МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему:

Програмний модуль класифікації блюд меню

Галузь знань **12 «Інформаційні технології»**
Спеціальність **122 «Комп'ютерні науки»**
Освітня програма «**Комп'ютерні науки»**
Освітній рівень: бакалавр

Виконав: студент 4 курсу, групи КН- 41


Панченко_В.В.
(прізвище та ініціали)

Керівник:____Іларіонов_О.Є._____
(прізвище та ініціали)
_____к.т.н._доцент___
(науковий ступінь, звання)

Випускна кваліфікаційна робота бакалавра допущена до захисту
рішенням кафедри *інтелектуальних технологій*
Протокол № 11 від 06.06.2022 р.
зав. кафедри_____ доц. Іларіонов О.Є.

Київ - 2022

Факультет інформаційних технологій

Кафедра інтелектуальних технологій

Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ

Завідувач кафедри

інтелектуальних технологій

Іларіонов О.Є.

"___"_____2022 р.

ЗАВДАННЯ

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Панченка Владислава Володимировича

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи): Програмний модуль класифікації блюд меню

затверджена протоколом засідання кафедри від « 23 » грудня 2021 р. №4

2. Термін здачі студентом закінченого проекту (роботи) 29 травня 2022 року

3. Вихідні дані до проекту (роботи): Програмний модуль класифікації блюд меню

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

4.1. Налаштування та навчання нейронної мережі

4.2. Реалізація програмного модуля класифікації блюд з меню

4.3. Аналіз точності роботи програмного модулю

5. Перелік презентаційного матеріалу (з точним зазначенням обов'язкових презентацій)

5.1. Характеристика та особливості сфери класифікації зображень 1-4 слайди

5.2. Архітектура програмного модулю 5-8 слайди

5.3 Особливості реалізації програмного модулю 10-12 слайди

5.4. Технологічні особливості реалізації програмного модуля 12-13 слайди

5.5. Демонстрація роботи програмного модулю 14 слайд

5.6. Висновки 15 слайд

6. Консультанти з випускної кваліфікаційної роботи із зазначенням розділів випускної кваліфікаційної роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1			
2			
3			

7. Дата видачі завдання 15 лютого 2022 року

Керівник_____ / Іларіонов О.Є. /

(підпис) (ПІБ)

Завдання прийняв до виконання_____/ Панченко В.В. /

(підпис) (ПІБ)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів випускної кваліфікаційної роботи	Примітка
1	Обговорення з керівником постановки завдання та змісту пояснювальної записки	25.01.2022 - 22.02.2022	
2	Аналіз постановки задачі, формалізація задачі, вибір методів та засобів реалізації поставленої задачі, аналіз літературних джерел	23.02.2022- 01.03.2022	
3	Проектно-технічна реалізація програмного модулю класифікації зображення	01.03.22 – 18.03.22	
4	Розробка та тестування програмного модулю класифікації зображення	19.03.22 – 30.04.22	
5	Оформлення пояснювальної записки, підготовка презентації	01.05.2022 - 10.05.2022	

Студент-дипломник _____/ В.В. Панченко /

(підпис) (ПІБ)

Керівник випускної кваліфікаційної роботи _____/ О.Є.Іларіонов /

(підпис) (ПІБ)

Анотація

Панченко Владислав Володимирович виконав випускну кваліфікаційну роботу на тему «Програмний модуль класифікації блюд меню» за спеціальністю 122 – «Комп’ютерні науки».

У випускній роботі проведено аналіз методів класифікації зображень, розглянуто архітектуру модулю, розроблено програмний модуль, що виконує класифікацію блюд меню із використанням згорткових нейронних мереж.

Ключові слова: продукти харчування, машинне навчання, згорткова нейронна мережа, InceptionV3, Xception

Annotation

Panchenko Vladislav Volodimirovich completed the graduate qualification work on " Program module for the classification of dishes in the menu" in the specialty 122 - "Computer Science".

In the graduate work the analysis of methods of classification of images, the architecture of the module is considered, the software module that performs the classification of menu dishes using convolutional neural networks was developed.

Keywords: food, machine learning, convolutional neural network, InceptionV3, Xception

Зміст

Вступ

Зі збільшенням коштів та зусиль, що владуються до сфери штучного інтелекту постає все більше питань, щодо використання плодів цих витрат. На даний момент для звичайних людей не дуже цікаво, що за допомогою штучного інтелекту можна знайти більше галактик із даних, що відсилають космічні телескопи на землю, що за допомогою можна більш точно прогнозувати погоду чи покращувати якість зображень. Для багатьох технології штучного інтелекту вважаються іграшкою, інколи дуже лякаючою , наприклад вистежування обличь за допомогою камер в Китаї або штучний інтелект на складах Amazon для стежки за робітниками, що вже звільнив багато людей. Але штучний інтелект може покращити життя і для звичайних людей, наприклад допомога лікарям при діагностуванні захворювань, зменшення часу проведеного в чергах або для більш зручної покупки товарів на вагу(овочів, фруктів, круп). Не потрібно відноситися до технологій штучного інтелекту як до дуже дорогих іграшок, що допомагає тільки великим компаніям, навіть із обмеженими ресурсами можна покращити життя багатьом людям.

Розділ 1. Аналіз технологій класифікації зображень

1.1 Аналіз області класифікації зображень

Розпізнання образів є одною із найпоширеніших галузей штучного інтелекту, воно надає багато можливостей для багатьох сфер діяльності. Але для даної роботи буде розглянута задача класифікації зображень, а саме зображень продуктів харчування. Програмна реалізація вирішення цієї задачі може бути цікава багатьом людям: власникам магазинів, покупцям та туристам. Програмне забезпечення, що може класифікувати зображення дає змогу зменшити черги, допомогти покупцям із покупкою товарів на вагу або допомогти туристам знайти назву їжі, що вони бачать без намагань дізнатися щось від місцевих жителів. На рисунку 1 можна побачити деякі приклади практичного використання наявних рішень в цій області.

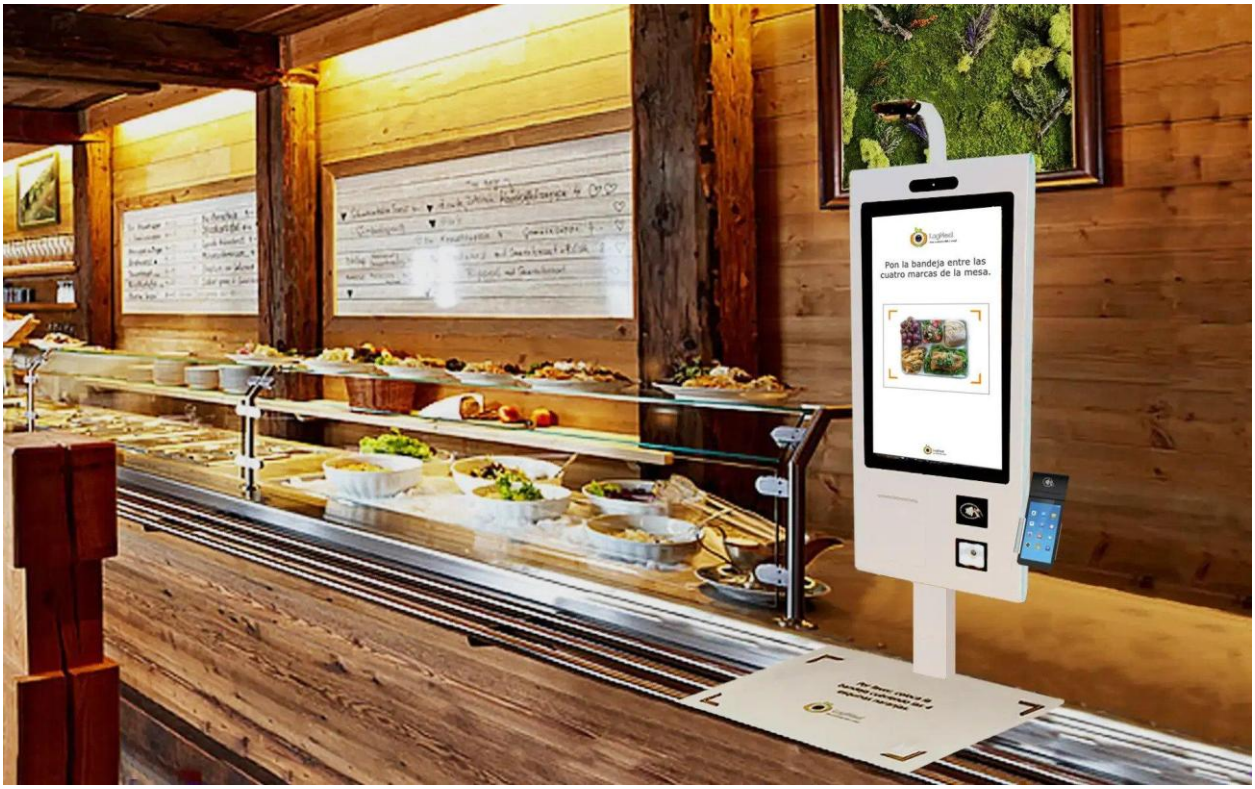


Рисунок 1 Безконтактний кіоск для самообслуговування в ресторані

У зв’язку з пандемією Covid-19 актуальність розробки подібних систем тільки зростатиме, бо це зменшує кількість людей, що перебуває в зоні ризику за допомогою більш швидкого обслуговування та меншої кількості контактів між собою та можливо зараженими поверхнями. Також для використання подібних систем не потрібно особливого обладнання , достатньо майже будь-якого комп’ютера та камери.

1.1.1 Класифікація зображень

Основна задача класифікації зображень – це виявлення класу об’єкта, що заданий на певному багатоканальному (червоний, зелений та синій) зображенні. Тобто, якщо на картинці зображено кота, то програмі необхідно видати результат, що збігається із тим, що представлено на картинці, приклад можна побачити на 2 рисунку[1][2].

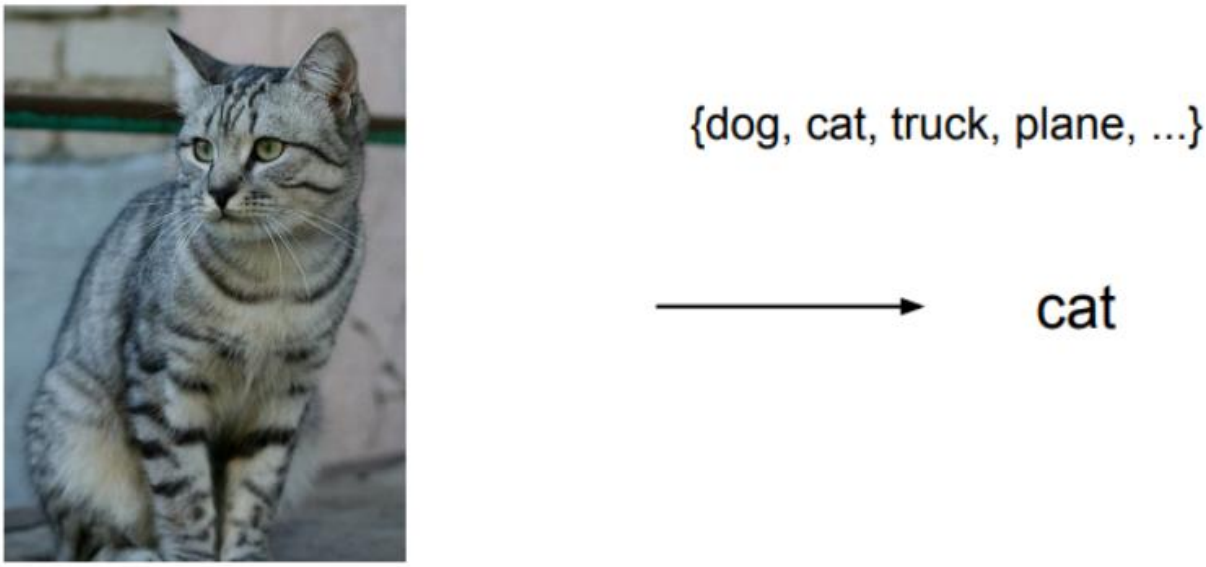


Рисунок 2 процес класифікації зображень

Проста, на перший погляд задача, має дуже багато складностей, що не дуже заважають людині, але роблять задачу вкрай тяжкою для програми. Це випливає з того, що зображення на комп'ютері – це матриця чисел і комп'ютер її сприймає так, а не як ціле зображення, тому будь-яка зміна положення, нахилу, повороту чи поява будь-якого шуму робить задачу класифікації невирішальною для звичайних методів таких як: порівняння еталонів, найближчого сусіда та к-найближчих сусідів. Ця проблема – називається «семантичний розрив» - нерозуміння інформації, що є в даних. Таким чином обирати такі прості методи недоречно і потрібно провести дослідження цих, та інших методів класифікації зображень.

1.1.2 Аналіз методів класифікації зображень

Кожен із методів класифікації зображень має свої переваги та недоліки такі як швидкість або точність роботи, складність використання, потреба у апаратних ресурсах та інші якості. Для даної дипломної роботи найбільш важливими є точність та швидкість роботи і невелика потреба у ресурсах. Для аналізу точності методів класифікації зображень буде використано дослідження «Image-based Food Classification and Volume Estimation for Dietary Assessment» в якому детально описано використання як «традиційних» методів класифікації зображення, так і методів класифікації зображень із використанням штучних нейронних мереж в області класифікації продуктів харчування. Результати дослідження роботи традиційних методів класифікації можна побачити на рисунку 3.

Authors	Methods	Types	Top 1
Kong et al. [18], 2011	Nearest-neighbour classifier; DoG and SIFT	61-food classes	84.0%
Kong et al. [19], 2012	Nearest-neighbour classifier; SIFT and visual words	5-food classes	92.0%
Kawano et al. [24], 2014	one-vs-rest classifier; Fisher vector with RootHoG	256-food classes	50.1%
He et al. [20], 2014	KNN classifier; DCD, MD-SIFT, SCD and SIFT	42-food classes	64.5%
Tamma. et al. [25], 2014	SVM classifier; BoF, SFTA and color histogram	5-food classes	70.0%
Anthim. et al. [3], 2014	SVM classifier; BoF, hsvSIFT and color moment invariant	11-food classes	78.0%
Pouladz. et al. [26], 2015	SVM classifier; color, texture, size, shape features	30-food classes	94.5%
Beijbom et al. [22], 2015	SVM classifier; color, HOG, SIFT, LBP, MR8 filter and LLC	50-food classes	77.4%

*Top 1 accuracy can be computed using (TP + TN)/(TP + TN + FP + FN)

Рисунок 3 Результати дослідження точності "традиційних методів"

[18]Виходячи із результатів дослідження можна побачити, що «традиційні» методи можуть непогано працювати на невеликій кількості класів, видаючи непогану точність top 1, але в основному точність роботи методів недостатня для цієї роботи, треба зазначити, що можна побачити на рисунку деякі методи, що досягли дуже непоганої точності, але це відбулося на невеликих за кількістю класів датасетах, окрім методу SVM. Необхідно також згадати те, що використання традиційних методів пов’язано із декількома проблемами, що не мають простого рішення, головною з яких є виділення ознак з малюнку, що класифікується, без вирішення цієї проблеми ніякий із цих методів працювати не буде, теж саме можна сказати і про звичайні штучні нейронні мережі, що також не дуже добре працюють без вирішення цієї проблеми.

Таким чином ця проблема є серйозною перешкодою для побудови моделі, що буде виконувати класифікацію. Отже для використання цих методів має бути дуже поважна причина, якою може бути точність роботи моделі.

Таким чином можна перейти до наступного етапу розвитку цієї галузі – появи штучних нейронних мереж глибинного, для яких вищезгадана проблема не є серйозною. Результати використання цих методів в вищезгаданому дослідженні можна побачити на 4 рисунку.

Authors	Methods	Dataset	Top 1
Kagaya et al. [27], 2014	CNN	Own dataset	73.7%
Christ. et al. [28], 2015	Patch-wise model+CNN	Own dataset	84.9%
Meyers et al. [29], 2015	CNN+inception	Food-101	79.0%
Liu et al. [30], 2016	CNN+inception	Food-101	77.4%
Pandey et al. [31], 2017	Ensemble Net	Food-101	72.1%
Liu et al. [17], 2018	CNN+inception	Food-101	77.0%
Yu et al. [32], 2018	DLA	Food-101	90.0%
Foresti et al. [33], 2018	WISeR	Food-101	90.3%
Cui et al. [34], 2018	DSTL	Food-101	90.4%
Qiu et al. [35], 2019	PAR-Net	Food-101	90.4%
Min et al. [36], 2019	IG-CMAN	Food-101	90.4%
Tan et al. [37], 2019	EfficientNet	Food-101	93.0%
Kawano et al. [21], 2014	Pre-trained CNN	UEC-FOOD100	72.3%
Yanai et al. [38], 2015	CNN	UEC-FOOD100	78.8%
Hassane. et al. [39], 2016	Inception V3	UEC-FOOD100	81.5%
Liu et al. [30], 2016	CNN+inception	UEC-FOOD100	76.3%
Foresti et al. [33], 2018	WISeR	UEC-FOOD100	89.6%
Yanai et al. [38], 2015	CNN	UEC-FOOD256	67.6%
Hassane. et al. [39], 2016	Inception V3	UEC-FOOD256	76.2%
Liu et al. [30], 2016	CNN+inception	UEC-FOOD256	54.7%
Foresti et al. [33], 2018	WISeR	UEC-FOOD256	83.2%

*Top 1 accuracy can be computed using (TP + TN)/(TP + TN + FP + FN)

Рисунок 4 Результати дослідження точності методів глибинного навчання

Отже з результатів дослідження можна з легкістю стверджувати, що підстав використовувати «традиційні» методи класифікації зображень не є доцільно, що підтверджується високою точністю методів на великій кількості класів. Але всі методи пов’язані із глибинним навчанням мають один дуже серйозний недолік – це велика потреба у апаратних ресурсах при навчанні, але подолання цієї проблеми набагато легше, ніж власноручна реалізація виділення ознак, до речі методами глибинного навчання і вирішуються ця проблема автоматично, що можна побачити на рисунку 5.

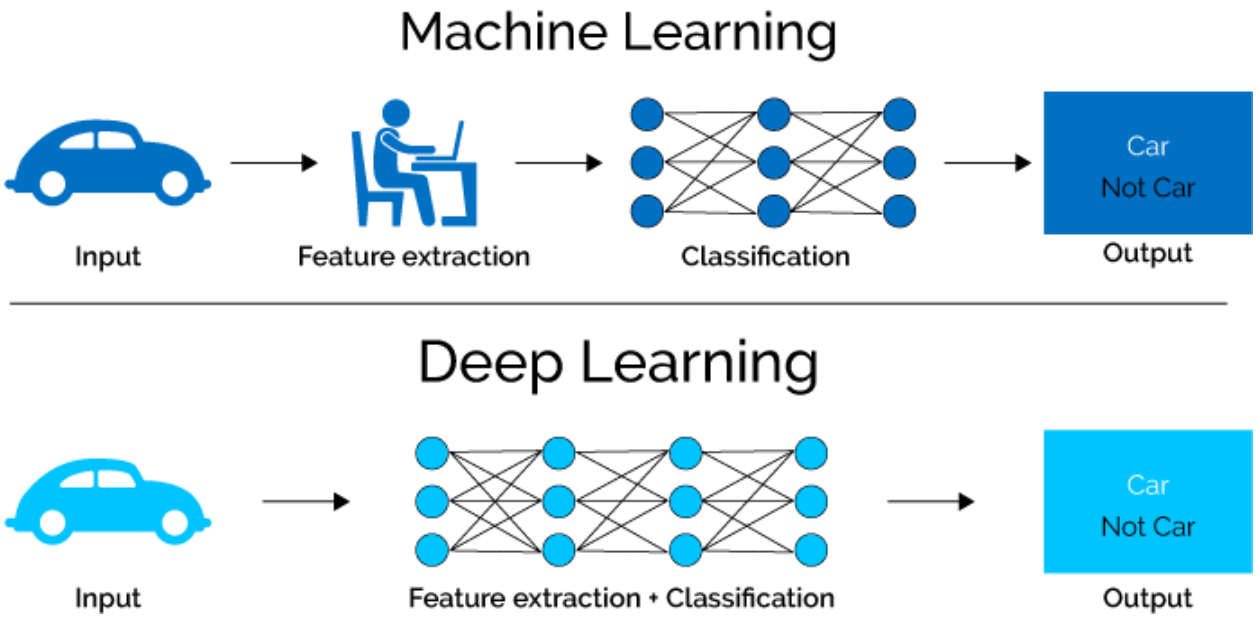


Рисунок 5 Порівняння процесу класифікації обох типів методів

Таким чином доречно для виконання цієї роботи обрати штучні нейроні мережі глибинного навчання.

1.2.1 Короткі історичні відомості щодо нейронних мереж та глибинного навчання

Першим кроком у галузь штучного інтелекту зробили Уоррен Маккалох і Уолтер Піттс написавши статтю у 1943 про роботу штучних нейронів та представлення моделі нейромережі на електронних схемах. Не слід забувати і прогрес у створені моделі людського навчання, що зможе дати надзвичайні результати у тодішньому майбутньому.

Перші штучні нейромережі з’явилися на переломі 50-60 років двадцятого століття, це було зроблено групою вчених та дослідників за керівництва Натаніеля Рочестера завдяки поєднанню вищеописаних досягнень.

Ці успіхи викликали спалах неабиякого оптимізму щодо штучного інтелекту, Розенблат, Відроу, Мінські та багато інших вчених розробили перші штучні нейроні мережі – перцептрони(рис 6), що складалися з одного прошарку штучних нейронів. Вони використовувалися для розв’язання великого переліку задач: штучний зір, прогноз погоди, аналіз кардіограм та багато інших.

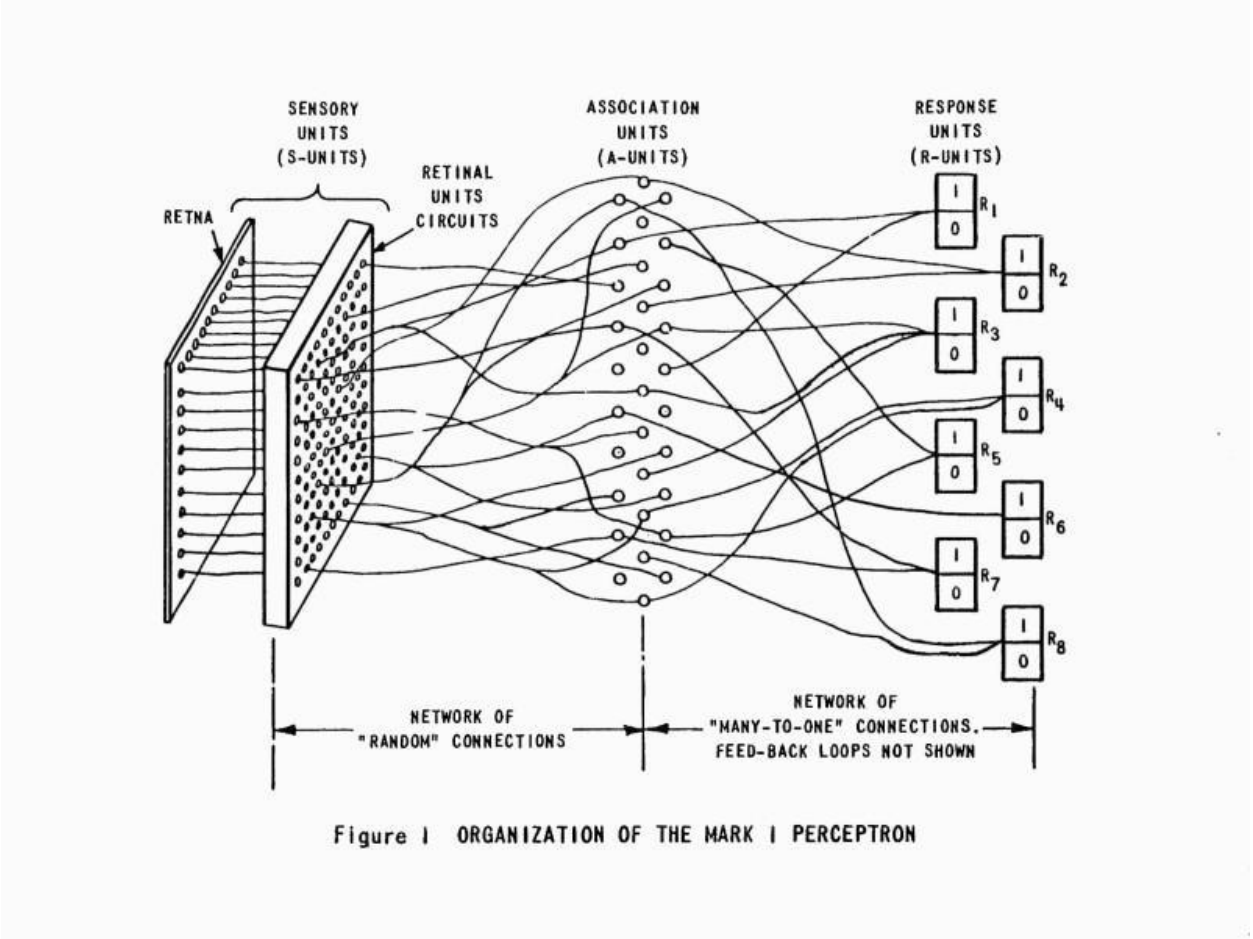


Рисунок 6 Перші перцептрони

Але під час вибуху великого оптимізму виникають надвеликі очікування, які зазвичай дуже рідко виправдовуються. Згодом так і трапилося, оптимізм схлинув, а песимізм стосовно технологій штучного інтелекту став зростати, так продовжувалося до 80 років двадцятого століття, коли всі ці песимістичні аналізи дали результат. В 80 роках до технологій штучного інтелекту різко зріс інтерес, спричинений зацікавленістю людей із інших сфер діяльності – філософів, інженерів-конструкторів тощо.

Надзвичайно великий внесок у розвиток галузі штучних нейронних мереж було зроблено у 1986 році використанням алгоритму backpropagation, в статті Румельхарта, Хінтона і Вільямса, що дало поштовх всій галузі і спричинило появу у 1989 році багатошарової штучної нейронної мережі прямого поширення. Ці дві події стали початком нового розквіту галузі штучного інтелекту. З цих років події стали розвиватися настільки швидко, що за ними неможливо було слідкувати.

Останньою надзвичайно важливою подією стало створенням при університеті Торонто у 2007 році Джефрі Хінгтоном алгоритмів глибинного навчання для багатошарових нейронних мереж. Що спричинило новий спалах інтересу до нейромеж, що сильно пропав у 2000 роки.

1.2.2 Нейронні мережі

Нейромережі, або штучні нейроні мережі – це галузь машинного навчання, що є програмною імітацією роботи біологічних нейронів людського мозку для вирішення певних задач. Основними поняттями для розуміння штучних нейронних мереж є: штучний нейрон, шари та навчання[3].

Штучний нейрон – це математична імітація роботи біологічного нейрону, цей елемент є базовим для штучної нейронної мережі. Цей елемент має структуру, подібну до біологічного нейрону, яка складається із входів, ваг, активаційної(передаточної) функції та виходу, детально структуру штучного нейрону можна побачити на рисунку 7.

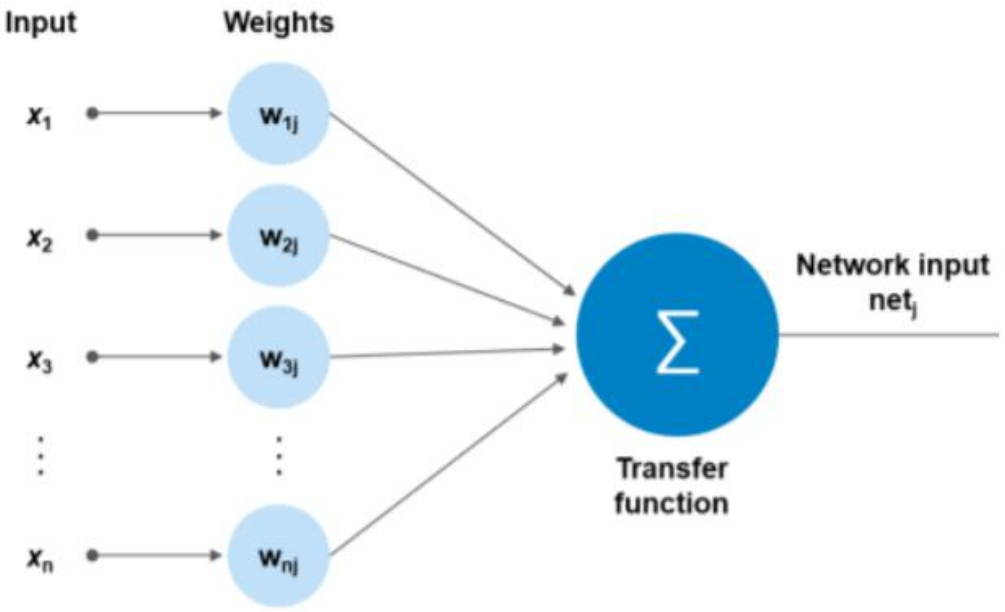


Рисунок 7 Структура штучного нейрону

Як можна побачити на 7 рисунку, сигнал(число) находить до штучного нейрону через вхід, потім потрапляє до ваг, де він множиться на значення цих ваг, потім всі сигнали, що вийшли із ваг додаються та отриманий результат надходить до активаційної функції, де він трансформується до потрібного вигляду та надходить до входу іншого штучного нейрону.

Архітектура нейронної мережі це з’єднання декількох типів шарів: вхідного, прихованого та вихідного, приклад можна побачити на рисунку 8. Кожен шар це сукупність штучних нейронів, що передають та перетворюють сигнали. Таким чином проводяться необхідні математичні розрахунки, що дозволяють вирішити певну задачу.

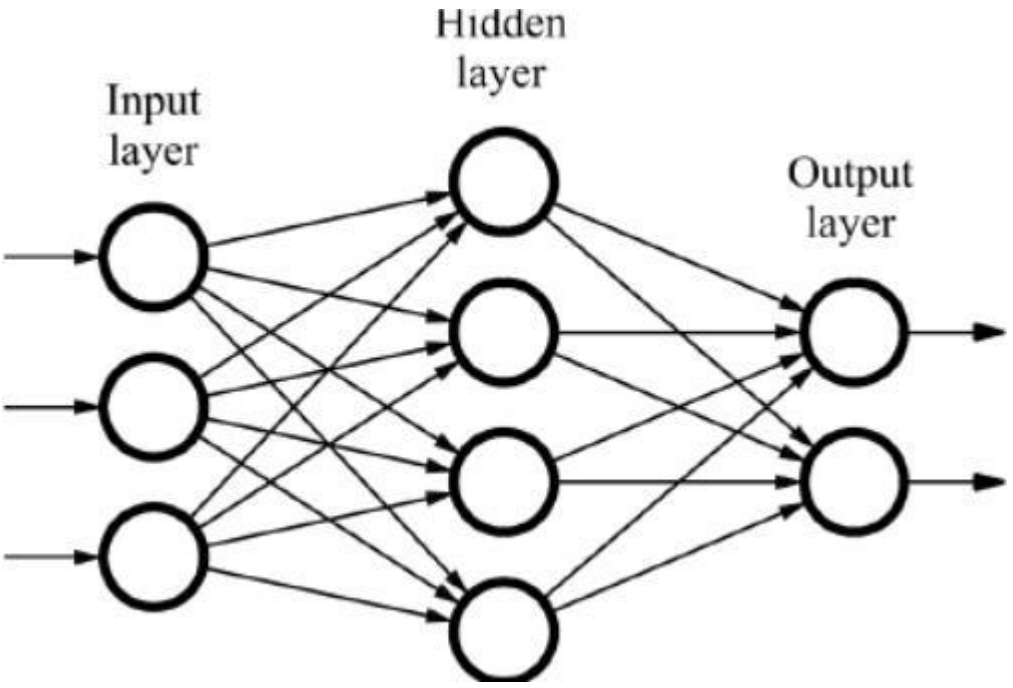


Рисунок 8 Архітектура нейромережі

Для того, що б математичні обрахунки сприяли розв’язку задачі виконується процес навчання, який необхідний для того, що б результат отриманий мережею був наближений до необхідного.

Навчання нейронної мережі це процес оптимізації функції похибки штучної нейронної мережі в якості аргументів якої ваги нейромережі. Функція похибки – показник того наскільки результат, що видає нейромережа відрізняється від реального. Оптимізація функції може виконуватися різними методами починаючи від градієнтного спуску і закінчуючи генетичними алгоритмами, але найбільш поширеним методом оптимізації функції є метод зворотного поширення похибки(backpropagation).Основна ідея якого в поширенні значення похибки по всім вагам мережі. Процес навчання може бути як із вчителем(supervised learning), так і без вчителя(unsupervised learning), але другий варіант не може бути застосований до задачі класифікації, бо не будуть відомі класи на які необхідно розподілити варіанти. Але, якщо навіть дуже добре навчити нейромережу, то задача класифікації зображень не може бути вирішена через її надзвичайно велику складність. Тому і виник підвид машинного навчання як глибинне навчання(deep learning), основна ідея якого в підвищенні кількості прихованих шарів, приклад цієї архітектури можна побачити на 9 рисунку.

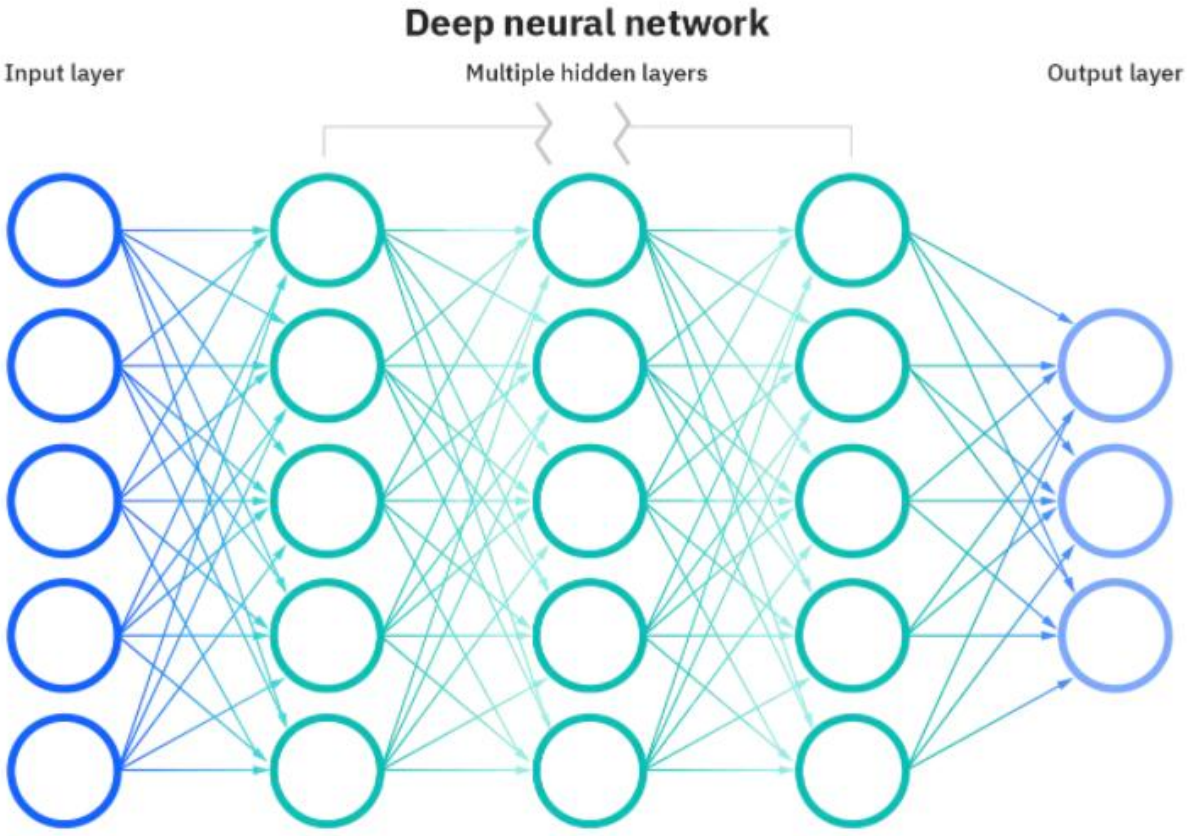


Рисунок 9. Архітектура глибинної нейромережі

Для вирішення задачі поставленої в даній роботі буде використовуватися підвид штучних нейронних мереж глибинного навчання відомий як згорткова нейронна мережа(convolutional neural network).

1.2.3 Згорткові нейронні мережі

Згорткова нейрона мережа – це підвид мереж глибинного навчання, що є дуже використовуваним у задачах комп’ютерного зору, завдяки її можливості витягувати із зображення певні образи, аспекти, з якими і буде відбуватися подальша робота, що надає їй можливості обробляти зображення з дуже високою варіативністю позицій, нахилів, кольорі тощо. Згорткова нейронна мережа складається із 3 типів шарів: вхідного, згорткового, об’єднуючого та повно зв’язного та вихідного шару, що можна побачити на 7 рисунку[10][11].

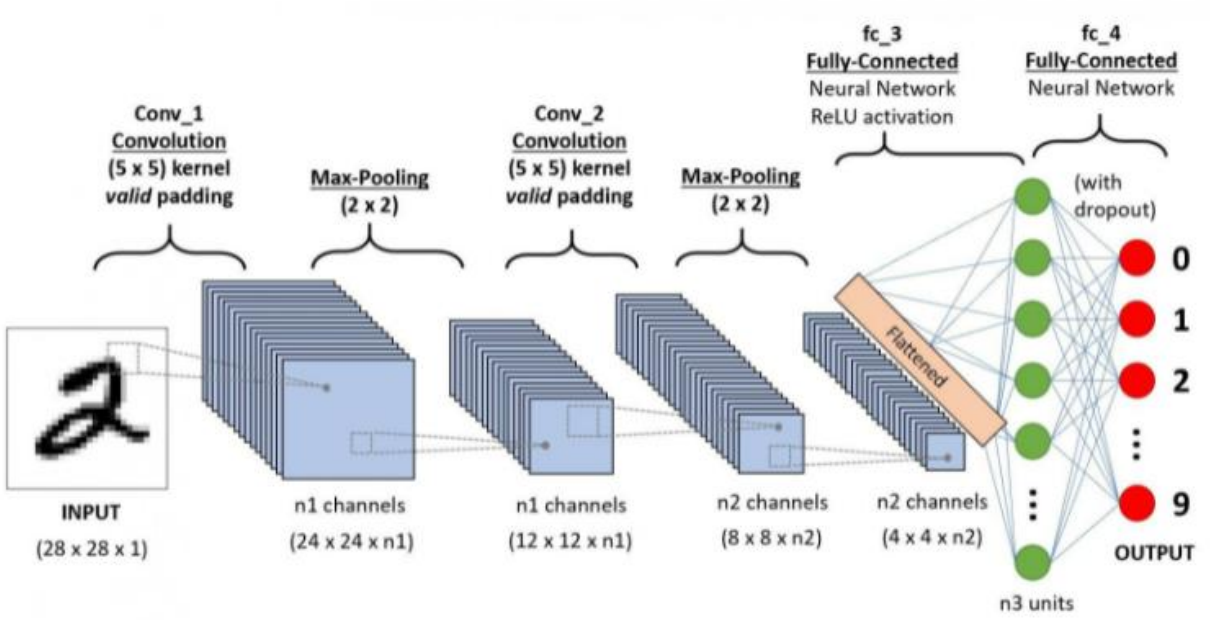


Рисунок 10 Архітектура згорткової нейронної мережі

Згортковий шар в цій мережі виконує пониження кількості даних без втрати значень, що мають велику роль у класифікації зображення. Це виконується завдяки фільтру/ядру(kernel), який певним чином обробляє певний шматок зображення і зберігає його у виді одного значення, таким чином, зменшуючи кількість значень в подальшій роботі. Цей процес показано на 8 рисунку.

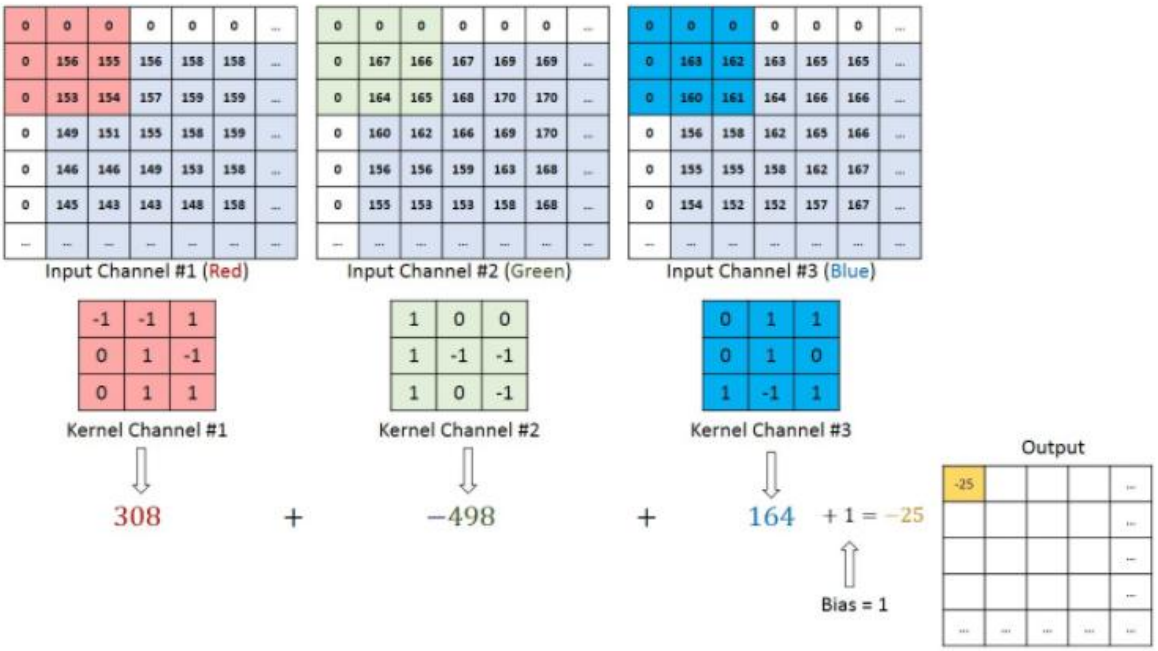


Рисунок 11 Приклад роботи згорткової нейронної мережі

Другим особливим шаром в згортковій штучній мережі є об’єднуючий шар який виконує схожу роль із попереднім шаром, але іншим чином. Також використовується ядро певного розміру, але обрахунки тут значно легші і бувають двох типів: максимальне(max-pooling) та середнє(average-pooling), які беруть максимальне або середнє значення та зберігають його. Приклад роботи цього шару показано на 12 рисунку.

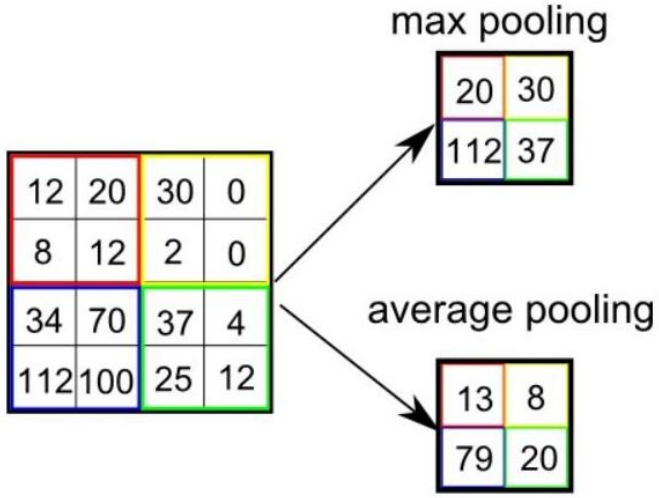


Рисунок 12 Приклад роботи об'єднуючого шару

Повно зв’язний шар це звичайний багаторівневий перцептрон який і необхідно буде навчати для класифікації зображень.

1.2.4 Вибір архітектури нейронної мережі

Для згорткових штучних нейронних мереж розроблено багато архітектур, що мають різну ефективність в різних умовах та певні параметри швидкості навчання та швидкості дії, через обмеженість ресурсів більш точні архітектури такі як Inception-V4, Inception-ResNet та NASNet-A-Large використовуватися не будуть, замість цього будуть використовуватися інші архітектури такі як: Inception-V3 та Xception. На 13 рисунку можна побачити графічне порівняння точності та необхідного апаратного ресурсу і кількості параметрів.

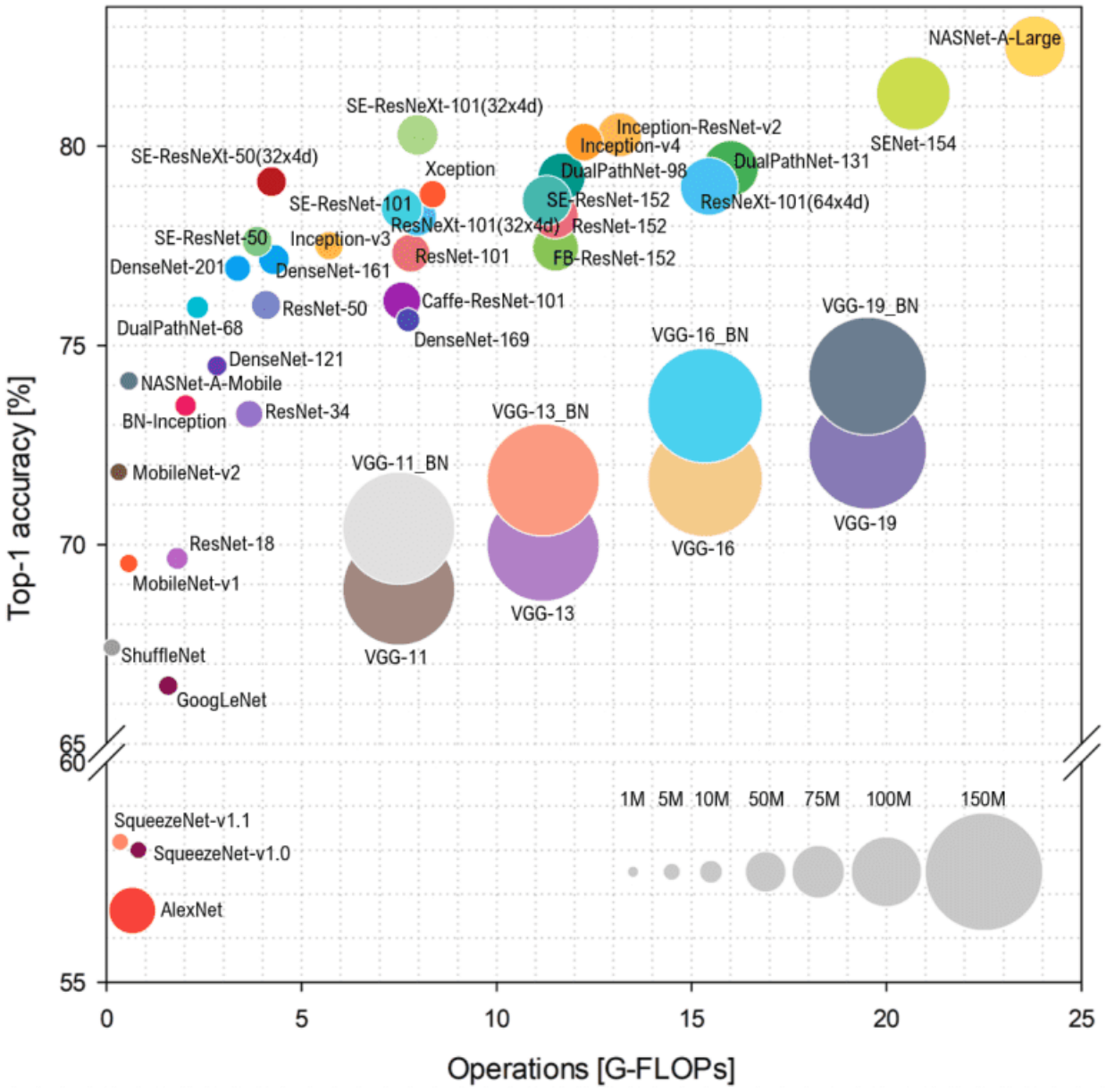


Рисунок 13 Графічне порівняння ефективності архітектур мереж

Як можна побачити з графіку не завжди збільшення складності моделі дає збільшення точності.Далі буде коротко описано кожну із використаних архітектур.Inception-V3 – це архітектура згорткових мереж, що була запропонована в 2015 році в статті «Rethinking the Inception Architecture for Computer Vision» командою із 5 науковців. Основна мета цієї архітектури в зменшенні кількості обрахунків при збережені точності нейромеррежі, це вдалося зробити завдяки деяким ідеям на яких ця архітектура заснована. Перша ідея, що використана і цій архітектурі – це заміщення великих фільтрів декількома меншими фільтрами, що дозволяє отримати перевагу в швидкості розрахунків при відсутності втрати якості.

Друга не менш важлива ідея в тому, що якщо фільтри NxN замінити на фільтри 1xN та Nx1 то кількість параметрів зросте, що буде добре впливати на точність роботи. Ще однією ідеєю, що скористалися науковці було добавлення допоміжного класифікатора помилка якого додавалася до основного при навчанні. На 14 рисунку показано саму архітектуру нейронної мережі[1].

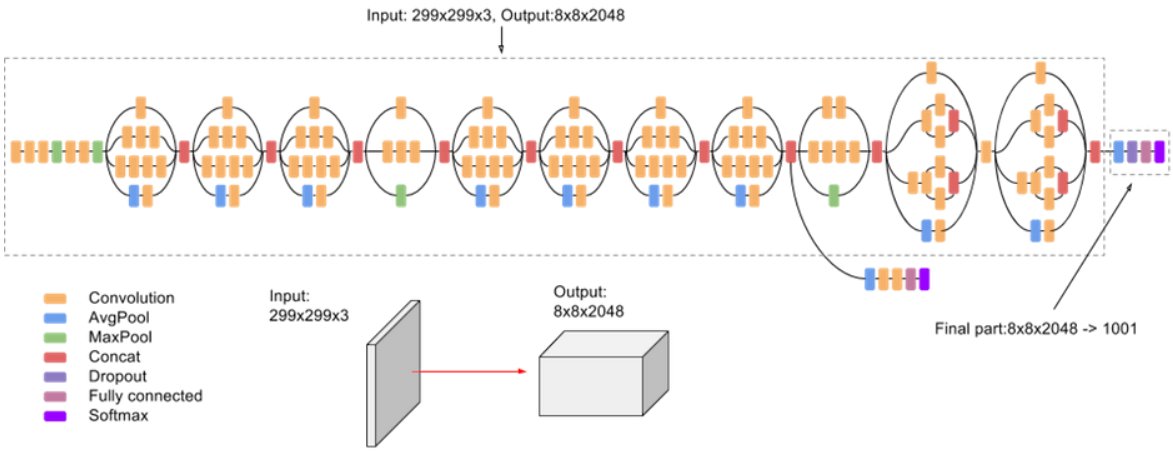


Рисунок 14 Архітектура ШНМ Inception-V3

Xception – це архітектура, що була запропонована в статті 2016 року «Xception: Deep Learning with Depthwise Separable Convolutions». Як можна побачити за назвою вона схожа на попередню, така логіка простежується не тількі в назві, але й в самій архітектурі. В цій архітектурі використовуються змінені inception модулі які мають архітектуру, що показано на 11 рисунку[15].

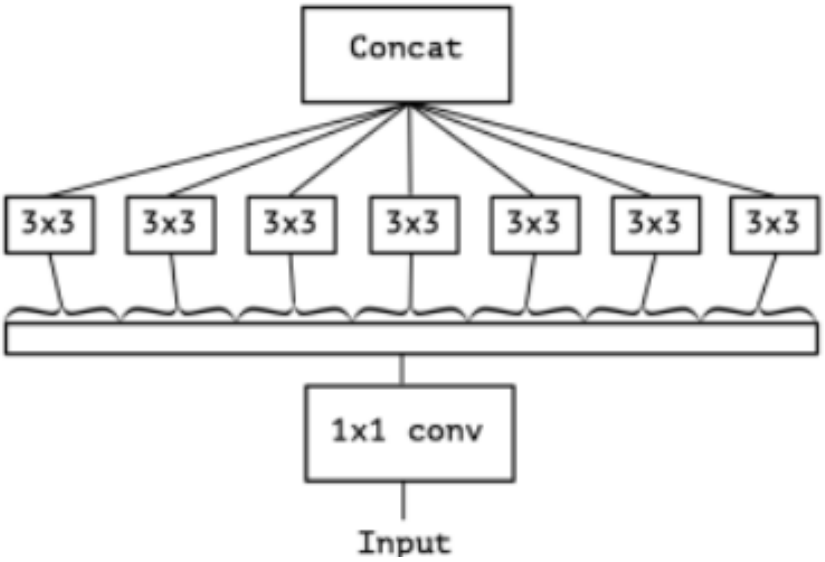


Рисунок 15 Модуль Inception

Така архітектура дозволяє не збільшуючи кількість параметрів отримати додаткову точність роботи. Повністю архітектура показана на 16 рисунку.

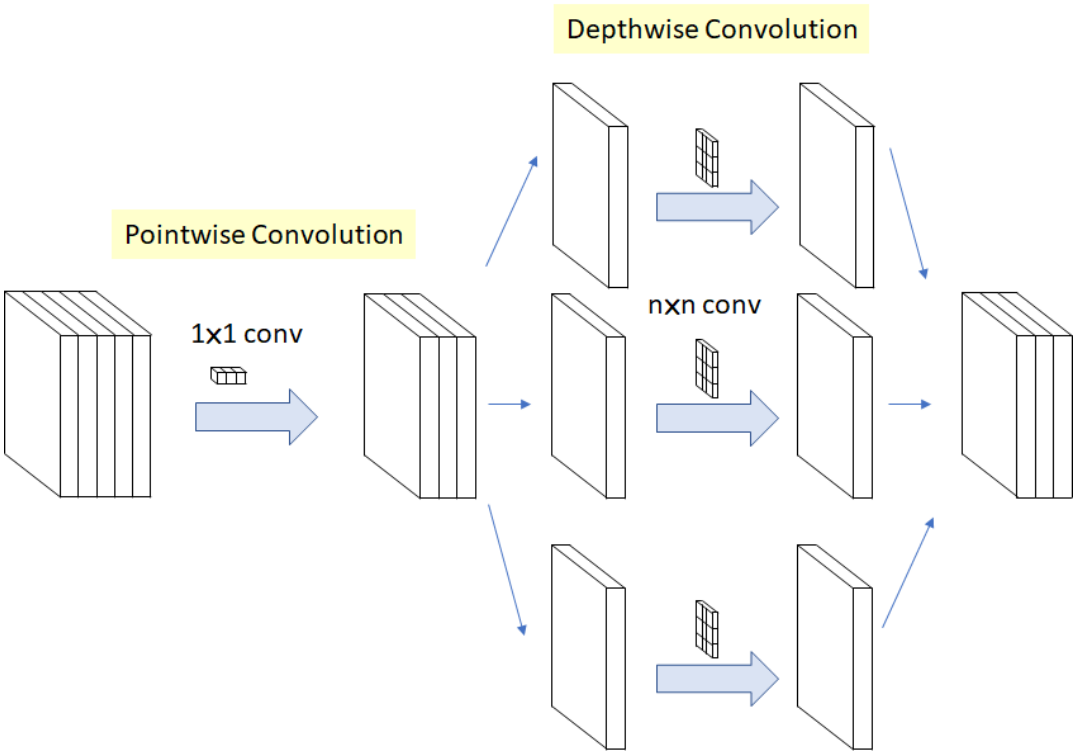


Рисунок 16 Архітектура Xception

1.3 Аналіз існуючих рішень

Розпізнавання продуктів харчування за зображеннями є дуже потрібною та швидко розвивається. Тому багато компаній вкладають зусилля та кошти в цю галузь, нижче буде наведено найяскравіші приклади програмних продуктів, що були розроблені цими компаніями, кожен з яких дуже якісно виконує ті ці, для яких створений. Але у всіх цих продуктів є декілька недоліків: вони коштують великих грошей або є вузько-спеціалізованими в одній сфері і не можуть бути використані в інших, тому більшість покупців не можуть скористуватися цими продуктами. Далі будуть наведені приклади цих продуктів:

Foodvisor – Програма, що створена для слідкування за дієтою, що за допомогою розпізнавання продуктів харчування може повністю розкрити зміст продуктів по компонентам: білки, жири, вуглеводи та клітковина. З даною можливостю спортсмени, люди, що слідкують за своїм харчуванням та тим людям, що створюють дієти можуть планувати свій раціон з вражаючою точністю та ефективністю, що дуже позитивно впливає на їх здоров’я. На рисунку 17 показано приклад роботи даного застосунку[9].

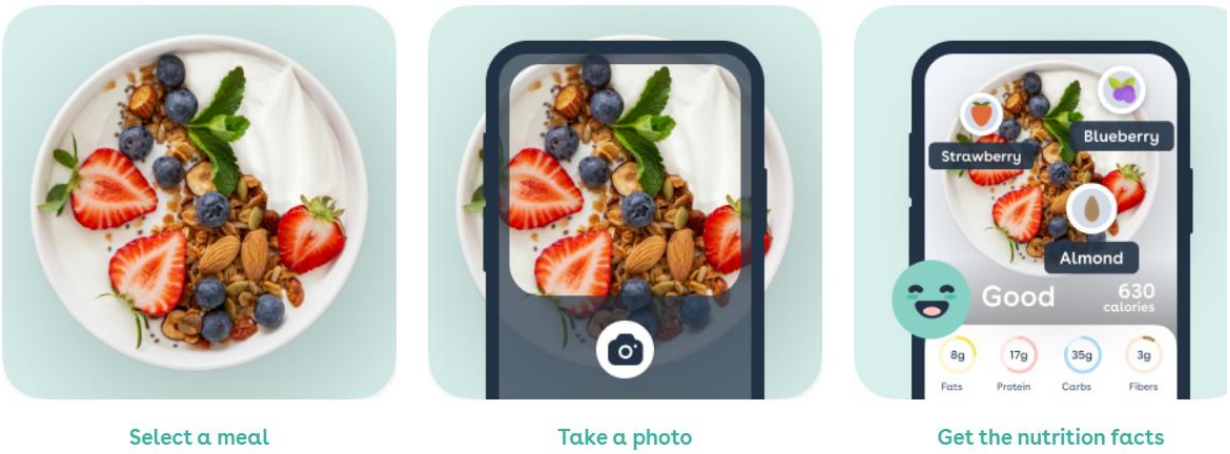


Рисунок 17 приклад роботи застосунку

LogMeal – API-шка, що створена командою науковців із 30 річним стажем в галузі комп’ютерного зору, яка написали 25 наукових статей в сфері розпізнавання продуктів харчування. Дана API-шка, за допомогою технологій штучного інтелекту, може дуже точно розпізнавати близько 880 різних продуктів харчування. Цей вражаючий результат було досягнуто завдяки власно розробленому, найбільшому в світі датасету різних варіантів їжі. Як неважко здогадатися цей процес потребував великої кількості часу, зусиль та коштів. Тому цей програмний продукт не є безкоштовним та не є доступним для більшості користувачів[10].

Smart Scales – програмний продукт, що розроблений чеською компанією Inference Technologies, для розпізнавання овочів, фруктів та кондитерських засобів на вагах. Даний продукт може за допомогою ваг та камери швидко та зручно представити ціну покупки, без необхідності запам’ятовувати цифри над прилавками. Особливо це важливо в умовах пандемії, бо це пришвидшує процес покупки, що запобігає створенням черг в магазинах і таким чином зменшує ризик підхопити або рознести хворобу. Як можна побачити на 18 рисунку програма виводить список найбільш вірогідних продуктів, з яких дуже легко і швидко зробити вибір[11].

Рисунок 18 приклад роботи системи Smart Scales

1.4 Постановка задачі

Об’єктом дослідження дипломної роботи є технології класифікації об’єктів за зображенням.

Предметом дослідження дипломної роботи є класифікація блюд меню з використанням штучних нейронних мереж.

Мета роботи полягає у розробці програмного модуля для класифікації блюд в меню за його зображенням.

Зацікавлені сторони:

- Власники магазинів
- Власники закладів масового харчування
- Туристи

У процесі буде реалізовано класифікацію виду продукту харчування за зображенням. Необхідно реалізувати програмний модуль, що буде спроможній провести класифікацію зображення з достатньою точністю для використання.

Задачі дипломного проектування:

- Провести аналіз літературних джерел
- Аналіз існуючих рішень
- Дослідити можливі методи класифікації зображень
- Дослідити ефективність методів реалізації класифікації зображень
- Провести аналіз застосування нейронних мереж
- Спроектувати архітектуру програмного модуля для класифікації блюд з меню
- Сформувати функціональні/нефункціональні вимоги до програмного модулю
- Реалізувати програмний модуль згідно з вимогами для класифікації блюд з меню

Функціональні вимоги:

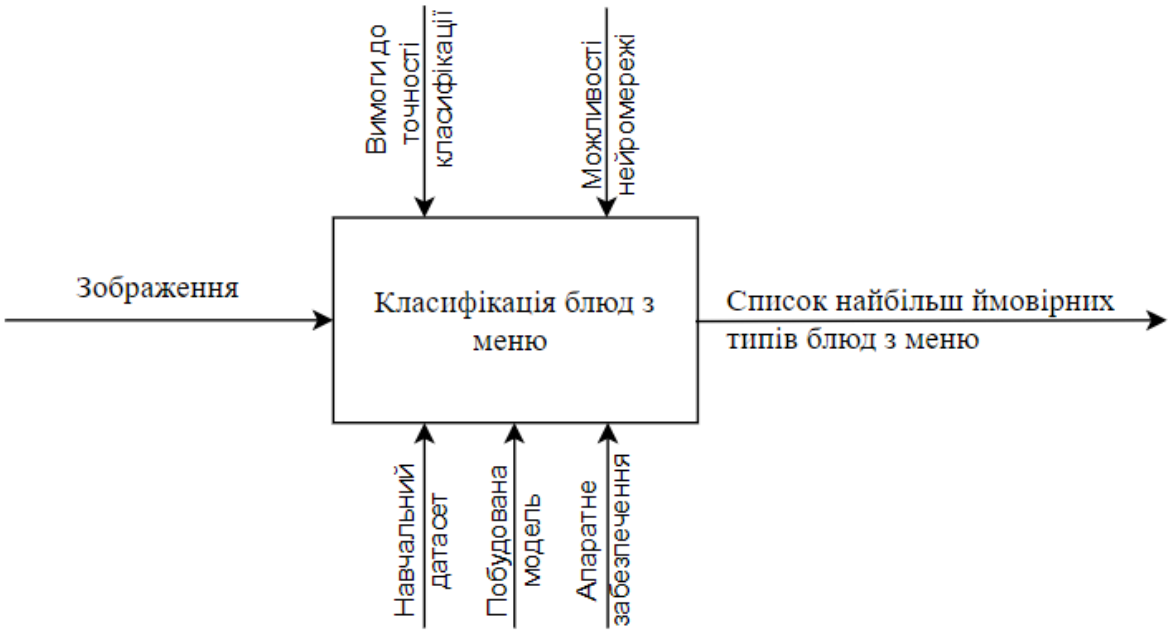
- Повинен працювати із будь-яким формат растрового зображення

- Програмний модуль повинен працювати із зображенням розміром 256/256 та більше
- Програмний модуль має класифікувати блюдо з меню на зображенні
- Програмний модуль повинен мати можливість додатково видавати 5 найбільш ймовірних блюд з меню

Нефункціональні вимоги:

- Низькі апаратні вимоги до роботи програмного модулю
- Швидкість класифікації (Програмний модуль повинен швидко видавати результат)
- Точність (Результати роботи модулю мають бути достатньо точними)
- Зрозуміла побудова програмного модулю (Використання програмного модулю має бути максимально зручним і простим)

Представимо програмний модуль у вигляді «чорної скрині», що можна побачити на 19 рис'унку.



На вхід до програмного модулю подаються такі дані:

- Зображення

Вихідні дані:

- Список найбільш ймовірних типів блюд з меню

Механізм:

- Навчальний датасет
- Побудована моделі
- Апаратне забезпечення

Управління:

- Вимоги до точності класифікації
- Можливості нейромережі

Висновки до першого розділу

Отже, у ході виконання першого розділу було проведено аналітичний огляд кваліфікаційної роботи. Проаналізовано область застосування для розробленого програмного модулю, визначені зацікавлені сторони, було визначену мету дипломної роботи. Проаналізовано існуючі методи та рішення для досягнення поставленого результату.

Було описано об’єкт, предмет, мету, актуальність кваліфікаційної роботи. Сформовано функціональні та нефункціональні вимоги, наведено програмний модуль у вигляді «чорної скрині».

Проаналізувавши існуючі методи для класифікації зображень, було виявлено, що найбільш ефективним методом є згорткові нейронні мережі. Було обрано 2 найбільш вдалі для поставленої задачі архітектури нейронних мереж, вони були обрані за компромісом між точністю, швидкістю роботи та швидкістю навчання.

Розділ 2. Проектування архітектури програмного модуля класифікації продуктів харчування

2.1 Архітектура програмного модулю

Програмний модуль для класифікації блюд з меню має включати такі функції:

- Класифікація блюд з меню
- Навчання моделі нейромережі
- Аналіз моделі нейромережі

Класифікація блюд з меню включатиме такі функції:

- Загрузка моделі нейромережі
- Отримання зображення
- Віднесення зображення до певного класу
- Видача найбільш ймовірного класу
- Видача списку 5 найбільш ймовірних класів

Навчання моделі нейромережі включатиме такі функції:

- Розподілення датасету на навчальну та тестову вибірки
- Створення/загрузка моделі нейромережі
- Навчання моделі нейромережі
- Збереження моделі нейромережі
- Збереження даних про навчання нейромережі

Видача даних для аналізу навчання нейромережі включатиме такі функції:

- Побудова графіків навчання моделі нейромережі
- Валідація моделі нейромережі

На основі описаного функціонального аналізу побудовано дерево функцій, що можна побачити на рисунку 20

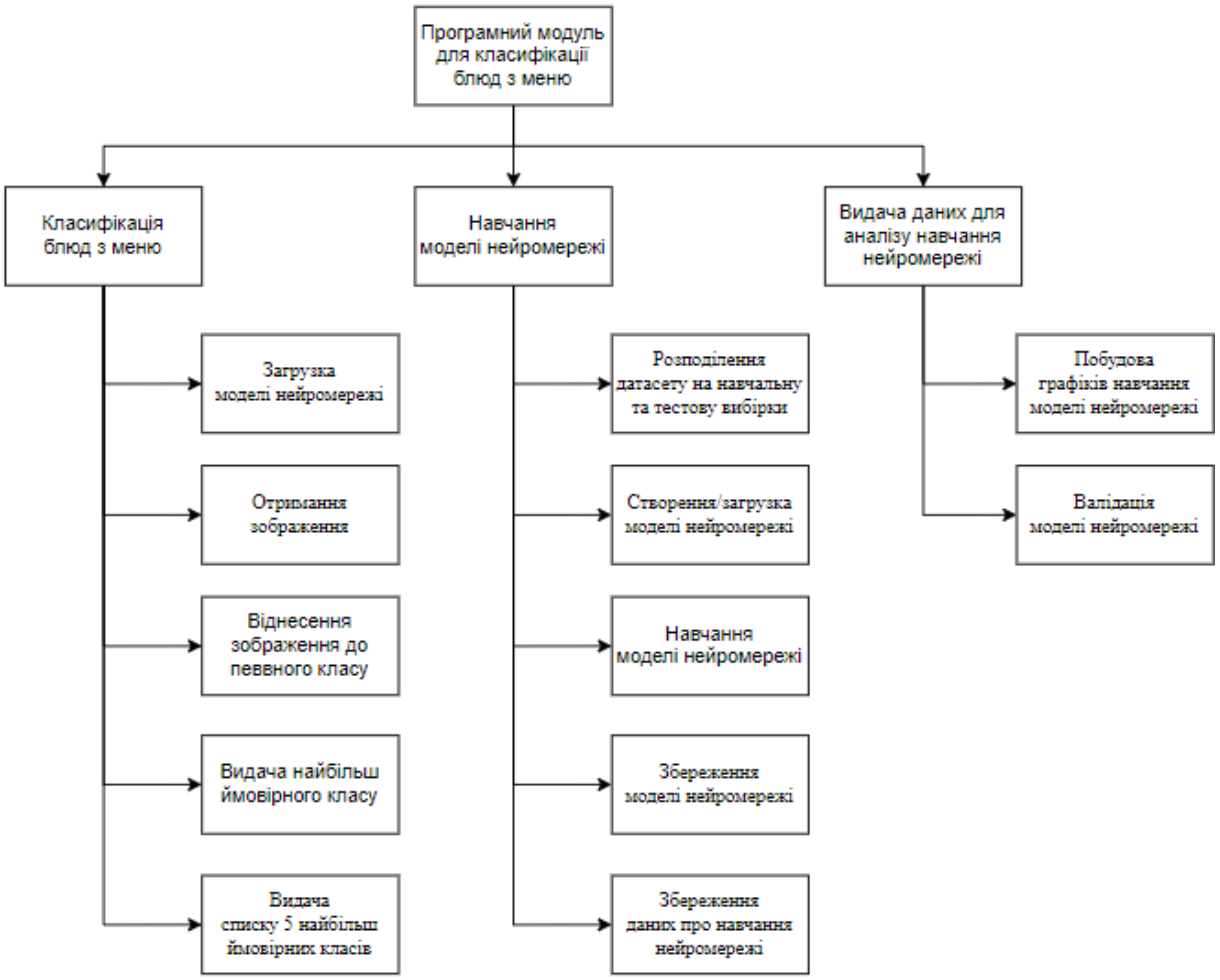


Рисунок 20 дерево функцій програмного модулю класифікації блюд меню

Із вищеописаного дерева функцій можна скласти карту процесів, що буде показано на рисунку 21.

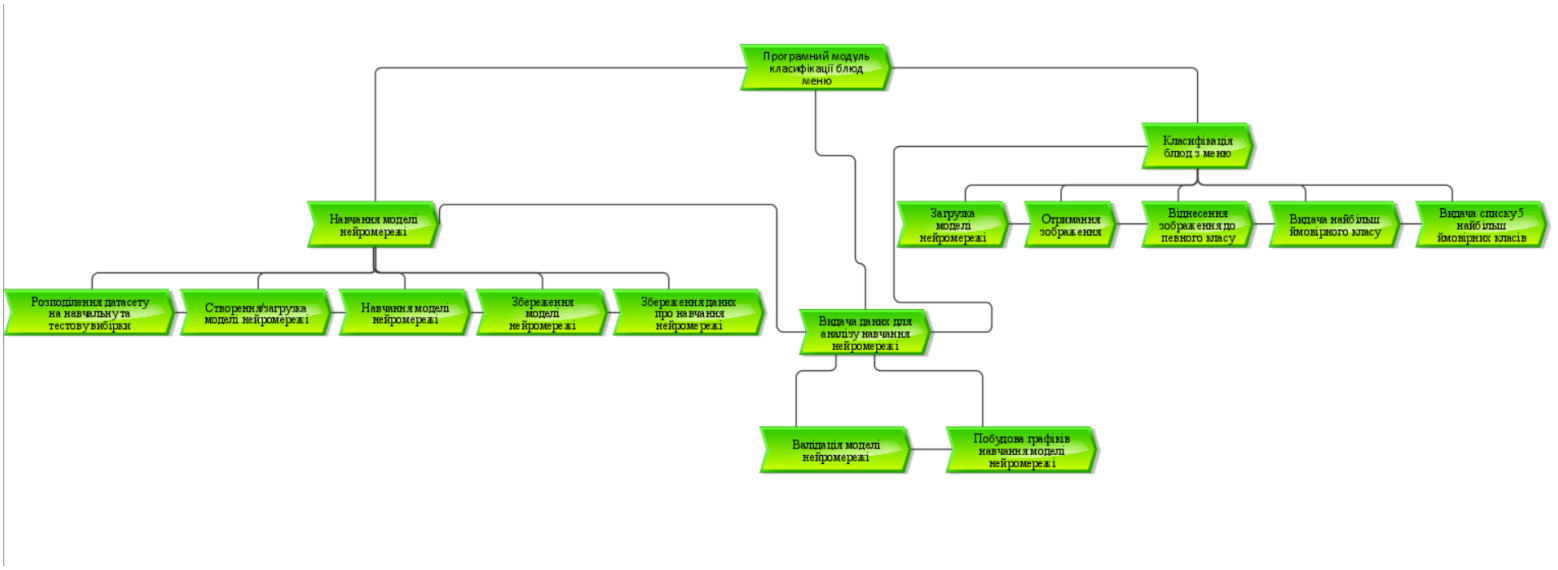


Рисунок 21 Карта процесів

Карта процесів починається із блоку навчання нейромережі, в якому відбувається розподілення датасету на навчальну та тестову вибірку, далі іде створення або загрузка моделі нейромережі. Після створення моделі нейромережі відбувається її навчання, після чого вона зберігається разом із даними про навчання зберігається.

Після навчання нейромережі видаються дані для аналізу навчання нейромережі, що складається із валідації нейромережі а потім побудови графіку навчання.

Для кращого уявлення роботи розробляємого програмного модулю, буде побудована діаграма, що буде показувати взаємодію між модулем та користоувачем, та його архітектуру, що можна побачити на рисунку 22:

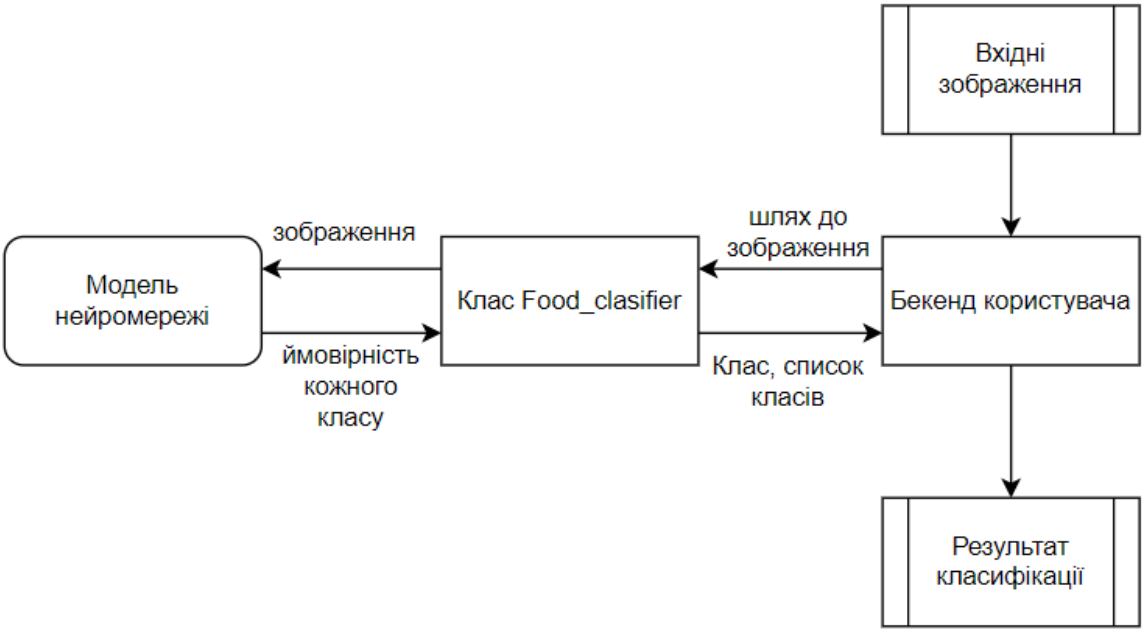


Рисунок 22 Діаграма архітектури та взаємодії з модулем

Як можна побачити, то для вирішення поставленої задачі необхідності в складній, багатокласовій архітектурі немає, тому доцільно буде описати цей клас, та як він працює більш детально, для цього буде побудована діаграма, що показує які методи є у класі, та які методи викликають інші методи і як(рисунок 23)

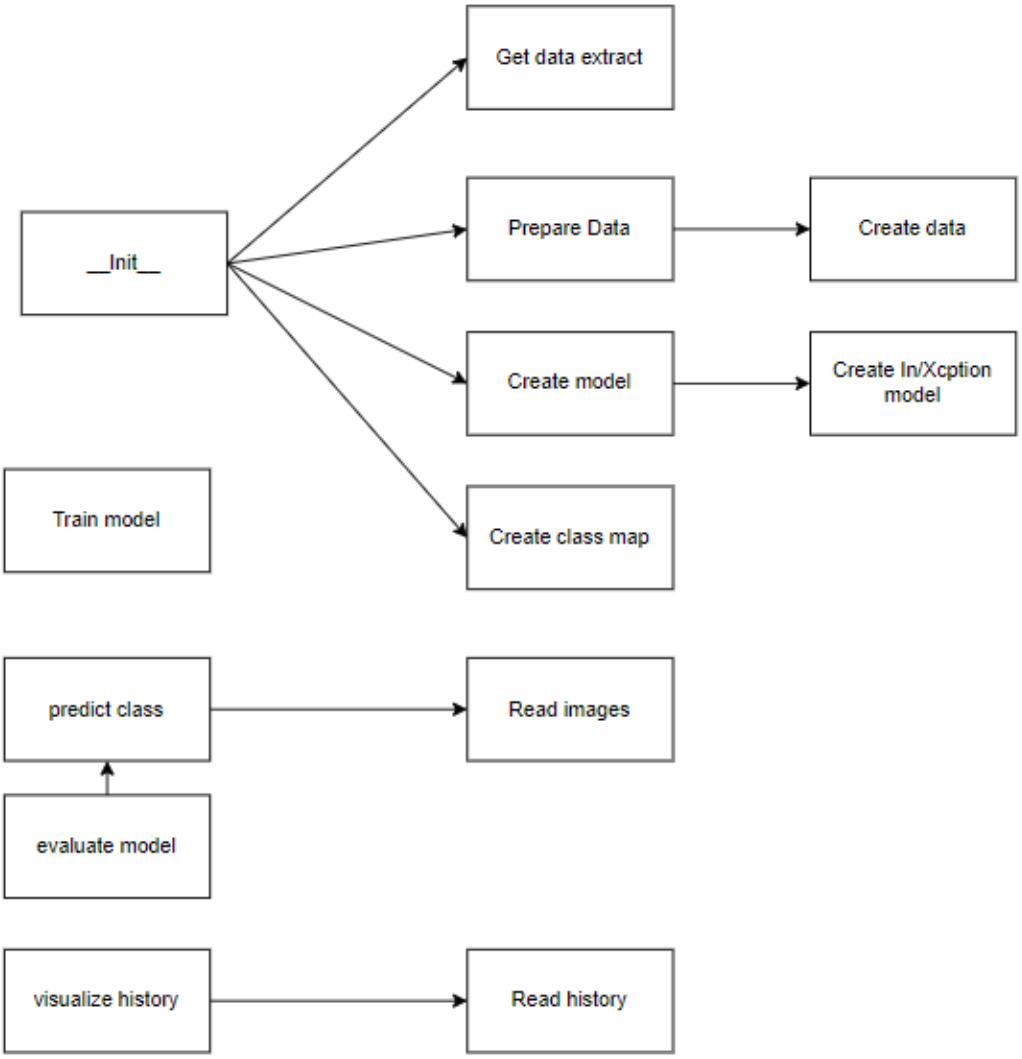


Рисунок 23 Діаграма, що показує функціонування класу

Як можна побачити, є методи, що викликають інші методи, що дозволяє не втрачати функціаналу кожного метода, замість того, що б створювати декілька «великих» методів, що будуть реалізовувати більший функціанал.

Доречно також створити діаграму життєвого циклу, для того, щоб показати як працює програма(рис 24). На цій діаграмі можна побачити, що робота починається із ініалізування класу програмного модуля далі виклнуєтсья загрузка моделі та створення словника класів пілся чого зчитуються зображення, що потім будть проходити класифікацію після чого модуль закінчує роботу, або залежно від використання продовжує класифікувати інші зображення.

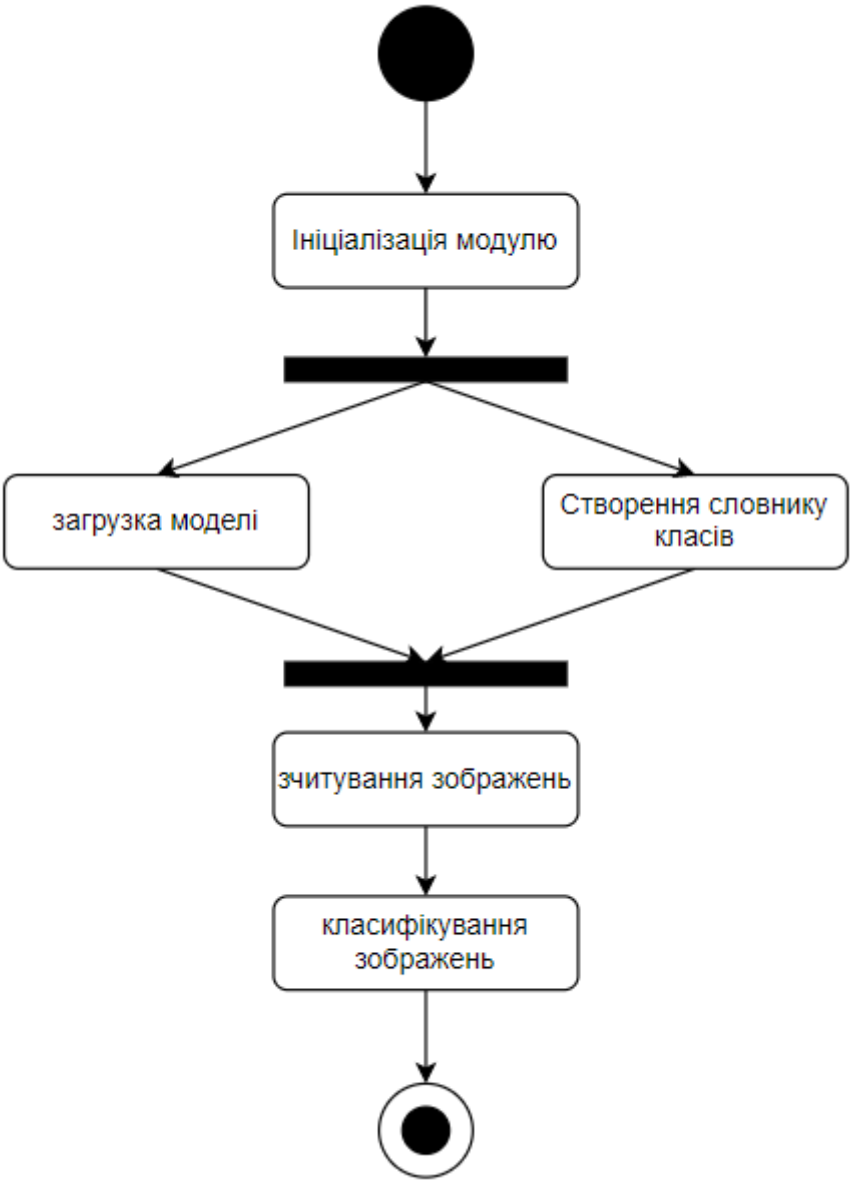


Рисунок 24 Діаграма життєвого циклу програмного модулю для класифікації зображень

2.2 Опис класу програмного модулю

У програмному модулі існує один клас – FoodClassifier який має методи та поля, далі будуть описані методи, та поля класу.

Поля класу:

path – шлях до папки з програмним модулем

data_dir – шлях до папки з даними

train_files – кількість зображень для навчальної вибірки

test_files – кількість зображень для тестової вибірки

model – модель нейромережі – саме це буде робити класифікацію

class_map – словник із назвою блюд та їх нумерацією

Методи класу:

__init__ - метод для ініціювання класу, в якому просто викликаються методи, що загружають датасет, готують його до використання, створють/загружають модель та створють словник із класами блюд

get_data_extract – метод для загрузки та розпаковки архіву із датасетом

create_data – метод для розподілу датасету на тренувальну та тестову вибірку.

prepare_data – метод для підрахунку кількості тренувальних/тестувальних зображень, для виклику методу create_data при ненаявності папок train/test

create_InceptionV3_model / create_Xception_model – метод для створення моделі InceptionV3 або Xception.

create_model_IV3 / create_model_Xc – метод для загрузки моделі при наявності, створення моделі

train_model_Xc / train_model_IV3 – метод для навчання моделі

create_class_map – створення словника із класами блюд

read_images – зберігає список із шляхами до зображень для класифікації

predict_class – класифікація зображення

evaluate_model – валідація моделі

read_history – читання логу навчання моделі

visualize_history – створення графіку навчання моделі

2.3 Опис використаного набору даних

Для вирішення задач поставлених в дипломній роботі було обрано штучну нейрону мережу, для навчання якої необхідна певна вибірка даних, що називається датасетом.

Був обраний набір даних із назвою «Food 101», що складається із 101 категорії продуктів харчування. Для кожної із категорій існує 750 навчальних і 250 тестових зображень, що в сумі становить 101000 різних варіантів. Даної кількості достатньо як для навчання, так і для валідації створеної нейромережі. Максимальна довжина сторони кожного зображення не перебільшує 512 пікселів. Деякі зображення містять шуми та інші недоліки, що робить використання цього датасету більш наближеним до реальних задач. На рисунку 25 можна побачити приклади зображень із цього набору даних.

Рисунок 25 Приклади зображень із датасету

Також даний набір даних має приклади дуже схожих між собою зображень, але різних за класами, що можна побачити на 26 рисунку.

Рисунок 26 зліва страва «Жарений рис», праворуч: «Пад тай»

Як можна побачити, то різницю між ними побачити дуже складно, а таких прикладів у житті дуже багато, що дуже добре впливає на якість датасету.

Набір даних представляється у вигляді архіву «Food-101.tar», для подальшої роботи його необхідно розархівувати. Набір даних має такі файли та папки:

Images – папка із паками з зображеннями

Meta – папка із файлами з даними про датасет в двох форматах txt та json:

Classes – файл із назвами класів блюд

Train/Test – файл із шляхами до зображеннями для навчання/валідації

2.4 Вибір засобів розробки програмного забезпечення

Для реалізації програмного модуля було обрано мову програмування Python, що має велику кількість бібліотек для роботи із штучними нейронними мережами та зображеннями такі як TensorFlow, Keras, PIL, OpenCV та інші. Так як навчання нейромереж потребує дуже потужного апаратного забезпечення то буде використовуватися відкритий сервіс Google Colaboratory. Далі буде коротко описано кожну із вище описаних засобів:

- TensorFlow - бібліотека, що створена для роботи із машинним навчанням компанією Google.
- Keras – бібліотека, що працює поверх бібліотеки TensorFlow і дуже сильно спрощує експерименти із глибокими нейромережами.
- OpenCV – бібліотека для роботи із зображеннями
- Google Colaboratory – це сервіс, що дозволяє виконувати код Python в браузері на дуже потужних CPU/GPU/TPU.
- Pandas - бібліотека для роботи із даними

2.4.1 Google Colaboratory

Google Colab – це безкоштовний хмарний сервіс на основі Jupyter Notebook. Google Colab надає все необхідне для машинного навчання прямо в

браузері, дає безкоштовний доступ до швидких GPU та TPU(Tensor Processing Unit). При цьому для його роботи, необхідно просто зареєструватися в сервісі google, що є дуже простим способом використання. Для початку роботи в Google Colab не потрібно ніяеих додаткових налаштувань. Також Google Colab надає можливість швидко і просто надавати доступ до документу іншим людям, що є дуже корисним для студентів, викладачів та дослідників

Середовище розробоки в Google Colab це майже повністю всім відомий Jupyter Notebook, що і можна побачити на рисунку 27.

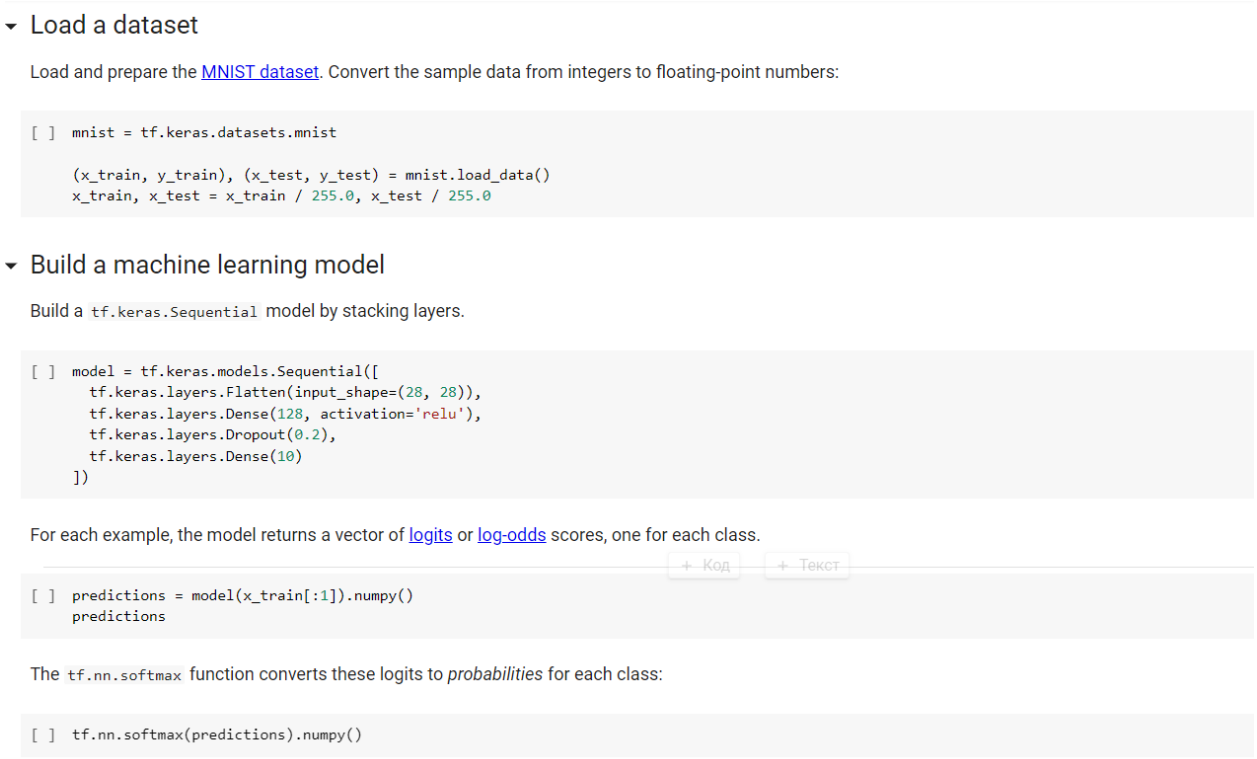


Рисунок 27 скріншот нотатку Google Colab

Також слід зауважити, що безкоштва версія цього сервісу має декілька обмежень:

- Місце на диску 80 Гб
- Оперативна пам’ять 12 Гб
- Відео пам’ять 12Гб
- Тривалість однієї сесії 12 годин
- Обмеження на використання GPU/TPU

Отже, беручи до уваги ці обмеження, можна сказати, що Google Colab не дуже підходить до використання заради комерційних цілей(навіть платна версія), але досить добре підходить для навчальних да дослідницьких цілей.

Висновки до другого розділу

Отже у ході виконання другого розділу дипломної роботи було проведено функціональний аналіз та побудовано дерево функцій програмного модулю. Було побудовано карту процесів програмного модулю. Було обрано датасет та інструменти розробки. Було описано клас програмного модулю.

Розділ 3. Технологічні особливості реалізації програмного модуля

3.1 Налаштування та навчання нейронної мережі

За допомогою хмарного сервісу Google Colab можна набагато швидше навчити нашу нейрону мережу. Для початку необхідно скачати датасет та приготовити його для використання, що і було зроблено за допомогою вбудованого методу в tensorflow – get_file.(рисунок 28).

```
def get_data_extract():
  if "food-101" in os.listdir():
    print("Dataset already exists")
  else:
    tf.keras.utils.get_file(
    'food-101.tar.gz',
    'http://data.vision.ee.ethz.ch/cvl/food-101.tar.gz',
    cache_subdir='/content',
    extract=True,
    archive_format='tar',
    cache_dir=None
    )
    print("Dataset downloaded and extracted!")
```

Method to download dataset and extract dataset

```
[5] get_data_extract()
    data_dir = "food-101/images/"

    Dataset already exists
```

```
[6] def prepare_data(filepath, src,dest):
      classes_images = defaultdict(list)
      with open(filepath, 'r') as txt:
          paths = [read.strip() for read in txt.readlines()]
          for p in paths:
            food = p.split('/')
            classes_images[food[0]].append(food[1] + '.jpg')

      for food in classes_images.keys():
        print("\nCopying images into ",food)
        if not os.path.exists(os.path.join(dest,food)):
          os.makedirs(os.path.join(dest,food))
        for i in classes_images[food]:
          copy(os.path.join(src,food,i), os.path.join(dest,food,i))
      print("Copying Done!")
```

Рисунок 28 Скріншот методів для підготовки датасету

Також за допомогою методів makedirs, copy бібліотеки os було створено папки train/test та скопійовано туди файли.

Далі необхідно створити , або загрузити модель нейромережі певної архітектури, в цьому випадку InceptionV3 та Xception для подальшого навчання. Для створення моделі використовувалися методи tensorflow/keras згідно з назвами InceptionV3, Xception що являють собою переднавчені моделі нейромереж на датасеті imagenet, далі було створено сам класифікатор на основі моделей цих мереж на, що являє собою повнозв'язний шар з функцією активації 'relu', після чого його було додано до моделі з регуляризацією L2 та функцією активації 'softmax' і 101 виходами. Так як, в подальшому зберігання моделей буде відбуватися на Google drive, то необхідно його під'єднати до цього проекту, що і було зроблено за допоги бібліотеки google та методу drive (рисунок 29)

Рисунок 29 Скріншот створення моделі нейромережі

Також необхідно не тільки створювати моделі а й загружати їх з диску, то необхідно створити метод для їх загружки, що і було зроблено за допомоги методу load_model бібліотеки tensorflow/keras.

Для навчання було створено окремий метод, в якому відбувається налаштування нейромережі для початку навчання, саме навчання, збереження натренованих моделей, збереження проміжних результатів, та ведення логу навчання(рисунок 30). Було задано папки із навчальними та тестовими зображеннями, шляхи зберігання моделей та логу. За допомоги методу ImageDataGenerator було проведено декодування зображень в прийнятний вид для нейромереж, а за допогою методу flow_from_directory було зроблено генератор батчей(вибірки навчальних прикладів).

```
def train_model_Xc(model, n_classes,num_epochs, nb_train_samples,nb_validation_samples):
  K.clear_session()
  img_width, img_height = 299, 299
  train_data_dir = 'food-101/train'
  validation_data_dir = 'food-101/test'
  batch_size = 32
  bestmodel_path = '/content/gdrive/MyDrive/models/Xcbestmodel_'+str(n_classes)+'class.hdf5'
  trainedmodel_path = '/content/gdrive/MyDrive/models/Xctrainedmodel_'+str(n_classes)+'class.hdf5'
  history_path = '/content/gdrive/MyDrive/models/Xchistory_'+str(n_classes)+'.log'

  train_datagen = ImageDataGenerator(
      preprocessing_function=preprocess_input,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True)

  test_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)

  train_generator = train_datagen.flow_from_directory(
      train_data_dir,
      target_size=(img_height, img_width),
      batch_size=batch_size,
      class_mode='categorical')

  validation_generator = test_datagen.flow_from_directory(
      validation_data_dir,
      target_size=(img_height, img_width),
      batch_size=batch_size,
      class_mode='categorical')

  model.compile(optimizer=SGD(lr=0.01, momentum=0.9), loss='categorical_crossentropy', metrics=['accuracy'])
  checkpoint = ModelCheckpoint(filepath=bestmodel_path, verbose=1, save_best_only=True)
  csv_logger = CSVLogger(history_path)

  history = model.fit_generator(train_generator,
                      steps_per_epoch = nb_train_samples // batch_size,
                      validation_data=validation_generator,
                      validation_steps=nb_validation_samples // batch_size,
                      epochs=num_epochs,
                      verbose=1,
                      callbacks=[csv_logger, checkpoint])

  model.save(trainedmodel_path)
  class_map = train_generator.class_indices
  return history, class_map
```

Рисунок 30 Скріншот методу для навчання нейромережі

За допомогою методу compile класу model бібліотеки tensorflow/keras було проведено налаштування нейромережі перед навчанням, було встановлено параметри learning rate та momentum, оптимізатор та визначено метрики. Також було зроблено зберігання моделей та ведення логу методами ModelCheckpoint, save та CSVLpgger, fit_generator.

Після проведення цих дій, можна перейти до навчання нейромережі методом fit_generator (рисунок 31):

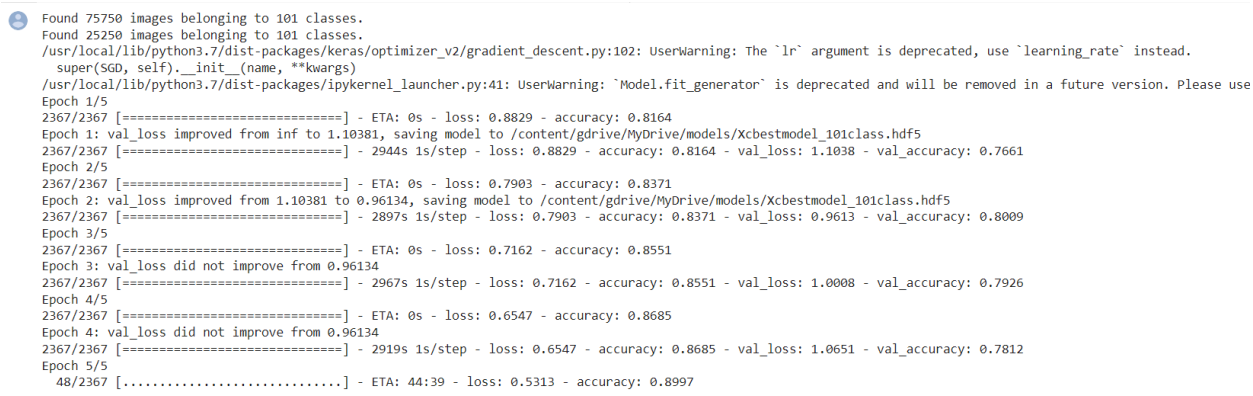


Рисунок 31 Скріншот процесу навчання нейромережі в сервісі Google Colaboratory

Через використання безоплатної версії цього сервісу та через великий обсяг датасету (100100 зображень), навчання нейромережі доводилося роботи в декілька підходів, для чого і необхідно було під'єдняння до гугл диску. Таким чином було отримано дві навчені моделі, з достатньо високою точністю класифікації.

3.2 Реалізація програмного модуля класифікації блюд з меню

Під час розробки проекту в Google Colaboratory було створено декілька методів, що з невеликими доробками можуть бути використані в програмному модулі. В першу чергу було розроблено основний функціонал програмного модулю, що включає в себе: загрузку моделі, створення словнику класів, загрузка зображень та класифікація зображень. Для цього було написано 3 методи: create_model_IV3 / create_model_Xc, read_images, create_class_map та predict_class.

В першому методі з вищеназваних було реалізовано загрузку моделей, а в подальшому виклик іншого методу для їх створення. Моделі, що загружаються були отримані під час етапу навчання. Створення моделей було реалізовано іншим методом, що буде описаний далі.

В другому методі, було реалізовано загрузку зображень, точніше – створено список із повними шляхами, до цих зображень, що дозволяє їх використання у класифікаційній моделі. Хоча, фактично в цьому методі не відбувається збереження зображень в пам’ять, але обраний шлях доцільніший, з огляду на зменшення використання оперативної пам’яті.

В тертім методі було реалізовано створення словнику класів, що складається з номеру в якості ключа та назви класу в якості значення. Найпростішим варіантом реалізації цього методу стало зчитування файлу classes.txt з датасету.

В методі predict_class відбувається класифікація зображень по класам, для цього необхідно використати метод predict класу Model бібліотеки tensorflow.keras. Цей метод віддає найймовірніший клас зображення та список із 5 наймовірніших класів зображення.

Після реалізації обсягу дій, що був описаний вище, було проведено тестування його на працездатність та виконання вимог до модулю. Тестування відбувалося таким чином: бралося довільні зображення блюд взяті з інтернету, що відносилися до наявних класів в датасеті, потім програмний модуль запускався та результати його роботи перевірялися із очікуваними данними(рисунки 32, 33).

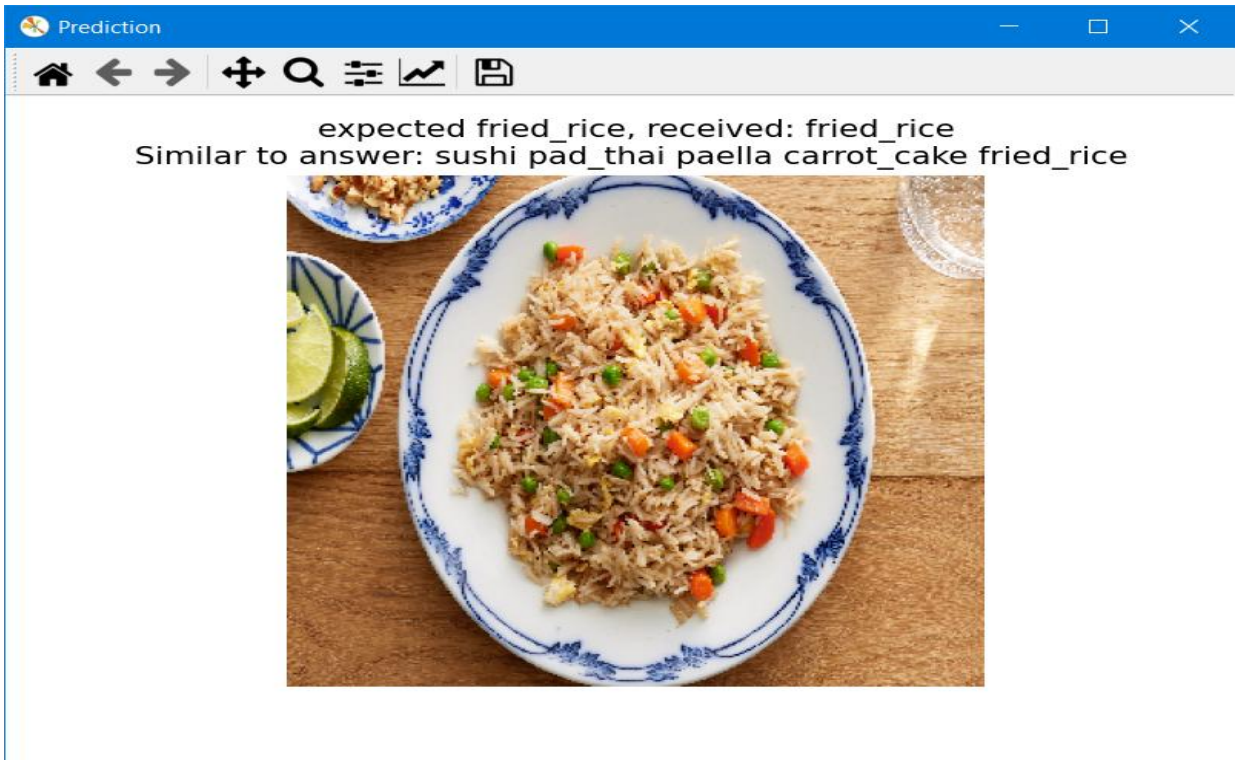


Рисунок 32 Скріншот графічного представлення результатів

```
Loaded model
([44], [[95, 70, 71, 14, 44]])
```

Рисунок 33 Скріншот консолі, в якій показано дані, що видає метод

Незважаючи на те, що у всіх випадках, в яких було протестовано модуль видавав правильний результат, або він знаходився у 4 додаткових класах, неможна сказати, що програмний модуль є достатньо точним або недостатньо точним, бо кількість тестів була незначаною. Ця проблема буде вирішена у підрозділі 3.1.Також треба зробити акцент на тому, що може бути подане зображення, класу якого в датасеті не існує, або взагалі не зрозуміло що. В такому разі програмний модуль поверне клас з номером -1. Це можна побачити на рисунку 34.

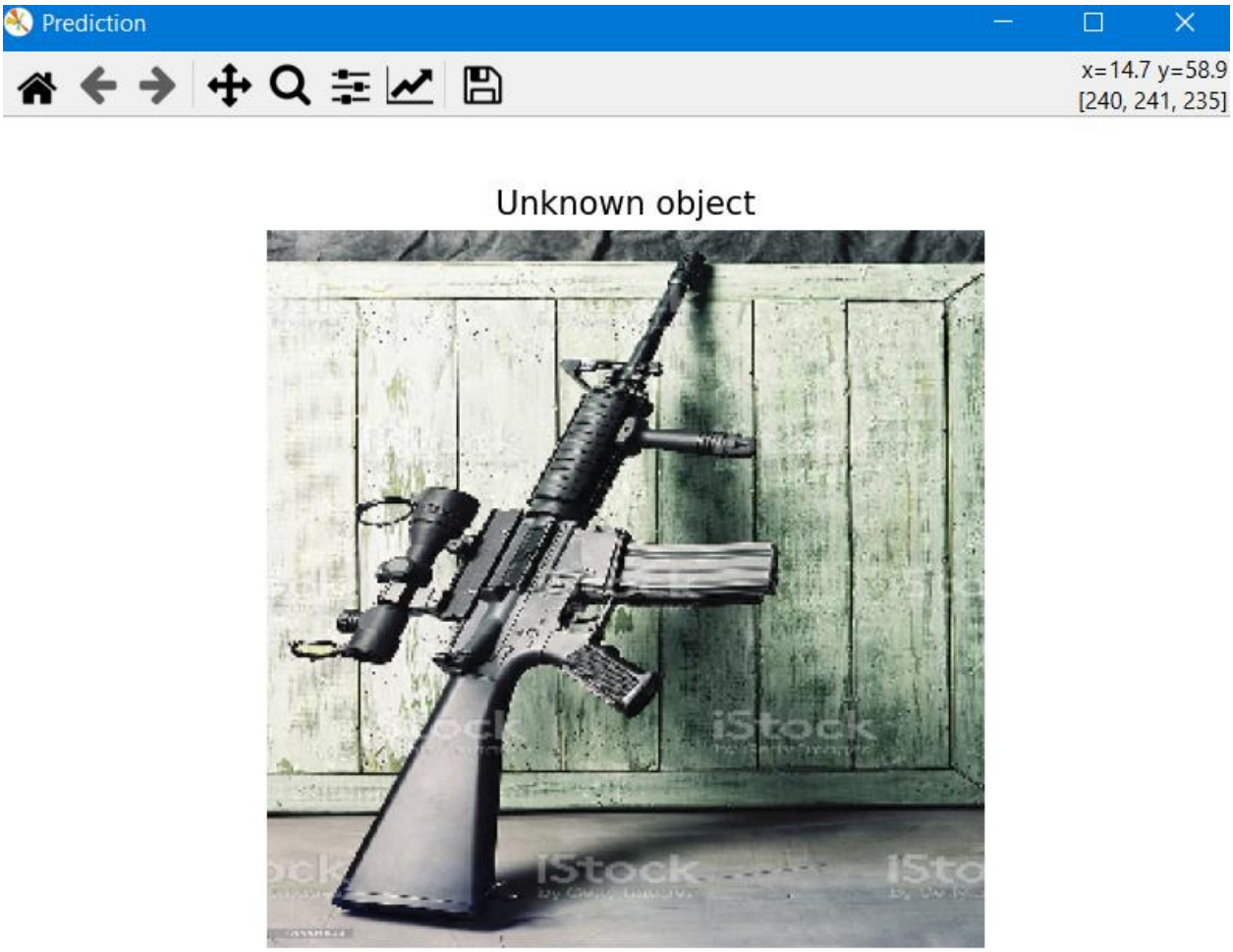


Рисунок 34 Приклад неправильного подання даних

Після цього в програмний модуль було додано декілька методів, що відповідають за навчання нейромережі: get_data_extract , create_data, prepare_data, create_InceptionV3_model / create_Xception_model, train_model_Xc / train_model_IV3. Так як ці методи були здебільшого розроблені під час навчання моделі, то доцільно було їх використання в програмному модулі із незначними змінами.

3.3 Аналіз точності роботи програмного модулю

Для валідації моделей було обрано той же самий датасет, що і для тренування, але ту частину на якій навчання не відбувалося – на частині test. Зображення в цій частині відносяться до тих же класів, що і зображення в тренувальній частині.(рис 35).

Рисунок 35 зображення із тестової

Для валідації за метриками top1 та top5 accuracy було розроблено метод evaluate_model, що дозволяє її провести, але повна перевірка на датасеті займає приблизно 4 години, що не є зручним. Таким чином була додана можливість використання не деякої частини датасету, але слід зауважити, що у цю частину мають входити усі класи зображень. Результати роботи методу зображені на рисунку 36

```
Tested 5.0% of validation files, summary 1313 files
Top 1 accuracy = 0.78
Top 5 accuracy = 0.92
```

Рисунок 36 Скріншот результату тестування моделі на 5% від всієї тестової вибірки

Для більшої об’єктивності валідації, необхідно збільшити кількість тестових зображень, це буде виконано для двох моделей, що були навчені на етапі 3.1. На рисунку 37 зображено результати валідації моделі InceptionV3, а на рисунку 38 зображено результати валідації моделі Xception.

```
Tested 15.0% of validation files, summary 3838 files
Top 1 accuracy = 0.78
Top 5 accuracy = 0.93
```

Рисунок 37 Результати валідації моделі inceptionV3

```
Tested 15.0% of validation files, summary 3838 files
Top 1 accuracy = 0.79
Top 5 accuracy = 0.94
```

Рисунок 38 Результати валідації моделі Xception

Але виникає питання, чи достатньо 15 % файлів для висновків, чи ні. Для вирішення цього питання було проведено дослідження на 1%, 2%, 5%, 10%, 15%, 20%, 25% кількості файлів.(результати можна побачити у додатку Г) З результатів дослідження можна побудувати графік, для кращого розуміння результату(рисунок 39).

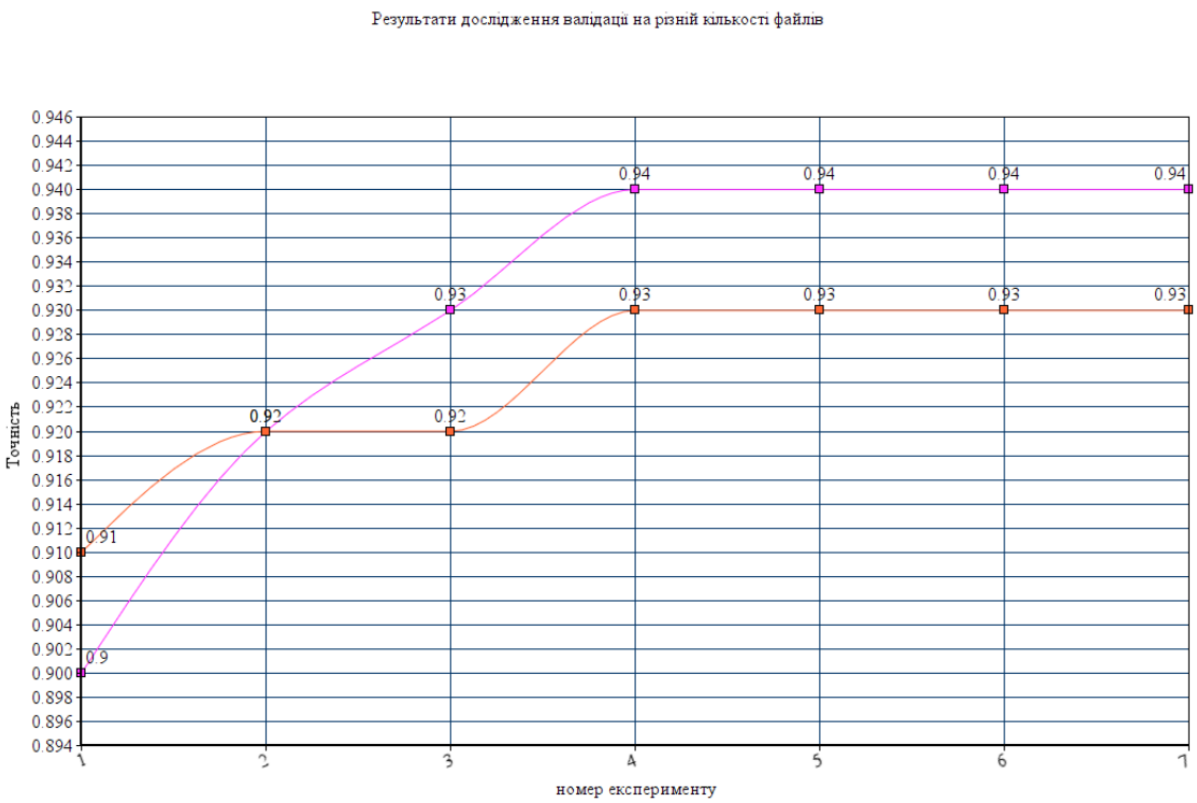


Рисунок 39 Графік точності валідації моделі

Отже, можна зробити висновок, що доцільніше використовувати лише 15% файлів, бо це дає найбільшу точність валідації при менший кількості затраченого часу, бо після 15% результати перестають змінюватися

Таким чином, можна зробити висновок, що обидві архітектури справляється зі своєю задачею приблизно однаково, хоча Xception показала трохи кращий результат у top 1 accuracy.

Але крім результатів валідації нейромережі треба порівняти час навчання нейромереж, для цього необхідно проаналізувати логи з навчання. Для цього було розроблено 2 методи read_history, visualize_history.

Перший метод необхідний для завантаження логу навчання в пам’ять, дані заносяться в тип Dataframe, що доданий бібліотекою pandas. Другий метод будує графіки, щодо наданих даних, це виконується методами бібліотеки matplotlib. На рисунках 40-43 можна побачити ці графіки.

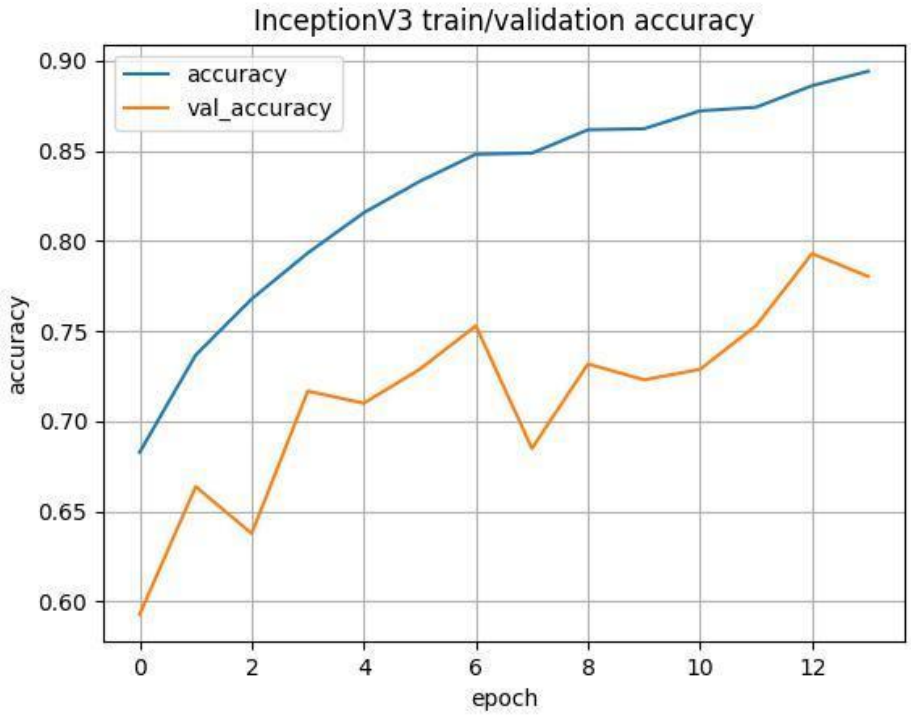


Рисунок 40 Графік змінення точності роботи моделі InceptionV3 під час навчання

Рисунок 41 Графік змінення значення функції втрат моделі InceptionV3 під час навчання

Як можна побачити на графіках, поліпшення результатів «точності валідації» відбувається не дуже очевидно, бо вона не завжди збільшується при збільшені часу навчання, це дуже відома проблема. Поясняється тим, що мережа перенавчається на тренувальному датасеті, також тут може впливати те, що моделі базуються на переднавченій нейромережі на датасеті «imagenet».

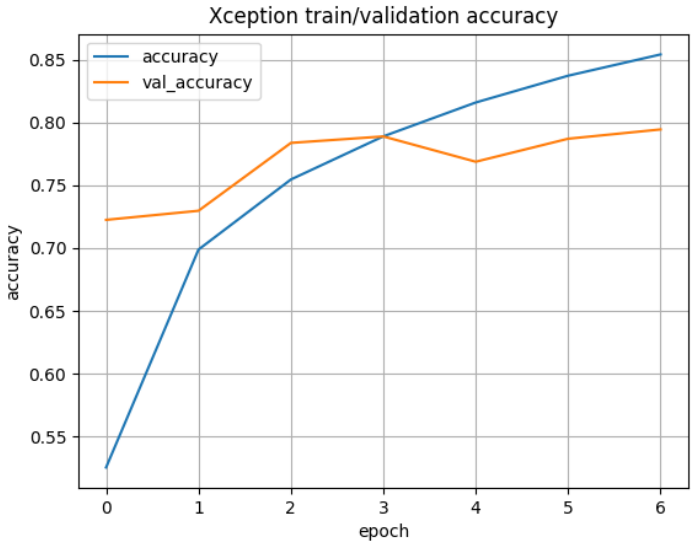


Рисунок 42 Графік змінення точності роботи моделі Xception під час навчання

Рисунок 43 Графік змінення значення функції втрат моделі Xception під час навчання

Як можна побачити, тут ситуація з цією проблемою краще, також можна побачити, що при меншій кількості епох архітектура Xception, краще справляється з поставленою задачею через більш швидке навчання, хоча і кожна епоха в цій архітектурі витрачає більше часу на 30% за епоху в архітектурі InceptionV3, різниця все-одно значна. Слід зауважити і те, що при меншій точності на тренувальній вибірці, точність на тестовій вибірці приблизно однакова. З цього можна зробити висновок, про меншу схильність цієї архітектури до перенавчання.

Висновки до третього розділу

Отже, в результаті виконання третього розділу дипломної роботи було описано та навчено обрані моделі нейромереж, було реалізовано програмний модуль класифікації блюд з меню, було проведено аналіз точності роботи нейромережі, аналіз швидкості навчання.

У подальшій розробці припустимо вдосконалення модулю шляхом додавання нової архітектури, подальшого навчання нейромереж, можливо додати алгоритм попередньої обробки зображень, що може допомогти в класифікації.

Важливою ціллю у майбутньому буде досягнення top 1 accuracy 90%, що можливо, при подальшому навчанні нейромережі та додаванні алгоритмів обробки зображень. Також, дуже важливим кроком, для розвитку програмного модулю буде додавання нових класів зображень, що потребує великої кількості часу та можливої переробки всієї архітектури мережі.

Також дуже важливим доповненням до цього було б зауваження про те, що було виявлено, що сервіс Google Colab не дуже підходить для подібних дуже ресурсоємних задач через динамічний дозволений час роботи(чми більше працюєш, ти менше дозволено).

Висновки

Отже, через підвищену необхідність в скорочені витрат та часу перебування людей в черзі, необхідність систем класифікації продуктів харчування або товарів буде тільки зростати. Пришвидшене обслуговування людей задоволить як покупців та і власників закладів харчування, магазинів.

У ході виконання даної роботи було проведено аналітичний огляд кваліфікаційної роботи. Проаналізовано область застосування для розробленого програмного модулю, визначені зацікавлені сторони, було визначену мету, предмет, об’єкт і актуальність дипломної роботи. Проаналізовано існуючі методи та рішення для досягнення поставленого результату. Було сформовано функціональні і нефункціональні вимоги, проведено функціональний аналіз та спроектовано архітектуру програмного модулю.

Було проведено вибір, налаштування та навчання моделі нейронної мережі, було розроблено програмний модуль згідно функціональних і не функціональних вимог за допомогою мови програмування Python, бібліотеки tensorflow, keras та інших . Було проведено аналіз точності роботи програмного модулю, було проведено аналіз швидкості та якості навчання обраних нейронних мереж.

Було реалізовано програмний модуль, що можна буде використовувати в подальших розробках додатків та систем. Хоча модуль не є більш точним за існуючі рішення, він є готовим до використання.

Список використаних джерел

1. Джордж Стокман, Линда Шапиро. [Computer Vision Компьютерное зрение]. —М. : Бином. Лаборатория знаний, 2006. — 752 с.
2. Стэнфордский курс: лекция 2. Классификация изображений [Електронний ресурс] – Режим доступу до ресурсу: https://www.reg.ru/blog/stehnfordskij-kurs-lekciya-2-klassifikaciya-izobrazhenij/.
3. What are neural networks? [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/cloud/learn/neural-networks.
4. Sumit Saha. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way [Електронний ресурс] / Sumit Saha. – 2018. – Режим доступу до ресурсу: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.
5. Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach [Електронний ресурс]. – 2018. – Режим доступу до есурсу: https://www.sciencedirect.com/science/article/pii/S1877050918308019.
6. Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens. Rethinking the Inception Architecture for Computer Vision. – 2015 – Режим доступу до ресурсу: http://arxiv.org/abs/1512.00567
7. Franc¸ois Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. – 2016 – Режим доступу до ресурсу: https://arxiv.org/pdf/1610.02357
8. Kaggle | Food 101 [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://www.kaggle.com/dansbecker/food-101.
9. Офіційний сайт Foodvisor [Електронний ресурс] – Режим доступу до ресурсу: https://www.foodvisor.io/en/.
10. Офіційний сайт Logmeal [Електронний ресурс] – Режим доступу до ресурсу: https://www.logmeal.es/.
11. Офіційний сайт SmartScales [Електронний ресурс] – Режим доступу до ресурсу https://smartscales.ru/en/.

12. TensorFlow 2 quickstart for beginners [Електронний ресурс] // The TensorFlow Authors. – 2019. – Режим доступу до ресурсу: https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/quickstart/beginner.ipynb.
13. Matplotlib 3.5.2 documentation [Електронний ресурс] // The Matplotlib development team. – 2012. – Режим доступу до ресурсу: https://matplotlib.org/stable/index.html.
14. TensorFlow Core v2.9.1 Documentation [Електронний ресурс] // Google Brain Team. – 2021. – Режим доступу до ресурсу: https://www.tensorflow.org/api_docs.
15. Keras API documentation [Електронний ресурс] // François Chollet. – 2020. – Режим доступу до ресурсу: https://keras.io/api/.
16. TensorFlow Image Classification : All you need to know about Building Classifiers [Електронний ресурс] // edureka!. – 2019. – Режим доступу до ресурсу: https://www.edureka.co/blog/tensorflow-image-classification.
17. os — Miscellaneous operating system interfaces [Електронний ресурс] // Python Software Foundation. – 2001. – Режим доступу до ресурсу: https://docs.python.org/3/library/os.html.
18. OpenCV-Python Tutorials [Електронний ресурс] // doxygen. – 2022. – Режим доступу до ресурсу: https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html.
19. Image-based Food Classification and Volume Estimation for Dietary Assessment: A Review / P.Frank, S. Yingnan, Q. Jianing, L. Benny. // Journal of Biomedical and Health Informatics 1. – 2020. – №10.
20. Jaspreet. A Concise History of Neural Networks [Електронний ресурс] / Jaspreet // Towards Data Science. – 2016. – Режим доступу до ресурсу: https://towardsdatascience.com/a-concise-history-of-neural-networks-2070655d3fec#:~:text=The%20idea%20of%20neural%20networks,McCulloch%20and%20mathematician%20Walter%20Pitts..

Додатки

Додаток А: Лістинг класу FoodClassifier

```
class FoodClassifier:
    path = "Your path"
    data_dir = ""
    train_files = 0
    test_files = 0
    model = 0
    class_map = {}
    def __init__(self, chmodel = "IV3"):
        self.get_data_extract()
        self.prepare_data()
        if chmodel == "IV3":
            self.create_model_IV3()
        elif chmodel == "Xc":
            self.create_model_Xc()
        else:
            self.create_model_IV3()
            print("Chosen IV3 model")
        self.create_class_map()

    def get_data_extract(self):
        if "food-101" in os.listdir(self.path):
            print("Dataset already exists")
        else:
            print("Downloading dataset...")
            tf.keras.utils.get_file(
                'food-101.tar.gz',
                'http://data.vision.ee.ethz.ch/cvl/food-101.tar.gz',
                cache_subdir=self.path,
                extract=True,
                archive_format='tar',
                cache_dir=None
            )
            print("Dataset downloaded and extracted!")
        self.data_dir = self.path + "/food-101/images/"

    def create_data(self, filepath, src, dest):
        classes_images = defaultdict(list)
        with open(filepath, 'r') as txt:
            paths = [read.strip() for read in txt.readlines()]
            for p in paths:
                food = p.split('/')
                classes_images[food[0]].append(food[1] + '.jpg')

        for food in classes_images.keys():
            if not os.path.exists(os.path.join(dest, food)):
                os. makedirs (os. path. join (dest, food))
            for i in classes_images[food]:
                copy (os. path. join (src, food, i), os. path. join (dest, food, i))
        print ("Copying Done!")
```

```
def prepare_data(self):
    if 'train' in os. litter (self. path+'/food-101'):
        print ('Train data is already existing')
    else:
        print ("Creating train data...")
        self. create_data (self. path+'/food-101/meta/train.txt', self.data_dir, self.path+'/food-101/train')
    if 'test' in os.listdir(self.path+'/food-101'):
        print('Test data is already exist')
    else:
        print("Creating test data...")
        self.create_data(self.path + '/food-101/meta/test.txt', self.data_dir, self.path + '/food-101/test')
    self.train_files = sum([len(files) for i, j, files in os.walk(self.path+'/food-101/train')])
    print("Total number of samples in train folder: " + str(self.train_files))
    self.test_files = sum([len(files) for i, j, files in os.walk(self.path+'/food-101/test')])
    print("Total number of samples in test folder: " + str(self.test_files))

def create_InceptionV3_model(self):
    K.clear_session()
    n_classes = 101
    inception = InceptionV3(weights='imagenet', include_top=False)
    x = inception.output
    x = GlobalAveragePooling2D()(x)
    x = Dense(128, activation='relu')(x)
    x = Dropout(0.2)(x)
    predictions = Dense(n_classes, kernel_regularizer=regularizers.l2(0.005), activation='softmax')(x)
    model = Model(inputs=inception.input, outputs=predictions)
    self.model = model
    return model

def create_Xception_model(self):

    K.clear_session()
    n_classes = 101
    xception = Xception (weights='imagenet', include_top=False)
    x = xception.output
    x = GlobalAveragePooling2D()(x)
    x = Dense(128, activation='relu')(x)
    x = Dropout(0.2)(x)
    predictions = Dense(n_classes, kernel_regularizer=regularizers.l2(0.005), activation='softmax')(x)
    model = Model(inputs=xception.input, outputs=predictions)
    self.model = model
    return model

def create_model_IV3(self):
    if os.path.exists(self.path+'/models/InceptionV3_bestmodel.hdf5'):
        K.clear_session()
        self.model = load_model(self.path+'/models/InceptionV3_bestmodel.hdf5')
        print("Loaded model")
    else:
        self.model = self.create_InceptionV3_model()
        print("Created model")
    return self.model

def create_model_Xc(self):
```

```
    if os.path.exists(self.path+'/models/Xception_bestmodel.hdf5'):
        K.clear_session()
        self.model = load_model(self.path+'/models/Xception_bestmodel.hdf5')
        print("Loaded model")
    else:
        self.model = self.create_InceptionV3_model()
        print("Created model")
    return self.model

def train_model_Xc(self, num_epochs):
    K.clear_session()
    img_width, img_height = 299, 299
    train_data_dir = self.path+'/food-101/train'
    validation_data_dir = self.path+'/food-101/test'
    batch_size = 64
    bestmodel_path = self.path+'/models/Xception_bestmodel'
    trainedmodel_path = self.path+'/models/Xception_trainedmodel'
    history_path = self.path+'/models/Xception_history'

    train_datagen = ImageDataGenerator(
        preprocessing_function=preprocess_input,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True)

    test_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)

    train_generator = train_datagen.flow_from_directory(
        train_data_dir,
        target_size=(img_height, img_width),
        batch_size=batch_size,
        class_mode='categorical')

    validation_generator = test_datagen.flow_from_directory(
        validation_data_dir,
        target_size=(img_height, img_width),
        batch_size=batch_size,
        class_mode='categorical')

    self.model.compile(optimizer=SGD(lr=0.01, momentum=0.9), loss='categorical_crossentropy', metrics=['accuracy'])
    checkpoint = ModelCheckpoint(filepath=bestmodel_path, verbose=1, save_best_only=True)
    csv_logger = CSVLogger(history_path)

    history = self.model.fit_generator(train_generator,
                            steps_per_epoch=self.train_files // batch_size,
                            validation_data=validation_generator,
                            validation_steps=self.test_files // batch_size,
                            epochs=num_epochs,
                            verbose=1,
                            callbacks=[csv_logger, checkpoint])

    self.model.save(trainedmodel_path)
    return history
```

```
def train_model_IV3(self, num_epochs):
    K.clear_session()
    img_width, img_height = 299, 299
    train_data_dir = self.path+'/food-101/train'
    validation_data_dir = self.path+'/food-101/test'
    batch_size = 64
    bestmodel_path = self.path+'/models/InceptionV3_bestmodel'
    trainedmodel_path = self.path+'/models/InceptionV3_trainedmodel'
    history_path = self.path+'/models/InceptionV3_history'

    train_datagen = ImageDataGenerator(
        preprocessing_function=preprocess_input,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True)

    test_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)

    train_generator = train_datagen.flow_from_directory(
        train_data_dir,
        target_size=(img_height, img_width),
        batch_size=batch_size,
        class_mode='categorical')

    validation_generator = test_datagen.flow_from_directory(
        validation_data_dir,
        target_size=(img_height, img_width),
        batch_size=batch_size,
        class_mode='categorical')

    self.model.compile(optimizer=SGD(lr=0.01, momentum=0.9), loss='categorical_crossentropy', metrics=['accuracy'])
    checkpoint = ModelCheckpoint(filepath=bestmodel_path, verbose=1, save_best_only=True)
    csv_logger = CSVLogger(history_path)

    history = self.model.fit_generator(train_generator,
                                       steps_per_epoch=self.train_files // batch_size,
                                       validation_data=validation_generator,
                                       validation_steps=self.test_files // batch_size,
                                       epochs=num_epochs,
                                       verbose=1,
                                       callbacks=[csv_logger, checkpoint])

    self.model.save(trainedmodel_path)
    return history

def create_class_map(self):
    i = 0
    class_map = {}
    with open(self.path+'/food-101/meta/classes.txt') as f:
        classes = [read.strip() for read in f.readlines()]
        for clas_s in classes:
            class_map.update({i:clas_s})
            i+=1
    self.class_map = class_map
```

```
        return class_map

    def read_images(self):
        images = os.listdir(self.path+'/examples')
        for ind in range(len(images)):
            images[ind] = self.path+'/examples/' + images[ind]
        #print(images)
        return images

    def predict_class(self, images, show=False, write=False):
        answer_top1 = []
        answer_top5 = []
        for img in images:
            timg = img
            img = image.load_img(img, target_size=(299, 299))
            img = image.img_to_array(img)
            img = np.expand_dims(img, axis=0)
            img = preprocess_input(img)

            pred = self.model.predict(img)
            #print(pred)
            index = np.argmax(pred)
            answer_top1.append(index)
            title = "expected " + os.path.basename(timg)[:-4] + ", received: " + self.class_map[index]+'\n' + "Similar to answer: "
            pred = np.array(pred)
            predtop5 = np.ndarray.tolist(pred.argsort())[0]
            predtop5 = predtop5[-5:]
            answer_top5.append(predtop5)
            for ind in predtop5:
                title += self.class_map[ind] + " "

            if write:
                print(title)
            timg = image.load_img(timg, target_size=(299, 299))
            if show:
                plt.figure("Prediction")
                plt.imshow(timg)
                plt.axis('off')
                plt.title(title)
                plt.show()
        return answer_top1, answer_top5

    def evaluate_model(self, val_numb):
        vall_true = 0
        vall_true_top5 = 0
        classes_images = defaultdict(list)
        validation_data_dir = self.path+'/food-101/test/'
        with open(self.path+'/food-101/meta/test.txt', 'r') as txt:
            paths = [read.strip() for read in txt.readlines()]
            for p in paths:
                food = p.split('/')
                classes_images[food[0]].append(food[1] + '.jpg')
        for food in classes_images.keys():
            print("Testing " + food)
```

```
            val_numbt = val_numb * self.test_files/len(classes_images)
            for img in classes_images[food]:
                if val_numbt <= 0: break
                val_numbt-=1
                images = [validation_data_dir + food + '/' + img]
                testtop1, testtop5 = self.predict_class(images)
                if self.class_map[testtop1[0]] == food:
                    vall_true+=1
                for ind in testtop5[0]:
                    if self.class_map[ind] == food:
                        vall_true_top5 += 1
        top1_acc = vall_true/ (val_numb * self.test_files)
        top5_acc = vall_true_top5 / (val_numb * self.test_files)
        print("Tested " + str(val_numb*100) + "% of validation files, summary " + str(val_numb * self.test_files) + " files")
        print("Top 1 accuracy = " + str("%.2f" % top1_acc))
        print("Top 5 accuracy = " + str("%.2f" % top5_acc))
        return top1_acc, top5_acc

    def read_history(self, path):
        history = pd.read_csv(path)
        return history

    def visualize_history(self, history, title, column1, column2):
        plt.figure("Graphics")
        plt.title(title)
        plt.grid(visible=True)
        plt.plot(history[column1])
        plt.plot(history[column2])
        plt.ylabel(column1)
        plt.xlabel('epoch')
        plt.legend([column1, column2], loc='best')
        plt.show()
```

Додаток Б: Лістинг використання класу

```
import pandas as pd
import tensorflow as tf

import tensorflow.keras.backend as K
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from tensorflow.keras import regularizers
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.applications.xception import Xception
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.layers import GlobalAveragePooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint, CSVLogger
from tensorflow.keras.optimizers import SGD

from tensorflow.keras.applications.inception_v3 import preprocess_input
import os
from collections import defaultdict
from shutil import copy
import numpy as np
import matplotlib.pyplot as plt

Fc = FoodClassifier("IV3") #створення об’єкту класу FoodClassifier, викликає «конструктор»
epochs = 5
Fc.train_model_Xc(epochs) #навчання моделі
print(Fc.predict_class(Fc.read_images(), True)) # вивід результатів класифікації #в консоль та на екран
Fc.evaluate_model(0.15) # Валідація моделі за 15% від тестових файлів
history = Fc.read_history ('Your path to/historyXc.log') #зчитування логу
Fc.visualize_history(history,'Xception train/validation accuracy','accuracy', 'val_accuracy')#побудова графіків по логу
Fc.visualize_history(history,'Xception train/validation loss', 'loss', 'val_loss')  #побудова графіків по логу
```

Додаток В: лістинг проекту в google colab

```
import tensorflow as tf

import tensorflow.keras.backend as K
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from tensorflow.keras import regularizers
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.applications.xception import Xception
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.layers import GlobalAveragePooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint, CSVLogger
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.regularizers import l2

from tensorflow import keras
from tensorflow.keras import models
from tensorflow.keras.applications.inception_v3 import preprocess_input

import cv2
import os
import random
import collections
from collections import defaultdict

from shutil import copy
from shutil import copytree, rmtree

import numpy as np

import matplotlib.pyplot as plt
import matplotlib.image as img
%matplotlib inline
print(tf.__version__)
print(tf.test.gpu_device_name())

def get_data_extract():
  if "food-101" in os.listdir():
    print("Dataset already exists")
  else:
    tf.keras.utils.get_file(
    'food-101.tar.gz',
    'http://data.vision.ee.ethz.ch/cvl/food-101.tar.gz',
    cache_subdir='/content',
    extract=True,
    archive_format='tar',
    cache_dir=None
    )
    print("Dataset downloaded and extracted!")

get_data_extract()
```

```
data_dir = "food-101/images/"

def prepare_data(filepath, src,dest):
  classes_images = defaultdict(list)
  with open(filepath, 'r') as txt:
      paths = [read.strip() for read in txt.readlines()]
      for p in paths:
        food = p.split('/')
        classes_images[food[0]].append(food[1] + '.jpg')

  for food in classes_images.keys():
    print("\nCopying images into ",food)
    if not os.path.exists(os.path.join(dest,food)):
      os.makedirs(os.path.join(dest,food))
    for i in classes_images[food]:
      copy(os.path.join(src,food,i), os.path.join(dest,food,i))
  print("Copying Done!")

print("Creating train data...")
prepare_data('food-101/meta/train.txt', 'food-101/images', 'food-101/train')
print("Creating test data...")
prepare_data('food-101/meta/test.txt', 'food-101/images', 'food-101/test')
train_files = sum([len(files) for i, j, files in os.walk("food-101/train")])
print("Total number of samples in train folder")
print(train_files)
test_files = sum([len(files) for i, j, files in os.walk("food-101/test")])
print("Total number of samples in test folder")
print(test_files)
def create_model(n_classes):
  K.clear_session()

  inception = InceptionV3(weights='imagenet', include_top=False)
  x = inception.output
  x = GlobalAveragePooling2D()(x)
  x = Dense(128,activation='relu')(x)
  x = Dropout(0.2)(x)
  predictions = Dense(n_classes,kernel_regularizer=regularizers.l2(0.005), activation='softmax')(x)
  model = Model(inputs=inception.input, outputs=predictions)
  return model
def create_model_Xc(n_classes):
  K.clear_session()
  n_classes = 101
  xception = Xception(weights='imagenet', include_top=False)
  x = xception.output
  x = GlobalAveragePooling2D()(x)
  x = Dense(128, activation='relu')(x)
  x = Dropout(0.2)(x)
  predictions = Dense(n_classes, kernel_regularizer=regularizers.l2(0.005), activation='softmax')(x)
  model = Model(inputs=xception.input, outputs=predictions)
  return model
from google.colab import drive
drive.mount('/content/gdrive')

n_classes = 101
```

```
if os.path.exists('/content/gdrive/MyDrive/models/Xcbestmodel_'+str(n_classes)+'class.hdf5'):
  %%time
  K.clear_session()
  model = load_model('/content/gdrive/MyDrive/models/Xcbestmodel_'+str(n_classes)+'class.hdf5')
  print("Loaded model")
else:
  #model = create_model(n_classes)
  model = create_model_Xc(n_classes)
  print("Created model")

def train_model(model, n_classes,num_epochs, nb_train_samples,nb_validation_samples):
  K.clear_session()
  img_width, img_height = 299, 299
  train_data_dir = 'food-101/train'
  validation_data_dir = 'food-101/test'
  batch_size = 64
  bestmodel_path = '/content/gdrive/MyDrive/models/bestmodel_'+str(n_classes)+'class.hdf5'
  trainedmodel_path = '/content/gdrive/MyDrive/models/trainedmodel_'+str(n_classes)+'class.hdf5'
  history_path = '/content/gdrive/MyDrive/models/history_'+str(n_classes)+'.log'

  train_datagen = ImageDataGenerator(
      preprocessing_function=preprocess_input,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True)

  test_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)

  train_generator = train_datagen.flow_from_directory(
      train_data_dir,
      target_size=(img_height, img_width),
      batch_size=batch_size,
      class_mode='categorical')

  validation_generator = test_datagen.flow_from_directory(
      validation_data_dir,
      target_size=(img_height, img_width),
      batch_size=batch_size,
      class_mode='categorical')

  model.compile(optimizer=SGD(lr=0.01, momentum=0.9), loss='categorical_crossentropy', metrics=['accuracy'])
  checkpoint = ModelCheckpoint(filepath=bestmodel_path, verbose=1, save_best_only=True)
  csv_logger = CSVLogger(history_path)

  history = model.fit_generator(train_generator,
                    steps_per_epoch = nb_train_samples // batch_size,
                    validation_data=validation_generator,
                    validation_steps=nb_validation_samples // batch_size,
                    epochs=num_epochs,
                    verbose=1,
                    callbacks=[csv_logger, checkpoint])

  model.save(trainedmodel_path)
  class_map = train_generator.class_indices
```

```
  return history, class_map

def train_model_Xc(model, n_classes,num_epochs, nb_train_samples,nb_validation_samples):
  K.clear_session()
  img_width, img_height = 299, 299
  train_data_dir = 'food-101/train'
  validation_data_dir = 'food-101/test'
  batch_size = 32
  bestmodel_path = '/content/gdrive/MyDrive/models/Xcbestmodel_'+str(n_classes)+'class.hdf5'
  trainedmodel_path = '/content/gdrive/MyDrive/models/Xctrainedmodel_'+str(n_classes)+'class.hdf5'
  history_path = '/content/gdrive/MyDrive/models/Xchistory_'+str(n_classes)+'.log'

  train_datagen = ImageDataGenerator(
      preprocessing_function=preprocess_input,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True)

  test_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)

  train_generator = train_datagen.flow_from_directory(
      train_data_dir,
      target_size=(img_height, img_width),
      batch_size=batch_size,
      class_mode='categorical')

  validation_generator = test_datagen.flow_from_directory(
      validation_data_dir,
      target_size=(img_height, img_width),
      batch_size=batch_size,
      class_mode='categorical')

  model.compile(optimizer=SGD(lr=0.01, momentum=0.9), loss='categorical_crossentropy', metrics=['accuracy'])
  checkpoint = ModelCheckpoint(filepath=bestmodel_path, verbose=1, save_best_only=True)
  csv_logger = CSVLogger(history_path)

  history = model.fit_generator(train_generator,
                    steps_per_epoch = nb_train_samples // batch_size,
                    validation_data=validation_generator,
                    validation_steps=nb_validation_samples // batch_size,
                    epochs=num_epochs,
                    verbose=1,
                    callbacks=[csv_logger, checkpoint])

  model.save(trainedmodel_path)
  class_map = train_generator.class_indices
  return history, class_map

epochs = 5
nb_train_samples = train_files
nb_validation_samples = test_files

history, class_map_101 = train_model_Xc(model,n_classes,epochs, nb_train_samples,nb_validation_samples)
print(class_map_101)
```

Додаток Г: результати валідації на різній кількості файлів

```
Tested 1.0% of validation files, summary 303 files
Top 1 accuracy = 0.77
Top 5 accuracy = 0.91
```

```
Tested 2.0% of validation files, summary 505 files
Top 1 accuracy = 0.77
Top 5 accuracy = 0.92
```

```
Tested 5.0% of validation files, summary 1313 files
Top 1 accuracy = 0.78
Top 5 accuracy = 0.92
```

```
Tested 10.0% of validation files, summary 2525 files
Top 1 accuracy = 0.79
Top 5 accuracy = 0.93
```

```
Tested 15.0% of validation files, summary 3838 files
Top 1 accuracy = 0.78
Top 5 accuracy = 0.93
```

```
Tested 20.0% of validation files, summary 5050 files
Top 1 accuracy = 0.78
Top 5 accuracy = 0.93
```

```
Tested 25.0% of validation files, summary 6363 files
Top 1 accuracy = 0.78
Top 5 accuracy = 0.93
```

```
Tested 1.0% of validation files, summary 303 files
Top 1 accuracy = 0.77
Top 5 accuracy = 0.90
```

```
Tested 2.0% of validation files, summary 505 files
Top 1 accuracy = 0.79
Top 5 accuracy = 0.92
```

```
Tested 5.0% of validation files, summary 1313 files
Top 1 accuracy = 0.79
Top 5 accuracy = 0.93
```

```
Tested 10.0% of validation files, summary 2525 files
Top 1 accuracy = 0.79
Top 5 accuracy = 0.94
```

```
Tested 15.0% of validation files, summary 3838 files
Top 1 accuracy = 0.79
Top 5 accuracy = 0.94
```

```
Tested 20.0% of validation files, summary 5050 files
Top 1 accuracy = 0.78
Top 5 accuracy = 0.94
```

```
Tested 25.0% of validation files, summary 6363 files
Top 1 accuracy = 0.78
Top 5 accuracy = 0.94
```