МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних технологій

# МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: Розробка штучної нейронної мережі для визначення емоції людини за фотографією

Виконав студент 2 курсу групи МІС-21
спеціальності 122 Комп'ютерні науки
Перчун Микола Михайлович

Керівник к.т.н., доцент
Фразе-Фразенко Олексій Олексійович

Рецензент к.т.н., начальник ЦІТ
ОНЕУ, Домаскін О.М.

Одеса 2022

# АНОТАЦІЯ

до кваліфікаційної магістерської роботи

за темою

«Розробка штучної нейронної мережі

для визначення емоції людини за фотографією»

студента групи МІС21

Перчуна Миколи Михайловича

Штучні нейронних мережі були винайдені, щоб полегшити людям життя, заощадити їх час, а також автоматизувати процеси таким чином, щоб участь людини була мінімальна і спрощена наскільки це можливо. Штучна нейронна мережа яка вміє визначати емоцію людини за фотографією легко та швидко можна інтегрувати у інші системи, наприклад системи відеоспостереження.

Мета і задачі дослідження.

Створити штучну нейронну мережу, яка здатна з великою вірогідністю визначати емоцію людини за допомогою фотографії, бути розширюваною та працювати при великих навантаженнях.

Для тренування штучної нейронної мережі було використано дата сет FER-2013, який знаходиться в публічному доступі та складається з 28709 фотографій та має 7 категорій емоцій:

- злість;
- огида;
- страх;
- радість;
- нейтральна;
- сум;

- здивування.

У кваліфікаційній магістерській роботі було використано нову версію .NET 6, у поєднанні з останньою версією бібліотеки ML.NET, що забезпечує стабільну та швидку роботу продукту. А також набір інструментів Swagger, що в свою чергу дозволяє легко інтегрувати її у інші програми.

Враховуючи досить велику точність визначення штучною нейронною мережею емоції людини за допомогою фотографії її було б корисно використовувати в охоронних системах з інтеграцію у системи відеоспостереження особливо при великій кількості людей, наприклад банки.

Структура та обсяг роботи:

- повний обсяг сторінок пояснювальної записки – 76;
- кількість рисунків – 39;
- кількість таблиць – 0;
- кількість посилань – 17.

ANN – Artificial Neural Network

API – Application Programming Interface

ASP.NET – технологія створення веб-застосунків і веб-сервісів

.NET 6 – модульна платформа для розробки програмного забезпечення з відкритим кодом

ML.NET – бібліотека машинного навчання програмного забезпечення для мов програмування C#

Swagger – набір інструментів для розробників API від SmartBear Software

**SUMMARY**

of the qualifying master's thesis

on the topic

"Development of an artificial neural network for

recognizing human emotions by image"

by Mykola Perchun,

a student of the MIS21 group.

Artificial neural networks were invented to make people's lives easier, save their time, and automate processes in such a way that human involvement is minimal and simplified as much as possible. An artificial neural network capable of recognizing human emotions by image can be easily and quickly integrated into other systems, such as video surveillance systems.

The purpose and objectives of the study.

To create an artificial neural network that is able to determine human emotion with a high probability using a photo, be expandable and work under heavy loads.

The FER-2013 data set, which is publicly available and consists of 28,709 photos and has 7 categories of emotions, was used for training the artificial neural network:

- angry;
- disgust;
- fear;
- happy;
- neutral;
- sad;
- surprise.

The qualifying master's thesis used the new version of .NET 6, combined with the latest version of the ML.NET library, which ensures stable and fast operation of the

product. As well as a set of Swagger tools, which in turn allows you to easily integrate it into other programs.

Given the fairly high accuracy of the artificial neural network's determination of human emotion using a photograph, it would be useful to use it in security systems with integration into video surveillance systems, especially when there are a large number of people, such as banks.

Structure and scope of work:

- total count pages of the explanatory note - 76;
- number of drawings - 39;
- number of tables - 0;
- number of links - 17.

ANN – Artificial Neural Network

API – Application Programming Interface

ASP.NET – technology for creating web applications and web services

.NET 6 – modular platform for developing open source software

ML.NET – machine learning software library for C# programming languages

Swagger – toolkit for API developers from SmartBear Software

# ЗМІСТ

## СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ANN – Artificial Neural Network

API – Application Programming Interface

AOT – метод компіляції перед виконанням

ADO.NET – набір бібліотек, що поставляється з Microsoft .NET Framework

ASP.NET – технологія створення веб-застосунків і веб-сервісів

CLI – інтерфейс командного рядка

CMS – Content Management System

CSS – Cascading Style Sheets

GIT – система контролю версіями

HTML – HyperText Markup Language

HTTP – Hyper Text Transfer Protocol

IDE – інтегрована среда розробки

JS – Java Script

PGO – техніка оптимізації програми компілятором

PHP – Personal Home Page Tools

LTS – довгострокова підтримка програмного продукту

UI – інтерфейс користувача

URL – Uniform Resource Locator

SDK – набір для розробки програмного забезпечення

XML – Extensible Markup Language

VS – інтегроване середовище розробки програмного забезпечення та низка інших інструментальних засобів від фірми Майкрософт

# ВСТУП

У наш час ми все частіше чуємо про штучні нейронні мережі, а саме що вони все частіше інтегруються у веб-додатки чи використовуються у різних сервісах пошуку і т. п. Штучні нейронних мережі були винайдені, щоб полегшити людям життя, заощадити їх час, а також автоматизувати процеси таким чином, щоб участь людини була мінімальна і спрощена наскільки це можливо. Штучна нейронна мережа яка вміє визначати емоцію людини за фотографією могла би полегшити роботу працівників охоронних об’єктів (наприклад банків) з великим скупченням людей, коли через камери услідкувати за усіма неможливо, та завчасно визначивши загрозу можна було би запобігти скоєнню злочинів. Або її можна було б використати психологам під час консультацій з людьми, щоб в свою чергу полегшило їх роботу, бо емоції людини автоматично розпізнавалися під час бесіди та запитань між ними, варіантів де її можна залучити безліч.

Але нажаль немає одного якісного та зручного веб-додатку для реалізації цієї задачі. Існують деякі аналоги вже натренерованих штучних нейронних мереж які здатні визначати емоцію людини за фотографією, але вони російського походження та не розширювані, або не мають зручного інтерфейсу для розробників та користувача продукту.

Мета кваліфікаційної роботи створення програмного продукту для швидкого та зручного визначення емоції людини з високою ймовірністю шляхом отримання фотографії обличчя людини.

Для виконання кваліфікаційної роботи необхідно створити програмний продукт який за допомогою штучної нейронної мережі матиме змогу з достатньо великою вірогідністю визначати базові емоції людини за фотографією. Інтерфейс веб-додатку повинен бути зручним у використанні та інтуітивно-зрозумілим. Створити можливість користувачам розширювати датасет власноруч для

покращення якості роботи штучної нейронної мережі. Також додаток повинен бути стабільним у роботі та витримувати велику кількість запитів до сервісу.

Загальні характеристики кваліфікаційної роботи:

- повний обсяг сторінок пояснювальної записки – 76;
- кількість рисунків – 39;
- кількість таблиць – 0;
- кількість посилань – 17.

# 1 ШТУЧНА НЕЙРОННА МЕРЕЖА ЇЇ ВИДИ ТА ТЕХНОЛОГІЇ СТВОРЕННЯ

В цьому розділі надаються загальні положення, щодо методів розробки сучасних штучних нейронних мереж, їх відмінності, а також їх технології створення.

## 1.1 Поняття штучна нейронна мережа

Штучна нейронна мережа (ANN - artificial neural network) є обчислювальною архітектурою для обробки складних даних за допомогою безлічі пов'язаних між собою процесорів і обчислювальних шляхів. Штучні нейронні мережі, створені за аналогією з людським мозком, здатні навчатися та аналізувати великі та складні набори даних, які за допомогою більш лінійних алгоритмів обробити вкрай складно [1].

Традиційний цифровий комп'ютер здатний успішно вирішувати багато різних завдань. Він робить це досить швидко і точно відповідно до вказівок користувача. На жаль, він безсилий у ситуаціях, коли сам користувач не до кінця розуміє проблему, яку він має вирішити. Гірше того, стандартні алгоритми не можуть працювати із «зашумленими» чи неповними даними, при тому що в реальному житті найчастіше доводиться мати справу саме з такою інформацією. Тут може допомогти штучна нейронна мережа - здатна до навчання обчислювальна система.

Перша штучна нейронна мережа була створена в 1958 психологом Френком Розенблатом. Вона отримала назву Perceptron і була призначена для моделювання діяльності мозку людини при обробці візуальних даних та навчанні

розпізнавання об'єктів. Згодом було розроблено аналогічні штучні нейронні мережі з метою вивчення процесу пізнання.

Згодом стало зрозуміло, що, крім аналізу діяльності мозку людини, вони можуть виконувати й інші дуже корисні функції. Завдяки здатності встановлювати відповідності шаблонам і навчатися ці мережі знайшли застосування для аналізу багатьох проблем, які дуже складно чи неможливо вирішити за допомогою традиційних обчислювальних чи статистичних методів. Наприкінці 80-х почалося активне використання штучних нейронних мереж у різних цілях.

Штучні нейронні мережі часто для простоти називають нейронними мережами, проте це вираз відноситься до головного мозку біологічних істот, діяльність якого мережі ANN спочатку були покликані моделювати.

Принцип дії штучної нейронної мережі полягає у формуванні зв'язків між безліччю різних обробних елементів, кожен з яких є аналогом одного нейрона в головному мозку біологічної істоти. Нейрони можуть бути відтворені фізично або змодельовані цифровим комп'ютером. Кожен нейрон отримує безліч вхідних сигналів, а потім з урахуванням внутрішньої системи вагових коефіцієнтів породжує один вихідний сигнал, який, як правило, служить вхідним для іншого нейрона.

Нейрони тісно взаємопов'язані один з одним і організовані на кілька різних рівнів. Вхідний рівень отримує вхідні дані, а вихідний породжує кінцевий результат. Зазвичай між цими двома рівнями є один або кілька прихованих рівнів. У такій структурі неможливо передбачити або точно дізнатися, як передаються дані.

Спочатку для штучних нейронних мереж, як правило, створюється система випадковим чином призначених вагових коефіцієнтів. Це означає, що мережі нічого не знають, і для вирішення конкретної проблеми їх потрібно навчити.

Взагалі кажучи, існують два методи навчання, які застосовуються залежно від того, для вирішення якої проблеми призначена мережа.

Самоорганізована штучна нейронна мережа (її часто називають мережею Кохонена на ім'я творця) розрахована на обробку великих обсягів даних і повинна знаходити закономірності та визначати взаємозв'язки між ними. Вчені часто застосовують такі мережі для аналізу експериментальних даних.

Мережа із зворотним поширенням помилки, навпаки, навчається людиною до виконання конкретних завдань. Під час навчання людина оцінює, чи коректний результат, отриманий штучною нейронною мережею. Якщо він коректний, збільшуються вагові коефіцієнти, які використовувалися при його отриманні. Якщо результат некоректний, ці вагові коефіцієнти зменшуються. Мережі такого типу часто застосовуються для вивчення процесу пізнання та додатків, що вирішують конкретні завдання.

Штучна нейронна мережа, створена з урахуванням одного комп'ютера, зазвичай, працює повільніше, ніж традиційне алгоритмічне рішення. Однак паралельна природа такого обчислювального середовища дає можливість використовувати для обробки кілька процесорів і отримувати значно більшу швидкість при дуже низьких витратах на розробку. Паралельна архітектура також дозволяє ефективно обробляти великі обсяги даних. Працюючи з великими безперервними потоками інформації, як у разі розпізнавання промови чи обробки даних, які з машинних датчиків, штучні нейронні мережі можуть діяти значно швидше, ніж їх лінійні «конкуренти».

Штучні нейронні мережі виявляються корисними в різних додатках, де використовуються складні і часто неповні дані. Так, вони застосовуються для розпізнавання образів та мовлення. Крім того, штучні нейронні мережі використовуються і в останніх версіях програм, що виконують перетворення текстів на мовлення. На них базуються, зокрема, багато програм аналізу

рукописних текстів (наприклад, що застосовуються в популярних кишенькових комп'ютерах).

Моніторинг автоматизованого виробництва також виконується за допомогою штучних нейронних мереж, які контролюють роботу машин, стежать за температурним режимом, діагностують несправності тощо. Ці штучні нейронні мережі розширять можливості працівників або дозволять частково замінювати досвідчених робітників, завдяки чому зусиллями меншої кількості співробітників можна виконувати більший обсяг робіт.

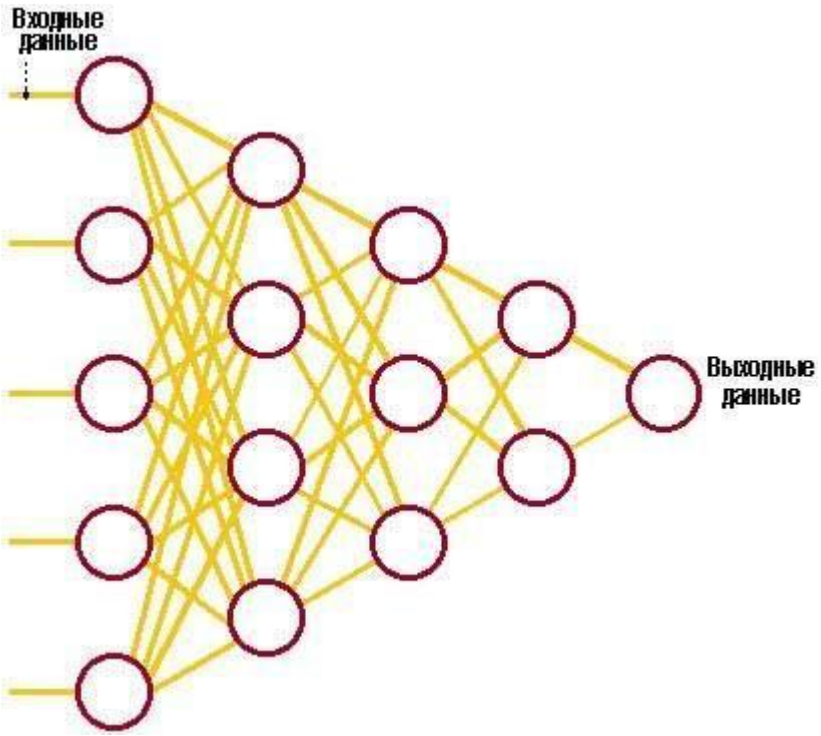


Рисунок 1 – Ієрархічна нейронна мережа

Тут наведено один із можливих способів об'єднання процесорів у нейронну мережу. У цій багаторівневій ієрархії кожен процесор передає свій результат усім процесорам наступного рівня. Як наслідок у такій структурі неможливо

визначити, яким був шлях обробки даних, який дозволив отримати конкретний результат.

### 1.2 Переваги використання штучних нейронних мереж

Нині існує величезна різноманітність систем, основу яких лежать методи, використовують елементи штучного інтелекту. Продукція, заснована на використанні тих чи інших методів штучного інтелекту, так міцно увійшла в наше повсякденне життя, що практично будь-яка сучасна людина вже не представляє комфортного існування без неї, а тим часом далеко не кожен навіть замислюється, як працює, той чи інший пристрій, програма.

Звичайно, це тільки вершина айсберга, наскільки сучасна людина залежить від інтелектуальних систем. Варто розуміти, що застосування методів ІІ зовсім не обмежується тими прикладами, що були наведені раніше, величезний інтерес становить використання цих методів у промислових системах, енергетиці та багато іншого [2].

Зупинимося докладніше на особливостях застосування нейронних мереж у сучасних системах автоматичного керування. Вимоги до якості управління, що забезпечується регуляторами, постійно зростають, цей процес визначається, насамперед, економічною необхідністю:

- збільшуються ціни на енергетичні та матеріальні ресурси, що ставить перед виробниками питання ефективного їх використання, економії;
- найточніша робота регулятора, як відомо дозволить якісніше здійснювати процес управління випуском продукції, що обов'язково позначиться її якості і відповідно конкурентоспроможності.

Як видно з наведених вище вимог у сучасних умовах жорсткої конкуренції підприємствам виробникам просто необхідно забезпечувати якісне управління

технологічними процесами. Зауважимо, що крім щодо безболісного перевитрати ресурсів можуть виникати й серйозніші проблеми, наприклад підвищений знос устаткування з наступним виходом його з ладу тощо.

Розглянемо докладніше переваги нейронних мереж, які дозволять підняти новий рівень системи автоматичного управління.

В процесі своєї роботи об'єкт управління зазнає різних змін, які можуть бути непомітні людському оку, але їх вплив може значно позначатися на якості управління, іншими словами відбувається деяке зношування обладнання і природно зміна його параметрів. Стає очевидно, що налаштований один раз на оптимальну роботу звичайний ПІД-регулятор через деякий час не зможе забезпечити нам бажаної якості управління. Проблема здається легко вирішуваною, адже фахівцю в даній предметній галузі в принципі нічого не варто зробити перерахунок параметрів регулятора, тобто внести деякі необхідні зміни до його роботи, тим самим повернувши керування до оптимального. При більш детальному розгляді стає ясно, що такий підхід хоч і може дати певні результати, але є щонайменше неефективним. Випадок, коли за кожним регулятором стежить певний фахівець, до обов'язків якого входить постійний перерахунок його параметрів дуже малоймовірний у сучасних умовах виробництва, оскільки на підприємствах зараз одночасно можуть працювати сотні і навіть тисячі автоматичних регуляторів. Уявіть скільки людських ресурсів може знадобитися за такої організації роботи.

Величезна кількість автоматичних регуляторів, які постійно вимагають до себе уваги, може бути не єдиною проблемою. Ще одним головним болем є розділеність сучасних виробництв, тобто. частини вашого підприємства можуть бути один від одного на значній відстані і тоді проблема налаштування параметрів систем управління стає просто непереборною.

Вирішенням проблеми описаної вище може бути тільки пристрій, що самоналаштовується, реагує автоматично на зміни, що відбуваються з об'єктом. Прикладом таких пристроїв може стати регулятор на основі нейромережі. Перед тим як говорити про використання нейромереж коротко пояснимо, що таке штучний, природний нейрони, нейромережі.

Нейрон – це нервова клітина, він є особливою біологічною клітиною, яка обробляє інформацію. Вона складається з тіла клітини або соми, і двох типів зовнішніх деревоподібних гілок: аксона та дендритів. Тіло клітини включає ядро, що містить інформацію про спадкові властивості, і плазму, що має молекулярні засоби для виробництва необхідних нейрону матеріалів. Нейрон отримує сигнали від інших нейронів через дендрити і передає сигнали, згенеровані тілом клітини, вздовж аксона, який наприкінці розгалужується на волокна. На закінчення цих волокон знаходяться синапси. Синапс є елементарним структурним та функціональним вузлом між двома нейронами. Коли імпульс досягає синаптичного закінчення, вивільняються певні хімічні речовини, які називаються нейротрансмітерами.

Нейрони взаємодіють за допомогою короткої серії імпульсів, як правило, тривалістю кілька мілісекунд. Повідомлення передається за допомогою частотно-імпульсної модуляції. Частота може змінюватися від кількох одиниць до сотень герц, що в мільйон разів повільніше, ніж швидкодіючі перемикальні електронні схеми. Проте складні рішення щодо сприйняття інформації, як, наприклад, розпізнавання особи, людина приймає за кілька сотень мілісекунд. Точність та якість, з якою людина виконує розпізнавання інформації не під силу навіть сучасним високопродуктивним комп'ютерам.

Вище було розглянуто особливості роботи біологічного нейрона, перейдемо тепер до його штучного аналогу. Штучний нейрон є деякою подобою природного нейрона, він складається з певної кількості входів, на які надходять

сигнали від інших нейронів, має лінії зв'язку з відповідними їм вагами і підсумовуючий блок, який відповідає тілу біологічного нейрона.

Не можна говорити, що штучний і природний нейрони навчаються і функціонують однаково, оскільки всі спроби створення штучних нейронів, які є копією справжніх, зазнали провалу внаслідок поганої поінформованості про те, як функціонує людський мозок. Справді, робота штучних мереж чимось схожа на роботу людського мозку, але на цьому подібність у більшості випадків закінчується.

Для більш повного розуміння, що таке штучний нейрон, розглянемо малюнок, на якому він зображений. З малюнка видно, що штучний нейрон має N входів, які одночасно є виходами інших нейронів, вагові коефіцієнти зв'язків, що відповідають синоптичним зв'язкам природного нейрона, підсумовуючий блок.

У процесі роботи на входи нейрона надходить безліч сигналів X={X1,X2,X3,…Xn}, кожен вхідний сигнал Xn множиться на відповідний йому ваговий коефіцієнт Wn, зрештою підсумовуючий блок виробляє підсумування алгебри, створюючи вихідний сигнал, який ми будемо називати NET.

У векторних позначеннях це може бути компактно записано так: NEW = XW. Таким чином, видно, що штучний нейрон має досить просту розуміння структуру, проте навіть один нейрон здатний виконувати корисну роботу. Далі в процесі роботи сигнал NET обробляється активаційною функцією, яка в залежності від свого виду здійснює перетворення вхідного сигналу до певного виду.

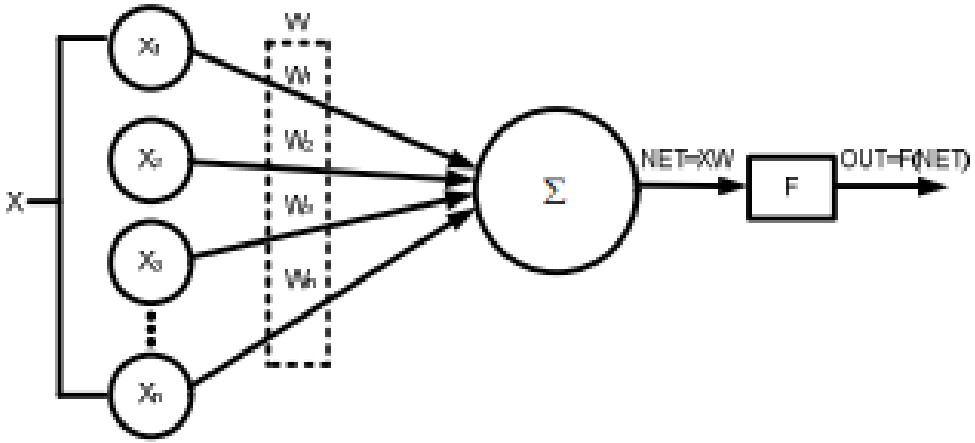


Рисунок 2 – Штучний нейрон з активаційною функцією

Нейромережі є за своєю суттю групою нейронів пов'язаних між собою певним чином, відзначимо, що в даний час існує кілька архітектур побудови мереж, що добре зарекомендували себе.

Підхід до побудови автоматичних регуляторів на основі штучного інтелекту дозволяє проектувати дуже потужні системи управління завдяки своїм унікальним властивостям. Однією з таких властивостей нейромереж є здатність до навчання, тобто. в процесі своєї роботи регулятор, створений із застосуванням даної технології, матиме здатність реагувати на зміни, що відбуваються з об'єктом управління, автоматично коригувати власні параметри, що, безсумнівно, позитивно позначиться на функціонуванні системи в цілому.

Крім перерахунку параметрів регулятора, нейронні мережі дозволяють визначати в якому з режимів роботи знаходиться система, що спостерігається, і з великою точністю передбачати майбутні значення керуючого параметра. Такі можливості, що надаються даним методом ІІ, дають розробникам дуже потужний інструмент, який дозволить вирішувати завдання раніше їм недоступні, підвищити якість роботи вже існуючих систем.

Споживачі ж отримають продукт із низкою унікальних властивостей, економічний ефект від використання якого буде дуже високий і в першу чергу виявиться у більшій економії ресурсів, що використовуються, якості управління. Ще однією важливою особливістю запропонованого методу ІІ є можливість і відносна простота побудови з його застосуванням, про гібридних систем, тобто. таких систем у яких можуть функціонувати не один і не два регулятори, а деяка їх кількість.

Наведемо кілька прикладів, як перший приклад задоволена проста і поширена ситуація. У нас є ПІД-регулятор, робота якого нас повністю влаштовує за невеликим винятком, а саме як згадувалося вище необхідно проводити час від часу корекцію його параметрів.

З цим чудово впорається нейромережа спеціальним чином спроектована та навчена. Як видно у нас вийшов зовсім новий пристрій, що складається з двох частин, ПІД-регулятор і коригуюча нейромережа. Так само можуть траплятися такі випадки, стан системи, коли кожен з наявних регуляторів може забезпечити необхідну нам якість управління тільки при певному режимі роботи.

Уявимо таку ситуацію: існує система, в якій містяться одночасно ПІД-регулятор, нечіткий контролер, заснований на прикладі апарату нечіткої логіки та нейронної мережі. Як відомо у ПІД-регулятора та нечіткого контролера є свої переваги та недоліки, там, де один забезпечує відмінне управління, інший відчуває певні проблеми і навпаки. Таким чином, щоб отримати необхідний результат, потрібно комбінувати їх роботу певним чином. Саме тут нам буде корисним використання однієї з можливостей нейромереж.

### 1.3 Види штучних нейронних мереж

Розвиток штучних нейронних мереж надихається біологією. Тобто розглядаючи мережеві зміни та алгоритми, дослідники мислять їх у термінах організації мозкової діяльності. Але на цьому аналогія може закінчитися. Наші знання про роботу мозку настільки обмежені, що мало знайшлося б керівних орієнтирів для тих, хто став би йому наслідувати. Тому розробникам мереж доводиться виходити межі сучасних біологічних знань у пошуках структур, здатних виконувати корисні функції. У багатьох випадках це призводить до необхідності відмови від біологічної правдоподібності, мозок стає просто метафорою, і створюються мережі, неможливі в живій матерії або потребують неправдоподібно великих припущень про анатомію та функціонування мозку [3].

Незважаючи на те, що зв'язок з біологією слабкий і часто несуттєвий, штучні нейронні мережі продовжують порівнюватися з мозком. Їхнє функціонування часто нагадує людське пізнання, тому важко уникнути цієї аналогії. На жаль, такі порівняння є неплідними і створюють невиправдані очікування, які неминуче ведуть до розчарування. Дослідницький ентузіазм, заснований на хибних надіях, може випаруватися, зіткнувшись із суворою дійсністю, як це вже одного разу було в шістдесяті роки, і багатообіцяюча область знову занепадає, якщо не буде дотримуватися необхідна стриманість.

Незважаючи на зроблені попередження, корисно все ж таки знати дещо про нервову систему ссавців, оскільки вона успішно вирішує завдання, до виконання яких лише прагнуть штучні системи.

Нервова система людини, побудована з елементів, званих нейронами, має приголомшливу складність. Близько 1011 нейронів беруть участь у приблизно 1015 передавальних зв'язках, що мають довжину метр і більше. Кожен нейрон має багато якостей, загальними з іншими елементами тіла, але його унікальною здатністю є прийом, обробка та передача електрохімічних сигналів по нервових шляхах, які утворюють комунікаційну систему мозку.

На (рис. 3) показано структуру пари типових біологічних нейронів. Дендрити йдуть від тіла нервової клітини до інших нейронів, де вони приймають сигнали в точках з'єднання, які називають синапсами. Прийняті синапс вхідні сигнали підводяться до тіла нейрона. Тут вони підсумовуються, причому одні входи прагнуть порушити нейрон, інші перешкодити його збудженню. Коли сумарне збудження у тілі нейрона перевищує певний поріг, нейрон збуджується, посилаючи сигналом іншим нейронам по аксону. Ця основна функціональна схема має багато ускладнень і винятків, проте більшість штучних нейронних мереж моделюють лише ці прості властивості.

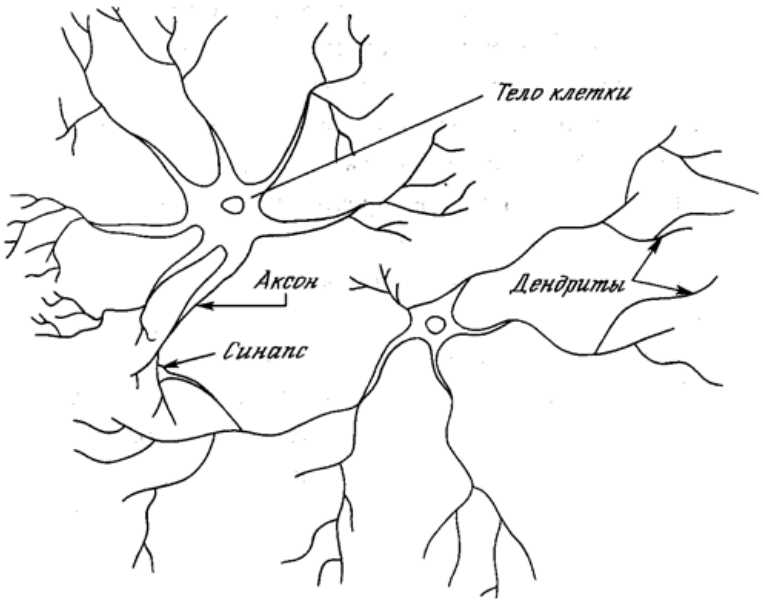


Рисунок 3 – Біологічний нейрон

Штучний нейрон імітує у першому наближенні властивості біологічного нейрона. На вхід штучного нейрона надходить кілька сигналів, кожен із яких є виходом іншого нейрона. Кожен вхід множиться на відповідну вагу, аналогічну синаптичній силі, і всі твори підсумовуються, визначаючи рівень активації

нейрона. На (рис. 2) представлено модель, що реалізує цю ідею. Хоча мережеві парадигми дуже різноманітні, в основі багатьох їх лежить ця конфігурація. Тут безліч вхідних сигналів, позначених x1, x2, ..., xn надходить на штучний нейрон. Ці вхідні сигнали, що сукупно позначаються вектором X, відповідають сигналам, що приходять в синапси біологічного нейрона. Кожен сигнал множиться на відповідну вагу w1, w2, ..., wn, і надходить на підсумовуючий блок, позначений?. Кожна вага відповідає силі одного біологічного синаптичного зв'язку. (Багато ваг у сукупності позначається вектором W.) Підсумовуючий блок, що відповідає тілу біологічного елемента, складає зважені входи алгебраїчно, створюючи вихід, який ми називатимемо NET. У векторних позначках це може бути компактно записано таким чином (рис. 4):

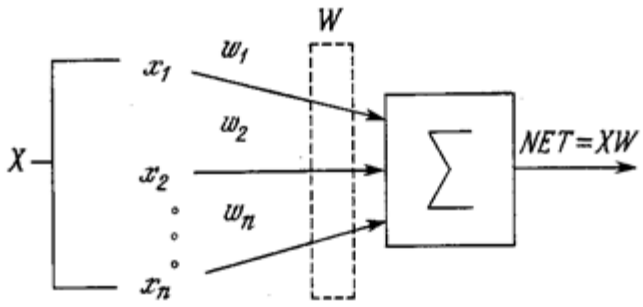


Рисунок 4 – Штучний нейрон

Сигнал NET далі, як правило, перетворюється на активаційну функцію F і дає вихідний нейронний сигнал OUT. Активаційна функція може бути звичайною лінійною функцією

OUT = K(NET), (1.2)

де К - постійна, порогова функція

OUT = 1, якщо NET > T,

OUT = 0 в інших випадках,

де T - деяка стала порогова величина, або функцією, більш точно моделює нелінійну передатну характеристику біологічного нейрона і представляє нейронної мережі великі можливості.

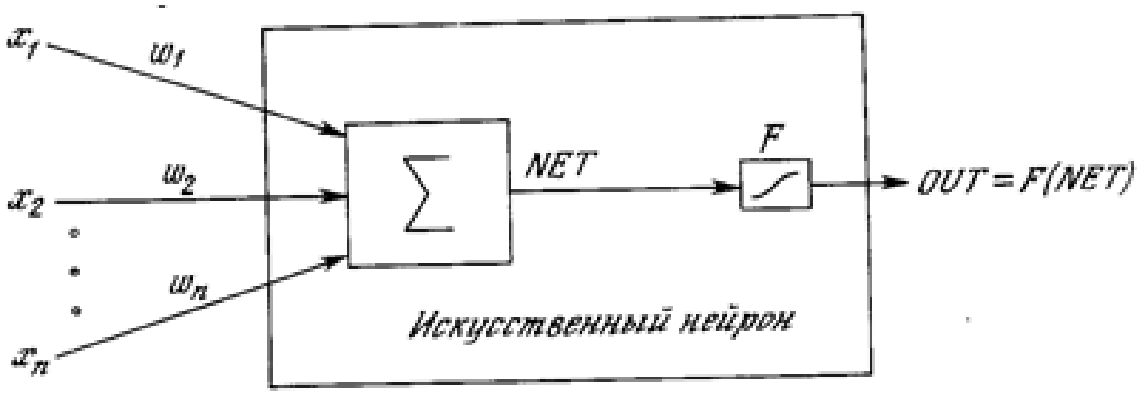


Рисунок 5 – Штучний нейрон з активаційною функцією

На (рис. 5) блок, позначений F приймає сигнал NET і видає сигнал OUT. Якщо блок F звужує діапазон зміни величини NET так, що при будь-яких значеннях NET значення OUT належать деякому кінцевому інтервалу, то F називається функцією, що «стискає». Як «стискаючу» функцію часто використовується логістична або «сигмоїдальна» (S-подібна) функція, показана на (рис 6). Ця функція математично виражається як F(x) = 1/(1 + e-x).


Рисунок 6 – Сігмоїдальна логістична функція

За аналогією з електронними системами активаційну функцію вважатимуться нелінійною підсилювальною характеристикою штучного нейрона. Коефіцієнт посилення обчислюється як відношення збільшення величини OUT до невеликого збільшення величини NET, що викликало його. Він виражається нахилом кривої при певному рівні збудження і змінюється від малих значень при великих негативних збудженнях (крива майже горизонтальна) до максимального значення при нульовому збудженні і знову зменшується, коли збудження стає більшим позитивним. Таким чином, нейрон функціонує з великим посиленням у широкому діапазоні рівня вхідного сигналу

Інший широко використовується активаційною функцією є гіперболічний тангенс. За формою вона подібна до логістичної функції і часто використовується біологами як математична модель активації нервової клітини. Як активаційну функцію штучної нейронної мережі вона записується наступним чином

Як показано на (рис. 7) подібно до логістичної функції гіперболічний тангенс є S-подібною функцією, але він симетричний щодо початку координат, і в точці NET = 0 значення вихідного сигналу OUT дорівнює нулю. На відміну від логістичної функції гіперболічний тангенс набуває значень різних знаків, що виявляється вигідним для низки мереж.

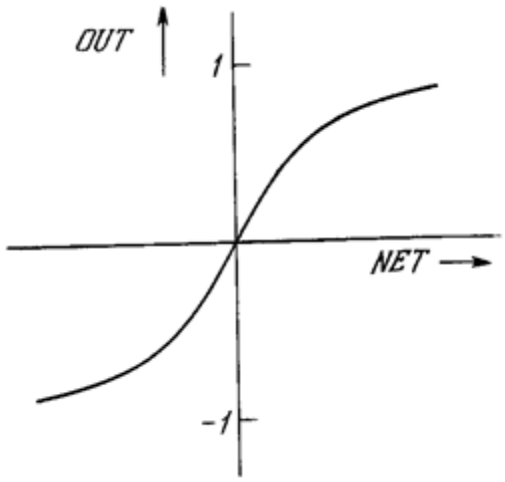


Рисунок 7 – Функція гіперболічного тангенсу

Хоча один нейрон і здатний виконувати найпростіші процедури розпізнавання, сила нейронних обчислень походить від сполук нейронів у мережах. Найпростіша мережа складається з групи нейронів, що утворюють шар, як показано у правій частині (рис. 8).

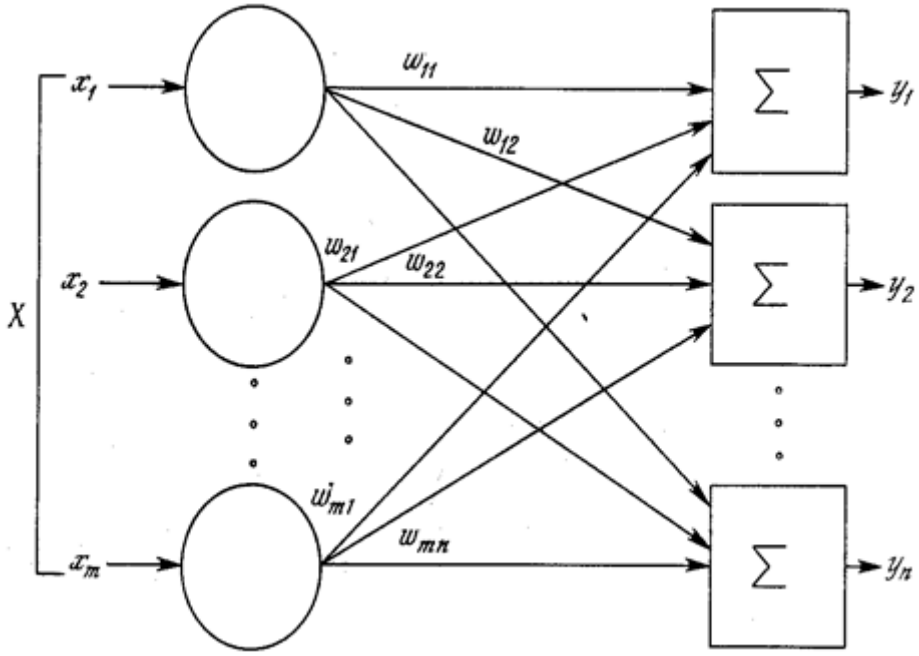

Рисунок 8 – Одношарова нейронна мережа

У штучних та біологічних мережах багато сполук можуть бути відсутніми, всі сполуки показані з метою спільності. Можуть мати місце також з'єднання між виходами та входами елементів у шарі.

Зручно вважати ваги елементами матриці W. Матриця має рядків і п стовпців, де m - число входів, а n - число нейронів. Наприклад, w2,3 - це вага, що зв'язує третій вхід із другим нейроном. Таким чином, обчислення вихідного вектора N, компонентами якого є виходи нейронів OUT, зводиться до матричного множення N = XW, де N і X - вектори-рядки.

Більші і складніші нейронні мережі мають, як правило, і великі обчислювальні можливості.

Багатошарові мережі можуть утворюватися каскадами шарів. Вихід одного шару є входом наступного шару. Така мережа показана на (рис. 9) і знову зображена з усіма сполуками.

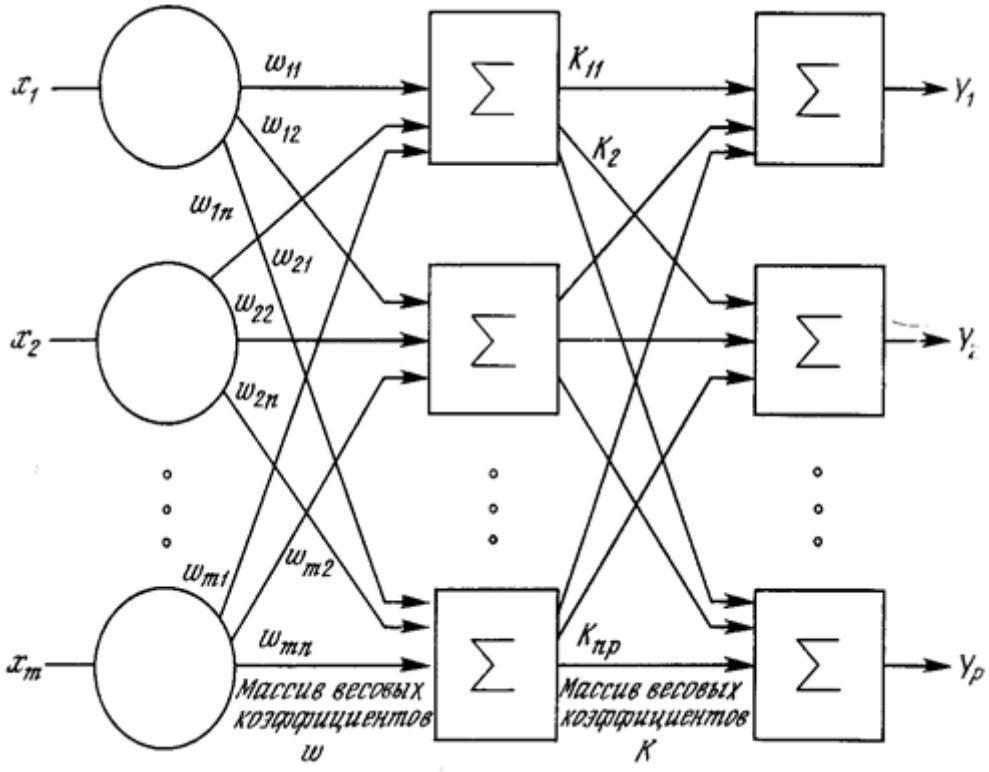


Рисунок 9 – Двошарова нейронна мережа

Багатошарові мережі не можуть призвести до збільшення обчислювальної потужності порівняно з одношаровою мережею лише в тому випадку, якщо активаційна функція між шарами буде нелінійною. Обчислення виходу шару полягає у множенні вхідного вектора на першу вагову матрицю з наступним множенням (якщо відсутня нелінійна активаційна функція) результуючого вектора на другу вагову матрицю (XW1) W2. Оскільки множення матриць асоціативно, то X(W1W2).

Це показує, що двошарова лінійна мережа еквівалентна одному шару з ваговою матрицею, що дорівнює добутку двох вагових матриць. Отже, будь-яка багатошарова лінійна мережа може бути замінена еквівалентною одношаровою мережею. Одношарові мережі дуже обмежені за своїми обчислювальними

можливостями. Таким чином, для розширення можливостей мереж порівняно з одношаровою мережею необхідна активна нелінійна функція.

### 1.4 Технології створення штучних нейронних мереж

Досить ефективний спосіб побудови нейронної мережі на основі відомих структурованих знань під назвою KBANN – Knowledge Based Artificial Neural Networks – розроблений Дж. Тоуелл [4-7]. Концепція цього методу полягає у генерації топології нейронної мережі на основі відомих структурованих знань. Початковою інформацією для побудови нейромережі може бути безліч кон'юнктивних правил. Вибрано формат правил «якщо …, то …», що найбільш підходить до моделюваного завдання. Перша причина цього вибору – безліч правил швидко перетворюється на дерево рішень. Створене дерево рішень перетворюється на нейронну мережу – «персептрон». Друга причина в тому, що вхідні та вихідні змінні проходять процедуру «шкал» [8].

При цьому значення шкал порівнюються з відповідними вхідними та вихідними нейронами, розподіляючи за рівнями вкладеності проміжні вузли дерева рішень. Звичайно, кількість цих рівнів визначає безліч прихованих шарів у мережі. Всім вузлам дерева рішень, які не є входом або виходом, прирівнюється у відповідність прихований нейрон. Зрештою виходить необхідна архітектура типу «персептрон» [4-7]. Важливим досягненням даного методу буде процедура ініціалізації значень ваги зв'язків ненавченої нейромережі та силі зв'язків початкового дерева рішень. Завдяки методу KBANN вагам міжнейронних зв'язків, що дорівнює позитивному параметру, що входить до правил, дається певне значення D1, а негативним вагам прирівнюється значення D2 (D2<Ω. Отримана умова рівності деякого вхідного вектора нейромережі активності одного з вихідних нейронів:

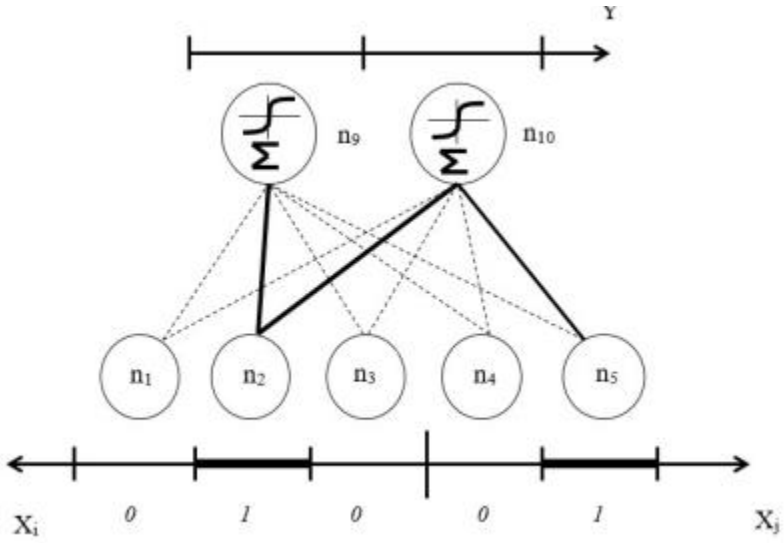


Рисунок 10 – Поширення вхідного сигналу в одношаровій нейромережі

У формулі застосовується функція активації A, в якій нелінійне відображення функції виходу суматора sett реалізується завдяки схожості значення sett конкретного інтервалу шкали значень виходу функції. Тоді щодо значення функції «A» можна сказати, що вона є функцією належності значення виходу суматора конкретному інтервалу або середнім значенням функції на цьому інтервалі, що належить вихідному нейрону. Формула визначає вагу зв'язку у вхідного та вихідного нейрона для семантично зв'язаних нейронів та показує силу впливу активації вхідного нейрона на значення виходу суматора вихідного нейрона.

У зв'язку з нейроном присутня інформація. При її докладнішому розгляді слід звернути увагу на залежність змінної та вихідного нейрона. Вага зв'язку i-го вхідного нейрона, рівного t інтервалу значень j даної змінної в рамках заданого інтервалу, відображає потужність впливу змінної на значення виходу суматора вихідного нейрона, який дорівнює інтервалу значень моделюється функції F.

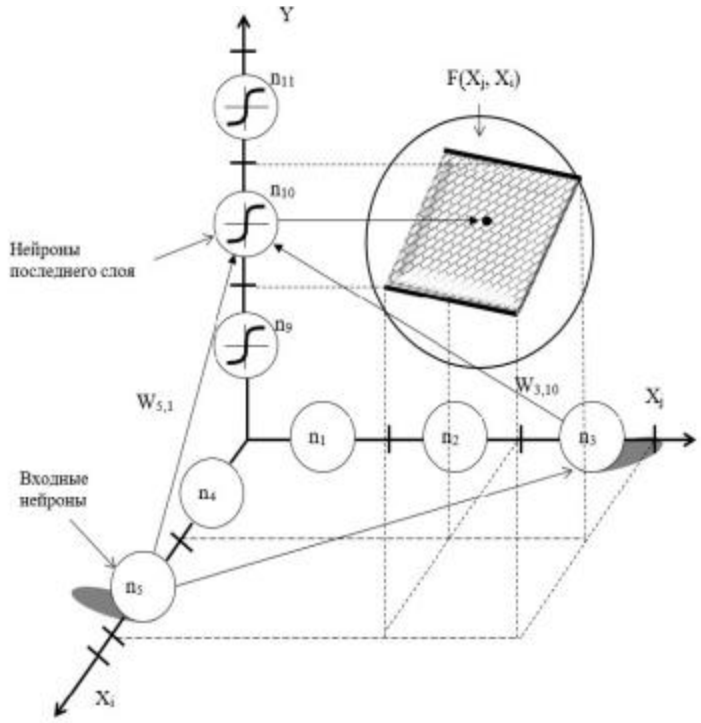


Рисунок 11 – Схема відображення нейромережею функціональної залежності виду Y = F (Xj, Xi)

На (рис. 11) відображено одношарову нейронну мережу з 2D функціональною залежністю.

Ваги зв'язків показують порівняльну межу впливу активації вхідних нейронів, рівних фіксованим інтервалам змінних вхідних, показання виходу суматора вихідного нейрона, який дорівнює потрібному інтервалу значень функції. Формула (2) Y = F (Xj, Xi) показує обмеження для значень ваг сполучених нейронів, що демонструють семантику впливу змінних моделюється нейромережею функції. Отримана нейромережа відбиває постійну функціональну залежність. Демонструє це безліч рівносильних лінійних функцій, якщо кожну функцію, що диференціюється, декількох змінних на області можливих або значних показників (робочої області) розібрати на комплекс

лінійних ділянок. Помилка лінеаризації знаходиться точністю апроксимації та регулюється кількістю інтервалів лінійності.

Інтервали всіх аргументів, рівні обраній ділянці лінійності та відповідної лінійної функції, можна прирівняти до інтервалів шкал дискретизації первинних аргументів у нейромережевій моделі. Зони простору аргументів, що показують будь-яку лінійну функцію з комплексу, описуються рівнянням лінійної множинної регресії.

Приватні коефіцієнти еластичності та бета-коефіцієнтів допомагають порівнювати незалежні змінні за рівнем їхнього впливу на залежну змінну, тобто порівнюють їх між собою за величиною даного впливу. Щоб визначити рівень впливу незалежної змінної на значення безперервної на інтервалі функції виду y=f(X), де X – вектор незалежних змінних, будемо застосовувати коефіцієнт еластичності

Коефіцієнт еластичності характеризує:

- Відносна зміна залежної змінної при зміні аргументу на 1%;
- силу впливу незалежних змінних значення функції на області допустимих значень.

Формула може застосовуватись для одношарової нейромережі. Багатошарові нейронні мережі відрізняються від одношарових присутністю шарів із прихованими нейронами. Тому використання даної формули для багатошарових нейромереж залежить від присутності проміжних змінних, що є залежними від вхідних змінних моделі та одночасно незалежними змінними для виходів. В результаті нейронна мережа стає логічно прозорою і просто декомпозується на безліч одношарових нейромереж. Створена формула застосовується на формування необхідних умов розрахунку первинних значень ваг зв'язків багатошарової нейронної мережі. На продуктивність методу «нейросетевого апроксиматора», заснованого на отриманій формулі алгебри,

можуть впливати різні фактори. У цій роботі запропоновано покращений метод розробки структури нейромережі, новизна якого ґрунтується на наближеній лінійній формулі. Також розглянуто застосування нейромережевого моделювання функціональної залежності показника для будь-якого процесу, що має багато вхідних та внутрішніх змінних параметрів. Цю методику можна застосовувати у класі методів створення нейромереж на основі знань. Застосування методу «нейросетьовий апроксиматор» за допомогою відомих структурованих знань створює можливість отримати:

- оптимальну модель процесу, розроблену для створення експериментів, що модулюються, в якій є пропуски і спотворення;
- прозору нейромережну структуру, а також виявити всі приховані закономірності у процесі модуляції;
- необхідні знання про структуру та потужність зв'язків параметрів імітаційного процесу при розробці нейромоделі.

За допомогою імітаційного процесу із застосуванням цього методу необхідно виконати такі правила:

- створення математичної моделі у форматі лінійного багаточлена;
- правильність набору висновків вимірювань параметрів процесу, виведених у вигляді таблиць значень змінних та рівнозначних значень функції;
- необхідне використання експерта у цій предметній галузі завдання.

Експертні знання дозволять покращити точність та об'ємність створеної моделі бази знань. Головне завдання при використанні методу «нейросетельного апроксиматора» за допомогою відомих структурованих знань полягає в перетворенні структури нейромережі, заснованої на базі даних із застосуванням наближеної формули, і ваги зв'язків, що вводяться, детермінованими отриманими значеннями в систему адекватних знань.

Внаслідок чого ваги, що вводяться, за створеним методом і перенавчена нейронна мережа отримують можливість відтворювати функціональну залежність, показану початковою математичною формулою. Звичайно, початкове навчання розробленої нейромережевої структури на основі таблиць даних модернізує початкову модель для рівності з цими даними.

# 2 ВИЗНАЧЕННЯ ПРОГРАМНИХ ЗАСОБІВ ДЛЯ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ ТА ЇХ ОБҐРУНТУВАННЯ

У своїй кваліфікаційній дипломній роботі я використовував середовище розробки програмного забезпечення Visual Studio 2022 Community edition. Було вирішено використовувати модульну платформу для розробки програмного забезпечення з відкритим вихідним кодом .NET 6, яка сумісна з такими операційними системами як Windows, Linux та macOS. Була випущена компанією Microsoft. У поєднані з технологією ML.NET та Swagger у якості фронтенду для веб-додатку. Далі більш детально ознайомимося і розглянемо особливості, а також переваги і недоліки кожної з обраних середовищ розробки та технологій.

## 2.1 Середовище розробки Visual Studio Community 2022

Visual Studio Community 2022 — це професійний інструмент розробки програмного забезпечення для Windows, Mac, iOS, Android, а також веб-додатків і хмарних сервісів. Visual Studio Community поширюється безкоштовно[9].

Microsoft Visual Studio – повнофункціональна інтегрована среда розробки (IDE) з підтримкою популярних мов програмування, серед яких C, C++, VB.NET, C#, F#, JavaScript, Python.

Функціональність Visual Studio охоплює всі етапи розробки програмного забезпечення, надаючи сучасні інструменти для написання коду, проектування графічних інтерфейсів, збірки, налагодження та тестування програм. Можливості Visual Studio можуть бути доповнені шляхом підключення потрібних розширень.

Основні особливості та переваги середи розробки Visual Studio [10]:

IntelliCode – це потужний набір засобів автоматичного завершення коду, які розпізнають контекст вашого коду: імена змінних, функції та тип

створюваного коду. Це дозволяє IntelliCode відразу завершувати цілий рядок, допомагаючи вам впевненіше та точніше створювати код.

CodeLens допомагає легко знаходити важливі аналітичні відомості, наприклад, які зміни зроблено, до чого вони привели, а також було б проведено модульне тестування вашого методу. Основні відомості, такі як посилання, автори, тести та історія фіксацій, завжди доступні та допоможуть вам прийняти виважені робочі рішення.

Інтегрована налагодження - найважливіша складова всіх продуктів Visual Studio. Можна проводити розбір коду, вивчати значення, що зберігаються в змінних, налаштовувати контрольні значення змінних, щоб відстежувати зміну значень, вивчати шлях виконання вашого коду та інші особливості роботи програмного забезпечення.

Аналізуйте обсяг протестованого коду та переглядайте миттєві результати у наборі тестів, оптимізованих для підвищення ефективності. Просунуті функції, що тестують код прямо під час введення, дозволяють дізнатися наслідки кожної зміни, що вноситься. Завдяки інтеграції WSL можна проводити тестування у Windows та Linux, щоб переконатися, що ваш додаток працює на обох платформах.

Visual Studio 2022 має вбудовану підтримку керування версіями Git для клонування, створення та відкриття власних репозиторіїв. Вікно інструментів Git містить все необхідне для фіксації та відправки змін до коду, керування гілками та вирішення конфліктів злиття. Якщо у вас є обліковий запис GitHub, можна керувати цими репозиторіями безпосередньо у Visual Studio.

## 2.2 Платформа для розробки .NET 6

.NET 6 – це швидка веб-платформа комплексної розробки, яка знижує витрати на обчислювальні ресурси під час роботи у хмарі.

У .NET 6 доступні заключні частини плану уніфікації .NET, який був представлений у .NET 5. .NET 6 поєднує пакет SDK, базові бібліотеки та середовище виконання для мобільних, класичних, хмарних програм та додатків Інтернету речей [11].

Крім цього екосистема .NET 6 пропонує такі можливості:

- Спрощена технологія: розпочати роботу дуже просто. Нові можливості мови C# 10 дозволяють скоротити обсяг створюваного коду. А інвестиції в веб-рішення та мінімальні API дозволяють швидко створювати невеликі та швидкодіючі мікрослужби.
- Вища продуктивність: .NET 6 - це найшвидша веб-платформа комплексної розробки, яка знижує витрати на обчислювальні ресурси при роботі в хмарі.
- Максимальна продуктивність: NET 6 та Visual Studio 2022 надають можливості гарячого перезавантаження, пропонують нові засоби Git, інструменти інтелектуального редагування коду, надійні засоби діагностики, тестування та більш ефективної спільної роботи груп.

Крім того .NET 6 буде підтримуватися протягом трьох років як випуск із довгостроковою підтримкою (LTS).

Розглянемо основні переваги платформи .NET 6.

.NET 6 містить безліч покращень продуктивності. У цьому розділі наведено деякі поліпшення в FileStream, оптимізацію за допомогою профілів і компіляції AOT. Нижче буде наведено лише частину покращень які є у новій платформі порівняно з попередніми її версіями.

Тип System.IO.FileStream був перероблений для .NET 6, щоб забезпечувати кращу продуктивність та надійність у Windows. Тепер FileStream ніколи не блокується під час створення асинхронного введення-виведення у Windows.

Сенс профільної оптимізації (PGO) полягає в тому, що JIT-компілятор створює оптимізований код з урахуванням найпоширеніших типів і шляхів коду. У .NET 6 з'явилася динамічна профільна оптимізація. Вона працює в тісній зв'язці з багаторівневою компіляцією для подальшої оптимізації коду на основі додаткового інструментарію, що розміщується на рівні 0. За замовчуванням динамічна функція PGO вимкнена, але її можна ввімкнути за допомогою змінного середовища DOTNET_TieredPGO.

Щоб зменшити розмір SDK для .NET, деякі компоненти були поміщені в нові, необов'язкові робочі навантаження для SDK. До цих компонентів відносяться .NET MAUI та Blazor WebAssembly AO. Якщо ви використовуєте Visual Studio, середовище встановить всі необхідні робочі навантаження для SDK. При використанні .NET CLI можна керувати робочими навантаженнями за допомогою нових команд dotnet workload.

У C# 10 включені такі нововведення, як Global using директиви, оголошення просторів імен з областю дії файлу та структури записів [12].

Відповідно до цієї роботи шаблони проектів у пакеті SDK для .NET для C# були модернізовані для використання деяких нових можливостей мови, а саме:

- Метод async Main;
- Інструкції верхнього рівня;
- Нові вирази цільового типу;
- Неявні директиви global using;
- Простір імен з областю дії файлу;
- Типи посилань, що допускають значення null.

System.Text.Json тепер підтримує серіалізацію та десеріалізацію з використанням екземплярів IAsyncEnumerable<T>. Асинхронні методи серіалізації перераховують усі екземпляри IAsyncEnumerable у графі об'єктів, а потім серіалізують їх як масиви JSON. Для десеріалізації був доданий новий метод JsonSerializer.DeserializeAsyncEnumerable<TValue>(Stream, JsonSerializerOptions, CancellationToken).

У .NET 6 доданий новий генератор вихідного коду System.Text.Json. Генератор вихідного коду працює з JsonSerializer і може бути налаштований кількома способами. Він може підвищити продуктивність, скоротити використання пам'яті та спростити усічення збірок. Для отримання додаткових відомостей див. стаття про вибір відображення або створення вихідного коду в System.Text.Json та створення вихідного коду в System.Text.Json [13].

ASP.NET Core містить удосконалення для мінімальних API-інтерфейсів, попередньої компіляції (AOT) для додатків Blazor WebAssembly та односторінкових додатків. Крім того, компоненти Blazor тепер можна візуалізувати з JavaScript та інтегрувати з наявними програмами на основі JavaScript [33].

У .NET 6 було додано такі дві структури: System.DateOnly і System.TimeOnly. Вони представляють частину дати і час DateTime, відповідно. DateOnly можна використовувати для днів народження та річниці, а TimeOnly підходить для щоденних оповіщень та щотижневих робочих годин.

Тепер у будь-якій операційній системі із встановленими даними про часовий пояс можна використовувати функцію керування адресними просторами Інтернету (IANA) або ідентифікатори часових поясів Windows. Метод TimeZoneInfo.FindSystemTimeZoneById(String) був оновлений для автоматичного перетворення вхідних даних з часового поясу Windows на часовий пояс IANA (або навпаки), якщо запитаний часовий пояс не знайдено в системі.

Крім того, були додані нові методи TryConvertIanaIdToWindowsId(String, String) та TryConvertWindowsIdToIanaId для сценаріїв, коли, як і раніше, необхідно вручну виконувати перетворення з одного формату часового поясу в інший [14].

## 2.3 Бібліотека ML.NET

ML.NET — це безкоштовна відкрита бібліотека із засобами машинного навчання мов програмування C# і F#. Вона також підтримує моделі на Python під час використання спільно з NimbusML. Попередній випуск ML.NET включав рішення для конструювання ознак (наприклад, створення N-грам), двійкової та мультикласової класифікацій, регресійного аналізу [15].

ML.NET дозволяє додавати до програми .NET можливості машинного навчання в автономному та підключеному режимах. Використовуючи цю функцію, ви зможете отримувати автоматичні прогнози на основі даних, доступних вашому додатку. Програми машинного навчання використовують для прогнозування закономірності, знайдені в даних, не будучи явно запрограмованими.

### 2.3.1 Принцип роботи ML.NET

В основі ML.NET є модель машинного навчання. Ця модель визначає кроки, які потрібно виконати для отримання прогнозів на основі вхідних даних. За допомогою ML.NET ви можете навчити модель користувача, вказавши відповідний алгоритм, а також імпортувати попередньо навчені моделі TensorFlow і ONNX [16].

Створену модель можна додати до програми та використовувати її для отримання прогнозів.

ML.NET працює у Windows, Linux та macOS з .NET Core або у Windows з .NET Framework. 64-розрядна версія підтримується на всіх платформах. 32-розрядна версія підтримується у Windows, за винятком функцій, пов'язаних з TensorFlow, LightGBM та ONNX.

За допомогою ML.NET можна отримувати прогнози таких типів:

- Класифікація/категоризація (автоматичний поділ відгуків клієнтів на позитивні та негативні);
- Регресія/прогнозування безперервних значень (прогнозування вартості будинків на основі розміру та розташування);
- Виявлення аномалій (виявлення шахрайських банківських транзакцій);
- Рекомендації (пропозиція продуктів, які можуть сподобатися покупцям Інтернет-магазину, на основі їхніх попередніх покупок);
- Тимчасові ряди/послідовності (прогнози погоди та обсягів продажу);
- Класифікація зображень (класифікація патологій на медичних зображеннях);
- Класифікація тексту (класифікувати документи за вмістом);
- Подібність пропозицій (вимірювати, наскільки схожі дві пропозиції).

На наведеній нижче схемі показано структуру коду програми, а також ітеративний процес розробки моделі (рис. 12).

- Збір та завантаження навчальних даних в об'єкт IDataView;
- Вказівка конвеєра операцій для отримання функцій та застосування алгоритму машинного навчання;
- Навчання моделі шляхом виклику функції Fit() для конвеєра;
- Оцінка моделі та ітерації для її покращення;
- Збереження моделі у двійковому форматі для використання у додатку
- Завантаження моделі назад в об'єкт ITransformer;
- Прогнозування за допомогою функції CreatePredictionEngine.Predict().

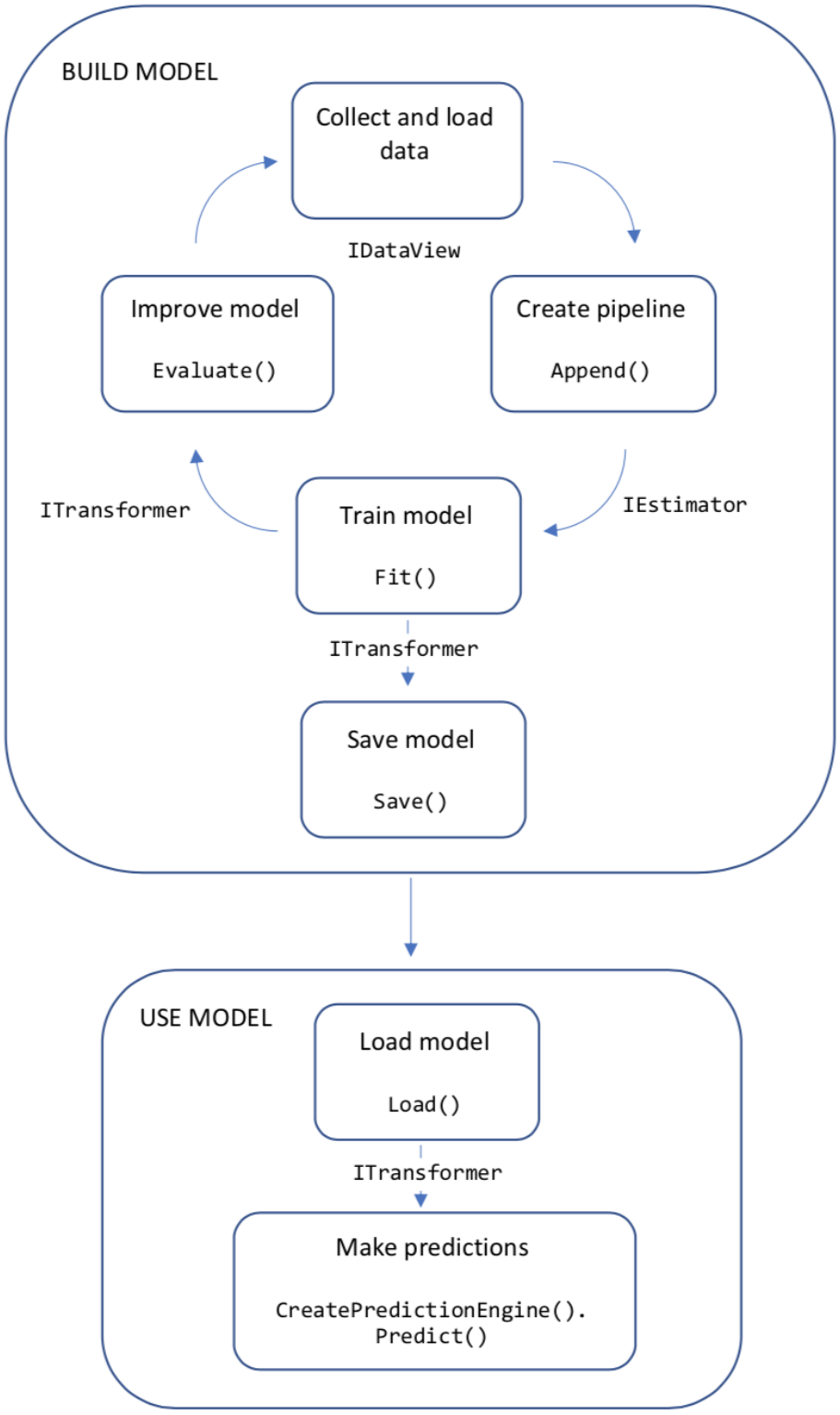


Рисунок 12 – Ітеративний процес розробки моделі.

### 2.3.2 Інтеграція ML.NET з TensorFlow

ML.NET не надає способу побудови нейронної мережі, на основі простого персептрона. По суті, це не інструмент для цього, як TensorFlow і Pythorch є в Python. Однак можна використовувати ML.NET у поєднанні з TensorFlow (точніше TensorFlow.NET), щоб використовувати попередньо навчені моделі, які надає TensorFlow. Як зображено на наступній діаграмі, додаємо посилання на пакети NuGet ML.NET до програми .NET Core або .NET Framework. Під обкладинкою ML.NET містить і посилається на свою вбудовану бібліотеку TensorFlow, яка дозволяє писати код, який завантажує наявний вже навчений файл моделі TensorFlow [17].

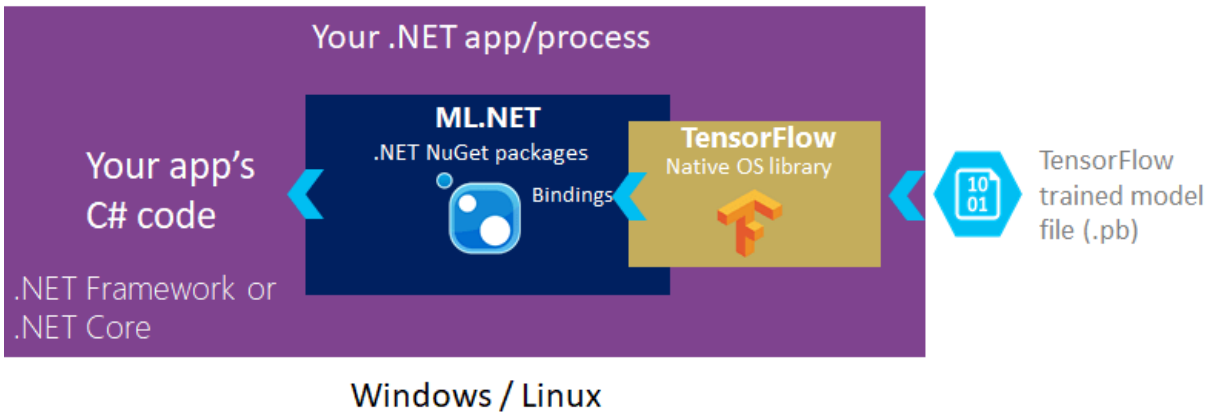


Рисунок 13 – Взаємодія навчених моделей з платформою .NET

Таким чином, додаючи посилання на пакети NuGet ML.NET в додатках .NET, а ML.NET в свою чергу містить і посилається на рідну бібліотеку TensorFlow. Це дає можливість використовувати попередньо навчені моделі TensorFlow.

## 2.4 Набір інструментів Swagger

Swagger це набір інструментів, які допомагають описувати API. Завдяки йому користувачі та машини краще розуміють можливості REST API без доступу до коду. За допомогою Swagger можна швидко створити документацію та надіслати її іншим розробникам чи клієнтам.

У 2015 році проект Swagger зробили відкритим та передали OpenAPI Initiative. Тепер сама специфікація називається OpenAPI. Swagger – інструментарій для роботи з OpenAPI, назва якого використовується в комерційних та некомерційних продуктах. Специфікаційний документ OpenAPI використовує YAML, але також може бути написаний у форматі JSON. Сам собою він є об'єктом JSON.

### 2.4.1 Основні підходи використання

Swagger пропонує два основних підходи до генерування документації:

- Автогенерація з урахуванням коду;
- Самостійна розмітка-написання.

Перший підхід простіший. Ми додаємо залежності до проекту, конфігуруємо налаштування та отримуємо документацію. Сам код може стати менш читабельним, документація теж не буде ідеальною. Але завдання щонайменше вирішене — код задокументований.

Щоб скористатися другим підходом, потрібно знати синтаксис Swagger. Описи можна готувати у форматі YAML/JSON. Можна спростити це завдання, використовуючи Swagger Editor. Звичайно, другий підхід дозволяє зробити документацію якіснішою і кастомнішою для кожного конкретного проекту та його особливостей, це потребує додаткових зусиль та часу на розробку.

У кваліфікаційній дипломній роботі було використано підхід з автогенерацією документації, що дозволило суттєво зменшити час на розробку.

### 2.4.2 Swagger Codegen

Це проект для автоматичного генерування клієнтських бібліотек API, заглушок сервера та документації. Підтримує велику кількість технологій. Переглянути повний список можна у репозиторії Swagger Codegen на GitHub.

У цьому репозиторії наведені приклади того, як можна генерувати документацію, використовуючи різні шаблони, що значно полегшує використання цього продукту.

Переваги Swagger Codegen:

- Генерування серверів – дозволяє позбутися рутини, створюючи шаблонний код 20 і більше мовами.
- Спрощене генерування SDK – можна створювати клієнтські набори 40 і більше мовами для швидкої інтеграції з вашим API.
- Постійне оновлення інструментів.

Він дуже легко додається до проекту наприклад за допомогою наступного коду:

```
<dependency>
   <groupId>io.swagger.codegen.v3</groupId>
   <artifactId>swagger-codegen-maven-plugin</artifactId>
   <version>3.0.24</version>
</dependency>
```

Swagger Codegen надає такі корисні та важливі команди, як:

- Config-help – допомога з налаштуванням вибраної мови;
- Generate – генерування коду з вибраним генератором;
- Help – висновок довідкової інформації для openapi-generator;
- List – список доступних генераторів;
- Meta – генератор набору шаблонів та налаштувань Codegen. Використовує інформацію про вибрану мову;
- Validate – перевірка специфікації;
- Version – відображення версії, що використовується.

Найкорисніші команди — generate та validate. Перша дозволяє створити клієнт, а друга – перевірити його відповідність специфікації. Переглянути повний перелік можливостей можна за допомогою команди help після запуску jar-файлу.

### 2.4.3 Swagger UI

Swagger UI візуалізує ресурси OpenAPI та взаємодію з ними без відображення логіки реалізації. Цей інструмент бере специфікацію та перетворює її на інтерактивний проект, де можна виконувати різні запити для тестування API.

Swagger UI візуалізує ресурси OpenAPI та взаємодію з ними без відображення логіки реалізації. Цей інструмент бере специфікацію та перетворює її на інтерактивний проект, де можна виконувати різні запити для тестування API.

Переваги Swagger UI:

- Немає залежностей. Інтерфейс доступний у будь-якому середовищі розробки;
- Повна підтримка браузерами;
- Зручна взаємодія для інших розробників та клієнтів. Можна перевірити всі операції API, навіть не маючи спеціальних знань;

- Проста навігація. Ресурси та кінцеві точки представлені у вигляді акуратно розподіленого на категорії списку;
- Можна налаштовувати стиль і інтерфейс користувача так, як буде потрібно, доступна повна кастомізація.

Інтерфейс користувача Swagger повністю розміщений в SwaggerHub. Є можливість написати та візуалізувати API або імпортувати існуючі визначення для створення інтерактивного інтерфейсу, розміщеного у хмарі. Завдяки вбудованій інтерактивності SwaggerHub дозволяє безпечно надавати доступ до документації зовнішнім користувачам або іншим розробникам.

### 2.4.4 Swagger Editor

Це онлайн-редактор для зміни та перевірки API усередині браузера. Дозволяє переглядати документацію у реалтаймі. З його допомогою можна створити описи, а потім використовувати їх з повним набором інструментів для створення документації.

Переваги Swagger Editor:

- Працює у будь-якому середовищі розробки, локально та в мережі.
- Дає зворотний зв'язок – показує помилки в синтаксисі при написанні, перевіряючи його на відповідність OpenAPI.
- Дозволяє миттєво візуалізувати API через Swagger UI.
- Допомагає писати документацію швидше завдяки інтелектуальному автозаповненню.
- Надає гнучкі інструменти для конфігурації – від теми оформлення до міжрядкового інтервалу.
- Пропонує створення серверних заглушок та клієнтських бібліотек для створеного API всіма популярними мовами.

# 3 ОПИС МОЖЛИВОСТЕЙ ТА ІНСТРУКЦІЯ КОРИСТУВАННЯМ ДОДАТКУ

Уся взаємодія з штучною нейронною мережею відбувається через інструмент Swagger, який відправляє необхідні запити у API веб-додатку. Нижче ми розглянемо основні можливості додатку та способи взаємодії з ними. При запуску веб-додатку користувач автоматично потрапляє на головну сторінку – сторінку Swagger, вона має наступний вигляд (рис. 14, 15).

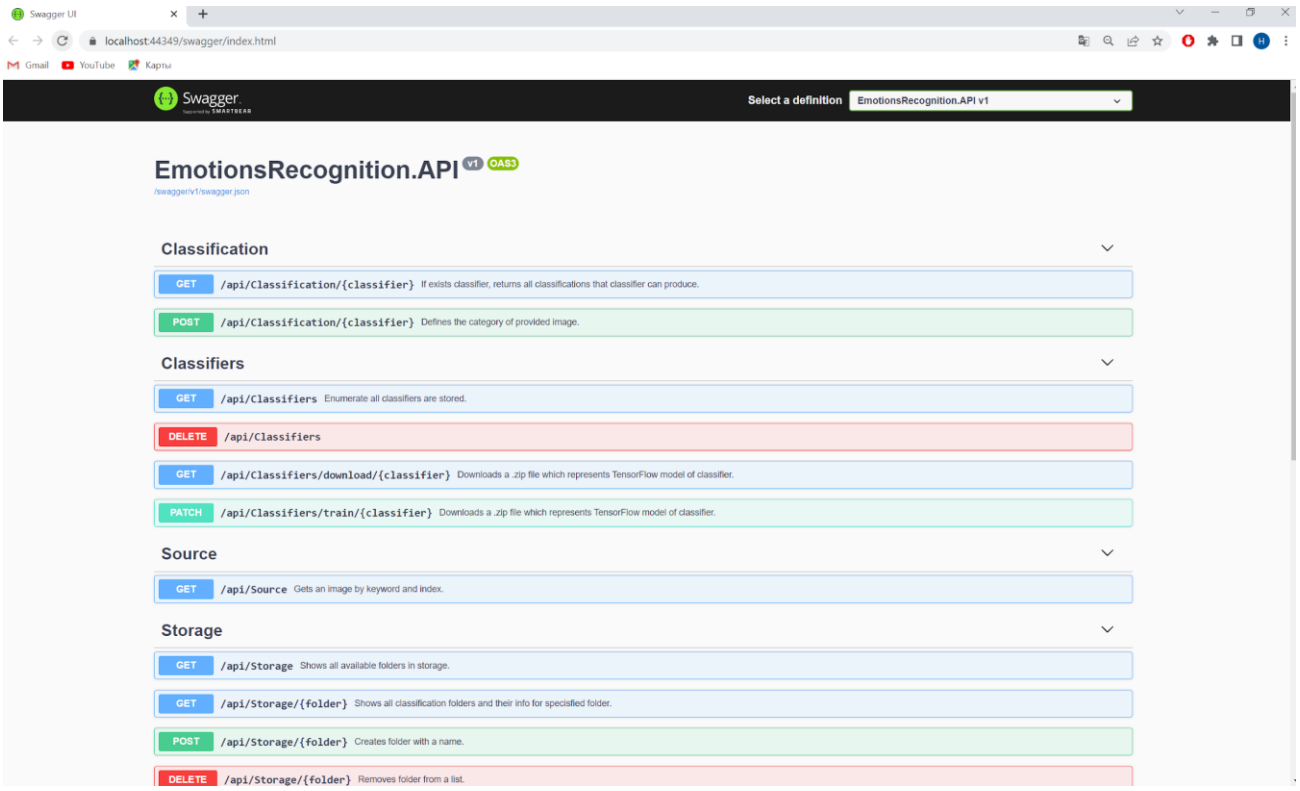


Рисунок 14 – Головна сторінка веб-додатку частина 1

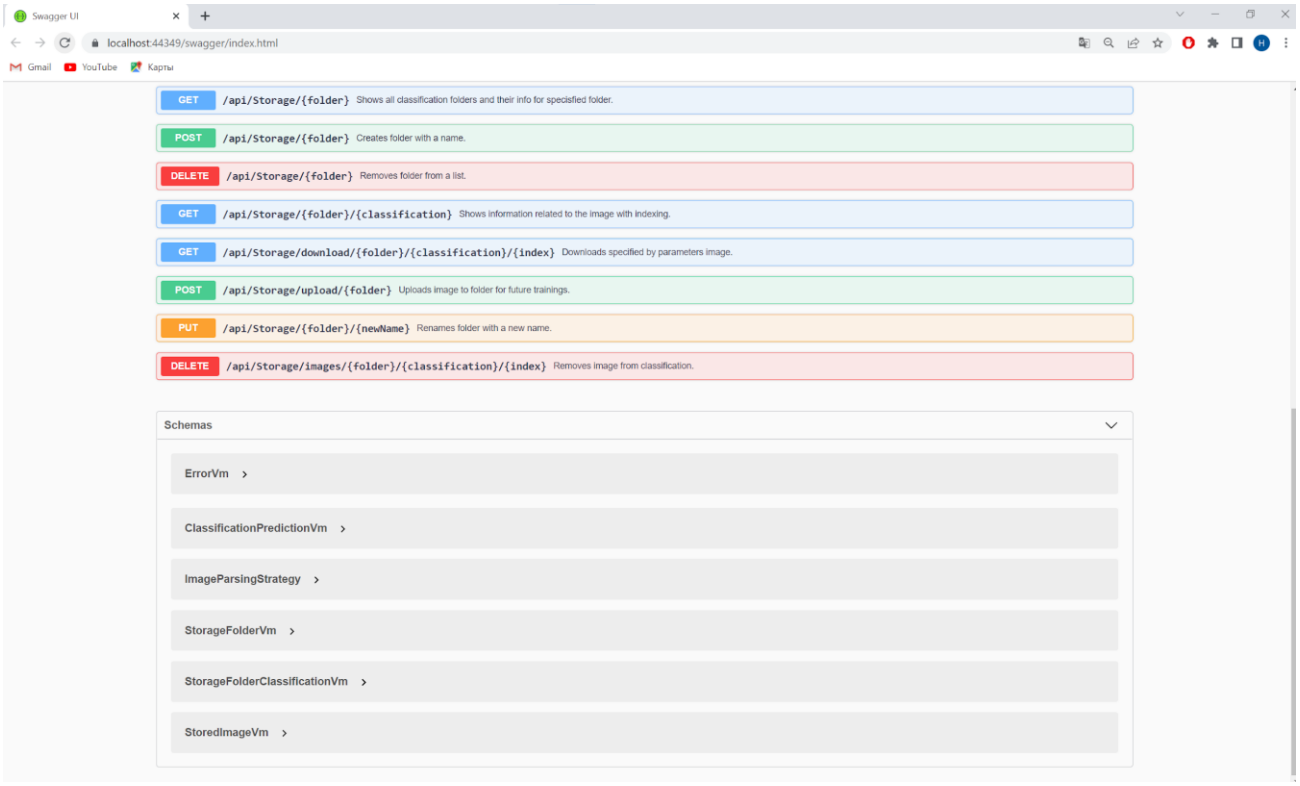


Рисунок 15 – Головна сторінка веб-додатку частина 2

Користувачу доступна повна інформація стосовно кожного API-endpoint, а саме:

- які дані очікує від користувача при виклику;
- тип запиту може бути;
- короткий опис що саме робить даний API-endpoint;
- помилки які можуть бути отримані у результаті виклику.

У додатку усього є 4 основних контролери (рис. 16):

- Classification;
- Classifiers;
- Source;
- Storage.

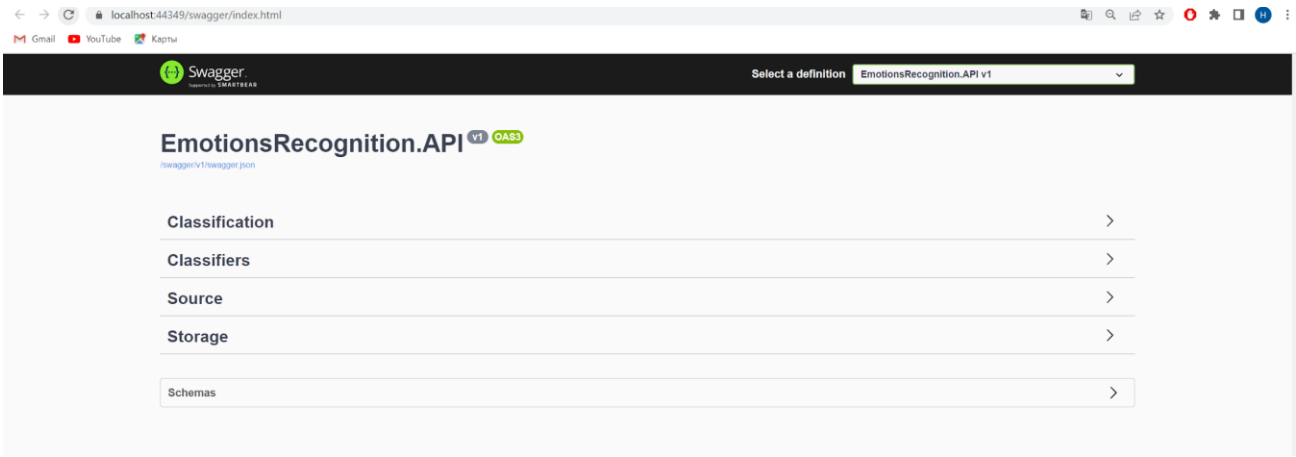


Рисунок 16 – Основні контролери веб-додатку

Розглянемо більш детально кожний з них, а саме для чого вони потрібні, які дані вони очікують та що роблять.

Щоб отримати назви усіх існуючих натренерованих моделей потрібно надіслати запит до «/api/Classifiers».

Зробити це можна наступним чином:

- обрати «/api/Classifiers» натиснувши на нього;
- натиснути на кнопку «Try it out»;
- потім натиснути кнопку «Execute».

Як бачимо на (рис. 17) зараз в нас існую лише одна натренерованих модель «emotions».

Якщо користувач хоче дізнатися більше детальну інформацію про вже існуючу модель модель тоді йому треба надіслати запит до «/api/Classification/{classifier}», де «classifier» – це назва існуючої моделі.

Натиснути на кнопку «Try it out» та заповненити поле «classifier» необхідною назвою моделі, наприклад з попереднього запиту. У відповіть користувач отримає список існуючих категорії у цій моделі (рис. 18).

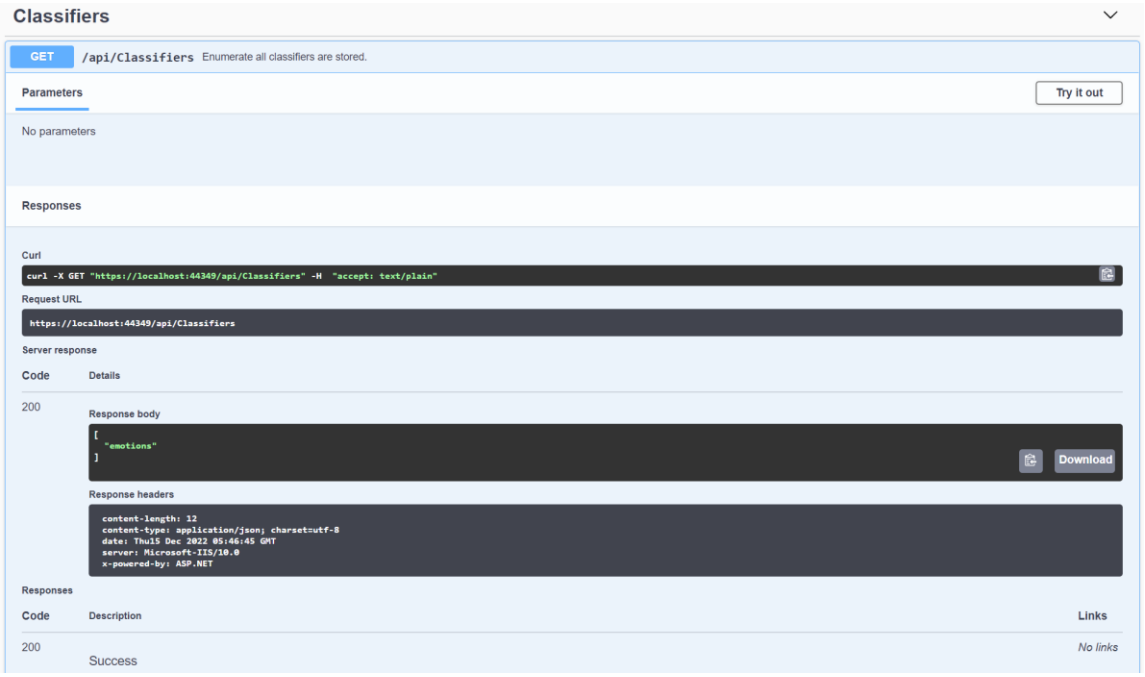


Рисунок 17 – Отримання списку усіх натренерованих моделей

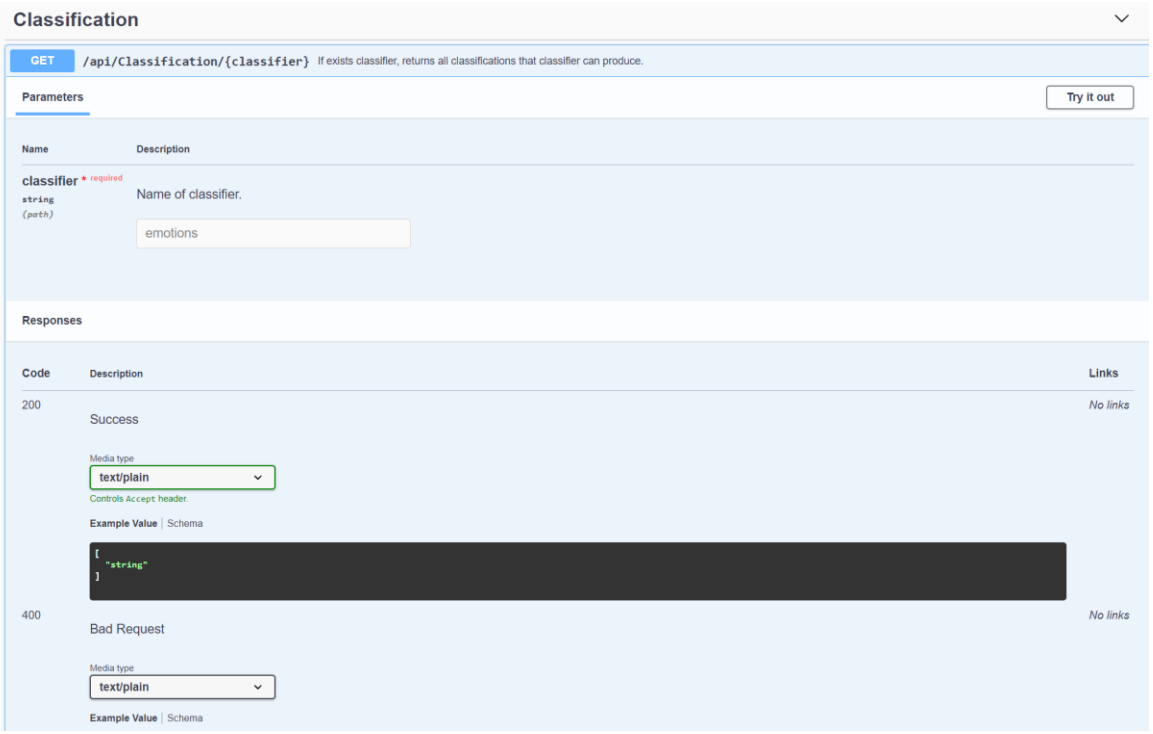


Рисунок 18 – Отримання списку існуючих категорії для моделі

Після того як ми ознайомилися з усіма існуючими моделями, та додатково подивилися категорії у певної моделі (за бажанням) ми можемо скористатися штучною нейронною мережею для оцінки емоції людини за фотографією. Щоб це зробити потрібно виконати наступні дії:

- обрати «/api/Classification/{classifier}», натиснувши на нього;
- натиснути на кнопку «Try it out»;
- у поле «classifier» внести назву моделі – «emotions»;
- у поле «image» завантажити фотографію яку ми бажаємо оцінити;
- та потім натиснути кнопку «Execute».

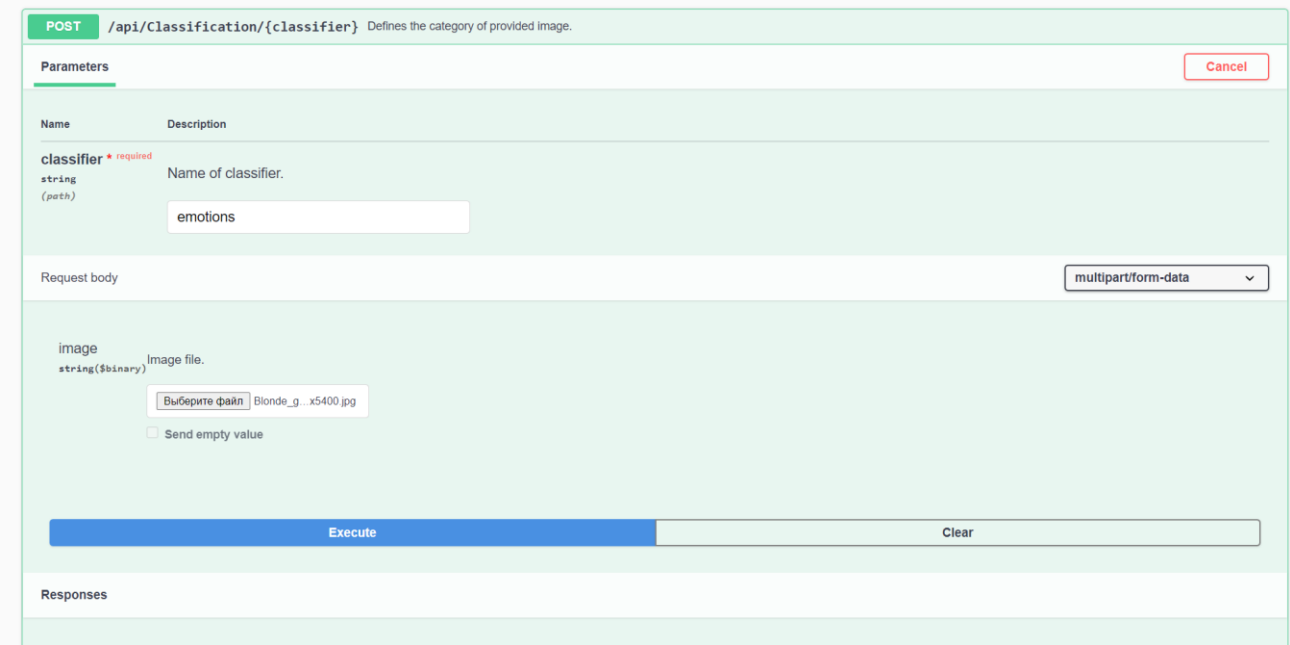


Рисунок 19 – Форма для визначення емоції людини за фотографією

Після виконання запиту ми отримаємо результат у якому буде наступна інформація:

- ім’я фотографії;
- передбачувана штучною нейронною мережею емоція людини;
- ймовірність емоції
- час на виконнання запиту.

Ми отримали наступні дані (рис. 20):

- емоція людини «щаслива»
- ймовірність 81.78%
- час на виконання 271 мілісекунда

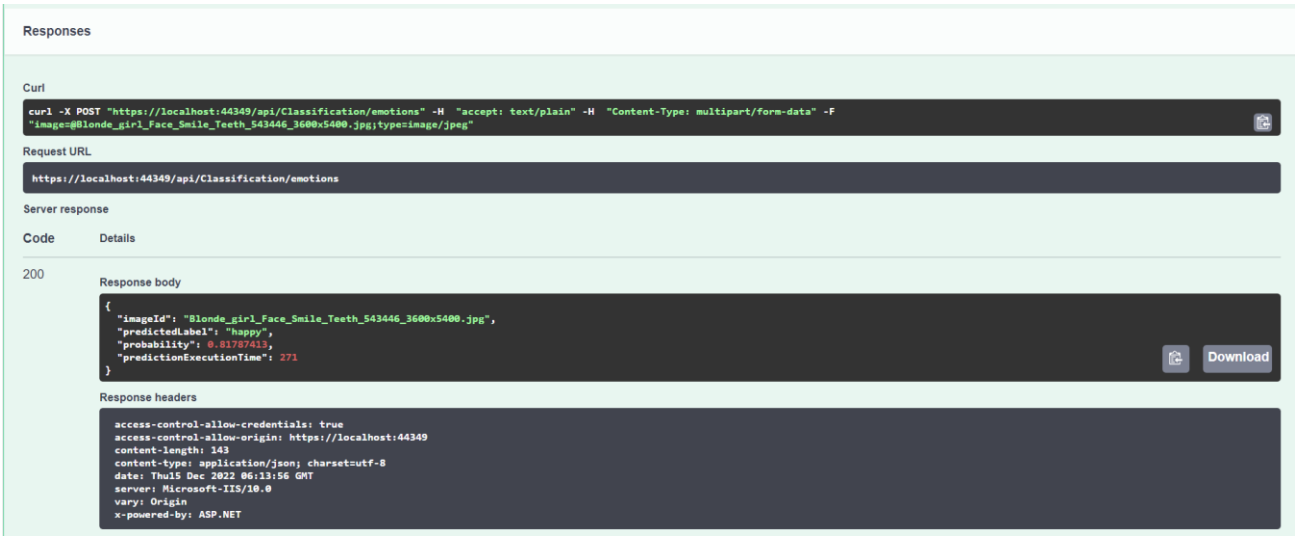


Рисунок 20 – Результат обробки фото нейронною мережею

Для того щоб завантажити модель у вигляді .zip файлу необхідно скористатися «/api/Classifiers/download/{classifier}» (рис. 21) та зробити наступне:

- обрати «/api/Classifiers/download/{classifier}», натиснувши на нього;
- натиснути на кнопку «Try it out»;
- у поле «classifier» внести назву моделі – «emotions»;
- та потім натиснути кнопку «Execute».

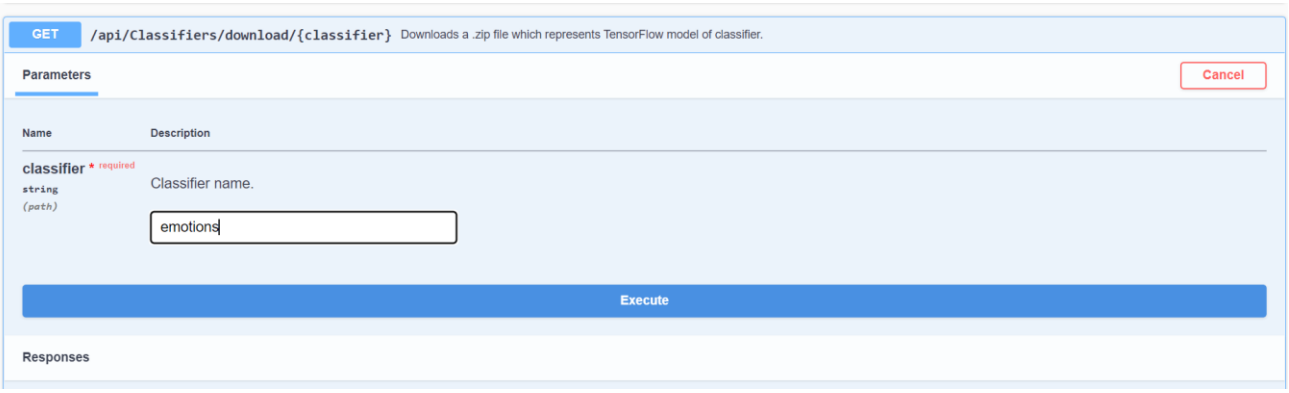

Рисунок 21 – Завантаження моделі у zip форматі

Щоб видалити модель необхідно скористатися «/api/Classifiers» (рис. 22) та зробити наступне:

- обрати «/api/Classifiers», натиснувши на нього;
- натиснути на кнопку «Try it out»;
- у поле «classifier» внести назву моделі – «emotions»;
- та потім натиснути кнопку «Execute».

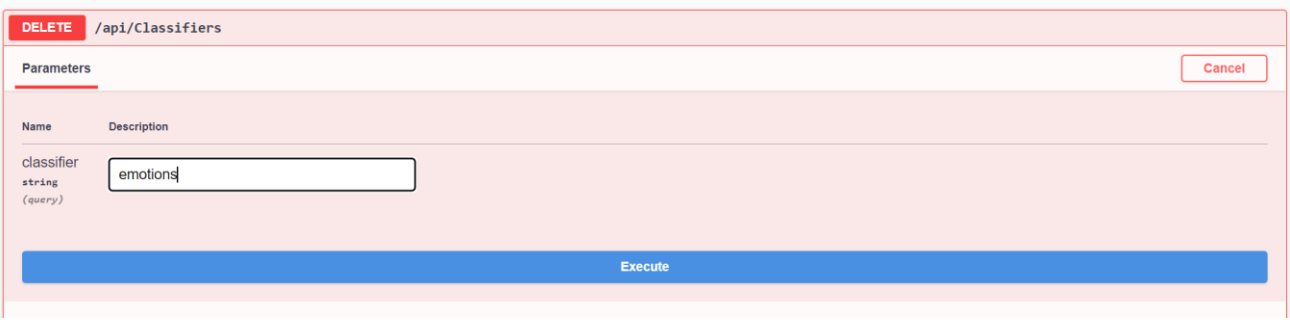


Рисунок 22 – Видалення моделі з веб-додатку

Також є можливість подивитися на вже завантажену фотографію у модель. Зробити це можна за допомогою ключового слова (наприклад назви зображення або за допомогою індексу). Для необхідно скористатися «/api/Source» (рис. 23) та зробити наступне:

- обрати «/api/Source», натиснувши на нього;
- натиснути на кнопку «Try it out»;
- у поле «keyword» внести назву ключового слова за яким буде виконаного пошук або у поле «index» відповідно номер індексу;

– для поля «source» вибрати з випадаючого списку значення «DefaultImageParsing»;

– та потім натиснути кнопку «Execute».

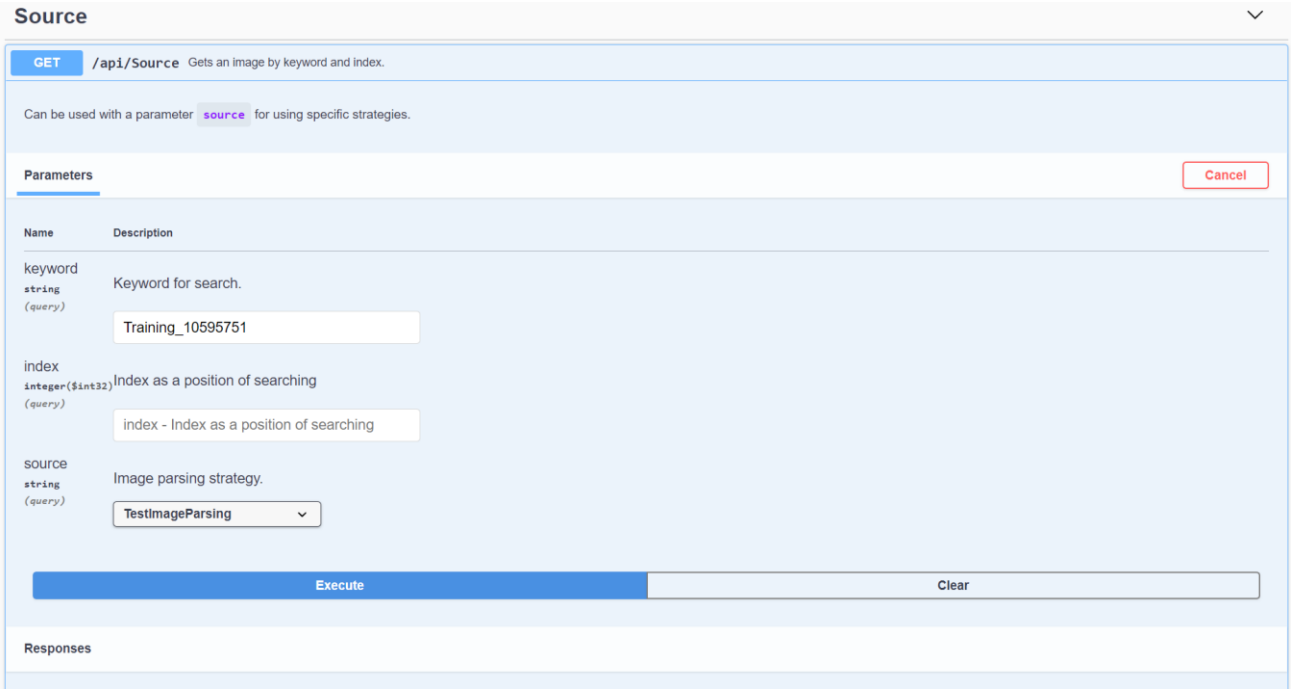


Рисунок 23 – Пошук зображення за ключовим словом

Для того щоб завантажити зображення потрібно зробити ті ж самі кроки що описані вище у для (рис. 23), але тільки скористатися «/api/Storage/download/{folder}/{classification}/{index}» (рис. 24).

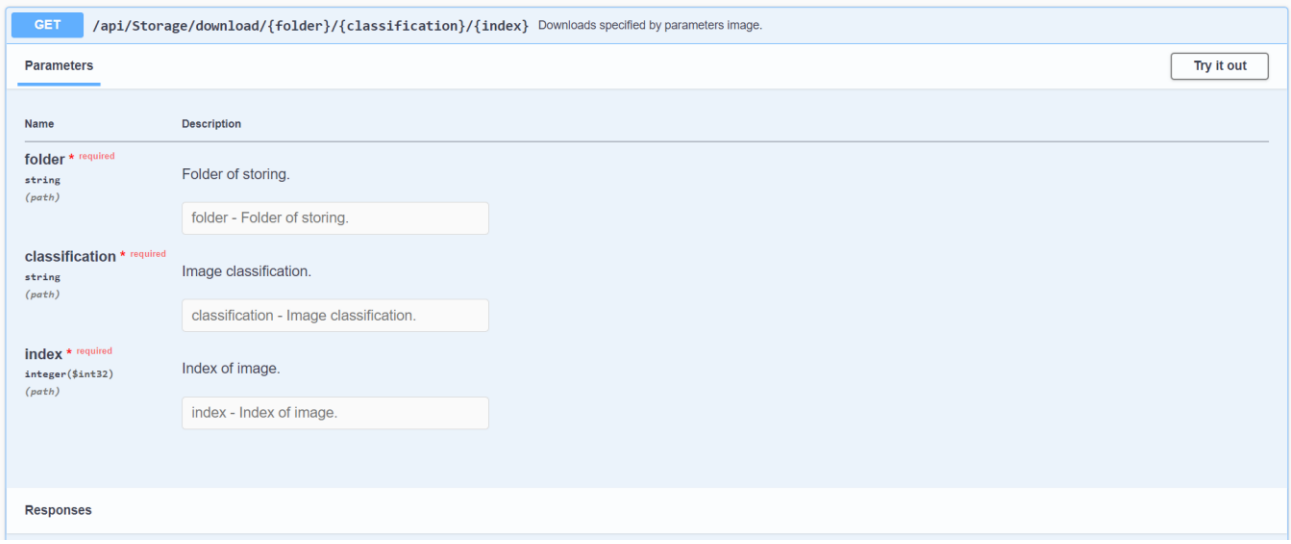


Рисунок 24 – Завантаження зображення за індексом

Якщо користувач  бажає додати нові зображення до штучної нейронної мережі це можна зробити кількома способами додати папки у проект власноруч за шляхом «...\EmotionsRecognition\EmotionsRecognition.API\Storage», а потім завантажити туди зображення, або зробити це за допомогою «/api/Storage/upload/{folder}» (рис. 25) та зробити наступне:

- обрати «/api/Storage/upload/{folder}», натиснувши на нього;
- натиснути на кнопку «Try it out»;
- у поле «folder» внести назву папки у яку буде завантажено зображення;
- у поле «classification» внести назву моделі категорії;
- у поле поле «image» відповідно завантажити саме зображення;
- та потім натиснути кнопку «Execute».

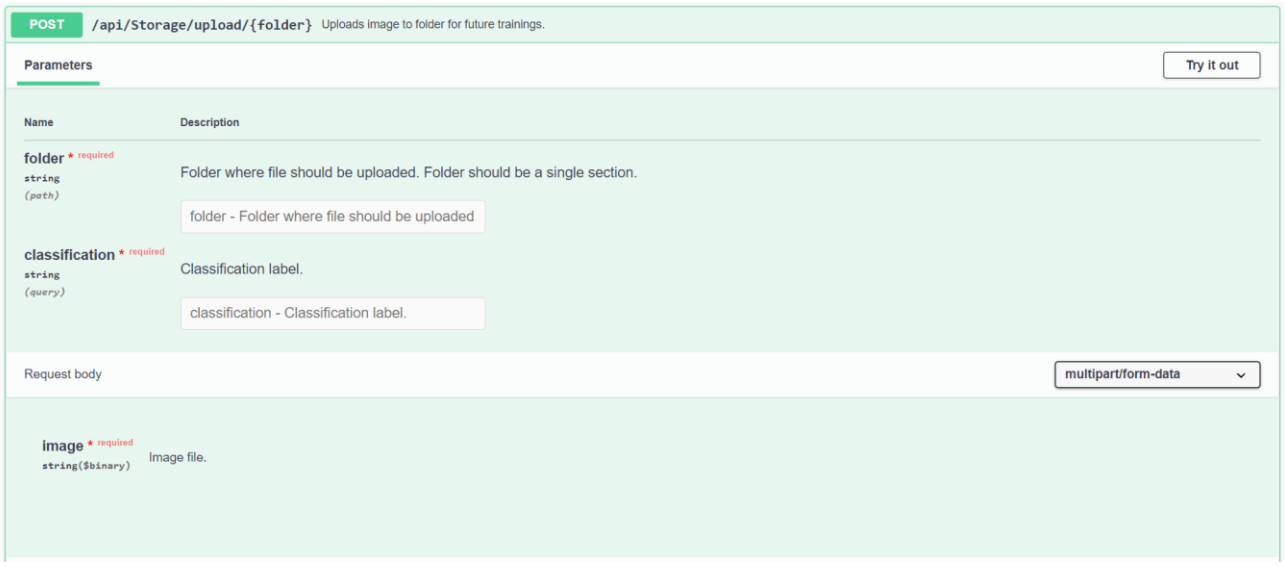


Рисунок 25 – Завантаження зображення у веб-додаток

Для того щоб навчити нову модель необхідно попередньо створити папки у веб-додатку одним із способів як це було зазначено вище. Назва і зміст папки повинні відтворювати назву категорії. Після цього додати необхідні зображення до цих папок. Та зверніть увагу, щоб навчити модель необхідно від двох і більше категорій (папок) у веб додатку для цього нового класифікатора.

Після того як всі зазначені вище умови виконані необхідно скористатися «/api/Classifiers/train/{classifier}» (рис. 26) та зробити наступне:

- обрати «/api/Classifiers/train/{classifier}», натиснувши на нього;
- натиснути на кнопку «Try it out»;
- у поле «imageFolder» внести назву папки у яку попередньо було завантажено категорії;
- у поле «classifier» внести нову назву класифікатора (моделі штучної нейронної мережі);
- та потім натиснути кнопку «Execute».

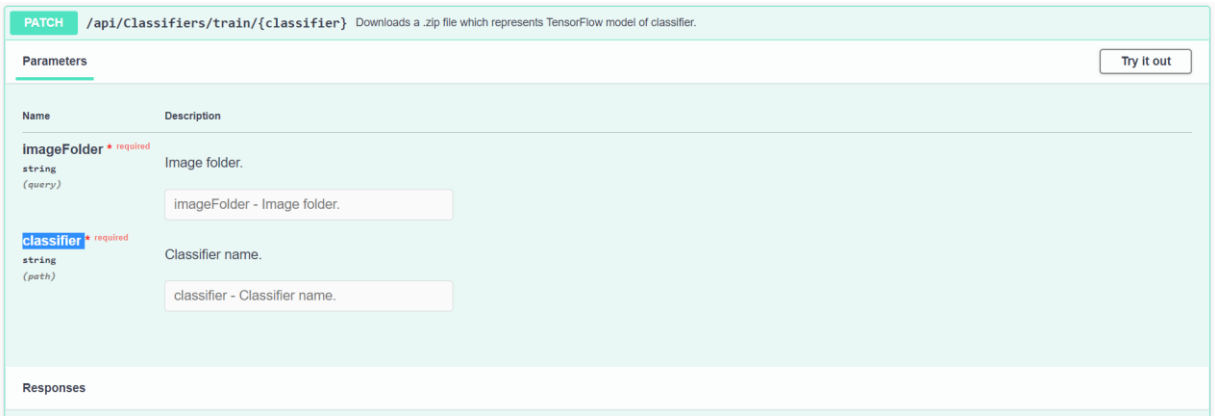


Рисунок 26 – Тренування нового класифікатора

Щоб перейменувати папку у веб-додатку, наприклад коли вона помилково була неправильно названа необхідно скористатися «/api/Storage/{folder}/{newName}» (рис. 27) та зробити наступне:

- обрати «/api/Storage/{folder}/{newName}», натиснувши на нього;
- натиснути на кнопку «Try it out»;
- у поле «folder» внести стару назву папки яку буде перейменовано;
- у поле «newName» внести нову назву папки;
- та потім натиснути кнопку «Execute».

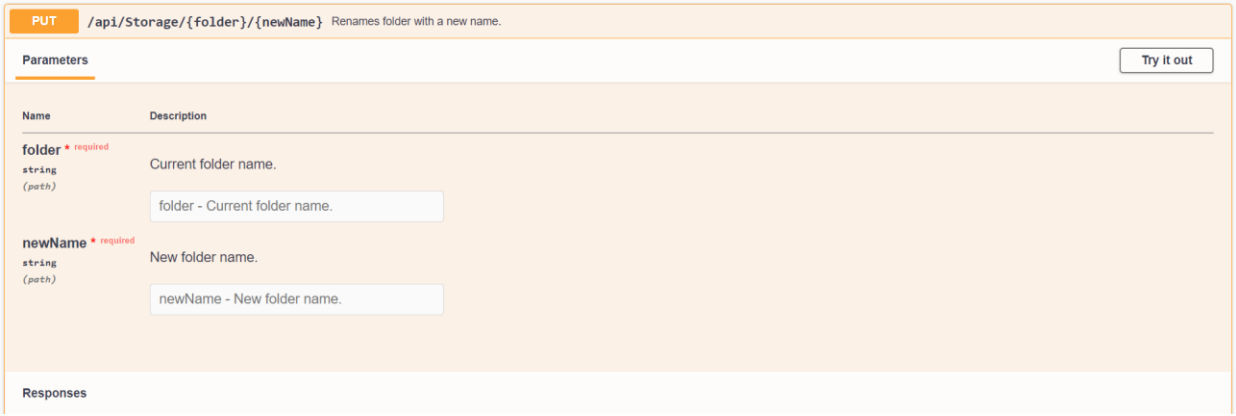

Рисунок 27 – Перейменування папки з зображеннями

Щоб видалити зображення з папки веб-додатку необхідно скористатися «/api/Storage/images/{folder}/{classification}/{index}» (рис. 27) та зробити наступне:

- обрати «/api/Storage/images/{folder}/{classification}/{index}», натиснувши на нього;
- натиснути на кнопку «Try it out»;
- у поле «folder» внести назву папки з якої буде видалено зображення;
- у поле «classification» внести назву моделі категорії;
- у поле поле «index» відповідно номер індексу зображення;
- та потім натиснути кнопку «Execute».

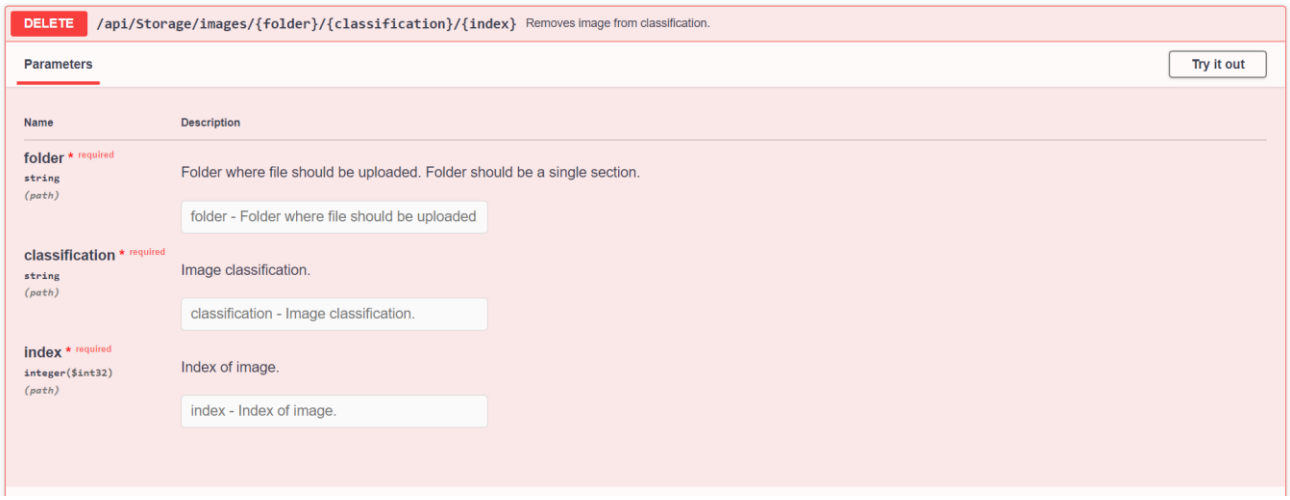


Рисунок 27 – Видалення зображення з папки за індексом

Якщо користувач зробив помилку під час виконання запиту, наприклад вказав неправильне ім’я не існуючого класифікатора під час тренування, чи неправильне ім’я папки при спробі її видалення в такому разі користувач отримає відповідний статус код та текст помилки.

# 4 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКУ

## 4.1 Підготування набору даних

Для розробки сценарію підготовки використовувалися ресурси для отримання колекції використовуючи стратегію Unsplash.

Розробка будь-якої моделі починається з підготовки двох масивів картинок, один з яких призначений для тренування та навчання, а інший для перевірки наскільки модель була натренована правильно. Для цього і був розроблений сценарій підготовки масивів даних для тестування або тренування моделей. Він спрощує пошук картинок використовуючи відкритий та безкоштовний сервіс Unsplash. Завдяки цьому можна отримувати та завантажувати великі об’єми даних та формувати їх в архіви необхідного формату.

Для підготовки архівів використовується модель Category, яка складається:

- Name – для загального найменування категорії;
- Keywords – для можливих ключових слів, які можуть використовуватися для пошуку картинок.

Для першого тестування будуть використовуватися категорії, такі як:

- angry;
- disgust;
- fear;
- happy;
- neutral
- sad;
- surprise.

Як видно з нижче зазначеного коду порядок виконання буде наступним:

- Ініціалізація;
- Прорахування відносних шляхів для завантаження;
- Реєстрація обробників подій;
- Отримання динамічних сторінок із картинками;
- Паралельне завантаження даних;
- Конвертація до одного формату;
- Архівація колекції;
- Обробник помилок та завершення сценарію.

```
public static async Task Main()
 {
    var parseRequest = new ParseRequest { Categories = categories, EstimatedCount =
estimatedCountOfImages };
    Directory.CreateDirectory(imagesDirectory);
    var files = Directory.GetFiles(imagesDirectory);
    var currentIndex = files.Any() ? (int?)files.Max(path => GetIndexFromPath(path, pattern,
indexBracers)) : null;
    var nextIndex = currentIndex.HasValue ? (int?)currentIndex.Value + 1 : null;
    var indexString = nextIndex.HasValue ? $"
{indexBracers.Left}{nextIndex}{indexBracers.Right}" : string.Empty;
    var archiveName = $"{pattern[0]}{indexString}.{pattern[^1]}";
    var archive = Path.Combine(imagesDirectory, archiveName);
    var context = new ParsingContext();
    var progress = GetProgress();
    var parsedImages = context.ParseImagesAsync(parseRequest, progress);
    await using (var fs = new FileStream(archive, FileMode.CreateNew))
    {
       using var zip = new ZipArchive(fs, ZipArchiveMode.Update);
       var indexes = categories.SelectMany(x => x.Keywords).ToDictionary(x => x, x =>
default(int));
       await foreach (var parsedImage in parsedImages)
       {
          var image = parsedImage.Image;
          var rawFormat = image.RawFormat;
          if (maxWidthOfImage < image.Width)
          {
             image = image.ProportionalResizeImageWidth(maxWidthOfImage);
          }
          var format = new ImageFormatConverter().ConvertToString(rawFormat).ToLower();
```

```
            var entryName = $"{parsedImage.Keyword}{(index == default ? string.Empty : $"-
{index}")}.{format}";
            var entryPath = Path.Combine(parsedImage.Category, entryName);
            var entry = zip.CreateEntry(entryPath, CompressionLevel.Optimal);
            await using var stream = entry.Open();
            try
            {
                image.Save(stream, rawFormat);
            }
            catch (Exception ex)
            {
                throw;
            }
            indexes[parsedImage.Keyword] += 1;
            image.Dispose();
            if (!image.Equals(parsedImage.Image))
            {
                parsedImage.Image.Dispose();
            }
        }
    }
}
```

В даному випадку архів буде завантажуватись паралельно, тому цей процес займає не більше декількох хвилин. Такий підхід дозволяє зменшувати час на пошук необхідних матеріалів, та єдине що потрібно зробити користувачу – це відфільтрувати отриманий архів та налаштувати під наступний етап – Тренування.

### 4.2 Тренування та створення моделі

Цей розділ відповідає на питання як натренувати модель та яким чином цей процес автоматизований. Починаючи відповідати на ці питання, необхідно зазначити, що процес тренування повністю автоматичний та готовий для використання з будь-якими архівами у відповідному форматі. Необхідно перевірити лише коректність завантаженого матеріалу, але з попереднім

сценарієм – це зробити набагато легше. Після отримання підготовленого архіву даних, необхідно запустити сценарій тренування.

Процес тренування складається з таких основних частин:

- Завантаження архіву з підготовленими даними;
- Разархівування та перехід до оперативної пам’яті;
- Підготовка даних для початку тренування;
- Завантаження картинок до пам’яті в побітовому форматі;
- Розділення даних для тренування та оцінки моделі;
- Визначення моделі та завдання опціональних значень;
- Початок тренування;
- Валідація та прогін епох;
- Оцінка моделі;
- Створення таблиці заплутаності;
- Збереження моделі у форматі ZIP.

Першим кроком на шляху тренування є завантаження підготовленого архіву даних. Зазвичай такі дані зберігаються на сервері, тому була створена утиліта, яка дозволяє бачити поточний прогрес завантаження.

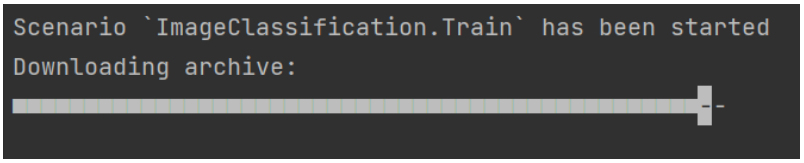


Рисунок 28 – Завантаження архіву

Наступним кроком буде разархівування завантаженого файлу. Зазвичай цей процес займає небагато часу, в даному випадку менше секунди, але це може

відрізнятися в залежності від потужності персонального комп’ютера та розміру архіву.

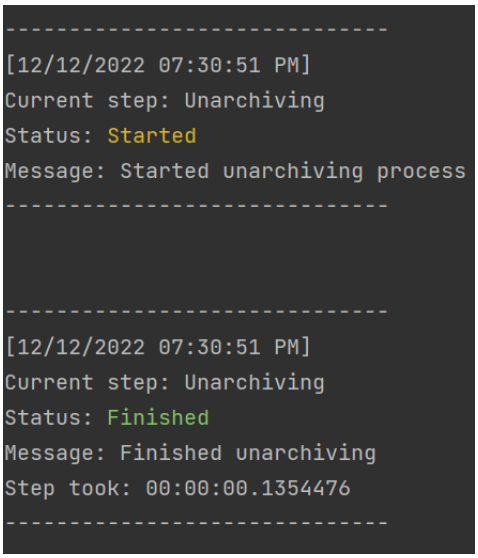


Рисунок 29 – Разархівування створеного масиву картинок

Слідом йде процес підготовки колекції даних для постачання до тренувального процесу моделі, на даному етапі колекція даних десеріалізується та переходить до однакового формату даних.

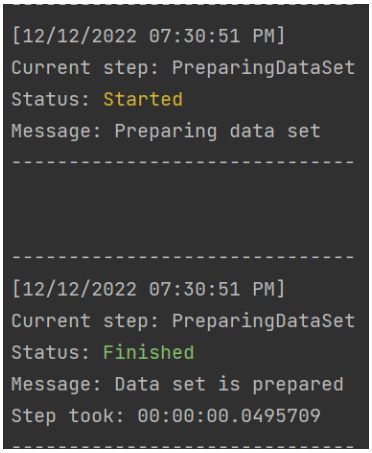

Рисунок 30 – Підготовка масиву даних для використання в тренуванні

Наступним йде процес завантаження картинок до оперативної пам’яті, звідки вони надалі потраплятимуть до тренування.

```
------------------------------
[12/12/2022 07:30:51 PM]
Current step: LoadingImages
Status: Started
Message: Loading images in memory
------------------------------

~~~ [Source=Term; Training, Kind=Trace] Channel started
~~~ [Source=StreamingDataView; Cursor, Kind=Trace] Channel started
~~~ [Source=Shuffle; Cursor, Kind=Trace] Channel started
~~~ [Source=StreamingDataView; Cursor, Kind=Trace] Channel finished. Elapsed 00:00:00.0404235.
~~~ [Source=StreamingDataView; Cursor, Kind=Trace] Channel disposed
~~~ [Source=Shuffle; Cursor, Kind=Trace] Channel finished. Elapsed 00:00:00.0411852.
~~~ [Source=Shuffle; Cursor, Kind=Trace] Channel disposed
~~~ [Source=Term; Training, Kind=Trace] Channel finished. Elapsed 00:00:00.0561555.
~~~ [Source=Term; Training, Kind=Trace] Channel disposed

------------------------------
[12/12/2022 07:30:51 PM]
Current step: LoadingImages
Status: Finished
Message: Loading images in memory has been finished
Step took: 00:00:00.0847199
```

Рисунок 31 – Завантаження картинок до оперативної пам’яті

Розподілення даних між двома процесами: тренуванням та оціненням необхідно для максимальної точності, крім того всі дані перемішуються довільним чином. Завдяки цьому модель не буде мати повторюваності та залежностей.

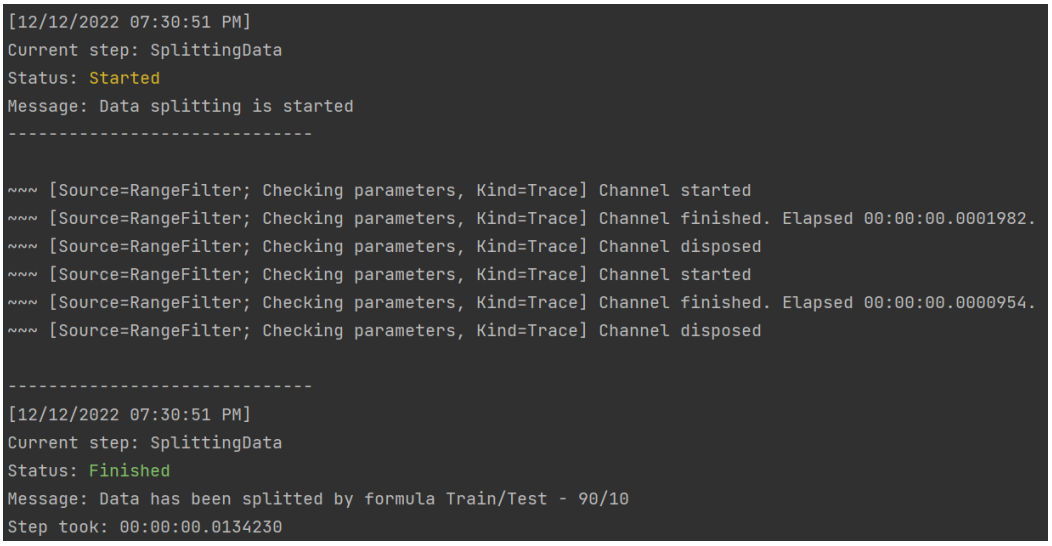


Рисунок 32 – Розподілення та перемішування даних

Визначення моделі дозволяє самостійно налаштувати процес навчання, де можна вказувати такі показники як:

- Епоха;
- Архітектура;
- Розмір пакету даних;
- Швидкість навчання;
- Назва стовпця функції;
- Назва стовпця мітки.

Такі параметри зазвичай вказують скільки епох потрібно для коректного навчання, архітектуру необхідну для використання, розмір одиничного пакету з картинками даних для завантаження до моделі, швидкість кожного кроку навчання та інше.

```
[12/12/2022 07:30:51 PM]
Current step: DefiningModel
Status: Started
Message: Model will be defined with options:
Epoch=200
Architecture=InceptionV3
BatchSize=10
LearningRate=0.01
FeatureColumnName=Image
LabelColumnName=Image

------------------------------

2022-12-12 19:30:51.812954: I tensorflow/core/platform/cpu_feature_guard.cc:142]
To enable them in other operations, rebuild TensorFlow with the appropriate comp
2022-12-12 19:30:51.829386: I tensorflow/compiler/xla/service/service.cc:168] XL
2022-12-12 19:30:51.829448: I tensorflow/compiler/xla/service/service.cc:176]

------------------------------
[12/12/2022 07:30:51 PM]
Current step: DefiningModel
Status: Finished
Message: Model is defined
Step took: 00:00:00.4999317
```

Рисунок 33 – Процес визначення моделі

Після визначення моделі, починається тренування, цей процес займає найбільший проміжок часу.

```
--------------------------------
[12/12/2022 07:30:51 PM]
Current step: Trainning
Status: Started
Message: Started training process
--------------------------------
```

Рисунок 34 – Початок навчання

Крім того до консолі будуть виводитися всі логи щодо валідації картинки, точності кожної епох та інші дані.

```
*****************************
Type of metrics: Train
Image with index 27 was processed out.
*****************************

~~~ [Source=RowToRowMapperTransform; Cursor, Kind=Trace] Channel finished. Elapsed 00:00:05.1096996.
~~~ [Source=RowToRowMapperTransform; Cursor, Kind=Trace] Channel disposed
~~~ [Source=GenerateNumber; Cursor, Kind=Trace] Channel finished. Elapsed 00:00:05.1037469.
~~~ [Source=GenerateNumber; Cursor, Kind=Trace] Channel disposed
~~~ [Source=RangeFilter; Cursor, Kind=Trace] Channel finished. Elapsed 00:00:05.1024455.
~~~ [Source=RangeFilter; Cursor, Kind=Trace] Channel disposed
~~~ [Source=SelectColumnsDataTransform; Cursor, Kind=Trace] Channel finished. Elapsed 00:00:05.1070234.
~~~ [Source=SelectColumnsDataTransform; Cursor, Kind=Trace] Channel disposed
*****************************
Accuracy: 0.73333335
- Accuracy of the batch on this Epoch. Higher the better.
Batch Processed Count: 3
- The number of batches processed in an epoch.
Cross-Entropy: 0.81684256
- Cross-Entropy (loss) of the batch on this Epoch. Lower the better.
Epoch: 0
- The training epoch index for which this metric is reported.
Learning Rate: 0.01
- Learning Rate used for this Epoch. Changes for learning rate scheduling.
*****************************
```

Рисунок 35 – Процес валідації та визначення точності епохи

Як видно в даному випадку процес тренування зайняв не більше 13 секунд. Але це стало можливим через невеликий архів даних. Іноді з архівами, які налічують більше тисячі файлів процес навчання може займати більше години.

```
------------------------------
[12/12/2022 07:31:04 PM]
Current step: Trainning
Status: Finished
Message: Finished training process
Step took: 00:00:12.8958002
------------------------------
```

Рисунок 36 – Завершення тренування

Наступним кроком буде оцінка моделі, яка дозволяє надати ефективність створеної моделі.

Цей процес необхідний для розуміння чи потрібно щось змінювати, чи ні.

```
[12/12/2022 07:31:06 PM]
Current step: EvaluatingModel
Status: Finished
Message: Finished evaluating model
Step took: 00:00:01.6975427
```

```
[12/12/2022 07:31:04 PM]
Current step: EvaluatingModel
Status: Started
Message: Started evaluating model...
```

Рисунок 37 – Оцінка створеної моделі

Завершенням оцінки моделі, виводиться таблиця невизначеності. В даному випадку, модель була натренована з такими показниками точності:

```
Confusion table
            ||========================
PREDICTED   ||     0 |     1 |     2 | Recall
TRUTH       ||========================
0.    happy ||     0 |     0 |     0 | 0.0000
1.      sad ||     0 |     0 |     1 | 0.0000
2. surprise ||     0 |     0 |     0 | 0.0000
            ||========================
Precision   ||0.0000 |0.0000 |0.0000 |
```

Рисунок 38 – Таблиця невизначеності

Та й останнім процесом є збереження моделі у форматі ZIP, що в свою чергу дозволяє легко переносити та завантажувати її на інші носії інформації. Крім того такий підхід відкриває можливості щодо подальшого розширення моделі за наявною необхідністю.

```
------------------------------
[12/12/2022 07:31:06 PM]
Current step: SavingModel
Status: Started
Message: Started saving model...
------------------------------

~~~ [Source=MulticlassPredictionTransformer; Saving train schema, Kind=Trace] Channel started
~~~ [Source=ImageClassificationTrainer; BinarySaver; Saving, Kind=Trace] Channel started
~~~ [Source=ImageClassificationTrainer; EmptyDataView; Cursor, Kind=Trace] Channel started
~~~ [Source=ImageClassificationTrainer; EmptyDataView; Cursor, Kind=Trace] Channel finished. Elapsed 00:00:00.0027003.
~~~ [Source=ImageClassificationTrainer; EmptyDataView; Cursor, Kind=Trace] Channel disposed
~~~ [Source=ImageClassificationTrainer; EmptyDataView; Cursor, Kind=Trace] Channel started
~~~ [Source=ImageClassificationTrainer; BinarySaver; Write, Kind=Trace] Channel started
~~~ [Source=ImageClassificationTrainer; EmptyDataView; Cursor, Kind=Trace] Channel finished. Elapsed 00:00:00.0010067.
~~~ [Source=ImageClassificationTrainer; EmptyDataView; Cursor, Kind=Trace] Channel disposed
~~~ [Source=ImageClassificationTrainer; BinarySaver; Write, Kind=Trace] Channel finished. Elapsed 00:00:00.0344629.
~~~ [Source=ImageClassificationTrainer; BinarySaver; Write, Kind=Trace] Channel disposed
~~~ [Source=ImageClassificationTrainer; BinarySaver; Saving, Kind=Trace] Channel finished. Elapsed 00:00:00.0615255.
~~~ [Source=ImageClassificationTrainer; BinarySaver; Saving, Kind=Trace] Channel disposed
~~~ [Source=MulticlassPredictionTransformer; Saving train schema, Kind=Trace] Channel finished. Elapsed 00:00:00.0722672.
~~~ [Source=MulticlassPredictionTransformer; Saving train schema, Kind=Trace] Channel disposed
~~~ [Source=Saving Schema, Kind=Trace] Channel started
~~~ [Source=BinarySaver; Saving, Kind=Trace] Channel started
~~~ [Source=EmptyDataView; Cursor, Kind=Trace] Channel started
~~~ [Source=EmptyDataView; Cursor, Kind=Trace] Channel finished. Elapsed 00:00:00.0002055.
~~~ [Source=EmptyDataView; Cursor, Kind=Trace] Channel disposed
~~~ [Source=EmptyDataView; Cursor, Kind=Trace] Channel started
~~~ [Source=BinarySaver; Write, Kind=Trace] Channel started
~~~ [Source=EmptyDataView; Cursor, Kind=Trace] Channel finished. Elapsed 00:00:00.0001
189.
~~~ [Source=EmptyDataView; Cursor, Kind=Trace] Channel disposed
Current step: SavingModel
Status: Finished
Message: Finished saving model
Step took: 00:00:04.4665177
------------------------------
------------------------------
Message: Total training took: 00:00:19.8574832
------------------------------
```

Рисунок 39 – Збереження моделі даних

### 4.3 Поєднання обох сценаріїв підготовки та тренування в єдиний сценарій

Поєднання сценаріїв дозволяє спростити процес створення моделі до одного кліку, але може бути не таким точним при використанні погано підібраного масиву даних. Але така можливість все одно є, та представлена у вигляді такого рішення:

```
public static async Task Main(string[] args)
{
   Console.WriteLine("Scenario `{0}` has been started",
typeof(Program).Assembly.GetName().Name);
   var stopwatch = Stopwatch.StartNew();
   await Preparation.Program.Main(args);
   await Train.Program.Main(args);
   stopwatch.Stop();
   Console.WriteLine();
   Console.WriteLine("Scenario `{0}` has been finished",
typeof(Program).Assembly.GetName().Name);
   Console.WriteLine("Scenario took: {0}", stopwatch.Elapsed);
}
```

Як бачимо з вище зазначеного коду, процес створення виглядає таким чином:

- Підготовка архіву (Preparation.Program.Main);
- Запуск процесу тренування (Train.Program.Main);
- Аналіз даних та звітування про використаний час (Scenario `{0}` has been finished);
- Завершення сценарію та збереження результату.

### 4.4 Розробка та використання функції API

Додаток ImageClassification.API складається з таких контролерів:

- ClassificationController;
- ClassifiersController;
- SourceController;
- StorageController;
- ErrorsController.

Кожен з цих контролерів відповідає за різні функції, але поєднує їх усіх один базовий контролер – ControllerBase.

Налаштування самого додатку виглядає наступним чином:

```
{
 "MLModel": {
  "MLModelFilePath": "ML/Classifiers/",
  "WarmupImagePath": "ML/TestImages/warmup.jpg"
 },
 "StorageOptions": {
  "StoragePath": "Storage/"
 },
 "ImageSourceUploadOptions": {
  "StringBefore": " ",
  "LeftBracer": "(",
  "RightBracer": ")"
 },
 "Logging": {
  "LogLevel": {
   "Default": "Information"
  }
 }
}
```

Основне завдання контролеру ClassificationController – це визначання категорію наданого зображення. Крім того він ще може отримувати дані щодо існуючих категорій на заданій моделі.

Наступний контролер ClassifiersController має такі функції:

- Перерахувати всі збережені класифікатори;
- Завантажує файл .zip, який представляє модель TensorFlow класифікатора.

SourceController використовується в свою чергу для отримання збережених зображень на сервері, а StorageController маємо для отримання таких функціональних завдань:

- Показує всі доступні папки в сховищі;
- Показує всі папки класифікації та інформацію про них для вказаної папки;
- Показує інформацію про зображення з індексацією;
- Завантаження, визначені параметрами зображення;
- Створює папку з назвою;
- Завантажує зображення в папку для подальшого навчання;
- Перейменовує папку з новою назвою;
- Видаляє папку зі списку;
- Видаляє зображення з класифікації.

Останній ErrorController використовується як фільтр помилок, для перехоплення виключень під час виконання додатку. Код який стосується обробки помилок виглядає наступним чином:

```
[Route("error")]
public IActionResult Error()
{
   var context = HttpContext.Features.Get<IExceptionHandlerFeature>();
   var exception = context?.Error;
   if (exception is null)
```

```
    {
        return NotFound();
    }
    IErrorData error = _exceptionMapper.Map(exception);
    _logger.LogError(exception, error.Message);
    var model = new ErrorVM
    {
        Message = error.Message,
        Data = error.Data
    };
    return StatusCode(error.StatusCode, model);
}
```

# ВИСНОВКИ

Під час виконання мого дипломного проекту було створено повноцінний веб–додаток для швидкого визначення емоції людини з готовою навченою моделлю штучної нейронні мережі.

Веб-додаток має зручний та інтуітивно-зрозумілий інтерфейс зроблений з використанням Swagger який в свою чергу має короткий опис усіх можливостей веб-додатку. Штучна нейронна мережа з високою ймовірністю може визначити емоцію людини. А саме одну з наступних:

- злість;
- огида;
- страх;
- щастя;
- нейтральна
- сум;
- здивування.

Серверна частина веб-додатку зроблена на основі сучасної многофункціональної платформи .NET 6 з використанням нової версії ML.NET, що в свою чергу забезпечує велику швидкість роботи штучної мережі, у середньому визначення емоції відбувається за 100 мілісекунд, а іноді ще швидше. Також є можливість розширювати існуючий датасет новими даними, або швидко додати новий з іншими категоріями.

Зрозуміла документація Swagger та використання API підходу розробки в свою чергу дає легку можливість розширення додатку, а саме інтеграцію його з різними фреймворками як Angular, View.js і тому подібних, а також інтеграцію з мобільними додатками.

Підводячи підсумки можна сказати, що було створено зручний веб-додаток для швидкого визначення емоції людини за фотографією з високою ймовірністю за допомогою штучної нейронні мережі.

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Штучні нейронні мережі URL: https://www.osp.ru/cw/2001/09/39289

2. Особливості нейромережевих рішень, переваги та недоліки, перспективи застосування URL: https://cyberleninka.ru/article/n/osobennosti-neyrosetevyh-resheniy-dostoinstva-i-nedostatki-perspektivy-primeneniya

3. Основні види штучних нейронних мереж URL: https://studbooks.net/2136467/matematika_himiya_fizika/osnovnye_vidy_iskusstvennyh_neyronnyh_setey

4. Handbook on Neural Information Processing / M. Bianchini, M. Maggini, L.C. Jain (eds.). Springer, 2013. P. 34-89.

5. Artificial Neural Networks – Application / C.-L. Hui(ed.). In Tech, 2011. P. 37-58.

6. Artificial Neural Networks. Methods and Applications in Bio-/Neuroinformatics / P. Koprinkova-Hristova, V. Mladenov, N.K. Kasabov (eds.). Springer, 2015. P. 13-19.

7. Steimer A., Maass W., Douglas R. Belief Propagation in Networks of Spiking Neurons // Neural Computation. 2009. 21. P. 2502-2523.

8. Створення нейромережевої моделі із застосуванням структурованих знань URL: https://cyberleninka.ru/article/n/sozdanie-neyrosetevoy-modeli-po-metodu-neyrosetevoy-approksimator-s-primeneniem-strukturirovannyh-znaniy

9. Особливості Visual Studio Community URL: https://visualstudio.microsoft.com/vs/community/

10. Переваги Microsoft Visual Studio URL: https://visualstudio.microsoft.com/vs/

11. Платформа розробки .NET 6 URL: https://learn.microsoft.com/en-us/dotnet/core/whats-new/dotnet-6

12. Нововведення у C# 10 URL: https://learn.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-10

13. Нововведення в System.Text.Json Namespace URL: https://learn.microsoft.com/en-us/dotnet/api/system.text.json?view=net-7.0

14. Нововведення у ASP.NET Core 6 URL: https://learn.microsoft.com/en-us/aspnet/core/release-notes/aspnetcore-6.0?view=aspnetcore-7.0

15. Фреймворк ML.NET URL: https://en.wikipedia.org/wiki/ML.NET

16. ML.NET API URL: https://learn.microsoft.com/en-us/dotnet/machine-learning/how-does-mldotnet-work

17. Навчання моделі класифікації ML.NET для категоризації зображень URL: https://learn.microsoft.com/en-us/dotnet/machine-learning/tutorials/image-classification