для аналізу сейсмоакустичних вимірів віддаленого сенсору, приймаючими елементами якого є геофони. Продукт повинен бути кросплатформенним та мати різні інтерфеси для роботи з даним продуктом. Нижче наведено аналіз різних варіацій реалізації програмного продукту, для вибору найоптимальнішого, з використання функціонально-вартісного аналізу.

5.2. Обґрунтування функцій та параметрів програмного продукту

Виходячи з конкретних цілей, які реалізуються :

F1 – завантаження вимірів:

а) завантаження даних з пам’яті пристрою частинами.

в) потокове завантаження запису

F2 – сегментація голосу:

а) спектральний анализ,

б) вейвлет-перетворення,

в) кореляційний аналіз.

F3 – обробка сегментів

а) нейронна мережа,

б)автоматизована обробказ евристичними параметрами.

F4 – отримання результатів:

а) визначення наявності кроків у сигналі,

б) визначення напрямку кроку, якщо він знайдений.

F5– збереження результатів роботи:

а) запис результатів у сховище та вивід користувачу,

б) лише вивід користувачу.

Виходячи з представлених варіантів будуємо морфологічну карту (рис.5.1).

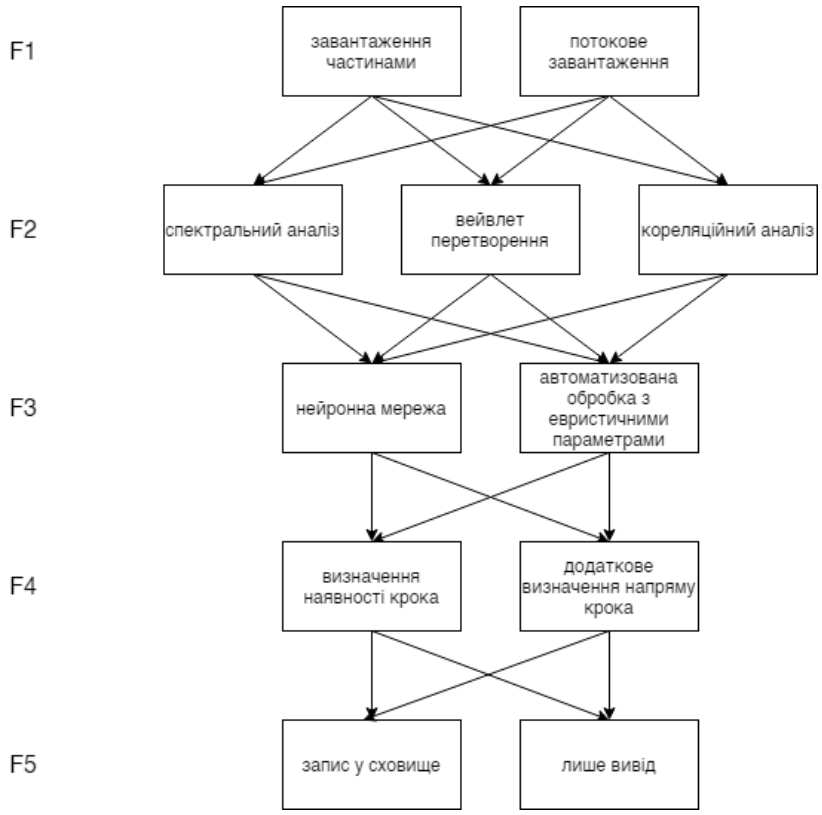


Рис. 5.1 Морфологічна карта.

Спираючись на морфологічну карту була побудована позитивно-негативна матриця (табл. 5.1).

Таблиця 5.1 Позитивно-негативна матриця

Основна функція	**Варіант реалізації**	**Переваги**	**Недоліки**
F1	А	Легкість реалізації	Неможливе швидке прийняття рішень
	Б	Прийняття рішень відбувається в режимі реального часу	Потрібна налагоджена процедура передачі даних
F2	А	Легкість реалізації	Чутливий до шумів
	Б	Дає задовільні результати, характеризує особливості сигналу в часі	Складна реалізація
	В	Легкість реалізації	Велика кількість помилок
F3	А	Досить точні результати	Складна реалізація, потреба в навчанні
	Б	Легкість реалізації	Низька точність результатів
F4	А	Легкість реалізації	Мала кількість інформації
	Б	Надання додаткової інформації	Складність реалізації

Продовження таблиці 5.1

F5	А	Можливість агрегування даних для подальшого використання	Потрібне сховище даних, складність реалізації
	Б	Легкість реалізації	Результати попереднії вимірів недоступні

Для характеристики прототипу програмного додатку використовуємо параметри X1 – X4. На основі даних, що представлені у літературі, визначаємо мінімальні, середні отримуванні та максимально допустимі значення(табл. 5.2).

Таблиця 5.2 Система параметрів додатку

Найменування параметру	Позначення параметру	Значення параметру		
		Мінімальне	Середнє	Максимальне
Час розробки, людина*год	X1	360	600	800
Час роботимоделі, мс	X2	50	300	800
Рекомендована частота процесору, ГГц	X3	1,9	2,05	3,2
Рекомендована швидкість запису на диск, МБ/с	X4	1	32	64

Оцінка кожного з параметрів по 10-ти бальній шкалі наведена окремо для кожного параметру на рис. 5.1-5.5.

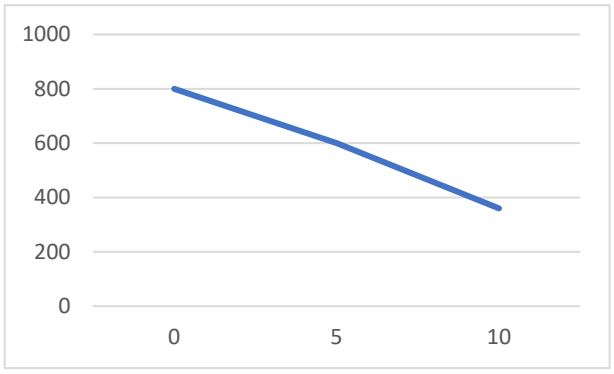


Рис. 5.2 Значення параметру X1

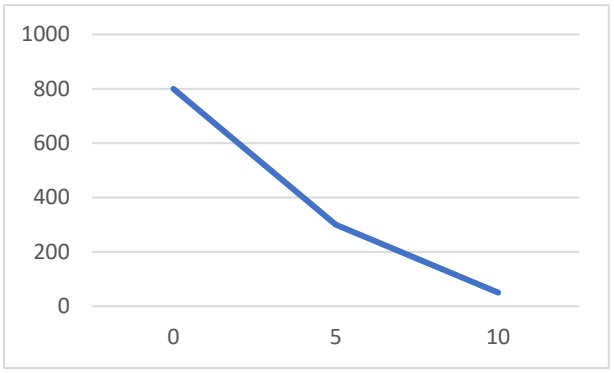


Рис. 5.3 Значення параметру X2

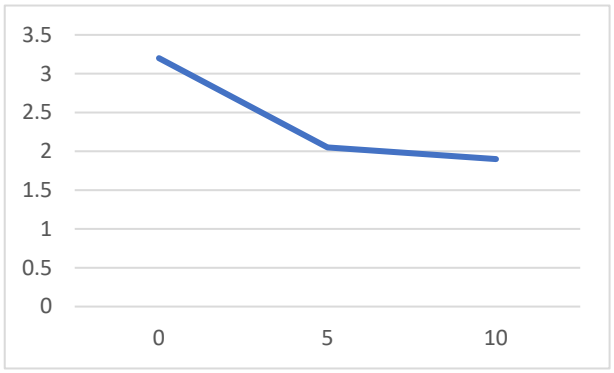


Рис. 5.4 Значення параметру X3

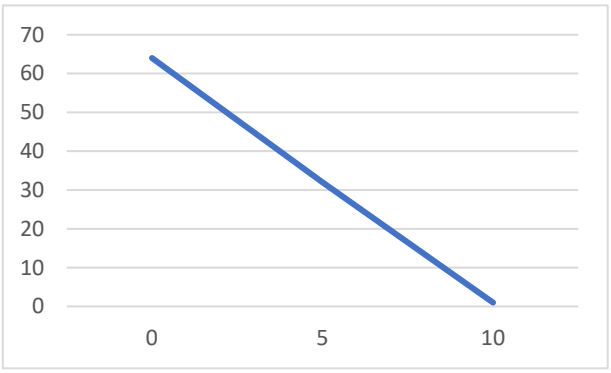


Рис. 5.5 Значення параметру X4

Вагомість параметрів оцінюється за допомогою методів попарного зрівняння. Ранги варіюються від 1 до 5. Результати наведені нижче в табл. 5.3 та в табл. 5.4.

Таблиця 5.3 Результат оцінки параметрів

Параметр	Ранг параметру по оцінці експерта							Сума рангів, R_i	Відхилення Δ_i	Квадрат відхилення, $(\Delta_i)^2$
	1	2	3	4	5	6	7			
X1	2	2	1	2	1	2	2	12	-5.5	30.25
X2	1	1	2	1	2	1	1	9	-8.5	72.25
X3	3	4	3	3	4	3	3	23	5.5	30.25
X4	4	3	4	4	3	4	4	26	8.5	72.25
Разом	10	10	10	10	10	10	10	70	0	205

Коефіцієнт узгодженості дорівнює:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 * 205}{7^2(4^3 - 4)} \approx 0.83 > W_{\text{норм}} = 0.67$$

Оскільки коефіцієнт більше нормативного значення, результати вважають достовірними. Експерти виставляли ранги від 1 до 4, де 1 це найбільш важливий на думку експерта параметр, а 4 – найменш. В табл. 5.4 наведене попарне зрівняння параметрів.

Таблиця 5.4

Параметри	**Експерти**							**Кінцева оцінка**	**Числове значення**
	1	**2**	**3**	**4**	**5**	**6**	**7**		
X1 та X2	<	<	>	<	>	<	<	<	0.5
X1 та X3	>	>	>	>	>	>	>	>	1.5
X1 та X4	>	>	>	>	>	>	>	>	1.5
X2 та X3	>	>	>	>	>	>	>	>	1.5
X2 та X4	>	>	>	>	>	>	>	>	1.5
X3 та X4	>	<	>	>	<	>	>	>	1.5

Використовуючи результати попарного порівняння обчислюється вагомість кожного з критеріївв табл. 5.5.

Таблиця 5.5 Розрахуноквагомостіпараметрів

Параметри	Параметрихj				Першийкрок		Другийкрок	
	X1	X2	X3	X5	b_i	$К_{вi}$	b_i	$К_{вi}$
X1	1	0.5	1.5	1.5	4.5	0.281	16.25	0.275
X2	1.5	1	1.5	1.5	5.5	0.344	21.25	0.36
X3	0.5	0.5	1	1.5	3.5	0.219	12.25	0.208
X4	0.5	0.5	0.5	1	2.5	0.156	9.25	0.157
Всього					16	1	59	1

Після другого кроку бачимо що зміна в відхиленні не перевищує 5%, тому можемо закінчити.

Враховуючи дані з порівнянь варіантів реалізацій функцій залишаємо наступні варіанти:

1) F1(Б) -F2(Б) -F3(А) - F4(А) - F5(А)

2) F1(Б) -F2(Б) -F3(Б) - F4(А) - F5(А)

Наступним кроком вирахувати коефіцієнт якості для відібраних варіантів реалізації. Для зручності обрахунків, запишемо характеристики обох ситуацій в табл. 5.6.

Таблиця 5.6

Основна функція	Варіант реалізації	Параметр	Абсолютне значення параметру	Бальна оцінка параметру	Коефіцієнт вагомості параметру	Коефіцієнт якості
F1	Б	X1	500	5.5	0.275	1.5125
F2	Б	X2	200	7	0.36	2.52
		X3	2.1	4	0.208	0.832
F3	А	X2	400	4	0.36	1.44
		X3	2.1	4	0.208	0.832
	Б	X2	700	2	0.36	0.72
		X3	2.4	3	0.208	0.624
F4	А	X4	16	7	0.157	1.099
F5	А	X4	40	3	0.157	0.471

Обрахуємо коефіцієнти якості кожного з варіантів розробки:

1) $K_{я1} = 1.5125 + 2.52 + 0.832 + 1.44 + 0.832 + 1.099 + 0.471 = 8.7065$

2) $K_{я2} = 1.5125 + 2.52 + 0.832 + 0.72 + 0.624 + 1.099 + 0.471 = 7.7785$

Оскільки варіант 1 має найбільший коефіцієнт якості, він є найкращим.

5.3 Економічний аналіз варіантів розробки

Для оцінки трудомісткості розробки спочатку проведемо розрахунок трудомісткості. Усі варіанти маютьнаступні основні завдання:

1) Сегментація сигналу
2) Побудова моделі
    a. Нейронної мережі
    b. Евристичного алгоритму
3) Вивід результатів на екран

Для першого завдання (ступінь новизни А, група складності алгоритму 1):

$$T_Р = 40,\ К_П = 1.28,\ К_{СК} = 1,\ К_{СТ.М} = 1$$

$$T_1^1 = 40 * 1.28 = 65.57 \text{ людино-днів}$$

Для другого завдання(при реалізації варіанту А алгоритм групи складності 3 , ступінь новизни В, вид використаної інформації — НДІ)

$T_р = 190$ людино-днів, $К_П = 1.7$, $К_{СК} = 1$, $К_{СТ} = 0.8$.

$T_2 = 190*1.7*0.8 = 258.4$ людино-днів.

Для другого завдання (за реалізації варіанту Б, алгоритм групи складності 1, ступінь новизни А, вид використаної інформації — НДІ)

$T_Р = 200$, $К_П = 1.26$ для НДІ, $К_{СК} = 1$, $К_{СТ.М} = 1.5$.

$T_1^2 = 200 * 1.26 * 1.5 = 378$ людино-днів

Для третього завдання(ступінь новизни Б, група складності алгоритму 3)

$T_р = 20$ людино-днів, $К_П = 0.7$, $К_{СТ} = 0.8$:

$T_З = 20*0.6*0.8 = 9.6$ людино-днів

Трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми

$T_I = (65.57 + 258.4 + 9.6) \cdot 8 = 2668.56$ людино-годин;

$T_{II} = (65.57 + 378 + 9.6) \cdot 8 = 3{,}625.36$людино-годин;

Найбільш високу трудомісткість має варіант І.

В розробці бере участь один програміст з окладом 15000 грн та один математик з окладом 19000.

Зарплата за годину:

$$С = \frac{34000}{2*22*8} = 96.590\text{грн.}$$

Заробітна плата для кожного з варіантів реалізації

$$С_{ЗП1} = 96.590 * 2668.56 = 257756.21$$

$$С_{ЗП2} = 96.590 * 3{,}625.36 = 350173.52$$

Відрахування ЄСВ складають 22%, або для кожного варіанту:

$$С_{від1} = 0.22 * 257756.21 = 56706.37$$

$$С_{від2} = 0.22 * 253823.07 = 77038.17$$

Обчислимо витрати на оплату однієї машино-години. Оскільки ЕОМ обслуговує один програміста з окладом 15000 грн., з коефіцієнтом зайнятості 0.4, то для однієї машини маємо:

$С_Г = 12*М*К_З = 12 * 15000*0.4 =72000$ грн.

З урахуванням додаткової заробітньої плати:

$С_{ЗП} =С_Г*(1 + К_З)= 72000*(1+0.4) = 100800$ грн.

Відрахування на соціальний внесок:

$С_{ВІД} = С_{ЗП} * 0.22 = 100800* 0.22 = 22176$грн.

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 90000 грн.

$С_А = К_{ТМ} * К_А*Ц_{ПР} = 1.15*0.25*90000 = 25875$грн.,

Витрати на ремонт та профілактику:

$С_Р = К_{ТМ}*Ц_{ПР}*К_Р = 1.15 * 90000 *0.05 = 5175$грн.

Ефективний годинний фонд часу ПК:

$Т_{ЕФ} = (Д_К — Д_В — Д_С — Д_Р) * t_З * К_В = (365 - 112 -8 -16)* 8 * 0.9 =$

$= 1648.8$ годин

Витрати на оплату електроенергії (з урахуванням $П_ДВ$):

$$С_{ЕЛ} = Т_{ЕФ} * N_С*К_З*Ц_{ЕН} = 1648.8 *0.28 * 0.9* 1.75 = 727.12 грн.$$

Накладні витрати

$$С_Н = Ц_{ПР} * 0.67 = 90000 *0.67 = 60300грн.$$

Річні експлуатаційні втрати:

$$С_{ЕКС} = 100800 + 22176 + 25875+ 5175+ 727.12+ 60300= 215053.12грн.$$

Собівартість однієї машино-години ЕОМ дорівнює:

$$С_{М-Г} = С_{ЕКС} / Т_{ЕФ} = 215053.12/ 1648.8 = 130.43грн./год.$$

Витрати на оплату машинного часу:

$$С_{М1} = 130.43 * 2668.56 = 348060.28 грн$$

$$С_{М2} = 130.43 * 3{,}625.36 = 472855.7 грн$$

Накладні витрати складають 67% від заробітньої плати:

$$С_{Н1} = 348060.28 * 0.67 = 233200.39 грн$$

$$С_{Н2} = 472855.70 * 0.67 = 316813.32 грн$$

Вартість розробки програмного продукту:

$$С_{ПП1} = 257756.21 + 56706.37 + 348060.28 + 233200.39 = \\ = 895723.25 грн$$

$$С_{ПП2} = 350173.52 + 77038.17 + 472855.7 + 316813.32 = \\ = 1216880.716 грн$$

Розрахуємо коефіцієнт техніко-економічного рівня:

$$К_{ТЕР1} = 8.7065/ 895723.25= 9.72 * 10^{-6};$$

$$К_{ТЕР2} = 7.7785/ 1216880.716= 6.39 * 10^{-6};$$

5.4 Висновки до розділу

Отже з результатів функціонально-вартісного аналізу в цьому розділі, можна зробити висновок, що найбільш ефективним є перший варіант з коефіцієнтом техніко-економічного рівня $9.72 * 10^{-6}$. Тобто було прийняте рішення про розробку програмного продукту з використанням нейронної мережі.

ВИСНОВКИ

Поширення сейсмоакустичні хвиль на невеликій глибині на сьогоднішній день не має математичної моделі, тому задача розпізнавання кроків є одним із перших етапів дослідження даних хвиль. Можливість розпізнавати кроки має і практичну цінність, оскільки на основі нього можна будувати охоронні або моніторингові системи.

У даному ДП було проведене дослідження сейсмоакустичних хвиль на основі даних вимірів отриманих під час експериментів, протягом яких записувалися сигнали кроків людини за допомогою віддаленого прийомного модуля в основі якого лежать геофони. З урахуванням особливостей даних виконано огляд відомих методів дослідження та попередньої обробки сейсмоакустичних сигналів.

Задачу класифікації запропоновано розв’язувати за допомогою нейронних мереж. За допомогою обраних ефективних інструментів було побудовано і навчено на вручну розміченій навчальній вибірці декілька моделей класифікаторів та реалізовано попередню обробку з використанням вейвлет перетворення для дослідження характеристик сигналів.

За допомогою попередньої обробки було визначено декілька закономірностей поведінки сигналів, проте деякі закономірності прослідковувалися не всюди, тому необхідна більша кількість експериментів, а також більше розмічених даних.

На тестовій вибірці було досягнуто точності в межах 85-95% для задачі бінарної класифікації (клас 0 – не крок, клас 1 – крок). Модель зі згортковими та рекурентими шарами виявилася точнішою. На основі саме цієї моделі побудовано СПР яка має досить високу точність.

Проте у навчених моделей є явний недолік – це заучування того факту, що крок має амплітудний перепад. Це призводить до не завжди коректного розпізнавання. В подальшому результати розпізнавання можна покращити за допомогою навчання більш складних моделей, а також детальнішого дослідження характеристик хвиль (наприклад дослідити енергію кожної складової сигналу), на основі яких можна будувати точніші СПР.

Подальшим розвитком даних досліджень є не тільки бінарна класифікація, а також визначення напряму руху об'єкта, знаходження положення об'єкта, а також визначення його характеристик, наприклад, маси.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Яковлев А.Н. Введение в вейвлет-преобразования. Новосибирск: Изд-во НГТУ, 2003. 104 с.

2. Хайкин С. Нейронные сети: полный курс, 2-е изд. Москва: Издательский дом "Вильямс", 2016. 1104 с.

3. Краснопрошин В.В., Головко В.А. Нейросетевые технологии обработки данных : учеб. пособие. Минск : БГУ, 2017. 263 с. URL: http://elib.bsu.by/handle/123456789/193558

4. GLOROT X., BORDES A., BENGIO Y. Deep sparse rectifier neural networks. Proceedings of the fourteenth international conference on artificial intelligence and statistics. 2011. № 15. С.315-323.

5. Нестеров Ю.Е. Методы выпуклой оптимизации. Москва: Издательство МЦНМО, 2010. 281 с. URL: http://premolab.ru/pub_files/pub5/MnexoB89z7.pdf

6. DUCHI J., HAZAN E., SINGER Y. Adaptive subgradient methods for online learning and stochastic optimization. Journal of machine learning research. 2011. № 12. С.2121-2159.

7. Ruder S. An overview of gradient descent optimization algorithms. CoRR. 2016. № 1609.04747. С.1-33 URL:

http://arxiv.org/abs/1609.04747

8. ZEILER M. Adadelta: an adaptive learning rate method. CoRR. 2012. № 1212.5701.

9. Andén, J., and S. Mallat. Deep Scattering Spectrum. IEEE Transactions on Signal Processing. 2014. № 16. С. 4114–4128.

10. Andén, J. Scatnet. Computer Software. 2014, №2. URL:

http://www.di.ens.fr/data/software/scatnet/

11. Сайт бібліотеки Kymatio. URL:

https://www.kymat.io/

12. MADIRAJU, Naveen Sai, et al. Deep temporal clustering: Fully unsupervised learning of time-domain features. arXiv preprint arXiv:1802.01059, 2018.

13. Документація Keras. URL:

https://keras.io/api/layers/recurrent_layers/

14. S. Kiranyaz, T. Ince and M. Gabbouj, "Real-Time Patient-Specific ECG Classification by 1-D Convolutional Neural Networks," in IEEE Transactions on Biomedical Engineering, vol. 63, no. 3, pp. 664-675, March 2016, doi: 10.1109/TBME.2015.2468589.

15. PARK, Hyung O.; DIBAZAR, Alireza A.; BERGER, Theodore W. Cadence analysis of temporal gait patterns for seismic discrimination between human and quadruped footsteps. In: 2009 IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE, 2009. p. 1749-1752.

16. SHARMA, Navdeep, et al. Detection of Various Vehicles Using Wireless Seismic Sensor Network. In: 2012 International Conference on Advances in Mobile Network, Communication and Its Applications. IEEE, 2012. p. 149-155.

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

Реалізація передобробки:

```
import os
import pandas as pd
import numpy as np
from scipy.signal import find_peaks
import scaleogram as scg
import pywt
from utils import build_plot
from tqdm import tqdm
import gc
scg.set_default_wavelet('cmor1-1.5')

DATA_PATH = os.getcwd()+'\\data\\'
SOURCE_PATH = os.getcwd()+'\\data\\source\\'
IMAGES_PATH = os.getcwd()+'\\data\\images\\'


labeled_steps = pd.read_csv(DATA_PATH +
"labeled_steps.csv",delimiter=';',encoding='window
s-1251')
files_list = labeled_steps['source_file'].unique() +
'.csv'
margin = 200
sep = '\\'
normalizations = ['freq','chan']
for filename in tqdm(files_list, desc = f"Filename
"):

    #---------------------------------------------
    file_path = SOURCE_PATH + filename
    data = pd.read_csv(file_path, delimiter=';',
names=['x','y','fake','z'])
    true_lables = ['x','y','z']
    data = data[true_lables]

    #---------------------------------------------
    steps = labeled_steps[labeled_steps['source_file']
== filename[:-4]]
    for _, row in tqdm(steps.iterrows(), desc="Steps
",total=steps.shape[0]):
        center = row["step_center"]
        step = data.iloc[center-margin:center+margin]
        record_name = row["source_file"]
        direction = row["direction"]
        step_num = row["step_num"]
        note = row["note"]
        if pd.isnull(note):
            note = ''
        image_name =
f'{record_name}_{step_num}_{direction}_{note}'
        for normalization in normalizations:
            image_path = IMAGES_PATH +
record_name + sep + normalization + sep
            if not os.path.exists(image_path):
                os.makedirs(image_path)
            build_plot(step,
image_path+image_name,normalization)
```

Навчання моделей, реалізація СПР і тд:

```
# -*- coding: utf-8 -*-

!pip install kymatio

import kymatio
from kymatio.keras import Scattering1D

import os


import pandas as pd
import numpy as np
import tensorflow as tf
import more_itertools

import matplotlib.pyplot as plt
import random
from tqdm.notebook import tqdm

from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Flatten,
Dense
from tensorflow.keras.optimizers import RMSprop

from sklearn.model_selection import
train_test_split
import matplotlib.pylab as pylab
pd.options.mode.chained_assignment = None

params = {'legend.fontsize': 'x-large',
          'figure.figsize': (15, 5),
         'axes.labelsize': 'x-large',
         'axes.titlesize':'xx-large',
         'xtick.labelsize':'x-large',
         'ytick.labelsize':'x-large'}
pylab.rcParams.update(params)

"""# Load and Preprocess
---
Mount drive if needed
"""

from google.colab import drive
drive.mount('/content/drive')

"""here is some constants"""

SRCFILES_PATH = "/content/drive/My
Drive/Diploma/experiments"
src_ext = '.csv'
```

```
path_to_steps = "/content/drive/My
Drive/Diploma/labeled_steps.csv"
steps_data =
pd.read_csv(path_to_steps,encoding='windows-
1251', delimiter=';')

"""Let`s see what we have"""

steps_data

def get_src_file(file_path):
    """
    There are 4 channels in given src files:

    x, y, z and "fake" channel

    this function takes only x,y,z channels
    """
    data = pd.read_csv(file_path, delimiter=';',
names=['x','y','fake','z'])
    true_lables = ['x','y','z']
    filtered_file = data[true_lables]

    return filtered_file

experiments_configs = {
    'exp_1':{
        'margin':64,'window':128,'step':64},
    'exp_2':{
        'margin':64,'window':512,'step':64},

    'exp_3':{
        'margin':128,'window':256,'step':64},
    'exp_4':{
        'margin':128,'window':512,'step':64},

    'exp_5':{
        'margin':256,'window':512,'step':128},
    'exp_6':{
        'margin':256,'window':512,'step':64},



    'exp_7':{
        'margin':100,'window':256,'step':64},
    'exp_8':{
        'margin':100,'window':512,'step':64},
    'exp_9':{
        'margin':100,'window':256,'step':128},
    'exp_10':{
        'margin':100,'window':512,'step':128},
    'exp_11':{
        'margin':100,'window':512,'step':256},
    'exp_12':{
        'margin':100,'window':512,'step':400},



    'exp_13':{
        'margin':150,'window':512,'step':64},

    'exp_14':{
        'margin':150,'window':512,'step':128},

    'exp_15':{
        'margin':150,'window':512,'step':256},
    'exp_16':{
        'margin':150,'window':512,'step':400},

    'exp_17':{
        'margin':200,'window':512,'step':64},

    'exp_18':{
        'margin':200,'window':512,'step':128},
    'exp_19':{
        'margin':200,'window':512,'step':256},
    'exp_20':{
        'margin':200,'window':512,'step':400},
}

margin, window, step =
experiments_configs['exp_3'].values()

steps_dict = {}
files = steps_data['source_file'].unique()
for filename in files:
    src_path = os.path.join(SRCFILES_PATH,
filename) + src_ext
    steps_dict[filename] = get_src_file(src_path)

def get_one_step(steps, center,axis):
    steps = steps_dict[filename]
    step = steps[axis].iloc[center - margin : center +
margin]
    return step.values

for ax in ['x','y','z']:
    steps_data[ax] = steps_data.apply(
            lambda x: get_one_step(x["source_file"],
                                   x["step_center"],
                                   ax),
                    axis=1,
                    raw=False)

non_steps_path = "/content/drive/My
Drive/Diploma/experiments/1_250c_Есть шум
поезда метро в 400м.csv"
non_steps =
pd.read_csv(non_steps_path,delimiter=';',names=['x
','y','fake','z'])
non_steps = non_steps['z']
non_steps_chunks =
list(more_itertools.windowed(non_steps,n=margin*
2, step=step))
non_steps =
random.sample(non_steps_chunks,300)

steps = steps_data['z'].to_numpy()
steps = np.stack(steps)

n_steps = len(steps)
n_non_steps = len(non_steps)

X_all = np.vstack((steps, non_steps))

Y_all = np.vstack((
```

```
    np.ones((n_steps,1)),
    np.zeros((n_non_steps,1))
            ))

# N = n_steps + n_non_steps # do i need it?
J = 6 #max scale
T = window

X_train, X_test, y_train, y_test =
train_test_split(X_all,Y_all, test_size=0.1,
random_state=42)

train_n = X_train.shape[0]
test_n = X_test.shape[0]

x_train = np.zeros((train_n,T))
for k, x in enumerate(X_train):
   m = len(x)
   start = (T - m) // 2
   x_train[k, start:start + m] = x

x_test = np.zeros((test_n,T))
for k, x in enumerate(X_test):
   m = len(x)
   start = (T - m) // 2
   x_test[k, start:start + m] = x

"""# Train"""

log_eps = 1e-8
tf.keras.backend.clear_session()

desired_acc = .92
class MyCallback(tf.keras.callbacks.Callback):
   def on_epoch_end(self,epoch,log={}):
      if log['val_accuracy'] >= desired_acc:
         print(f"\nAccuracy has reached
{desired_acc*100}%")
         self.model.stop_training = True

callback = MyCallback()

# model = tf.keras.models.Sequential([
#       tf.keras.layers.Input(shape=(T)),
#       Scattering1D(J),
#       tf.keras.layers.Lambda(lambda x: x[..., 1:,
:]),
#       tf.keras.layers.Lambda(lambda x:
tf.math.log(tf.abs(x) + log_eps)),
#
tf.keras.layers.GlobalAveragePooling1D(data_form
at='channels_first'),
#       tf.keras.layers.BatchNormalization(axis=1),
#       # tf.keras.layers.Flatten(),
#       tf.keras.layers.Dense(64,activation='relu'),
#       tf.keras.layers.Dense(1, activation='sigmoid')

#    ])
# OK? acc ~96%

model = tf.keras.models.Sequential([
      tf.keras.layers.Input(shape=(T)),
      Scattering1D(J),
      tf.keras.layers.Lambda(lambda x: x[..., 1:, :]),
      tf.keras.layers.Lambda(lambda x:
tf.math.log(tf.abs(x) + log_eps)),

tf.keras.layers.GlobalAveragePooling1D(data_form
at='channels_first'),
      tf.keras.layers.BatchNormalization(axis=1),
      tf.keras.layers.Dense(512,activation='relu'),
      tf.keras.layers.Dense(1, activation='sigmoid')

   ])

model.()

tf.keras.utils.plot_model(model,show_shapes=True,
show_layer_names=False)

x_val = x_test
y_val = y_test

from tensorflow.keras.optimizers import SGD

model.compile(loss='binary_crossentropy',

optimizer=RMSprop(learning_rate=0.001,
rho=0.9),
            metrics=['accuracy'])


# model.compile(loss='binary_crossentropy',
#
optimizer=RMSprop(learning_rate=0.001,
rho=0.9),
#               metrics=['accuracy'])

history = model.fit(x_train,
                y_train,
                epochs=100,
                validation_data=(x_val, y_val),
                batch_size=32,
                callbacks=[callback])

"""## Results"""

# import matplotlib.pyplot as plt
# plt.plot(history.history['accuracy'])
# plt.show()
# plt.plot(history.history['loss'])
# plt.show()

"""# Data processing

Now we are going to detect all steps in all files.
"""

def find_steps(model, data, window=256, step=64):
   chunks =
list(more_itertools.windowed(data,n=window,
step=step))
   steps = []
   indexes = []
```

```
    k = 0
    for i, chunk in enumerate(tqdm(chunks[:-
1],desc='Step detection: ')):
        step_array = np.array(chunk)
        prediction =
model.predict(step_array.reshape(1,-1))
        if prediction > 0.5:
            k +=1
            # steps.append(step_array)

indexes.append(np.arange(i*step,i*step+window))

    # print(f'{k} steps detected')
    # return steps,indexes
    return indexes

columns =
['exp_name','start_idx','start_time','x','y','z']

detected_steps = pd.DataFrame(columns=columns)

files = list(filter(lambda x: x[-3:] == 'csv',
                os.listdir(SRCFILES_PATH)))
files = files[0:3]
savefolder = '/content/drive/My
Drive/Diploma/results/window_256/'
if not os.path.exists(savefolder):
    os.makedirs(savefolder)
for filename in tqdm(files, desc="File: "):
    filepath = os.path.join(SRCFILES_PATH,
filename)
    file_data = get_src_file(filepath)
    file_data['res'] = 0
    steps_idxs = find_steps(file_data['z'], window,
step)

    for index in steps_idxs:
        height = np.max(file_data['z'].iloc[index])
        file_data['res'].iloc[index] = height

    file_data.to_csv(savefolder+filename)

"""#Results"""

ROOT = "/content/drive/My
Drive/Diploma/results/window_256/"

# results = os.lisdir(ROOT)

# for result in tqdm(results,desc="Result"):
# result_name = '1_250c_Есть шум поезда метро
в 400м.csv'
result_name = "2_120c.csv"
# result_name = '3_100c.csv'
result =
pd.read_csv(ROOT+result_name,index_col=0)

result = result[10000:30000]

result.plot()
plt.show()
```

```
"""# Experiments"""

def MyModel(x_train,y_train,x_test,y_test,T):
    J = 6
    log_eps = 1e-8
    tf.keras.backend.clear_session()

    desired_acc = .92
    class MyCallback(tf.keras.callbacks.Callback):
        def on_epoch_end(self,epoch,log={}):
            if log['val_accuracy'] >= desired_acc:
                print(f"\nAccuracy has reached
{desired_acc*100}%")
                self.model.stop_training = True
    callback = MyCallback()

    model = tf.keras.models.Sequential([
            tf.keras.layers.Input(shape=(T)),
            Scattering1D(J),
            tf.keras.layers.Lambda(lambda x: x[..., 1:,
:]),
            tf.keras.layers.Lambda(lambda x:
tf.math.log(tf.abs(x) + log_eps)),

tf.keras.layers.GlobalAveragePooling1D(data_form
at='channels_first'),
            tf.keras.layers.BatchNormalization(axis=1),
            tf.keras.layers.Dense(64,activation='relu'),
            tf.keras.layers.Dense(1,
activation='sigmoid')

        ])


    model.compile(loss='binary_crossentropy',
            optimizer=RMSprop(learning_rate=0.001,
rho=0.9),
            metrics=['accuracy'])


    history = model.fit(x_train,
                        y_train,
                        epochs=100,
                        validation_data=(x_test, y_test),
                        batch_size=32,
                        callbacks=[callback],
                        verbose=1)

    return model, history

class Experiment():
    def __init__(self,data,configs,exp_name):
        self.margin = configs['margin']
        self.window = configs['window']
        self.step = configs['step']
        self.data = data
        self.model = None
        self.exp_name = exp_name
        self.x_train,self.y_train = None, None
        self.x_test,self.y_test = None, None
```

```
    def _process_data(self):
        steps_data = self.data.copy()
        steps_dict = {}
        files = self.data['source_file'].unique()
        for filename in files:
            src_path = os.path.join(SRCFILES_PATH,
filename) + src_ext
            steps_dict[filename] =
get_src_file(src_path)

        def get_one_step(steps, center,axis):
            steps = steps_dict[filename]
            step = steps[axis].iloc[center - self.margin :
center + self.margin]
            return step.values

        for ax in ['x','y','z']:
            steps_data[ax] = steps_data.apply(
                    lambda x:
get_one_step(x["source_file"],
                                    x["step_center"],
                                    ax),
                        axis=1,
                        raw=False)



        non_steps_path = "/content/drive/My
Drive/Diploma/experiments/1_250c_Есть шум
поезда метро в 400м.csv"
        non_steps = get_src_file(non_steps_path)
        non_steps = non_steps['z']
        non_steps_chunks =
list(more_itertools.windowed(non_steps,
                                        n=self.margin*2,
                                        step=self.step))
        non_steps =
random.sample(non_steps_chunks,300)

        steps = steps_data['z'].to_numpy()
        steps = np.stack(steps)

        n_steps = len(steps)
        n_non_steps = len(non_steps)

        X_all = np.vstack((steps, non_steps))

        Y_all = np.vstack((
            np.ones((n_steps,1)),
            np.zeros((n_non_steps,1))
                    ))

        J = 6 #max scale
        T = self.window

        X_train, X_test, y_train, y_test =
train_test_split(X_all,
                                            Y_all,
                                            test_size=0.1,

random_state=42)
        train_n = X_train.shape[0]
        test_n = X_test.shape[0]


        x_train = np.zeros((train_n,T))
        for k, x in enumerate(X_train):
            m = len(x)
            start = (T - m) // 2
            x_train[k, start:start + m] = x


        x_test = np.zeros((test_n,T))
        for k, x in enumerate(X_test):
            m = len(x)
            start = (T - m) // 2
            x_test[k, start:start + m] = x

        self.x_train,self.y_train = x_train, y_train
        self.x_test,self.y_test = x_test, y_test


    def _train_model(self):
        self.model, history = MyModel(self.x_train,
self.y_train,
                        self.x_test, self.y_test,
                        self.window)

        filepath = '/content/drive/My
Drive/Diploma/metrics.txt'
        with open(filepath,'a') as fl:
            acc = history.history['accuracy'][-1]
            val_acc = history.history['val_accuracy'][-1]
            string = f"\n{self.exp_name} train acc:
{acc}, val acc {val_acc}"
            fl.write(string)



    def _predict(self):
        savefolder = '/content/drive/My
Drive/Diploma/results/'
        files = os.listdir(savefolder)

        for filename in tqdm(files, desc="File: "):
            filepath = os.path.join(savefolder, filename)
            file_data = pd.read_csv(filepath,
delimiter=',')
            res_col_name = 'res_' + self.exp_name
            file_data[res_col_name] = 0
            steps_idxs = find_steps(self.model,
file_data['z'], self.window, self.step)

            for index in steps_idxs:
                height = np.max(file_data['z'].iloc[index])
                file_data[res_col_name].iloc[index] =
height

            file_data.to_csv(savefolder+filename,
index=False)


    def run(self):
```

```
        self._process_data()
        self._train_model()
        self._predict()

exp_name = 'exp_20'
exp =
Experiment(steps_data,experiments_configs[exp_n
ame],exp_name)
exp.run()

from IPython.display import clear_output


for exp_name,exp_configs in
tqdm(experiments_configs.items()):
    exp =
Experiment(steps_data,exp_configs,exp_name)
    exp.run()

savefolder = '/content/drive/My
Drive/Diploma/results/'
files = os.listdir(savefolder)
for filename in files:
    path = savefolder + filename
    df = get_src_file(path)
    df.to_csv(savefolder+filename,index=False)

pd.read_csv(path)

savefolder = '/content/drive/My
Drive/Diploma/results/'
files = os.listdir(savefolder)

for filename in tqdm(files, desc="File: "):
    filepath = os.path.join(savefolder, filename)
    file_data = pd.read_csv(filepath, delimiter=',')
    res_col_name = 'res_'
    file_data[res_col_name] = 0
    print(type(file_data['z']))

path = '/content/drive/My
Drive/Diploma/results/1_250c_Есть шум поезда
метро в 400м.csv'
# path = "/content/drive/My
Drive/Diploma/results/2_120c.csv"
df = pd.read_csv(path)

df[['x','y','z','res_exp_1']][6000:7000].plot()
plt.plot()

steps_data['source_file'].unique()

model = tf.keras.models.Sequential([
        tf.keras.layers.Input(shape=(256)),

        tf.keras.layers.Dense(64,activation='relu'),
        tf.keras.layers.Dense(1,
activation='sigmoid')

    ])

tf.keras.utils.plot_model(model,
show_shapes=True)

from scipy.signal import find_peaks

data = df['z'][6000:7200]
plt.plot(df[6000:7200][['x','y','z']])

peaks, _
=find_peaks(data,distance=None,prominence=0.01)
plt.scatter(data.iloc[peaks].index,data.iloc[peaks]
,marker='x',color='r')

peaks

df[['x','y','z','res_exp_1']][16000:17000].plot()
plt.plot()



"""# Model with autoencoder"""

from tensorflow.keras.models import Model

from tensorflow.keras.layers import (
    Input,
    Dense,
    Conv1D,
    LeakyReLU,
    MaxPooling1D,
    LSTM,
    Bidirectional,
    UpSampling1D,
    Conv2DTranspose,
    RepeatVector
)

# latent_dim = 128
# timesteps,input_dim = (256,1)
# inputs = Input(shape=input_shape)
# encoded = LSTM(latent_dim)(inputs)

# decoded = RepeatVector(timesteps)(encoded)
# decoded = LSTM(input_dim,
return_sequences=True)(decoded)


# autoencoder = Model(inputs, decoded)

autoencoder.compile(optimizer='adadelta',
loss='mse')

autoencoder.fit(x_train, x_train,
                epochs=100,
                shuffle=True,
                batch_size=128,
                validation_data=(x_test, x_test)
               )

autoencoder.summary()

autoencoder.compile(loss='mse',
```

```
            optimizer=RMSprop(learning_rate=0.01,
rho=0.9))
history = autoencoder.fit(x_train, x_train,
            epochs=10,
            shuffle=True,
            batch_size=128,
            validation_data=(x_test, x_test)
           )

latent_dim = 128
timesteps,input_dim = (256,1)
inputs = Input(shape=(timesteps,input_dim))
x = Conv1D(512,32,padding='same')(inputs)
x = MaxPooling1D()(x)
x = Conv1D(128,3,padding='same')(x)
x = MaxPooling1D()(x)
encoded = LSTM(latent_dim)(x)
x = Dense(512)(encoded)
x = LeakyReLU()(x)
last = Dense(1,activation='sigmoid')(x)

classifier = Model(inputs, last)

classifier.summary()

classifier.compile(
              loss='binary_crossentropy',
              metrics='accuracy',

optimizer=RMSprop(learning_rate=0.001, rho=0.9)
              )

history = classifier.fit(x_train, y_train,
            epochs=100,
            shuffle=True,
            batch_size=32,
            validation_data=(x_test, y_test),
           )

# path = '/content/drive/My
Drive/Diploma/results/7_100c.csv'
path = '/content/drive/My
Drive/Diploma/results/2_120c.csv'
# path = '/content/drive/My
Drive/Diploma/results/1_250c_Есть шум поезда
метро в 400м.csv'
df = pd.read_csv(path)

data = df['z'][36000:40000]
steps = find_steps(classifier,data)
int_steps = intersect_steps(steps)
plt.plot(data)
# for step in steps:
plt.plot(data,'b')
for step in int_steps:

   plt.plot(data.iloc[step],'r')
plt.show()

# plt.plot(df[['x','y','z']][6000:7000])
# nsteps = np.array(steps)
# detected_idx =
data.index[list(set(nsteps.flatten()))]
# ones = np.zeros((len(data),))
# for step in nsteps:
#     max_ = np.max( data.iloc[step])
#     # val = [max_]*len(step)
#     ones[step] = max_
#     # plt.plot(data.index[step],val)
# plt.plot(data.index,ones)
# plt.show()

ones[nsteps[0]]

def intersect_steps(steps):
   res = []
   n = len(steps)
   for i in range(0,n):
      intersected_steps = []
      for j in range(i-3, n):
         a = steps[i]
         b = steps[j]
         intersection = np.intersect1d(a,b)
         if len(intersection) != 0:
            intersected_steps.append(intersection)
      if len(intersected_steps) >=3:
         intersection = intersected_steps[1]
         for step in intersected_steps:
            intersection =
np.intersect1d(intersection,step)
         res.append(intersection)
   return res

def intersect_steps(steps):
   res = []
   n = len(steps)
   for i in range(0,n-2):
      intersection =
np.intersect1d(steps[i],steps[i+1])
      if len(np.intersect1d(intersection, steps[i+2]))
!= 0:
         intersection = np.intersect1d(intersection,
steps[i+2])
      res.append(intersection)
   return res

tf.keras.utils.plot_model(classifier,show_shapes=Tr
ue,show_layer_names=False)
```

ДОДАТОК Б ІЛЮСТРАТИВНИЙ МАТЕРІАЛ

Система розпізнавання кроків на основі вимірів віддалених сенсорів

Виконав:
Студент 4 курсу
Групи КА-64
Папаш Олексій
Науковий керівник:
Данилов В.Я

1

- Об’єкт дослідження: виміри сенсорів сейсмоакустичних хвиль
- Предмет дослідження: методи попередньої обробки та нейронні мережі

- Мета роботи:

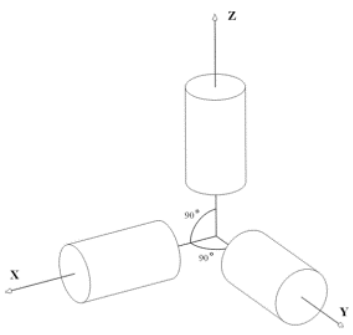
Дослідити існуючі методи обробки сейсмоакустичних хвиль; знайти методи класифікації сигналів та реалізувати систему розпізнавання кроків на основі даних отриманих із сенсора.

2

Приймач

В основі сенсору використовуються геофони.

Вимірами такого пристрою є градієнтні характеристики руху людини.

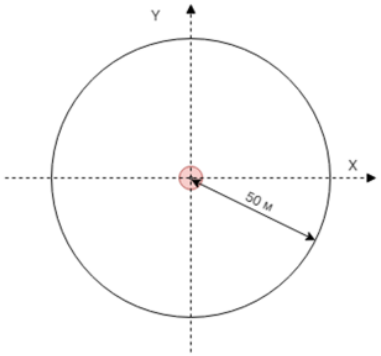


3

Експерименти та отримані дані

З даним сенсором було проведено 28 експериментів, під час яких приймальний прилад розміщували в ґрунті, а людина рухалася з різною швидкістю вздовж осі X або Y починаючи і закінчуючи ходьбу приблизно в 50 кроках від приладу.

При цьому прилад також фіксував шуми навколишнього середовища (віддалені метро, авто та люди).



4

Для дослідження була запропонована наступна постановка задачі:

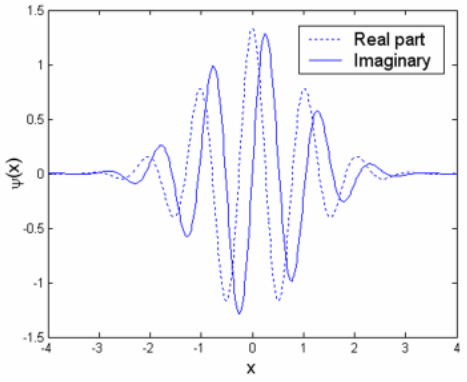
- Виконати огляд відомих методів дослідження та попередньої обробки сигналів.
- Для задачі дипломного проекту сформулювати задачу бінарної класифікації та провести огляд існуючих підходів для її розв'язання.
- Спроектувати СПР, яка містить наступні блоки: попередню обробку, класифікацію за допомогою нейронних мереж та локалізацію кроку.
- Підготувати дані експериментів, провести їх обробку та проаналізувати отримані результати.

5

Попередня обробка включає вибір типу вейвлета

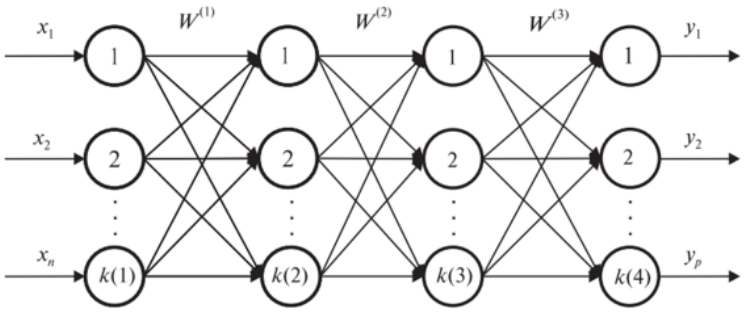
В результаті експериментів вибраний вейвлет Морле

$$\psi(t) = e^{-\frac{t^2}{2}} e^{i\omega t}$$



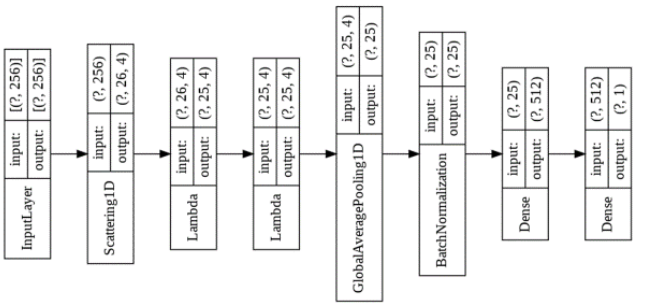
6

Класифікація
Класичний підхід MLP



7

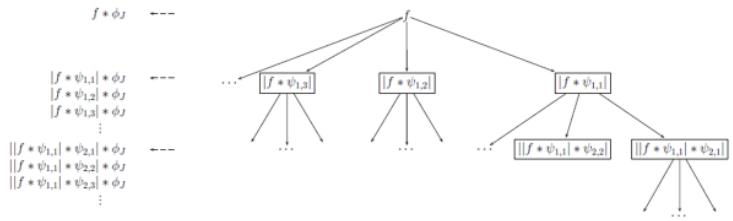
З використанням ідеології і методів сучаного машинного навчання згенеровано наступні архітектури моделей



8

Wavelet scattering

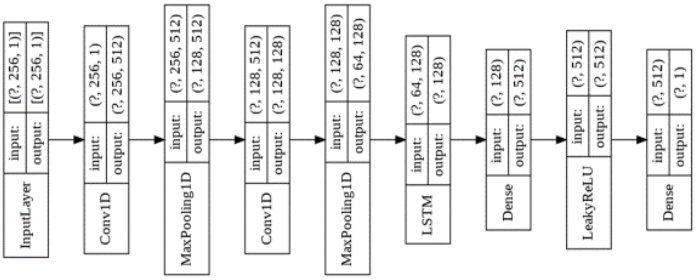
підхід до отримання за допомогою вейвлет перетворення
інваріативного до зсувів набору ознак сигналу, які в подальшому подаються на вхід класифікатора



На схемі $\{\psi_{j,k}\}$ – це вейвлети, ϕ_J – функції масштабування, а вхідні дані це f.

9

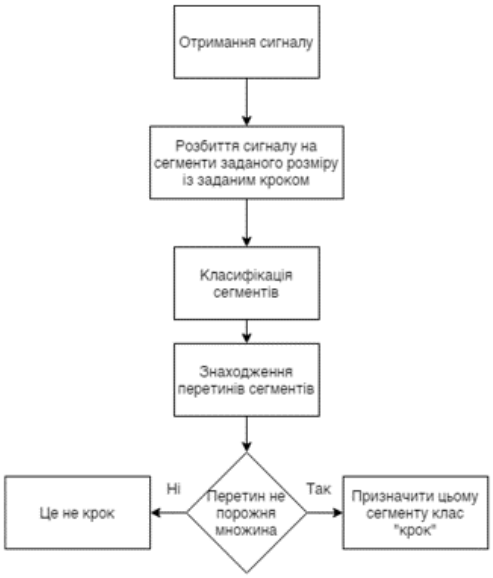
Мережа з LSTM енкодером



10

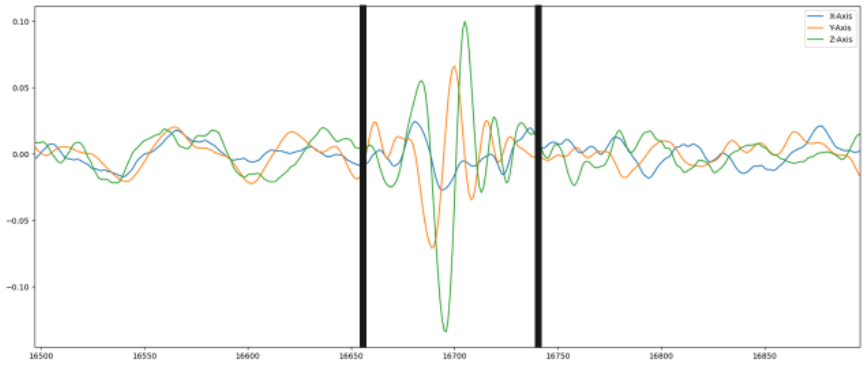
СПР

Остаточно в СПР був використаний другий варіант мережі, оскільки він мав кращу точність та менше помилявся на файлі із шумом навколишнього середовища.



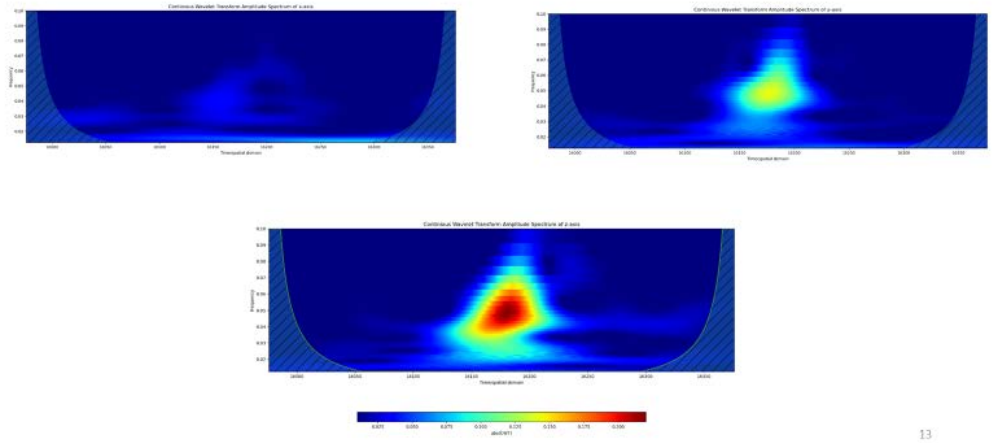
11

Дослідження сигналу сейсмоакустичного сенсора



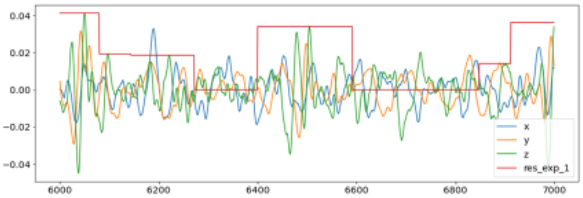
12

Результати вейвлет перевторення



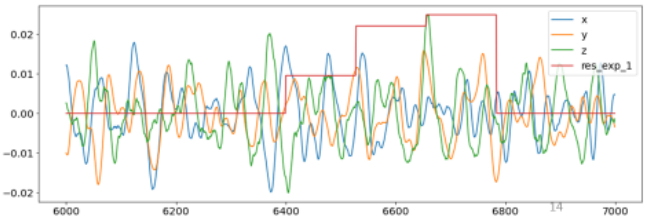
13

Робота нейронної мережі



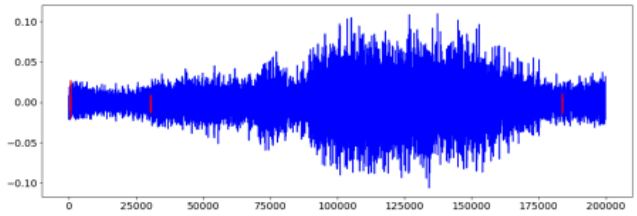
На фрагменті даних із кроками

На фрагменті даних тільки із шумом



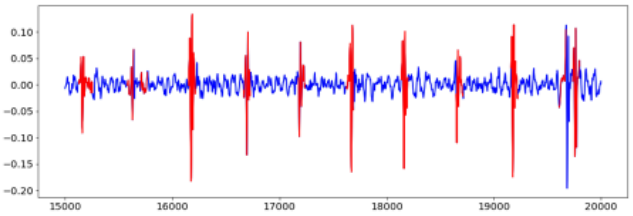
14

Робота СПР



На всіх даних тільки із шумом алгоритм знайшов всього лиш 3 неіснуючі кроки

Робота алгоримту на фрагменті даних з кроками



15

Висновки

- Розроблений алгоритм попередної обробки, яка визначає характер сеймоакустичних хвиль (кроків людини).
- Згенеровано дві різні сучасні архітектури нейронних мереж, які використані для класифікації кроків.
- Розроблена система для бінарної класифікації сигналів (виявлення кроків).
- Оброблено сумарно ~55 хв. вимірів, отриманих з 28 експериментів (1с – 800 відліків).
- Проаналізовано результати роботи кожної з моделей та СПР в цілому.
- Головним висновком є те, що на основі вимірів одного сенсору можна встановити присутність людини.

16

Дякую за увагу

17