Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки

«Затверджую»
Зав. кафедри теоретичної та
прикладної системотехніки
_____ д.т.н., проф. С. І. Шматков
«___» _____ 2023 р

Пояснювальна записка

до кваліфікаційної роботи
магістра

на тему: «Комп'ютерна модель аналізу біосигналів на основі глибокого навчання»

Захищено на засіданні
Атестаційної комісії № 40
протокол № __ від __.12.2023 р.
Оцінка _____ / _____
Голова Атестаційної комісії
_____ **СКОБ Ю. О.**

Виконав:

студентка 2 курсу, групи КІ– 61
за спеціальністю 123 – Комп'ютерна інженерія.
Галузь знань: 12 – Інформаційні технології
Кузікова Катерина Михайлівна_____

Керівник: кандидат техн. наук, доцент
Бакуменко Ніна Станіславівна_____

Рецензент: професор, професор кафедри комп'ютерної математики і аналізу даних Національного технічного університету «Харківський політехнічний інститут»
Погорєлов Станіслав Вікторович_____

Харків – 2023

АНОТАЦІЯ

Пояснювальна записка до магістерської атестаційної роботи складається зі вступу, трьох розділів, висновків, списку використаних джерел і чотирьох додатків. Загальний обсяг роботи складає 67 сторінок, із яких 50 сторінок основної частини з 13 рисунками, 17 найменуваннями списку використаних джерел та чотирма додатками.

У сучасному світі розробка комп'ютерних моделей для аналізу біосигналів здійснюється з метою вдосконалення моніторингу фізичного стану людини та попередження можливих травм під час фізичних вправ. У даній роботі пропонується комп'ютерна модель на основі глибокого навчання для класифікації дій людини під час виконання фізичних вправ. Для досягнення цієї мети використовується набір даних з Kaggle та методи комп'ютерного моделювання глибокого навчання.

Застосування комп'ютерних моделей у цій області має декілька переваг, включаючи автоматизацію та швидкість обробки даних, об'єктивність результатів, здатність виявляти навіть найменші аномалії та покращення точності аналізу. Особливу увагу слід звернути на глибоке навчання, яке є потужним методом для аналізу біосигналів завдяки його здатності виділяти важливі ознаки та робити прогнози з високою точністю на основі великої кількості даних.

В даній роботі докладно розглядається вибір методів та підходів до аналізу біосигналів, а також проводиться порівняльний аналіз з іншими методами класифікації. Вибір глибокого навчання на основі згорткових нейронних мереж обговорюється як потужний та ефективний метод для цієї задачі.

Ця робота має важливе практичне застосування в медицині, спорті, при реабілітації поранених військових, де важливий моніторинг фізичного стану та попередження травм.

Ключові слова: глибоке навчання, машинне навчання, аналіз біосигналів, згорткові нейронні мережі

ABSTRACT

The explanatory note to the master's attestation work consists of an introduction, three sections, conclusions, a list of used sources and two appendices. The total volume of the work is 68 pages, of which 50 pages of the main part with 13 figures, 16 names of the list of used sources and four appendices.

In the modern world, the development of computer models for the analysis of biosignals is carried out in order to improve the monitoring of the physical condition of a person and to prevent possible injuries during physical exercises. In this work, a computer model based on deep learning is proposed for the classification of human actions during physical exercises. To achieve this goal, a dataset from Kaggle and deep learning computer simulation methods are used.

The application of computer models in this area has several advantages, including automation and speed of data processing, objectivity of results, ability to detect even the smallest anomalies and improvement of accuracy of analysis. Special attention should be paid to deep learning, which is a powerful method for biosignal analysis due to its ability to extract important features and make predictions with high accuracy based on large amounts of data.

In this work, the choice of methods and approaches to the analysis of biosignals is considered in detail, and a comparative analysis is also carried out with other classification methods, such as logistic regression, the method of support vectors, decision trees, and others. The selection of deep learning based on convolutional neural networks is discussed as a powerful and efficient method for this task.

This work has an important practical application in medicine, sports, in the rehabilitation of wounded soldiers, where physical condition monitoring and injury prevention are important.

Keywords: deep learning, machine learning, biosignal analysis, convolutional neural networks

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ

PCA - аналіз основних компонентів;

CNN - convolutional neural network;

NA - пропущені значення;

ВСТУП

У наш час значний акцент робиться на розвиток комп'ютерних моделей, спрямованих на аналіз біосигналів, які відіграють важливу роль у вдосконаленні спостереження за фізичним станом людей. Цей напрямок досліджень зосереджений на розробці технологій, які не лише допомагають у контролі та відстеженні здоров'я в реальному часі, але й забезпечують профілактику травматизму під час занять фізичною активністю.

Інноваційні підходи у цій області сприяють створенню більш ефективних та інтуїтивно зрозумілих систем моніторингу, що дозволяють користувачам краще розуміти та управляти власними фізичними навантаженнями та станом здоров'я.

Ця робота вирізняється використанням згорткових нейронних мереж (CNN), які є одним із передових методів у галузі глибокого навчання, для аналізу біосигналів.

Згорткові нейронні мережі, як відомо, ефективні у виявленні складних шаблонів у великих наборах даних, що робить їх ідеально підходящими для обробки та аналізу біосигналів. Використання CNN у цьому контексті дозволяє розглядати більш глибокі та складні зв'язки між даними, що може виявити нові важливі інсайти та покращити точність передбачення.

Ця робота також стає важливою через її аналогію та відмінність від існуючих підходів. Вона демонструє альтернативні методи та підходи до аналізу біосигналів порівняно з традиційними методами. Розвиток та впровадження нових методів у машинному навчанні, таких як CNN, розширює можливості аналізу та інтерпретації біосигналів, що може мати значний вплив на медичні дослідження, діагностику та моніторинг.

Актуальність роботи. Комп’ютерні моделі і методи аналізу є важливим інструментом для розуміння та обробки біосигналів. Вони дозволяють автоматизувати процеси виявлення аномалій, класифікації станів і

використовуються для розробки нових методів діагностики та лікування. Вони дозволяють отримувати важливу інформацію з біосигналів, яку було б складно або навіть неможливо визначити вручну. Дана робота є актуальною, тому що спрямована на захист здоров'я та попередження травм під час виконання вправ з вагами, що є важливим завданням для фітнес-індустрії, спортивної медицини або, в наш час війни та після, використання для реабілітації поранених військових, дозволятиме покращити точність та швидкість аналізу порівняно з класичними методами за рахунок використання нейронних мереж глибокого навчання.

Метою даної кваліфікаційної роботи є підвищення точності розпізнавання біосигналів комп'ютерної системи мониторингу стану людини, яка виконує вправи з вагами на підставі нейромережевих моделей глибокого навчання.

Предмет дослідження — Предметом даної кваліфікаційної роботи є методи аналізу біосигналів, зокрема методи нейромережевого моделювання за допомогою глибокого навчання.

Об'єктом дослідження є процес аналізу біосигналів за допомогою комп'ютерних моделей з використанням методів штучного інтелекту та глибокого навчання, зокрема згорткових нейронних мереж.

Завдання дослідження:

1. Провести попередню обробку та стандартизацію даних: Виконати очищення, нормалізацію та стандартизацію даних біосигналів для забезпечення їх якості перед аналізом.
2. Розробити та удосконалити згорткову нейронну мережу (CNN): Створити модель CNN, яка ефективно аналізує біосигнали, та оптимізувати її архітектуру для підвищення точності та ефективності.
3. Аналізувати та інтерпретувати результати моделі: Ретельно проаналізувати результати, отримані від моделі CNN, і витягнути значущі висновки та інсайти з отриманих даних.

4. Оцінити робастність та надійність моделі: Перевірити здатність моделі стабільно та надійно функціонувати в різних умовах та з різними даними, оцінити її стійкість до помилок та варіабельності даних.

РОЗДІЛ 1
ОГЛЯД І ЗАСТОСУВАННЯ МЕТОДІВ ГЛИБОКОГО НАВЧАННЯ В АНАЛІЗІ БІОСИГНАЛІВ

1.1 Огляд проблематики аналізу біосигналів

У сучасному світі розвиток технологій у галузі обробки та аналізу біологічних сигналів відіграє важливу роль у медицині, спорті та наукових дослідженнях. Біосигнали, такі як ЕЕГ, ЕКГ і сигнали м'язової активності, є джерелом цінної інформації про стан організму людини. Вони допомагають виявляти патології, визначати фізичну підготовку та підтримувати якість життя. Глибоке навчання використовується для аналізу біосигналів, зокрема для автоматичного визначення правильності виконання вправ з вагами [1], [4].

Комп'ютерні моделі і методи аналізу є важливим інструментом для розуміння та обробки біосигналів. Вони дозволяють автоматизувати процеси виявлення аномалій, класифікації станів і використовуються для розробки нових методів діагностики та лікування.

Значення цих технологій особливо велике у контексті забезпечення ефективної реабілітації після травм чи хірургічних втручань, а також для використання у фітнес-індустрії. Особливу увагу заслуговує застосування цих методів у реабілітації військовослужбовців, які перенесли серйозні травми на війні, де точний і швидкий аналіз біосигналів може значно покращити процеси відновлення та адаптації.

Прогрес у цій галузі також стимулюється за рахунок інтеграції новітніх технологій штучного інтелекту та машинного навчання, які відкривають нові перспективи для розробки інноваційних діагностичних інструментів. Це, у свою чергу, сприяє покращенню якості медичних послуг та підвищенню ефективності лікування, роблячи внесок у загальне поліпшення якості життя пацієнтів.

Однак, незважаючи на значення біосигналів, їх аналіз часто стикається з викликами, пов'язаними з необхідністю точної та швидкої обробки великої кількості даних. Традиційні методи можуть бути не тільки часомісткими, але й вимагають значного внеску експертів, що іноді обмежує можливості оперативного реагування. В цьому контексті, використання методів глибокого навчання, зокрема згорткових нейронних мереж (CNN), може радикально трансформувати підходи до аналізу біосигналів.

Глибоке навчання, як передова галузь штучного інтелекту, вже показало свою ефективність у широкому спектрі застосувань, включаючи обробку зображень та аналіз мови. Застосування цих інноваційних технологій у сфері аналізу біосигналів відкриває перспективи для розробки більш точних, швидких та автоматизованих методів діагностики.

Ця кваліфікаційна робота зосереджена на розробці комп'ютерної моделі для аналізу біосигналів, використовуючи принципи глибокого навчання, з акцентом на згорткові нейронні мережі. Метою є дослідити, як ці технології можуть бути застосовані для покращення аналізу біосигналів.

1.2 Методи класифікації

В машинному навчанні класифікаційні методи використовуються для визначення категорій на основі аналізу вхідних даних. До таких методів відносяться [11]:

- логістична регресія;
- наївний байєсівський класифікатор;
- метод k-найближчих сусідів;
- дерева прийняття рішень;
- випадковий ліс;
- метод опорних векторів;
- нейронні мережі.

Логістична регресія. Цей метод є спеціалізованою формою множинної регресії, призначеною для прогнозування ймовірності появи певної категорії або події. Він ефективно використовується в ситуаціях, де залежна змінна є двійковою (наприклад, "так" або "ні", "позитивний" або "негативний"). Логістична регресія моделює ймовірність відповіді з використанням сигмоїдальної функції.

Наївний байєсівський класифікатор. Наївні методи Байеса – це набір алгоритмів навчання з учителем, заснованих на застосуванні теореми Байеса з «наївним» припущенням про умовну незалежності між кожною парою характеристик з урахуванням значення змінної класу.

Метод k-найближчих сусідів. Метод k-найближчих сусідів використовується для вирішення задачі класифікації. Він відносить об'єкти до класу, якому належить більшість з k його найближчих сусідів в багатовимірному просторі ознак. Це один з найпростіших алгоритмів навчання класифікаційних моделей. Число k – це кількість сусідніх об'єктів в просторі ознак, які порівнюються з класифікованим об'єктом.

Іншими словами, якщо k =8, то кожен об'єкт порівнюється з 8-ю сусідами.

Дерева прийняття рішень. Дерева рішень – це не параметричний контрольований метод навчання, що використовується для класифікації та регресії. Мета полягає в тому, щоб створити модель, яка передбачає значення цільової змінної шляхом вивчення простих правил прийняття рішень, виведених із особливостей даних. Дерево можна розглядати як кусочно постійне наближення. Це моделі, що використовують ієрархічну структуру правил для прийняття рішень. Вони розділяють набір даних на все менші підмножини на основі конкретних значень ознак, утворюючи деревоподібну

структуру. Цей метод зручний, оскільки легко інтерпретується, але може бути схильним до перенавчання.

Випадковий ліс. Випадковий ліс – це контрольований алгоритм навчання, який використовується як для класифікації, так і для регресії. Але, тим не менш, він в основному використовується для задач класифікації. Оскільки знаємо, що ліс складений з дерев, і більше дерев означає більш стійкий ліс. Так само алгоритм випадкового лісу створює дерева рішень для вибірок даних, а потім отримує прогноз по кожній з них і, нарешті, вибирає краще рішення за допомогою голосування .

Метод опорних векторів. Метод опорних векторів – один з найбільш популярних методів навчання, який застосовується для розв'язання задач класифікації і регресії. Основна ідея методу полягає в побудові гіперплощини, що розділяє об'єкти вибірки оптимальним способом.

Алгоритм працює в припущенні, що чим більше відстань (зазор) між роздільною гіперплощиною і об'єктами поділюваних класів, тим менше буде середня помилка класифікатора. Основна ідея класифікатора на опорних векторах полягає в тому, щоб будувати роздільну поверхню з використанням тільки невеликої підмножини точок, що лежать в зоні, критичної для поділу, тоді як інші, вірно класифіковані спостереження навчальної вибірки поза цією зоною, ігноруються (точніше, є "резервуаром" для оптимізаційного алгоритму) .

Нейронні мережі. Нейронні мережі вирізняються своєю здатністю до навчання та адаптації, ефективно розпізнаючи складні взірці у великих масивах даних. Вони складаються з великої кількості вузлів або "нейронів", які взаємодіють один з одним через мережу зв'язків, передаючи сигнали під час процесу обробки інформації. Ця структура дозволяє нейронним мережам ефективно моделювати складні залежності та розпізнавати взаємозв'язки в даних.

1.3 Глибоке навчання

1.3.1 Принципи роботи глибокого навчання

Глибоке навчання є розвинутою галуззю машинного навчання, яка використовує структури, засновані на архітектурі штучних нейронних мереж. Ці мережі наслідують функціональні аспекти людського мозку, дозволяючи машинам виявляти складні патерни та залежності у великих наборах даних. В основі глибокого навчання лежить концепція, що машина може навчатися з досвіду та самостійно вдосконалювати свої алгоритми.

Типова модель нейронної мережі складається з декількох ключових шарів, кожен з яких виконує певну функцію [10].

Вхідний шар (Input Layer): Це точка входу в мережу для вхідних даних.

Функціонал: Вхідний шар є першим контактним пунктом між даними та нейронною мережею. Кожен нейрон у цьому шарі представляє одну ознаку (feature) вхідних даних. Наприклад, у випадку зображення, кожен нейрон може представляти інтенсивність окремого пікселя.

Обробка даних: Вхідні дані можуть бути нормалізовані або стандартизовані, щоб мережа краще їх обробляла. Нормалізація забезпечує, що всі вхідні дані мають однаковий масштаб, що важливо для ефективного навчання.

Приховані шари (Hidden Layers): Це серце мережі, де відбувається обробка даних через взаємозв'язані нейрони.

Функціонал: Приховані шари формують основу нейронної мережі та є місцем, де відбувається фактичне "навчання". Вони можуть складатися з одного

або багатьох шарів і є відповідальними за виявлення складних патернів та взаємозв'язків у даних.

Обробка даних:

- Кожен нейрон у прихованому шарі отримує ваговані сигнали від нейронів попереднього шару.
- Дані обробляються за допомогою функції активації, яка вводить нелінійність, дозволяючи мережі вчитися більш складні шаблони.
- Нелінійність є критичною, оскільки без неї, незалежно від кількості шарів у мережі, вся мережа все одно функціонувала б як один лінійний класифікатор.

Вихідний шар (Output Layer): Це кінцева частина мережі, яка видає результат або передбачення.

Функціонал: Вихідний шар видає кінцеві передбачення мережі. В залежності від завдання, вихідний шар може мати один нейрон (наприклад, для бінарної класифікації) або кілька нейронів (наприклад, для багатокласової класифікації).

Обробка даних:

- Дані з останнього прихованого шару передаються до вихідного шару.
- Для задач класифікації часто використовується softmax функція активації, яка конвертує вихід нейронів у вірогідності належності до певних класів.

Кожен нейрон у прихованому шарі отримує вхідні дані, обробляє їх за допомогою вагових коефіцієнтів та функції активації, і передає результат наступному шару або вихідному шару.

Функції Активації

Це основні елементи нейронної мережі, які визначають, чи повинен активуватись нейрон. Найбільш поширені функції активації включають:

- ReLU (Rectified Linear Unit): Пропускає позитивні сигнали і блокує негативні, що забезпечує ефективне навчання.
- Сигмоїд: Перетворює вхід у значення між 0 і 1, корисно для бінарної класифікації.
- Тангенс гіперболічний (tanh): Перетворює вхід у значення між -1 і 1, що може бути корисним для збереження знаку інформації.
- Softmax: Використовується у вихідному шарі для мульти-класової класифікації для представлення вірогідності належності до кожного класу.

Навчання мережі

Навчання мережі включає в себе:

- Визначення функції втрати (Loss Function): вона вимірює, наскільки добре мережа виконує своє завдання.
- Оптимізація: використання алгоритмів оптимізації, таких як SGD (Stochastic Gradient Descent), Adam, або RMSprop, для мінімізації функції втрати.
- Зворотне поширення помилки: після кожної ітерації прямого поширення, зворотне поширення ефективно розповсюджує помилку з вихідного шару назад через мережу, щоб оновити ваги нейронів з метою покращення майбутніх передбачень.

Недоліки нейронних мереж

Незважаючи на свою потужність, глибоке навчання має кілька викликів:

- Обчислювальні витрати: Глибокі нейронні мережі вимагають значних обчислювальних ресурсів, особливо для великих датасетів.

- Потреба в даних: Ефективне навчання мережі вимагає великих обсягів високоякісних даних.
- Перенавчання та узагальнення: Мережі схильні до перенавчання на тренувальних даних, що може обмежувати їх здатність до узагальнення на нових даних.
- Інтерпретація: Велика складність моделей ускладнює їх інтерпретацію та розуміння того, як саме вони досягають певних результатів.

Незважаючи на ці виклики, глибоке навчання продовжує розвиватися, з появою нових архітектур, підходів до навчання та застосувань в різних доменах.

1.3.2 Основні типи нейронних мереж

- Повнозв'язні мережі (Fully Connected or Dense Networks): Кожен нейрон у одному шарі з'єднаний з усіма нейронами наступного шару.
- Згорткові нейронні мережі (Convolutional Neural Networks, CNNs): Використовують згорткові шари, які автоматично та ефективно визначають і вивчають просторові ієрархії у даних, наприклад, в зображеннях.
- Рекурентні нейронні мережі (Recurrent Neural Networks, RNNs): Мають зворотні зв'язки, що дозволяють зберігати попередню інформацію в "пам'яті", що ефективно для обробки послідовних даних, як-от текст або часові ряди.
- Мережі глибокого посилення навчання (Deep Reinforcement Learning): Ці мережі тренуються на основі системи винагород, де агент навчається виконувати дії в середовищі, щоб максимізувати деяку метрику винагороди.

Тренування мережі.

Основний процес тренування глибокої нейронної мережі має два етапи:

- Пряме поширення: Дані проходять від вхідного шару до вихідного, при цьому кожен шар мережі вносить свій вклад у вихідні дані.

- Зворотне поширення: Після отримання вихідних даних, мережа використовує алгоритм зворотного поширення для оновлення ваг, заснованих на помилці між вихідними даними та фактичними мітками. Це дозволяє мережі "вчитися" з своїх помилок.

Застосування глибокого навчання.

Глибоке навчання виявилося особливо ефективним у таких областях:

- Комп'ютерний зір: Від розпізнавання облич до автономних автомобілів.
- Обробка природної мови (NLP): Від чат-ботів до систем автоматичного перекладу.
- Аудіо та музика: Від генерації музики до розпізнавання мовлення.
- Медицина: Від діагностики захворювань до персоналізованої медицини.
- Ігри: Глибоке навчання використовується для створення більш складного штучного інтелекту в іграх.

Глибоке навчання продовжує розширювати свої кордони і відкривати нові можливості у всіх цих областях, збільшуючи своє значення в індустрії штучного інтелекту.

1.4 Згорткові нейронні мережі

1.4.1 Структура згорткових нейронних мереж

Згорткові нейронні мережі є одним з найбільш важливих класів теорії глибокого машинного навчання для вирішення задач комп'ютерного зору. Згорткові нейронні мережі складаються з декількох шарів обробки, які як правило чергуються, кожен з шарів містить, як лінійні, так і нелінійні оператори. Сьогодні використання згорткових мереж є одним з основних методів для вилучення ознак з аудіо, відео і текстових даних.

Згорткова нейронна мережа (CNN) складається з трьох основних видів шарів: згортковий шар, субдискредитуючий шар і вихідний шар (найчастіше повнозв'язний) [12].

Шари CNN розташовані один за одним: спочатку згортковий шар, а потім субдискредитуючий, за останніми згортковим шаром слідує вихідний шар. Згорткові і субдискредитуючі шари вважаються шарами двовимірної розмірності, а вихідний шар, як правило, являє собою вектор з простору $\mathbb{R}$. У CNN кожен двовимірний шар має декілька рівнів. Кожен рівень являє собою двовимірний масив. Вихід кожного рівня називають картою ознак.

Приклад архітектури згорткової нейронної мережі представлений на рис. 1.1.

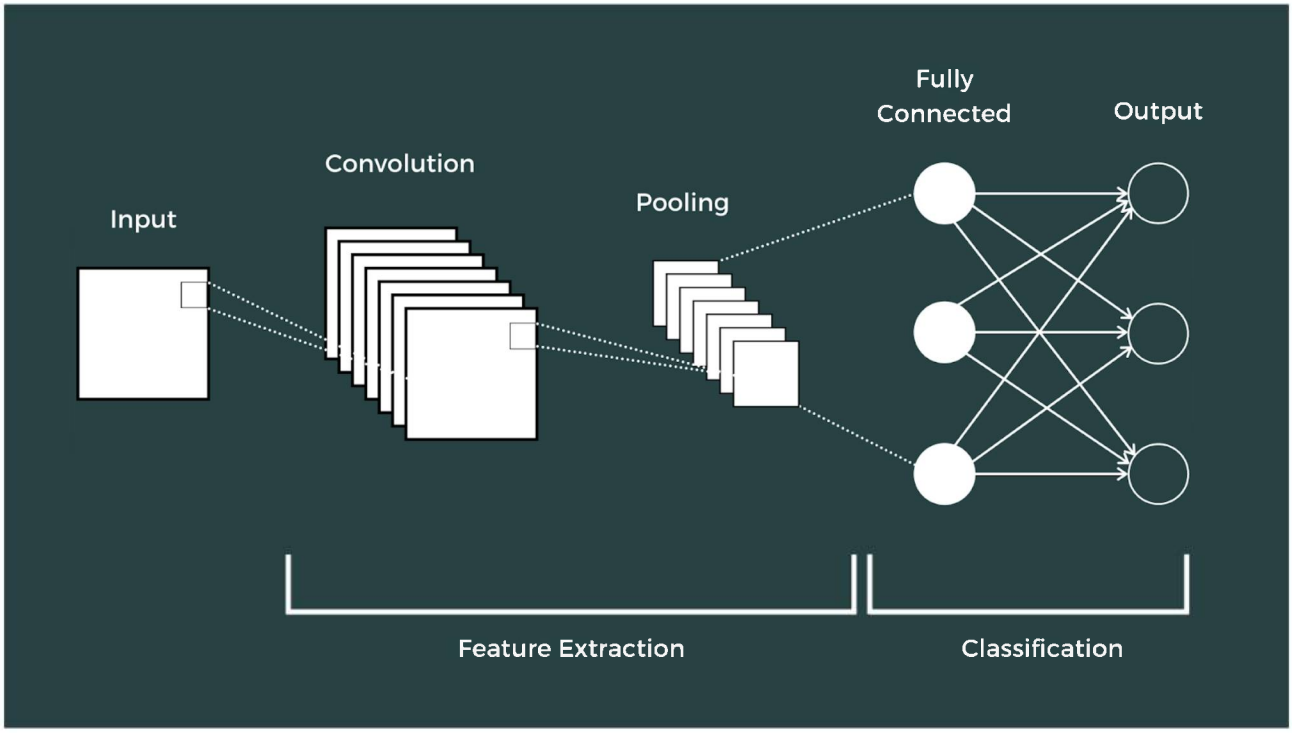


Рисунок 1.1 – Архітектура згорткової нейронної мережі

Основною ідеєю використання згорткового шару є застосування математичної операції згортки (фільтра) до зображення [9]. Згортка – це двовимірна матриця коефіцієнтів. Вхід такого фільтра – це фрагмент двовимірного зображення, а вихід – деяке число (рис. 1.2). Перевага

використання подібного роду фільтрів полягає в наступному: число на виході тим більше, чим більше елемент зображення схожий на застосовуваний до нього фільтр. Отже, використання операції згортки допомагає отримати на виході зображення, кожен піксель якого буде відповідати ступеню подібності шматочка зображення на фільтр. Іншими словами, ми отримаємо карту ознак.

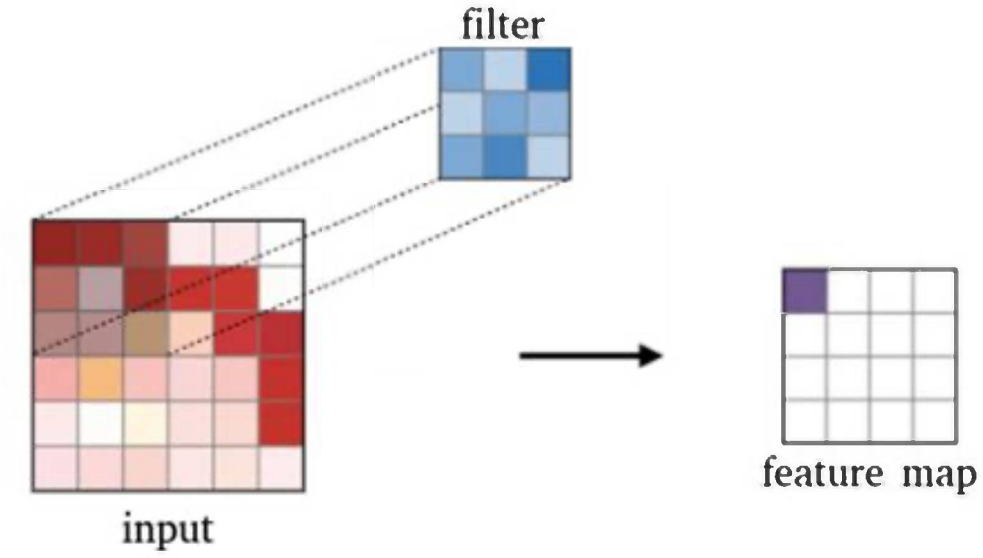


Рисунок 1.2 – Приклад використання двовимірної операції згортки

Щодо використання згорткової мережі у цій роботі:

Згорткові нейронні мережі (CNN) використовуються в роботі для класифікації фізичних вправ згідно з даними, отриманими з носимих пристроїв. CNN аналізує шаблони у даних, що стосуються варіацій біцепс-згинів, для виявлення типу виконуваної вправи. Оскільки дані включають часові послідовності рухів або сигнали сенсорів, згорткові шари в CNN можуть ефективно виявляти просторові та часові залежності в даних, що є критично важливим для розпізнавання патернів руху.

Опис застосування CNN включає:

- Використання глибини: CNN здатні ефективно виявляти складні залежності в даних завдяки використанню багатьох прихованих шарів.

- Локальне з'єднання та згортки: Замість того, щоб пов'язувати кожен нейрон з усіма вхідними даними, згорткові шари зосереджуються на невеликих областях, що дозволяє їм виявляти локальні ознаки, такі як краї, кути, та інші візерунки, що можуть бути важливими для класифікації.
- Субдискретизація (Pooling): Цей процес зменшує розмірність карти ознак, зберігаючи при цьому важливу інформацію. Це допомагає зменшити обчислювальні вимоги та запобігти перенавчанню.
- Розпізнавання шаблонів: У цьому сценарії CNN використовується для визначення шаблонів рухів або активності, які відповідають певним класам фізичних вправ.
- Повнозв'язні шари: Після згорткових і пулінг шарів мережа використовує один або декілька повнозв'язних шарів для класифікації функцій на основі виявлених патернів.

CNN застосовується для того, щоб знайти взаємозв'язки та патерни в даних, які є занадто складними для традиційних алгоритмів машинного навчання, дозволяючи точно класифікувати різні варіації вправ.

Висновки за розділом 1

Технології обробки та аналізу біосигналів відіграють вирішальну роль у розвитку сучасної медицини, спорту та наукових досліджень. Вони сприяють покращенню розуміння фізіологічних процесів в організмі людини та відіграють важливу роль у виявленні та лікуванні різних станів. Актуальність даної роботи обумовлена необхідністю захисту здоров'я та попередження травм під час виконання вправ з вагами, що є важливим завданням для фітнес-індустрії та спортивної медицини.

Виходячи з аналізу методів аналізу біосигналів, для розв'язку поставленої задачі були обрані згорткові нейронні мережі (CNN), оскільки:

- CNN ефективно виявляють просторові залежності в даних, що є критичним для аналізу біосигналів, які часто мають складну структуру та взаємозв'язки.
- Ці мережі здатні виділяти та розпізнавати певні взірці, що є ключовим для розпізнавання складних шаблонів активності м'язів або серцевого ритму.
- Вони забезпечують зменшення розмірності даних та запобігають перенавчанню, що є важливим для роботи з великими наборами даних, типовими для аналізу біосигналів.

РОЗДІЛ 2
ПІДГОТОВКА ДАНИХ ТА РОЗРОБКА МОДЕЛІ АНАЛІЗУ БІОСИГНАЛІВ

2.1 Вибір мови програмування

В цьому розділі розглядається процес підготовки даних, що є критичним етапом для ефективного аналізу біосигналів. Дані з датасету "Weight Lifting Exercises" із платформи Kaggle [3] були завантажені та оброблені за допомогою бібліотеки Pandas у Python.

Використання Python для обробки даних з датасету "Weight Lifting Exercises" з платформи Kaggle було обрано з кількох причин:

- **Широкий вибір бібліотек для обробки даних:** Python відомий своїми потужними бібліотеками для аналізу та обробки даних. Бібліотека Pandas, зокрема, є стандартом де-факто для обробки та аналізу даних у Python. Вона дозволяє зручно виконувати завантаження, очищення, перетворення та аналіз даних.
- **Інтуїтивно зрозумілий синтаксис:** Python відомий своїм чистим та зрозумілим синтаксисом, що робить код легким для читання та написання. Це особливо корисно у великих проектах з обробки даних, де чистота та зрозумілість коду мають велике значення.
- **Інтеграція з іншими бібліотеками та інструментами:** Python має відмінну сумісність з різними інструментами та бібліотеками для наукових розрахунків, машинного навчання та візуалізації даних, як-от NumPy, SciPy, Matplotlib, Seaborn, TensorFlow і Keras. Це дозволяє виконувати комплексний аналіз та обробку даних в одному програмному середовищі.
- **Спільнота та підтримка:** Python має одну з найбільших і найактивніших спільнот серед мов програмування. Це означає широку доступність ресурсів для навчання, активну підтримку на форумах та в спільнотах.

- **Гнучкість та масштабування:** Python ефективний як для швидкого прототипування, так і для розробки масштабованих систем. Це робить його ідеальним вибором для експериментів і розвитку складних проектів з обробки даних.
- **Сумісність з різними платформами:** Python є кросплатформною мовою, що дозволяє використовувати написані на ній програми в різних операційних системах без значних змін у коді.
- **Продуктивність та ефективність:** Незважаючи на те, що Python часто критикують за нижчу швидкість порівняно з компільованими мовами, такими як C++ або Java, його ефективність у сфері обробки даних часто покращується за рахунок оптимізованих бібліотек, які використовують C/C++ у своїх надрах.

Всі ці фактори роблять Python відмінним інструментом для роботи з великими наборами даних та розробки моделей машинного навчання, як у цьому випадку з аналізом біосигналів.

2.2 Підготовка і обробка даних для аналізу біосигналів

Процес підготовки та обробки даних для аналізу біосигналів відіграє ключову роль у розробці ефективних алгоритмів машинного навчання, особливо в контексті аналізу фізичних вправ.

Для роботи з даними використовувались такі бібліотеки:

Pandas (pd):

Використовується для завантаження, обробки та аналізу даних.

Scikit-Learn (sklearn):

- PCA
- StandardScaler: Стандартизує дані, нормалізуючи їх для забезпечення більш ефективного навчання.
- train_test_split: Розділяє дані на навчальні та тестові набори.

- Метрики (accuracy_score, classification_report, confusion_matrix): Використовуються для оцінки ефективності моделі.

TensorFlow та Keras:

Для побудови та тренування нейронної мережі.

- Sequential використовується для створення моделі, де шари додаються послідовно.
- Conv2D, MaxPooling2D, Flatten, Dense: Це різні типи шарів, що використовуються у CNN.

Matplotlib та Seaborn (sns):

Використовуються для візуалізації даних. У даному контексті, вони використовуються для побудови матриці невідповідності та інших графіків для аналізу результатів моделі.

Numpy (np):

Використовується для числових обчислень. Використовується для операцій з масивами.

LabelEncoder:

Використовується для перетворення категоріальних змінних у числовий формат, який можна використовувати в машинному навчанні.

EarlyStopping (з Keras):

Це механізм для запобігання перенавчання моделі. Навчання може бути зупинено раніше, якщо показники продуктивності перестають покращуватися на валідаційному наборі даних.

Перш за все, за допомогою бібліотеки pandas здійснюється завантаження та попередня обробка даних з CSV файлу. Важливим аспектом роботи з біосигналами є ефективна підготовка та обробка даних, яка включає управління пропущеними значеннями. На етапі попередньої обробки, завантажені з CSV-файлу дані проходять через кілька ключових етапів обробки.

Початковий крок включає ідентифікацію та видалення пропущених значень. В контексті даної роботи, пропущені значення (NA) видаляються

шляхом видалення стовпців, які містять хоча б одне NA. Цей підхід забезпечує цілісність датасету, оскільки використання неповних даних може спотворити результати аналізу.

Після видалення стовпців з NA, відокремлюється цільова змінна 'classe', яка представляє класифікацію виконання фізичної вправи (A - правильно виконана, B, C, D, E - помилки). Важливо, що видалення стовпців з NA відбувається до відокремлення 'classe', щоб забезпечити збереження кореляцій між цільовою змінною та іншими характеристиками[14].

Після цього, з основного набору даних видаляється стовпець 'classe', а сама змінна 'classe' перетворюється у числовий формат за допомогою LabelEncoder. Це перетворення дозволяє використовувати цільову змінну в алгоритмах машинного навчання, які вимагають числових вхідних даних.

Далі, видаляються стовпці з текстовими значеннями, оскільки вони можуть спотворити аналіз, особливо при використанні алгоритмів, які працюють із числовими даними. Залишені числові дані піддаються стандартизації за допомогою StandardScaler, що гарантує однорідність всіх числових характеристик датасету.

Цей підхід до обробки даних, який включає управління пропущеними значеннями та вибіркову стандартизацію, створює міцну основу для подальшого аналізу біосигналів за допомогою алгоритмів машинного навчання. Він дозволяє забезпечити високу якість вхідних даних, що є критичним для досягнення точних та надійних результатів у процесі аналізу.

Після стандартизації та очищення даних зберегли їх в новому файлі(рис.2.1.).

cleaned_data.csv

```
raw_timestamp_part_1,raw_timestamp_part_2,num_window,roll_belt,pitch_belt,yaw_belt,total_accel_belt,gyros_belt_x,gyros_belt_y,gyros_belt_z,acc
-1.6538631420973395,-1.6143821082025744,-1.7394582979495457,-0.9769583833590815,1.8382102430708986,-0.757749777979599,-1.0836590030320759,9.76
-1.6538631420973395,-1.517489105590082,-1.7394582979495457,-0.9775966249940744,1.8917810648748896,-0.7545912596922237,-1.2129896354892689,9.47
-1.6538631420973395,-1.4897351197858102,-1.7394582979495457,-0.9788731082640599,1.9319591812278831,-0.7524855808339735,-1.3423202679464619,9.0
-1.6538631420973395,-1.4481630299659067,-1.7394582979495457,-0.9791922290815563,1.963208827280211,-0.7503799019757235,-1.3423202679464619,8.70
-1.6538631420973395,-1.4205095016739069,-1.7394582979495457,-0.9790326686728081,1.9944584733325394,-0.7482742231174734,-1.3423202679464619,8.5
-1.6538631420973395,-1.143423434462486,-1.7394582979495457,-0.9809473935777865,2.0167796490842025,-0.7482742231174734,-1.3423202679464619,8.46
-1.6538631420973395,-1.0740072934824805,-1.7394582979495457,-0.9831812393002612,2.043565059986198,-0.7482742231174734,-1.0836590030320759,8.60
-1.6538631420973395,-0.8936756667976595,-1.7394582979495457,-0.9895636556501893,2.074814706038526,-0.7514327414048486,-0.9543283705748828,8.46
-1.6538631420973395,-0.8076701800841719,-1.7394582979495457,-0.9991372801750812,2.097135881790189,-0.7556440991213488,-1.2129896354892689,7.98
-1.6538631420973395,-0.7829437758566516,-1.7394582979495457,-1.0040836528462753,2.1105285872411867,-0.757749777979599,-1.0836590030320759,7.16
-1.6538631420973395,-0.7415171762269352,-1.7394582979495457,-0.9975416760875992,2.1149928223915193,-0.7524855808339735,-1.2129896354892689,6.2
-1.6538631420973395,-0.6721876365506126,-1.7394582979495457,-0.9962651928176136,2.1149928223915193,-0.7503799019757235,-1.2129896354892689,5.5
-1.6538631420973395,-0.6719867215260684,-1.7394582979495457,-0.998977719766333,2.1016001169405216,-0.7503799019757235,-1.2129896354892689,4.89
-1.6538631420973395,-0.5750244378706298,-1.7394582979495457,-1.0018498071238005,2.079278941188859,-0.7503799019757235,-1.0836590030320759,4.65
-1.6538631420973395,-0.5746156797172469,-1.7394582979495457,-1.0036049716200308,2.0569577654371956,-0.7514327414048486,-0.8249977381176898,4.8
-1.6538631420973395,-0.5745429346221533,-1.7394582979495457,-1.0056792569337574,2.0301723545352,-0.7514327414048486,-0.5663364732033038,5.2835
-1.6538631420973395,-0.5337156160139221,-1.7394582979495457,-1.0064770589774983,2.01231541393387,-0.7503799019757235,-0.5663364732033038,6.006
-1.6538631420973395,-0.5335770539280296,-1.7394582979495457,-1.0090300255174696,2.01231541393387,-0.7482742231174734,-0.5663364732033038,6.680
-1.6538631420973395,-0.492160846454755 16,-1.7394582979495457,-1.0155720022761456,2.0167796490842025,-0.7482742231174734,-0.9543283705748828,7.
-1.6538631420973395,-0.4920292124731573,-1.7394582979495457,-1.0240287039398002,2.0301723545352,-0.7493270625465984,-1.2129896354892689,7.4034
-1.6538631420973395,-0.43659398595970705,-1.7394582979495457,-1.0283368349760016,2.0480292951365304,-0.7472213836883482,-0.9543283705748828,7.
-1.6538631420973395,-0.2980947530058381,-1.7394582979495457,-1.0281772745672535,2.0703504708881932,-0.7430100259718481,-0.4370058407461108,6.6
-1.6538631420973395,-0.201093864776779,-1.7394582979495457,-1.0312089223334693,2.1149928223915193,-0.7387986682553477,0.20964732153985416,6.05
-1.6538631420973395,-0.15948020633110774,-1.7394582979495457,-1.0364744158221597,2.1774921144961756,-0.7366929893970975,1.373623013654591,5.81
```

Рисунок 2.1 - Дані, збережені в окремому файлі після стандартизації

Далі, використовуючи **train_test_split**, дані діляться на частини для навчання та перевірки моделі, що є критичним для перевірки ефективності моделі. Щоб наглядно подивитись пропорцію класів у навчальному наборі використаємо бібліотеку **matplotlib** для створення графіка, який візуалізує ці пропорції (рис.2.2).

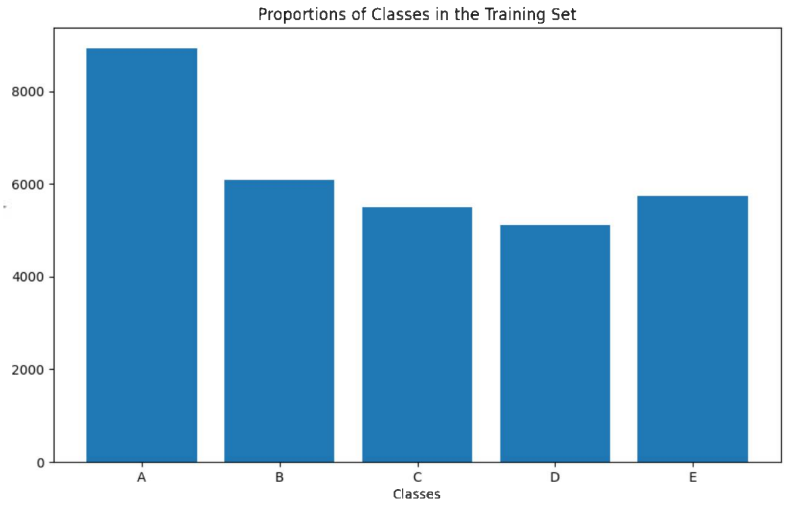


Рисунок 2.2 - Графік пропорції класів

Далі використовується метод головних компонент (PCA) для зменшення розмірності даних, що не тільки спрощує модель, але й допомагає уникнути перенавчання.

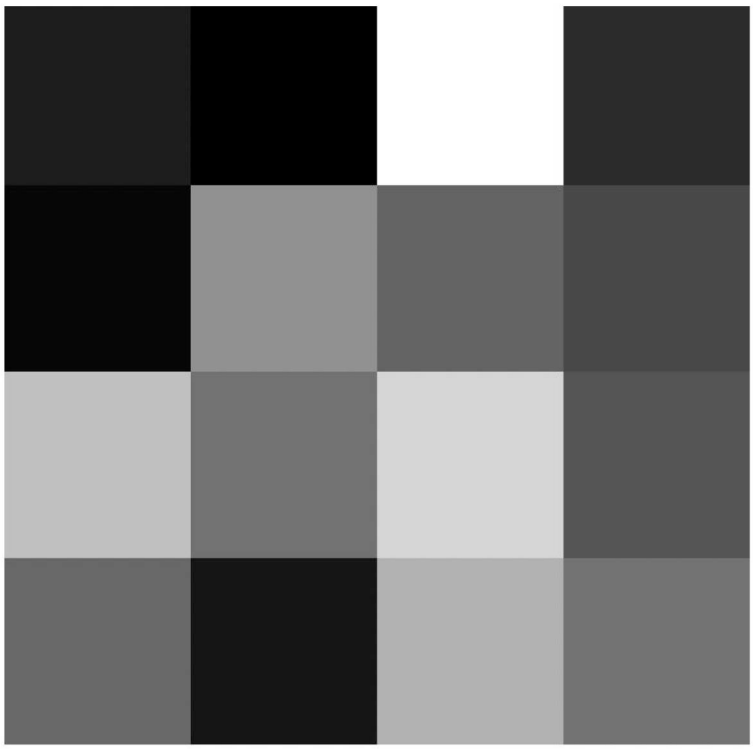
Однією з ключових причин застосування PCA в цій роботі є необхідність адаптації даних для використання в згортковій нейронній мережі (CNN). Згорткові мережі ефективно обробляють дані, що мають просторову структуру, як-от зображення. Оскільки біосигнали у нашому випадку не є візуальними даними з самого початку, використання PCA дозволяє трансформувати їх у формат, який може бути візуалізований та ефективно оброблений CNN. Конвертація даних у квадратні карти, такі як 4x4, є критичною для цього процесу, дозволяючи використовувати потужні можливості CNN для виявлення складних шаблонів та взаємозв'язків у біосигналах.

У PCA «компонент» відноситься до нової функції, створеної алгоритмом. Ось що відбувається під час PCA:

Зменшення розмірності: PCA зменшує розмірність даних, знаходячи новий набір функцій (компонентів), які є лінійними комбінаціями вихідних функцій. Ці нові функції фіксують найбільш значні розбіжності в даних.

Порядок компонентів: компоненти впорядковані за величиною дисперсії, яку вони фіксують у даних. Перший компонент фіксує найбільшу дисперсію, другий найбільше фіксує другий і так далі. Цей порядок визначається під час процесу підгонки PCA, коли алгоритм обчислює власні вектори та власні значення з коваріаційної матриці даних.

Візуалізація: коли візуалізуються ці компоненти в сітці (наприклад, сітку 4x4), кожен квадрат представляє значення одного компонента для одного спостереження (у нашому випадку, однієї вправи). Інтенсивність кольору відображає величину значення компонента після відображення в колірній шкалі. На карті(рис.2.3) світліші кольори представляють нижчі значення, а темніші — вищі.



Дисперсія вимірює, наскільки набір чисел (точок даних) розкинувся від їх середнього значення. У статистиці це кількісне вираження мінливості або розкиду в наборі даних. Чим вище дисперсія, тим більш розкидані точки даних.

PCA та дисперсія: PCA шукає напрямки (основні компоненти), де дисперсія даних є максимальною. Ідея полягає в тому, що висока дисперсія зазвичай відповідає важливим структурним особливостям у наборі даних, оскільки вона вказує, де дані найбільш розпорошені та де вони можуть бути найбільш інформативними.

Врахування дисперсії: перший головний компонент фіксує найбільшу дисперсію, тобто він враховує якомога більше варіабельності даних. Другий головний компонент фіксує наступну найбільшу величину дисперсії та є ортогональним (під прямим кутом) першому компоненту тощо. Цей процес триває, доки кількість компонентів не зрівняється з кількістю вихідних розмірів або доки компоненти не охоплять достатню дисперсію.

Важливість дисперсії в PCA: компоненти з найвищою дисперсією вважаються найважливішими, оскільки вони представляють напрямки максимальної варіативності даних. Це функції, які PCA використовує, щоб спробувати зрозуміти базову структуру даних.

2.3 Перетворення даних у формат, сумісний зі згортковою нейронною мережею (CNN)

Це є важливим кроком у підготовці для аналізу біосигналів. У даному випадку, датасет "Weight Lifting Exercises", який первинно містить числові дані, було перетворено на зображення 4x4. Ось ключові аспекти та міркування за цим підходом:

- **Перетворення даних у зображення:** CNN традиційно використовуються для аналізу візуальних даних, таких як зображення. У даному випадку, числові дані біосигналів перетворюються у візуальний

формат, створюючи зображення 4x4 пікселів. Кожен піксель у цьому зображенні відображає певну характеристику або вимірювання від існуючого датасету.

- **Збереження інформації:** Цей підхід забезпечує, що важливі характеристики даних зберігаються, навіть після трансформації. Кожен "піксель" або елемент у 4x4 зображенні несе інформацію про оригінальні характеристики біосигналу.
- **Сумісність з CNN:** CNN ефективно виявляють просторові залежності в зображеннях за допомогою своїх згорткових шарів. Перетворенням даних в зображення, можна використовувати CNN для виявлення складних взаємозв'язків та шаблонів в даних біосигналів.
- **Оптимізація архітектури мережі:** Вибір розміру зображення 4x4 є результатом оптимізації, яка забезпечує достатню деталізацію для аналізу, але також зберігає модель достатньо простою для ефективного навчання та висновку.
- **Візуалізація даних:** Цей підхід також відкриває можливості для візуалізації даних, які можуть бути корисними для аналізу та інтерпретації результатів. Зображення можуть бути відображені для визначення того, як модель "бачить" дані.
- **Експериментування та налагодження:** Під час процесу розробки моделі цей підхід дозволяє експериментувати з різними конфігураціями та налаштуваннями CNN, щоб знайти оптимальну архітектуру для конкретного набору даних.

Загалом, перетворення даних біосигналів у зображення 4x4 для використання в CNN дозволяє використовувати потужні алгоритми глибокого навчання для аналізу цих даних, відкриваючи нові можливості для їхнього розуміння та інтерпретації.

2.4 Вибір згорткових нейронних мереж (CNN) для аналізу біосигналів

Використання згорткових нейронних мереж (CNN) у цій роботі ґрунтується на їх винятковій здатності ефективно обробляти та аналізувати багатовимірні дані. CNN відрізняються своєю архітектурою, яка імітує механізм зору людини, здатного виділяти та розпізнавати певні взірці в візуальному вході. Ця здатність особливо корисна для аналізу біосигналів, які часто мають складну структуру та взаємозв'язки в даних.

Обґрунтування використання CNN [12]**:**

- **Локалізація ознак:** CNN використовують згорткові шари для виявлення локалізованих ознак у даних, таких як форми та краї. У контексті аналізу біосигналів, це означає, що модель може ефективно ідентифікувати важливі характеристики в сигналах, такі як піки та відхилення, які можуть вказувати на певні фізичні стани або відповіді.
- **Просторова ієрархія:** CNN здатні вчитися просторовим ієрархіям ознак, що дозволяє їм виявляти складніші шаблони на вищих рівнях. Для біосигналів це означає можливість розрізнення складних шаблонів активності м'язів або серцевого ритму.
- **Зменшення розмірності та запобігання перенавчанню:** Субдискретизація (Pooling) в CNN дозволяє зменшувати розмірність даних, зберігаючи при цьому важливу інформацію. Це знижує ризик перенавчання, одночасно зберігаючи здатність моделі виявляти ключові ознаки в біосигналах.

- **Ефективна обробка часових послідовностей:** Незважаючи на те, що CNN традиційно використовуються для аналізу зображень, їх можна адаптувати для роботи з часовими послідовностями, які є характерними для біосигналів. Це робить CNN ідеальними для аналізу даних, отриманих від датчиків руху або активності м’язів.

Докази ефективності CNN у аналізі біосигналів:

- **Дослідження в галузі:** Існують численні дослідження, які демонструють ефективність CNN у класифікації та аналізі медичних зображень та біосигналів. Наприклад, дослідження, проведене в області ЕКГ-аналізу, показало, що CNN може виявляти серцеві патології з високою точністю.
- **Практичні результати:** В цій роботі застосування CNN до датасету "Weight Lifting Exercises" дозволило з високою точністю класифікувати різні типи фізичних вправ, що підтверджує їх придатність для аналізу складних біосигналів.

2.5 Архітектура та налаштування моделі CNN

Модель CNN була спроектована з урахуванням специфіки даних біосигналів. Використання різних типів шарів (Conv2D, MaxPooling2D, Flatten, Dense) було спрямовано на оптимальне виявлення та класифікацію характеристик біосигналів. Особливу увагу було приділено налаштуванню параметрів кожного шару, включаючи кількість та розміри фільтрів у згорткових шарах, а також на вибір активаційних функцій, які максимально ефективно обробляють особливості біосигналів.

Структура та налаштування шарів CNN

Згорткові Шари (Conv2D):

- Функціональність: Згорткові шари є основою моделі CNN. Вони відповідають за виявлення локальних ознак у даних, таких як краї, форми та текстури, що є важливими для аналізу біосигналів.

- Налаштування: Визначення розміру фільтрів та кількості фільтрів у кожному згортковому шарі було ключовим. Наприклад, менші фільтри (наприклад, 3x3 або 5x5) використовувались для виявлення більш тонких і детальних ознак, тоді як більші фільтри допомагали виявляти більш загальні патерни.

Шари Субдискретизації (MaxPooling2D):

- Функціональність: Ці шари зменшують розмірність карти ознак, що допомагає зменшити обчислювальне навантаження та запобігає перенавчанню.
- Налаштування: Вибір розміру пулінгу (наприклад, 2x2 або 3x3) впливає на ступінь зменшення розмірності. Було обрано баланс, що дозволяє зберегти необхідні ознаки без надмірного втручання в інформацію.

Плоский Шар (Flatten):

- Функціональність: Цей шар перетворює двовимірні карти ознак у одновимірний вектор, готуючи дані для подальшої класифікації.
- Налаштування: Flatten шар служить важливим переходом від згорткової частини мережі до класифікаційної частини.

Повнозв'язні Шари (Dense):

- Функціональність: Ці шари відповідають за кінцеву класифікацію виходу моделі.
- Налаштування: Кількість нейронів у повнозв'язних шарах та вибір активаційних функцій (наприклад, ReLU або softmax) було адаптовано з урахуванням специфіки задачі.

2.6 Тренування та оцінка моделі

Процес оцінки та аналізу моделі глибокого навчання, зокрема згорткових нейронних мереж (CNN), є фундаментальною частиною розробки та використання моделей у практичних застосуваннях.

Тестування моделі:

- Після тренування модель перевіряється на окремому наборі тестових даних. Це забезпечує незалежну оцінку продуктивності моделі.
- Тестові дані не використовувалися під час навчання, що дозволяє перевірити здатність моделі до узагальнення на нових, невідомих даних.

Метрики оцінки:

- Точність (Accuracy)**:** Це відсоток випадків, де модель правильно класифікувала вхідні дані.
- Матриця Невідповідності (Confusion Matrix)**:** Візуалізує, як модель класифікує кожен клас порівняно з фактичними мітками. Це дозволяє зрозуміти типи помилок, які робить модель.
- Класифікаційний Звіт (Classification Report)**:** Включає інші важливі метрики, такі як точність (precision), відтворення (recall) та F1-бал (F1-score) для кожного класу.

Візуалізація результатів:

- Використання **matplotlib** та **seaborn** для створення графічних зображень результатів, включаючи матрицю невідповідності.
- Графіки надають інтуїтивно зрозуміле розуміння продуктивності моделі.

Збереження результатів

Збереження результатів оцінки:

- Результати оцінки моделі, включаючи метрики та вихідні дані, зберігаються у текстових файлах. Це забезпечує постійний доступ до результатів та можливість порівняння з майбутніми моделями.

Демонстрація практичного застосування:

- Код також може включати демонстрацію практичного застосування моделі, таку як візуалізація передбачень для конкретних випадків.
- Це дозволяє краще зрозуміти, як модель функціонує у реальних умовах.

Значення процесу

- Цей процес не тільки важливий для перевірки та валідації моделі, але й надає цінну інформацію для подальшого вдосконалення моделі.
- Отримані дані дозволяють аналізувати сильні та слабкі сторони моделі, зокрема її здатність правильно класифікувати різні класи біосигналів.
- Виявлення областей для покращення може привести до внесення коригувань у модель, наприклад, зміну структури шарів, налаштування гіперпараметрів або навіть перегляд підходу до обробки вхідних даних.

Висновки за розділом 2

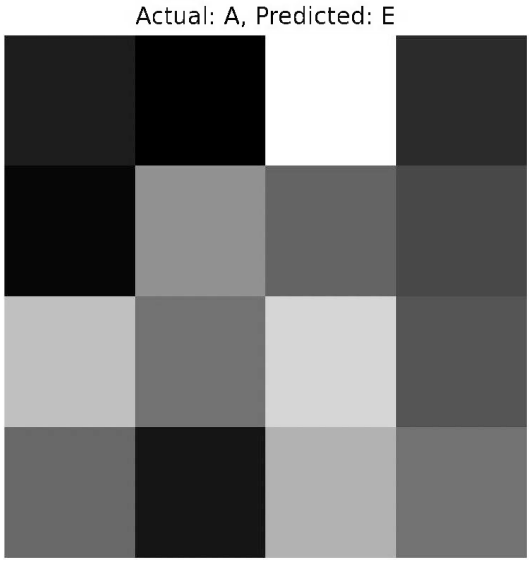
У цьому розділі обговорюється процес підготовки та аналізу даних для створення моделі глибокого навчання. Python був обраний через свої потужні бібліотеки та інтуїтивно зрозумілий синтаксис. Дані з Kaggle були завантажені, оброблені та перетворені в зображення 4x4 для сумісності з CNN. Використання CNN було обґрунтовано їхньою здатністю ефективно обробляти складні біосигнали. Модель була ретельно сконструйована з використанням різних типів шарів для оптимального аналізу. Важливим етапом було тренування та оцінка моделі, використовуючи метрики, такі як точність та матриця невідповідності. Процес завершувався візуалізацією результатів та збереженням даних оцінки, що забезпечило цілісне розуміння роботи моделі.

РОЗДІЛ 3
АНАЛІЗ РЕЗУЛЬТАТІВ НАВЧАННЯ МОДЕЛІ ТА ЇЇ ВДОСКОНАЛЕННЯ

3.1. Аналіз результатів

В результаті, ми бачимо візуалізацію даних (рис.3.1), змінених у сітку 4x4, яка представляє трансформовані функції з PCA, застосовуваного до набору даних.

Кожен квадрат у сітці відповідає одному з 16 головних компонентів для конкретного зразка з тестового набору. Колір градацій сірого вказує на величину значення компонента, причому темніші відтінки представляють вищі значення, а світлі відтінки представляють нижчі значення. PCA впорядковує компоненти за величиною дисперсії, яку вони пояснюють у даних, від найбільшої до найменшої. Коли змінюється форма компонентів PCA на сітку 4x4, вони розміщуються в цій сітці в тому самому порядку, у якому вони виводяться PCA, зазвичай з першим компонентом (що пояснює найбільшу дисперсію) у верхньому лівому куті, а останній компонент – (що пояснює найменшу дисперсію) у нижньому правому куті.



Заголовок «Actual: A, Predicted: E» говорить нам, що для цього конкретного випадку справжньою міткою було «A», тобто вправа важкої атлетики була виконана правильно, але модель неправильно передбачила її як «E», вказуючи на певний тип помилки у виконанні вправи.

Результат навчання показує, що точність моделі підвищилася за 10 епох, що вказує на те, що модель навчалася на даних навчання. Остаточна точність набору перевірки становила близько 81%. Класифікаційний звіт (рис.3.2) показує точність, запам'ятовування та оцінку F1 для кожного класу (A, B, C, D, E). Ці показники дають уявлення про ефективність моделі для кожного класу:

Клас A мав високий коефіцієнт запам'ятовування 0,94, тобто модель правильно визначила 94% усіх фактичних класів A.

Точність для класу E становила 0,90, що вказує на те, що 90% випадків, передбачених як клас E, були правильними.

Точність усіх прогнозів становила 0,81, тобто модель правильно передбачила клас у 81% випадків.

diploma.py | classification_report.txt

classification_report.txt

	precision	recall	f1-score	support
A	0.79	0.94	0.86	2225
B	0.81	0.71	0.76	1503
C	0.73	0.68	0.70	1346
D	0.83	0.77	0.80	1314
E	0.90	0.86	0.88	1461
accuracy			0.81	7849
macro avg	0.81	0.79	0.80	7849
weighted avg	0.81	0.81	0.81	7849

Рисунок 3.2 - Класифікаційний звіт

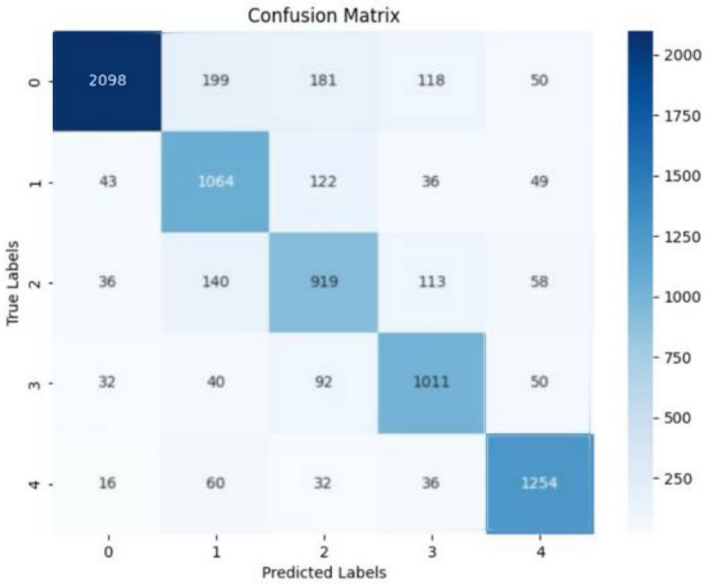


Рисунок 3.3 - Матриця невідповідності

Також для кращого аналізу навчання моделі було додано створення графіків, один для точності (accuracy) і один для помилки (loss), які показують, як ці показники змінювались протягом епох тренування моделі. Можна побачити, як модель поліпшується (або погіршується) із часом.

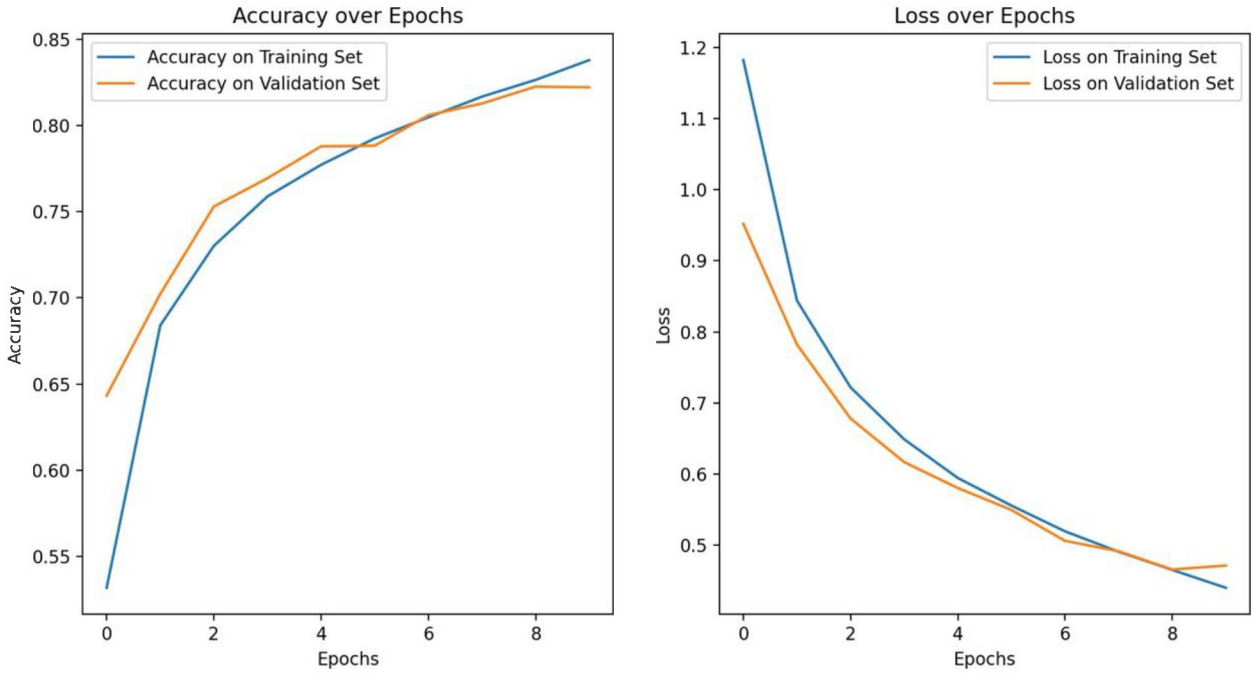


Рисунок 3.4 - Графіки точності та помилки для тестових/навчальних наборів

Набір графіків (рис.3.4) відображає початкові характеристики навчання моделі. Графік точності показує, що точність на навчальному наборі даних (синя лінія) та на тестовому наборі (помаранчева лінія) стабільно збільшується з кожною епохою, що демонструє здатність моделі до навчання. Відмітимо, що точність на тестовому наборі даних майже співпадає з точністю на навчальному наборі, що може вказувати на добру узагальнюючу здатність моделі на даному етапі.

Графік помилок відображає зниження значення функції втрат як для навчального, так і для тестового наборів, що свідчить про зменшення кількості помилок, які робить модель під час прогнозування. Спад функції втрат на обох наборах є швидким і послідовним, що підтверджує ефективність оптимізації моделі.

Загалом зображення та вихідні дані свідчать про те, що модель згорткової нейронної мережі працює досить добре, але може бути місце для вдосконалення, особливо щодо правильної класифікації всіх класів.

3.2 Вдосконалення моделі

Тож, базуючись на результатах я вирішила вдосконалити модель.

Щоб підвищити точність моделі та переконатися, що вона однаково враховує всі класи (A, B, C, D, E), я розглянула такі стратегії:

- Збільшити складність моделі: якщо модель не переобладнана, можна спробувати додати більше шарів або збільшити кількість нейронів у щільних шарах, щоб охопити складніші зв'язки в даних.
- Збільшити дані: ця техніка може збільшити різноманітність навчального набору шляхом застосування випадкових, але реалістичних модифікацій, таких як введення шуму у вхідні функції. Однак, оскільки дані не є даними зображення, потрібно думати про те, яке розширення має сенс для функцій,

трансформованих PCA. А так як дані беруться з датасету, то цей варіант був виключен.

• Ваги класів: якщо набір даних незбалансований (деякі класи мають набагато більше прикладів, ніж інші), можна використовувати ваги класів, щоб наказати моделі «звертати більше уваги» на класи з меншою кількістю прикладів під час навчання.

• Налаштувати гіперпараметри: варіативність з різними темпами навчання, оптимізаторами та розмірами пакетів, щоб знайти найкращу комбінацію для моделі.

• Перехресна перевірка: замість єдиного поділу навчання/тесту можна використати k-кратну перехресну перевірку, щоб забезпечити узгодженість продуктивності моделі в різних підмножинах даних.

• Рання зупинка: можна використати ранню зупинку, щоб припинити навчання, коли втрати перевірки перестануть покращуватися, запобігаючи переобладнанню.

Після роздумів і проб я додала такі зміни:

- Налаштування Ваг Класів для Незбалансованих Даних: використала ваги класів для компенсації можливої незбалансованості у наборі даних.
- Зміна Архітектури Конволюційної Нейронної Мережі (CNN): збільшила кількість фільтрів (64 замість 32) та нейронів (128 замість 64) у повнозв'язних шарах, що збільшує складність моделі.
- Застосування Ранньої Зупинки (Early Stopping): використала метод ранньої зупинки під час тренування моделі для запобігання перенавчання.
- Кількість Епох тренування Моделі: збільшила кількість епох з 10 до 50, хоча реальна кількість епох може бути меншою через ранню зупинку.
- Візуалізація Передбачень: змінила відображення 'Oranges' замість ‘gray' для більшої наглядності.

Після оновлення коду, в мене вийшли такі результати в класифікаційному звіті(рис.3.5):

diploma.py | confusion_matrix.png | classification_report.txt

classification_report.txt

	precision	recall	f1-score	support
A	0.98	0.96	0.97	2225
B	0.91	0.97	0.94	1503
C	0.94	0.85	0.89	1346
D	0.88	0.97	0.92	1314
E	0.99	0.95	0.97	1461
accuracy			0.94	7849
macro avg	0.94	0.94	0.94	7849
weighted avg	0.94	0.94	0.94	7849

Рисунок 3.5 - Класифікаційний звіт після внесених змін

Продуктивність моделі помітно покращилася.

Точність тесту: 94%

Спостерігається помітне підвищення точності та запам’ятовування в усіх класах, а не лише в A та E. Точність, макросереднє та середньозважене значення підвищилися, що вказує на більш збалансовану та точну модель.

Конкретні вдосконалення:

- Запам'ятовуваність класу C значно покращилася, вказуючи на менше хибно негативних результатів для цього класу.
- Загальний баланс між класами з точки зору точності та запам’ятовування кращий, що свідчить про те, що модель тепер більш вправна в ідентифікації всіх класів вправ, а не упереджена до підмножини.

• Матриця невідповідності показує більш діагональний шаблон, що вказує на кращу продуктивність, з меншою кількістю неправильних класифікацій між класами.

Матриця невідповідності (рис.3.6) має вищі значення вздовж діагоналі (справжні позитивні результати), що означає, що модель має менше помилкових позитивних і помилкових негативних результатів.

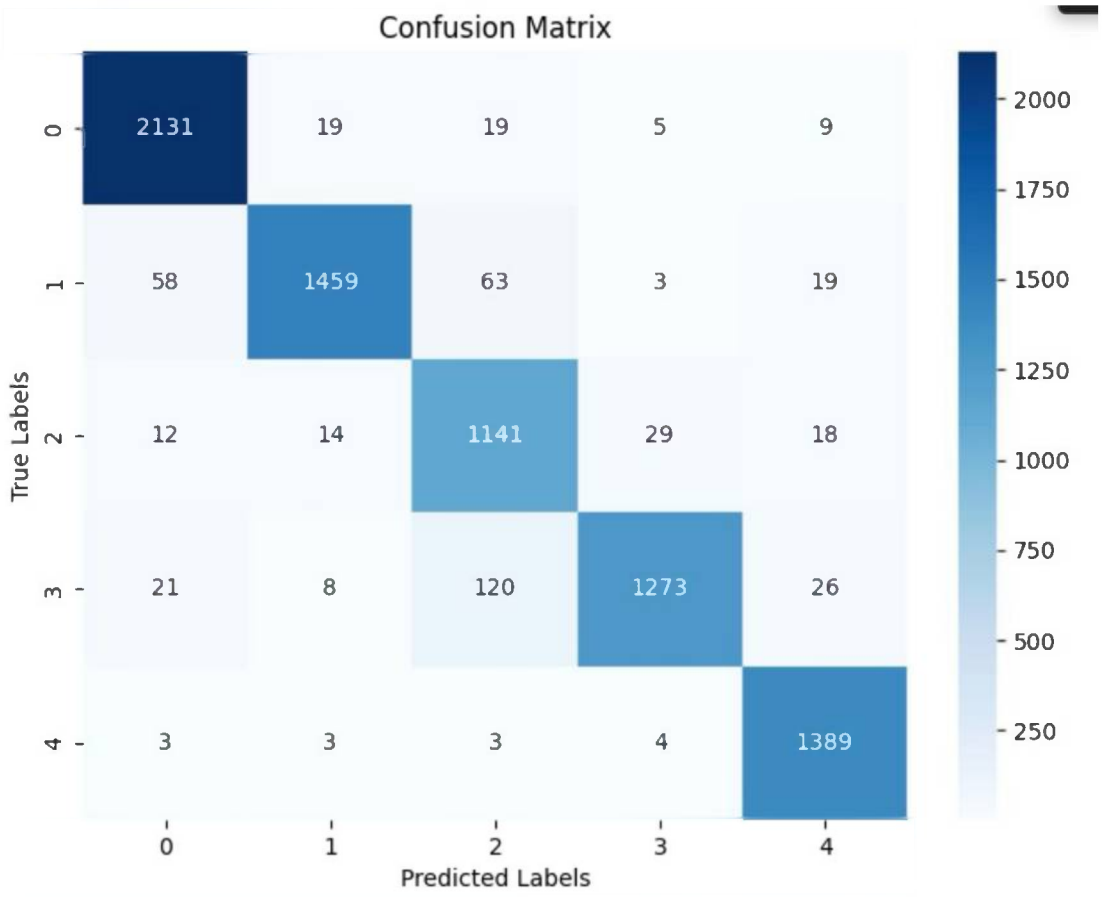


Рисунок 3.6 - Матриця невідповідності

Справжні мітки (вісь Y): це фактичні класи, як вони відображаються в тестовому наборі даних. Кожен рядок відповідає класу, який модель намагалася передбачити.

Прогнозовані мітки (вісь X): це класи, які передбачила модель. Кожен стовпець представляє передбачення, зроблені для класу.

Числа всередині матриці: кожна клітинка в матриці представляє кількість зразків, для яких прогнозований клас (стовпець) відповідає справжньому класу (рядок).

Наприклад, клітинка у верхньому лівому куті (2131) повідомляє, що було 2131 випадків, коли модель правильно передбачила клас 0.

Комірки по діагоналі матриці (від верхнього лівого кута до нижнього правого) представляють правильні прогнози (справжні позитивні для кожного класу).

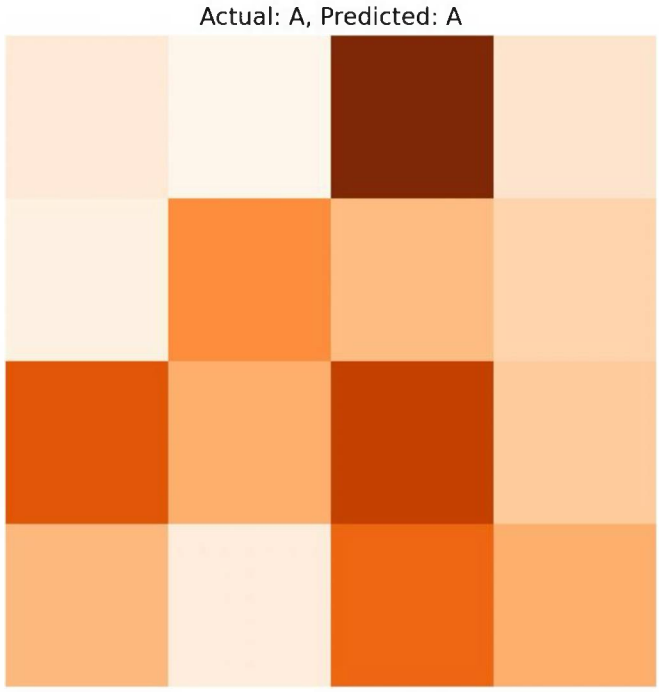
Недіагональні комірки представляють неправильні прогнози, при цьому номер рядка вказує на справжній клас, а номер стовпця вказує на прогнозований клас.

Наприклад, число 58 у другому рядку першого стовпця вказує на те, що модель неправильно передбачила 58 екземплярів класу 1 як клас 0.

Чутливість матриці невідповідності:

Матриця невідповідності надає розбивку прогнозів для кожного класу, показуючи, де модель допускає помилки.

Аналізуючи матрицю, можна визначити, чи плутає модель два класи (тобто систематично неправильно прогнозує один як інший).



для вправи, яка фактично була виконана як клас A, що узгоджується з вищою точністю для класу A у звіті про класифікацію.

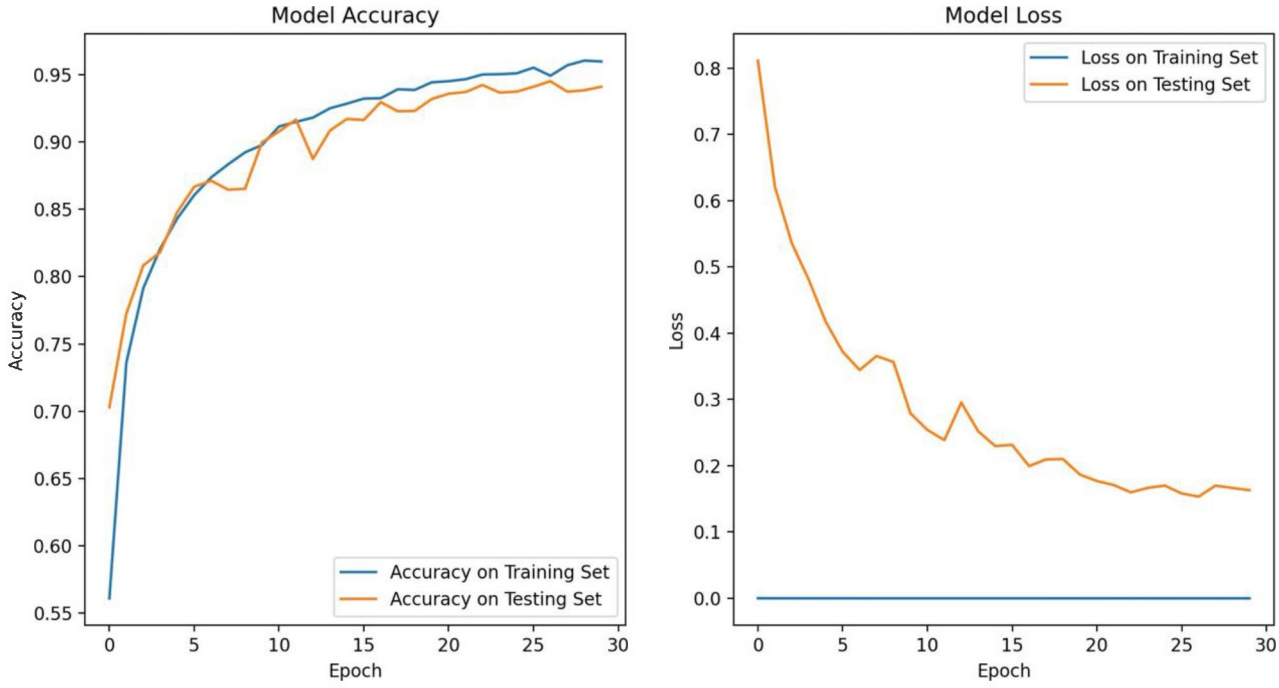


Рисунок 3.8 - Графіки точності та помилки для тестових/навчальних наборів

На графіку точності (рис.3.8) бачимо, що з підвищенням кількості епох, точність моделі на навчальному наборі даних збільшується стабільно і досягає плато, тоді як на тестовому наборі спостерігається плавне нарощування точності, що вказує на ефективність моделі в узагальненні навчених властивостей на нових даних.

На графіку втрат (помилки) ми бачимо, що величина втрат знижується як на навчальному, так і на тестовому наборах даних, що свідчить про покращення продуктивності моделі. На початкових етапах навчання спостерігається стрімке зниження втрат, після чого зниження помилки відбувається повільніше, наближаючись до мінімуму. Значення втрат на тестовому наборі даних відносно вище, ніж на навчальному, що може бути ознакою початкового перенавчання, але з часом різниця між ними зменшується, свідчачи про зростаючу здатність моделі до узагальнення.

Графіки демонструють, що модель добре адаптується до даних, покращуючи свою точність та зменшуючи втрати, і що вона здатна уникнути перенавчання завдяки використанню таких методів, як рання зупинка та налаштування гіперпараметрів.

Ці результати свідчать про те, що коригування, внесені до моделі, призвели до більш надійної моделі, яка краще узагальнює всі класи вправ. Розуміння моделлю даних і її здатність правильно класифікувати вправи покращилися.

Висновки за розділом 3

У третьому розділі висвітлено результати навчання та процес удосконалення моделі аналізу біосигналів. Виявлено, що модель ефективно класифікує фізичні вправи, демонструючи високий рівень точності та запам'ятовування в різних класах. Особливо вражаючими були показники для класу A, де модель виявила високу здатність правильно ідентифікувати вправи. Тим не менш, певні прорахунки в класифікації вказували на необхідність удосконалення.

Для підвищення ефективності моделі були розглянуті та впроваджені різні стратегії, включаючи збільшення складності моделі та впровадження ваг класів для збалансування навчання. Результатом цих змін стало значне покращення загальної продуктивності, зокрема, точності та балансу між різними класами. Це було підтверджено через класифікаційний звіт та матрицю невідповідності, які продемонстрували вищу точність та краще розуміння моделлю всіх класів вправ.А на основі графіків змогли зрозуміти що модель добре адаптується до даних, покращуючи свою точність та зменшуючи втрати, і що вона здатна уникнути перенавчання завдяки використанню таких методів, як рання зупинка та налаштування гіперпараметрів.

Оновлена модель показала, що коригування архітектури та підходів до тренування можуть значно покращити здатність моделі до узагальнення та класифікації. Це вказує на важливість гнучкого підходу до розробки моделей глибокого навчання, де постійна оцінка та удосконалення є ключовими для досягнення високої продуктивності.

ВИСНОВКИ

В цій кваліфікаційній роботі було досягнуто значних результатів у розробці та аналізі комп'ютерної моделі для обробки біосигналів з використанням методів глибокого навчання. Робота мала на меті вирішити складну задачу аналізу біосигналів, що має значний потенціал у медичній діагностиці та моніторингу здоров'я.

Було сформульовано чітку постановку задачі, визначені об'єкт та предмет дослідження, а також конкретна мета. Використання датасету "Weight Lifting Exercises" з Kaggle дозволило отримати цінні дані для тренування та тестування моделі.

Особлива увага була приділена розробці та оптимізації моделі згорткової нейронної мережі (CNN), що включало експерименти з різними архітектурами та параметрами моделі. Було здійснено ряд адаптацій, зокрема налаштування ваг класів, зміну кількості фільтрів та нейронів, а також використання методів ранньої зупинки та збільшення кількості епох для підвищення точності та ефективності моделі.

Детальний аналіз результатів моделі показав її високу точність та надійність. Було продемонстровано, що модель здатна ефективно класифікувати біосигнали, що відкриває шлях для її застосування у важливих областях, таких як медичний моніторинг, спорт-індустрія. Зокрема, модель показала обнадійливі результати у визначенні правильного виконання фізичних вправ, що є важливим для запобігання травмам та підвищення ефективності тренувань.

Ще однією задачею для майбутнього розвитку є створення інтерфейсу для зручного використання і відображення результатів.

Отже в даній кваліфікаційній роботі були виконані усі сформовані задачі та досягнута мета, а саме розпізнавання біосигналів на основі глибокого навчання для класифікації дій людини, яка виконує фізичні вправи на підставі значень датчиків, знятих під час руху.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Awais M., Chiari L., Ihlen E., Helbostad J., Palmerini L. Classical Machine Learning versus Deep Learning for the Older Adults Free-Living Activity Classification. Sensors. 2021. T. 21: 4669. DOI: 10.3390/s21144669

2. Aksyuk S. Weight Lifting Exercise Data Analysis. URL: https://rstudio-pubs-static.s3.amazonaws.com/245269_3531e86e522145f0951c35d87758cccb.html

3. Banerjee P. Dataset from Kaggle 'Weight Lifting Exercises'. URL: https://www.kaggle.com/datasets/prashant111/weight-lifting-exercises/data?select=example_wearablecomputing_weight_lifting_exercises_biceps_curl_variations.csv

4. Cai Y., Academic Editor. Physical Activity Monitoring and Classification Using Machine Learning Techniques. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9332439/ DOI: 10.3390/life12081103.

5. Fischer G., Academic Editor. Detection of Physical Activity Using Machine Learning Methods Based on Continuous Blood Glucose Monitoring and Heart Rate Signals. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9658555/ DOI: 10.3390/s22218568.

6. Gözde Dursun Development of convolutional neural networks for recognition of tenogenic differentiation based on cellular morphology. URL: https://www.sciencedirect.com/science/article/abs/pii/S0169260721003539 (September 2021, 106279)

7. Garg R. Types of Classification Algorithms. URL: https://analyticsindiamag.com/7-types-classification-algorithms/

8. Goodfellow, I. Deep learning / I. Goodfellow, Y. Bengio, A. Courville. – MIT Press, 2016 – 785p

9. K. O'Shea, Nash R. An Introduction to Convolutional Neural Networks — 2015

10. Nielsen, A. M. Neural Networks and Deep Learning / A. M. Nielsen – Determination Press, 2015

11. Rohit Garg Types of Classification Algorithms. URL: https:// analyticsindiamag.com/7-types-classification-algorithms/

12. Samer H. Kumar R. Rowen C. Using Convolutional Neural Networks for Image Recognition, 2015

13. Sensors (Basel). 2022 Nov; 22(21): 8568. Published online 2022 Nov 7. DOI: 10.3390/s22218568. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/ PMC9658555/

14. Suthampan E. Classification in Weight Lifting Exercises Dataset. URL: https://rstudio-pubs-static.s3.amazonaws.com/ 249735_a595d232b36e43f0b67f5de3314b6233.html

15. Velloso E., Bulling A., Gellersen H., Ugulino W., Fuks H. Qualitative Activity Recognition of Weight Lifting Exercises. URL: https://perceptualui.org/ publications/velloso13_ah.pdf

16. Zhou S. Machine Learning for Weight Lifting Exercises Data. URL: http://shinezhou9.github.io/MachineLearningProject1/

17. scikit-learn 1.3.2 Principal component analysis (PCA) URL: https:// scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

ДОДАТКИ

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) **Магістр**
Галузь знань: **12 – Інформаційні технології**
Спеціальність: **123 «Комп'ютерна інженерія»**

ЗАТВЕРДЖУЮ
Завідувач кафедри теоретичної та прикладної системотехніки
_____д.т.н., проф. Шматков С. І.

«08 » грудня 2022 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Кузікова Катерина Михайлівна
(прізвище, ім'я, по батькові студента)

1. Тема роботи «**Комп'ютерна модель аналізу біосигналів на основі глибокого навчання**»
керівник роботи: Бакуменко Ніна Станіславівна кандидат технічних наук, доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від «10» листопада 2023 року № 4101-5/3197

2. Строк подання студентом роботи 28 .11.2023

3. Перелік питань, які потрібно розробити
1)Обґрунтування та вибір методу розпізнавання станів медико-біологічних систем;
2)Вибір та обґрунтування архітектури нейромережевої моделі;
3)Розробка алгоритму попередньої обробки вхідних даних для покращення ефективності методу;
4)Навчання та тестування нейромережевої моделі;
5)Порівняння ефективності різних методів;
6)Перевірка на працездатність та оцінка результатів;
7)Підготовка та оформлення пояснювальної записки;

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Огляд методів машинного навчання для класифікації медико-біологічних даних	08.12.2022
2	Вибір та обґрунтування методу розпізнавання станів медико-біологічних систем	01.01.2023
3	Вибір та обґрунтування архітектури нейромережевої моделі	10.02.2023
4	Розробка алгоритму попередньої обробки вхідних даних для покращення ефективності методу	20.03.2023
5	Навчання та тестування нейромережевої моделі	01.05.2023
6	Опис програмного забезпечення для вирішення задачі	05.07.2023
7	Порівняння ефективності різних методів	10.08.2023
8	Огляд та аналіз отриманих результатів	15.09.2023
9	Підготовка та розробка програмної документації	10.10.2023
10	Розробка та оформлення пояснювальної записки	22.11.2023

5. Дата видачі завдання 06.12.2022р

Студент ___Кузікова К.М.____ _____
ініціали, прізвище підпис

Керівник роботи __Бакуменко Н.С.___ _____
ініціали, прізвище підпис

Додаток Б

Технічне завдання
на розробку програмного виробу

«Комп'ютерна модель аналізу біосигналів на основі глибокого навчання»

Назва розділу	**Назва і зміст підрозділу**
1. Введення	1.1. Назва програмного виробу: комп'ютерна модель аналізу біосигналів на основі глибокого навчання. 1.2. Галузь застосування: медицина, спорт.
2. Підстава для розробки	2.1. Навчальний план за спеціальністю 123 – «Комп'ютерна інженерія» 2.2. Завдання на кваліфікаційну роботу магістра затверджено наказом ректора No 4101-5/3107 від 10.11.2023.
3. Призначення розробки	3.1. Мета розробки програмного виробу: розпізнавання біосигналів на основі глибокого навчання для класифікації дій людини, яка виконує фізичні вправи на підставі значень датчиків, знятих під час руху. 3.2. Призначення програмного виробу: для класифікації дій людини, яка виконує фізичні вправи на підставі значень датчиків, знятих під час руху.

4. Технічні вимоги до програмного виробу	4.1. Вимоги до функціональних характеристик : точність аналізу: висока точність ідентифікації та класифікації біосигналів; швидкість обробки: здатність швидко обробляти великі обсяги даних; адаптивність: гнучкість моделі для адаптації до різних типів біосигналів. 4.2. Вимоги до надійності: запобігання помилкам: мінімізація помилкових позитивних та негативних результатів. 4.3. Вимоги до умов експлуатації: сумісність з різними платформами: модель має бути сумісною з різними середовищами для розробки, такими як Python, TensorFlow та Keras. 4.4. Вимоги до складу і параметрів технічних засобів: обладнання для збору даних: специфікації для датчиків та іншого обладнання, що використовується для збору біосигналів. 4.5. Вимоги до інформаційної та програмної сумісності: інтеграція з іншими системами: здатність інтегруватися з різними медичними та аналітичними системами. 4.6. Вимоги до транспортування і зберігання: умови зберігання: вказівки для зберігання обладнання, яке використовується для збору біосигналів. 4.7. Спеціальні вимоги: конфіденційність даних: Забезпечення захисту конфіденційності оброблюваних біосигналів; відповідність стандартам: дотримання відповідних медичних та технічних стандартів і нормативів.

<table>
<tr><td>5. Вимоги до програмної документації.</td><td colspan="2">Програмною документацією до виробу «комп'ютерна модель аналізу біосигналів на основі глибокого навчання» вважати:
1) Справжнє Технічне завдання на розробку програмного виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи).
2) Програму і методику випробувань розробленого програмного виробу (представити у вигляді Додатку В до пояснювальної записки до кваліфікаційної роботи).
3) Опис програмного виробу (представити в розділі 2 пояснювальної записки до кваліфікаційної роботи).
4) Текст програми (представити в Додатку Г до пояснювальної записки до кваліфікаційної роботи).</td></tr>
<tr><td>6. Техніко-економічні показники</td><td colspan="2">Орієнтовна оцінка ефективності та якості виконуваного аналізу: точність (accuracy) повинна бути вище 90%; модель повинна правильно передбачати класи для вправи.</td></tr>
<tr><td rowspan="8">7. Стадії і етапи розробки</td><td>Огляд методів машинного навчання для класифікації медико-біологічних даних</td><td>08.12.2022</td></tr>
<tr><td>Вибір та обґрунтування методу розпізнавання станів медико-біологічних систем</td><td>01.01.2023</td></tr>
<tr><td>Вибір та обґрунтування архітектури нейромережевої моделі</td><td>10.02.2023</td></tr>
<tr><td>Розробка алгоритму попередньої обробки вхідних даних для покращення ефективності методу</td><td>20.03.2023</td></tr>
<tr><td>Навчання та тестування нейромережевої моделі</td><td>01.05.2023</td></tr>
<tr><td>Опис програмного забезпечення для вирішення задачі</td><td>05.07.2023</td></tr>
<tr><td>Порівняння ефективності різних методів</td><td>10.08.2023</td></tr>
</table>

	Огляд та аналіз отриманих результатів	15.09.2023
	Підготовка та розробка програмної документації	10.10.2023
	Розробка та оформлення пояснювальної записки	22.11.2023
8. Порядок контролю і приймання	В даному розділі повинні бути вказані загальні вимоги до приймання розробленого програмного виробу наприклад: 1) Перевірка ходу розробки програмного виробу. Керівнику робіт виконувати 1 раз в 3 тижні. 2) Випробування програмного виробу відповідно до Програми і методики випробувань провести на базі комп'ютерного класу. 3) Захист розробленого програмного виробу провести на засіданні атестаційної комісії. 4) Пояснювальну записку надати на паперових носіях в одному примірнику, в електронному вигляді - на CD-диску в одному екземплярі.	

Виконавець:
студентка групи КІ-61
Кузікова Катерина Михайлівна

_____

Замовник:
кандидат техн. наук, доцент
Бакуменко Ніна Станіславівна

_____

Додаток В

Програма і методика випробувань
програмного виробу

«Комп'ютерна модель аналізу
біосигналів на основі глибокого
навчання»

1 Об'єкт випробувань

1. Назва програмного виробу : «Комп'ютерна модель аналізу біосигналів на основі глибокого навчання»
2. Галузь застосування : спорт-індустрія, реабілітація військових
3. Перераховані відомості запозичуються з відповідних розділів Технічного завдання.

2. Мета випробувань

Перевірка відповідності функціональності програмної реалізації системи заявленим функціональним можливостям в технічному завданні (Додаток Б до пояснювальної записки до кваліфікаційої роботи).

3. Загальні положення

1. Підстави для проведення випробувань

Підставою для проведення випробувань є наказ про призначення атестаційної комісії.

2. Місце і тривалість випробувань

Приймальні (приймально-здавальні) випробування проводяться на базі комп'ютерного класу кафедри в період роботи атестаційної комісії.

3. Обсяг випробувань

Приймальні випробування програмного вироби проводяться в обсязі відповідному цієї програми і методики випробувань.

4. Організації, які беруть участь у випробуваннях

Приймальні випробування проводяться атестаційною комісією напередодні засідання (або в процесі засідання) за участю Замовника, Виконавця та інших осіб, присутніх на засіданні.

4. Вимоги до програми або програмного виробу

Модель повинна задовольняти наступним вимогам:

1. працювати на основних операційних системах: Windows, Linux, MacOS;
2. висока точність ідентифікації та класифікації біосигналів;
3. швидкість обробки: здатність швидко обробляти великі обсяги даних;
4. адаптивність: гнучкість моделі для адаптації до різних типів біосигналів.
5. вимоги до маркування та упаковки (не висуваються);
6. вимоги до транспортування і зберігання (не висуваються).

Спеціальні вимоги (не пред'являються).

5. Вимоги до програмної документації

Програмною документацією до виробу «комп'ютерна модель аналізу біосигналів на основі глибокого навчання» вважати:

1) Справжнє Технічне завдання на розробку програмного виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи).

2) Програму і методику випробувань розробленого програмного виробу (представити у вигляді Додатку В до пояснювальної записки до кваліфікаційної роботи).

3) Опис програмного виробу (представити в розділі 2 пояснювальної записки до кваліфікаційної роботи).

4) Текст програми (представити в Додатку Г до пояснювальної записки до кваліфікаційної роботи).

6. Засоби і порядок випробувань

6.1 Засоби випробувань

Для проведення випробувань необхідна середа розробки яка підтримує мову Python та встановлені відповідні бібліотеки.

6.2 Порядок проведення випробувань

Як правило, випробування проводяться в два етапи:

-ознайомчий (1-й етап);

-випробування програмного виробу (2-й етап).

Перелік перевірок, що проводяться на 1 етапі випробувань, включає в себе:

1. Перевірку комплектності програмної документації.
2. Перевірка комплектності складу програмної документації здійснюється за критерієм наявності зазначеної в ТЗ документації.
3. Перевірку комплектності складу технічних і програмних засобів.
4. Методику проведення перевірок на 1 етапі випробувань.
5. Якість програмної документації перевіряється на відповідність вимогам стандартів ЕСПД.

Перелік перевірок, що проводяться на 2 етапі випробувань, включає в себе:

1. перевірку відповідності технічних характеристик програми вимогам технічного завдання;
2. перевірку ступеня виконання функціональних вимог до програми;
3. методику проведення перевірок, що входять до переліку по 2 етапу випробувань.

1. Програма працює відповідно до умов експлуатації операційних систем MS Windows, Linux та MacOS.
2. Для роботи необхідний компілятор мови програмування python, версії не нижчої ніж 3.0, файл програми, та датасет у форматі .csv .

3. Порядок проведення випробувань:
   3.1. Запуск програми здійснюється за допомогою командного рядка або терміналу;
   3.2. Перейти до директорії, де зберігається Python файл. Це можна зробити за допомогою команди **cd**.
   3.3. Ввести наступну команду для запуску програми: “python ім’я_файлу.py”.
   3.4. Після появи графіку співвідношення класів, треба його закрити, щоб програма продовжила своє виконання.

Для проведення випробувань пропонується тест 1.

Тест 1

1. Перевірка виконання програми;
2. Завантаження даних з датасету;
3. Отримання результатів про успішну роботу моделі у вигляді зображень, класифікаційного звіту та результату в консолі.

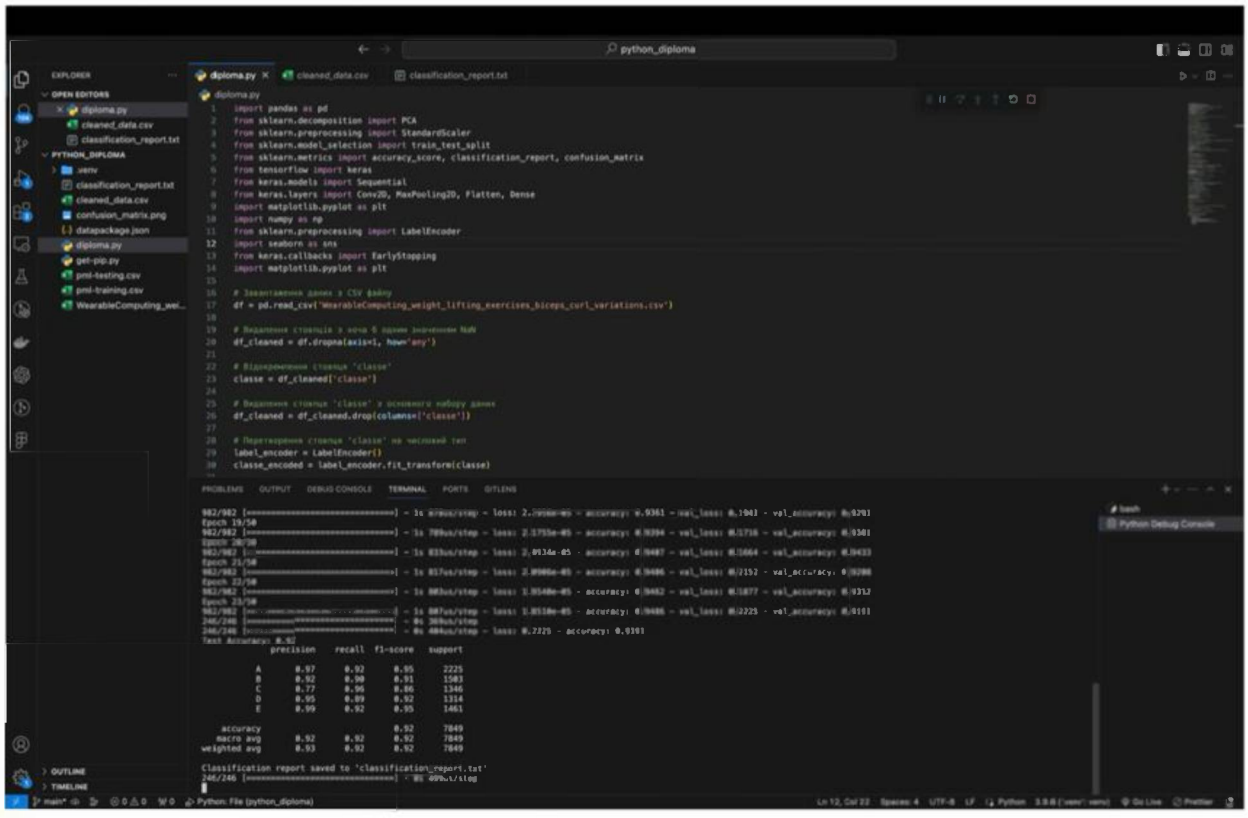


Рис. В.1 Тест 1

Тест вважається пройденим, якщо відбуваються вказані операції і їх відображення у програмному продукті.

Висновки: тест 1 успішно пройшов випробування показавши точність вище 90%. Випробування пройшло успішно.

Виконавець: студентка групи КІ-61, Кузікова К.М.

Додаток Г

Код програми:

```
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import LabelEncoder
import seaborn as sns
from keras.callbacks import EarlyStopping

# Завантаження даних з CSV файлу
df =
pd.read_csv('WearableComputing_weight_lifting_exercises_biceps_cur
l_variations.csv')

# Видалення стовпців з хоча б одним значенням NaN
df_cleaned = df.dropna(axis=1, how='any')

# Відокремлення стовпця 'classe'
classe = df_cleaned['classe']

# Видалення стовпця 'classe' з основного набору даних
df_cleaned = df_cleaned.drop(columns=['classe'])

# Перетворення стовпця 'classe' на числовий тип
label_encoder = LabelEncoder()
classe_encoded = label_encoder.fit_transform(classe)

# Видалення стовпців з текстовими значеннями
df_cleaned = df_cleaned.select_dtypes(exclude=['object'])

# Стандартизація залишених числових даних
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df_cleaned)

# Стандартизація залишених числових даних
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df_cleaned)

# Конвертація стандартизованих даних назад у DataFrame
df_scaled_df = pd.DataFrame(df_scaled, columns=df_cleaned.columns)

# Збереження очищених даних у новому CSV файлі
```

```
df_scaled_df.to_csv('cleaned_data.csv', index=False)


# Розділення даних на навчальні та тестові набори
X_train, X_test, y_train, y_test = train_test_split(df_scaled,
classe_encoded, test_size=0.2, random_state=42)

# Використовуйте метод головних компонент
pca = PCA(n_components=16)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)

# Додавання стовпця 'classe' назад до DataFrame
df_cleaned['classe'] = classe

# Перетворення даних у формат зображень 4x4
X_train_images = X_train_pca.reshape(-1, 4, 4, 1)
X_test_images = X_test_pca.reshape(-1, 4, 4, 1)

# Створення моделі CNN
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(4, 4,
1)))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(5, activation='softmax'))  # Вихідний шар з 5
класами

# Компіляція моделі
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Тренування моделі
history = model.fit(X_train_images, y_train, epochs=10,
validation_data=(X_test_images, y_test))

# Візуалізація залежності помилки від кількості епох
plt.figure(figsize=(12, 6))

# Побудова графіка точності
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Accuracy on Training
Set')
plt.plot(history.history['val_accuracy'], label='Accuracy on
Validation Set')
plt.title('Accuracy over Epochs')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
```

```
# Побудова графіка втрат
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Loss on Training Set')
plt.plot(history.history['val_loss'], label='Loss on Validation
Set')
plt.title('Loss over Epochs')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()

# Передбачення на тестовому наборі даних
y_pred = model.predict(X_test_images)
y_pred_classes = np.argmax(y_pred, axis=1)

# Оцінка точності моделі на тестовому наборі даних
accuracy = model.evaluate(X_test_images, y_test)[1]
print(f'Test Accuracy: {accuracy:.2f}')


# Обчислити матрицю невідповідності
confusion = confusion_matrix(y_pred_classes, y_test)

# Побудувати матрицю невідповідності
plt.figure(figsize=(8, 6))
sns.heatmap(confusion, annot=True, fmt="d", cmap="Blues")
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')

# Зберегти зображення матриці невідповідності як зображення
plt.savefig("confusion_matrix.png")

# Закрити графік для звільнення ресурсів
plt.close()

# Оцінка моделі
report = classification_report(y_test, y_pred_classes,
target_names=label_encoder.classes_)
print(report)

# Зберегти звіт класифікації у текстовому файл
with open("classification_report.txt", "w") as text_file:
    text_file.write(report)

# Вивести повідомлення, що звіт збережено
print("Classification report saved to
'classification_report.txt'")


# Візуалізація зображень та їх передбачень
```

```
predictions = model.predict(X_test_images)
class_names = ['A', 'B', 'C', 'D', 'E']

for i in range(len(y_test)):
    if not np.isnan(y_test[i]):
        plt.figure()
        plt.imshow(X_test_images[i].reshape(4, 4), cmap='gray')
        actual_class = class_names[int(y_test[i])]
        predicted_class = class_names[y_pred_classes[i]]
        plt.title(f'Actual: {actual_class}, Predicted:
{predicted_class}')
        plt.axis('off')
        plt.show()
        plt.close()
```

Код програми після вдосконалення моделі:

```
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import LabelEncoder
import seaborn as sns
from keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt

# Завантаження даних з CSV файлу
df =
pd.read_csv('WearableComputing_weight_lifting_exercises_biceps_cur
l_variations.csv')

# Видалення стовпців з хоча б одним значенням NaN
df_cleaned = df.dropna(axis=1, how='any')

# Відокремлення стовпця 'classe'
classe = df_cleaned['classe']

# Видалення стовпця 'classe' з основного набору даних
df_cleaned = df_cleaned.drop(columns=['classe'])

# Перетворення стовпця 'classe' на числовий тип
label_encoder = LabelEncoder()
classe_encoded = label_encoder.fit_transform(classe)
```

```
# Видалення стовпців з текстовими значеннями
df_cleaned = df_cleaned.select_dtypes(exclude=['object'])

# Стандартизація залишених числових даних
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df_cleaned)

# Конвертація стандартизованих даних назад у DataFrame
df_scaled_df = pd.DataFrame(df_scaled, columns=df_cleaned.columns)

# Збереження очищених даних у новому CSV файлі
df_scaled_df.to_csv('cleaned_data.csv', index=False)

# Розділення даних на навчальні та тестові набори
X_train, X_test, y_train, y_test = train_test_split(df_scaled,
classe_encoded, test_size=0.2, random_state=42)

# Обчислення кількості зразків у кожному класі
unique, counts = np.unique(y_train, return_counts=True)
class_counts = dict(zip(unique, counts))

# Візуалізація пропорцій класів
plt.figure(figsize=(10, 6))
plt.bar(class_counts.keys(), class_counts.values())
plt.xlabel('Classes')
plt.ylabel('Number of samples')
plt.title('Proportions of Classes in the Training Set')
plt.xticks(list(class_counts.keys()),
label_encoder.inverse_transform(list(class_counts.keys())))
plt.show()
plt.close()

# Використання метод головних компонент
pca = PCA(n_components=16)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)

# Додавання стовпця 'classe' назад до DataFrame
df_cleaned['classe'] = classe

# Перетворення даних у формат зображень 4x4
X_train_images = X_train_pca.reshape(-1, 4, 4, 1)
X_test_images = X_test_pca.reshape(-1, 4, 4, 1)

counts = np.bincount(classe_encoded)

# Додавання ваги класів для обробки незбалансованих наборів даних
class_weights = {i: 1.0/counts[i] for i in
range(len(np.unique(classe_encoded)))}
counts = np.bincount(classe_encoded)
class_weights = {i: 1.0/count for i, count in enumerate(counts)}
```

```
# Налаштування складності моделі, додавши більше шарів і нейронів
model = Sequential()
model.add(Conv2D(64, (3, 3), activation='relu', input_shape=(4, 4,
1)))  # Increased filters
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))  # Increased neurons
model.add(Dense(5, activation='softmax'))

# Рання зупинка
early_stopping = EarlyStopping(monitor='val_loss', patience=3)

# Компіляція моделі
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Тренування моделі та рання зупинка
history = model.fit(X_train_images, y_train, epochs=50,
validation_data=(X_test_images, y_test),
                    class_weight=class_weights,
callbacks=[early_stopping])

# Візуалізація залежності помилки від кількості епох
plt.figure(figsize=(12, 6))

# Побудова графіка точності
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Accuracy on Training
Set')
plt.plot(history.history['val_accuracy'], label='Accuracy on
Testing Set')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

# Побудова графіка втрат
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Loss on Training Set')
plt.plot(history.history['val_loss'], label='Loss on Testing Set')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.show()

# Передбачення на тестовому наборі даних
y_pred = model.predict(X_test_images)
y_pred_classes = np.argmax(y_pred, axis=1)
```

```
# Оцінка точності моделі на тестовому наборі даних
accuracy = model.evaluate(X_test_images, y_test)[1]
print(f'Test Accuracy: {accuracy:.2f}')

# Обчислення матриці невідповідності
confusion = confusion_matrix(y_pred_classes, y_test)

# Побудова матриці невідповідності
plt.figure(figsize=(8, 6))
sns.heatmap(confusion, annot=True, fmt="d", cmap="Blues")
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')

# Збереження зображення матриці невідповідності як зображення
plt.savefig("confusion_matrix.png")

# Закривання графіку для звільнення ресурсів
plt.close()

# Оцінка моделі
report = classification_report(y_test, y_pred_classes,
target_names=label_encoder.classes_)
print(report)

# Збереження звіту класифікації у текстовому файл
with open("classification_report.txt", "w") as text_file:
    text_file.write(report)

print("Classification report saved to
'classification_report.txt'")

# Візуалізація зображень та їх передбачень
predictions = model.predict(X_test_images)
class_names = ['A', 'B', 'C', 'D', 'E']

for i in range(len(y_test)):
    if not np.isnan(y_test[i]):
        plt.figure()
        plt.imshow(X_test_images[i].reshape(4, 4), cmap='Oranges')
        actual_class = class_names[int(y_test[i])]
        predicted_class = class_names[y_pred_classes[i]]
        plt.title(f'Actual: {actual_class}, Predicted:
{predicted_class}')
        plt.axis('off')
        plt.show()
        plt.close()
```