НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ННК "Інститут прикладного системного аналізу"
(повна назва інституту/факультету)

Системного проектування
(повна назва кафедри)

«На правах рукопису»
УДК 004:004.453

«До захисту допущено»
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
"___" _____2018 р.

Магістерська дисертація

зі спеціальності (спеціалізації) 122 – комп'ютерні науки та інформаційні технології (Системне проектування сервісів)
(код і назва спеціальності)

на тему: Класифікація зображень на GPU за допомогою нейронних мереж

Виконав (-ла): студент (-ка) 6 курсу, групи ДА-62м
(шифр групи)

Галушко Марія Олегівна _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник к.т.н. Харченко К. В._____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант Розробка стартап-проекту к.т.н. Харченко К. В
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент_____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації
немає запозичень з праць інших авторів
без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Інститут/факультет ННК "Інститут прикладного системного аналізу"
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною (освітньо-науковою) програмою

Спеціальність (спеціалізація) 122 – комп'ютерні науки та інформаційні технології (Системне проектування сервісів)
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
«___»_____2018 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Галушко Марії Олегівні
(прізвище, ім'я, по батькові)

1. Тема дисертації Класифікація зображень на GPU за допомогою нейронних мереж
науковий керівник дисертації К.т.н. Харченко Костянтин Васильович,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Строк подання студентом дисертації

3. Об'єкт дослідження архітектури нейронних мереж, підходи, що дозволяють отримати оптимальну мережу для використання її в реальних задачах з прийнятною точністю

4. Предмет дослідження (Вихідні дані – для магістерської дисертації за освітньо-професійною програмою) система, яка класифікує зображення на 69 класів(товари побуту)

5. Перелік завдань, які потрібно розробити
   1. Проаналізувати існуючі методи та засоби класифікації зображень.
   2. Розробка алгоритмів класифікації зображень за допомогою нейронних мереж на GPU.
   3. Огляд результатів тестування розроблених моделей.
   4. Розробка стартап-проекту.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу

презентація на тему “ Класифікація зображень на GPU за допомогою нейронних мереж”

7. Орієнтовний перелік публікацій
   1. Галушко М. Класифікація зображень на GPU за допомогою нейронних мереж // Системний аналіз та інформаційні технології: 20-th міжнародна конференція САІТ 2018, Київ, Україна.
   2. Halushko M. Knowledge distillation approach for image classification tasks / M. Halushko, K. Kharchenko. // IEEE First International Conference on System Analysis & Intelligent Computing (SAIC). – 2018.

8. Консультанти розділів дисертації[1*]

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розроблення стартап-проекту	Харченко К.В.		

9. Дата видачі завдання 01.02.2018

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	01.02.2018	
2	Огляд існуючих рішень та методів	15.02.2018	
3	Огляд та порівняння існуючих архітектур. Вибір чотирьох для аналізу	05.03.2018	
4	Побудова власної архітектури.	30.03.2018	
5	Оцінка розроблених моделей та налаштування параметрів	10.04.2018	
6	Налаштування параметрів, підбір параметрів для отримання	17.04.2018	

[1*] Консультантом не може бути зазначено наукового керівника

	оптимальної моделі методом обміну знань		
7	Оцінка результатів всіх отриманих моделей	20.04.2018	
8	Отримання допуску до захисту та подача роботи в ДЕК	10.05.2018	

Студент _____ ____М.О. Галушко____
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ ____К.В. Харченко____
(підпис) (ініціали, прізвище)

РЕФЕРАТ

магістерської дисертації Галушко Марії Олегівни на тему «Класифікація зображень на GPU за допомогою нейронних мереж»

У магістерській дисертації досліджується класифікація зображень на GPU за допомогою нейронних мереж, а саме на прикладі категоризації товарів побуту. Дана тема є актуальною, так як у повсякденному життя нас оточують зображення і людині їх легко інтерпретувати, а комп’ютеру набагто складніше, тим більше класифікувати чи сегментувати зображення.

Метою роботи є виявлення ефективних способів та засобів класифікації зображень у прикладних системах, таких як категоризація товарів на сайті. Об’єктом дослідження є аналіз способів і засобів використання нейронних мереж у прикладних системах та процес такої мережі, яка б відповідала поставленим цілям роботи.

Було виконано огляд існуючих аналогів готового продукту дипломної роботи – автоматичних систем категоризації товарів та зроблено висновки щодо їх недоліків та переваг при виконанні роботи. Створено систему, яка автоматично класифікує товари, модифікуючи існуючі підходи, отримано власний, який для даної задачі працює краще. Вдосконаливши продукт можна використовувати його для будь-яких організацій, де зручно було б автоматично класифікувати продукти.

Отже, науковою новизною є модифікований підхід використання архітектур нейронних мереж, підібрано параметри моделей та процес оцінки моделей.

Загальний обсяг роботи: 86 сторінки, 30 рисунків, 28 таблиць та 25 бібліографічних найменувань.

Ключові слова: машинне навчання, нейронні мережі, класифікація зображень.

РЕФЕРАТ

магистерской диссертации Галушко Марии Олеговны на тему
«Классификация изображений на GPU с помощью нейронных сетей»

В магистерской диссертации исследуется классификация изображений на GPU с помощью нейронных сетей, а именно на примере категоризации товаров быта. Данная тема является актуальной, так как в повседневной жизни нас окружают изображения и человеку их легко интерпретировать, а компьютеру намного сложнее, тем более классифицировать или сегментировать изображения.

Целью работы является выявление эффективных способов и средств классификации изображений в прикладных системах, таких как категоризация товаров на сайте. Объектом исследования является анализ способов и средств использования нейронных сетей в прикладных системах и процесс такой сети, соответствующей поставленным целям работы.

Было выполнено обзор существующих аналогов готового продукта дипломной работы - автоматических систем категоризации товаров и сделаны выводы относительно их недостатков и преимуществ при выполнении работы. Создана система, которая автоматически классифицирует товары, модифицируя существующие подходы, получено собственный, который для данной задачи работает лучше.

Итак, научной новизной является модифицированный подход, использование архитектур сетей, подобрано параметры и процесс их оценки.

Общий объем работы: 86 страницы, 30 рисунков, 28 таблиц и 25 библиографических наименований.

Ключевые слова: машинное обучение, нейронные сети, классификация изображений.

ABSTRACT

for master's thesis of Maria Olegivna Halushko

On “Image classification on the GPU using neural network”

In the master's dissertation, the classification of images on the GPU with the help of neural networks is investigated, namely on the example of the categorical of goods of everyday life. This topic is relevant because in everyday life we are surrounded by images and it is easy for them to interpret them, and it's more difficult for a computer to work up, the more it is classified or segmented.

The goal of the work is to identify effective ways and means of categorizing images in application systems, such as product categorization on the site.

The object of the study is to analyze the methods and means of using neural networks in application systems and the process of such a network that would meet the goals of the work.

A review of the existing analogs of the finished product of the thesis - automatic categorization of goods was made and conclusions were drawn regarding their drawbacks and advantages when performing the work. A system has been created that automatically classifies goods, modifying existing approaches, obtains own, which works better for the given task. After perfecting the product, you can use it for any organization where it would be convenient to automatically classify the products.

Consequently, the scientific novelty is a modified approach, the use of neural network architectures model parameters and model evaluation process are selected.

A total volume of work: 86 pages, 30 figures, 28 tables and 25 bibliographic titles.

Keywords: machine learning, neural networks, classification of images.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ВЛ	Випадкові ліси
ГС	Градієнтний спуск
ДР	Дерева рішень
НБК	Наївний баєсовий класифікатор
НМ	Нейронна мережа
ЗНМ	Згорткова нейронна мережа
ПЗ	Програмне забезвечення
СГС	Стохастичний градієнтний спуск
SVM (Support Vector Machine)	Метод опорних векторів
ReLU(The Rectified Linear Unit)	Прямокутний лінійний елемент

ВСТУП

Розвиток наукоємних галузей людської діяльності в сучасному суспільстві супроводжується зростанням ролі комп'ютерних технологій. Зараз значно збільшується потік інформації, з'явилася необхідність пошуку нових способів її зберігання, подання, формалізації і систематизації, а також автоматичної обробки.

Ми живемо в цікаву епоху, де все швидко росте та змінюється. Це стосується й інформації. Її кількість з кожним роком все більше і більше зростає. Сюди входять різні дані, аудіо та медіа контент, тексти.

Як наслідок це призводить до збільшення ентропії інформації. Що породжує проблему структуризації, підбору потрібного нам рішення, необхідної покупки і т.д. Тобто нас все важче й важче віднайти необхідно інформацію.

Вперше нейронні мережі привернули загальну увагу в 2012 році, коли Алекс Крижевський завдяки їм виграв конкурс ImageNet, побивши рекорд помилок класифікації з 26% до 15% (що майже в два рази відрізняється), і на той момент це стало проривом. На даний час глибинне навчання лежить в основі послуг багатьох компаній: Facebook використовує нейронні мережі для алгоритмів автоматичного створення тегів, Google - для пошуку серед фотографій користувача, Amazon - для генерації рекомендацій товарів, Pinterest - для персоналізації домашньої сторінки користувача, а Instagram - для пошукової інфраструктури.

Завдання класифікації зображень - це прийом початкового зображення і виведення його класу (кішка, собака і т.д.) або групи ймовірних класів, яка найкраще характеризує зображення. Для людей це один з перших навичок, який вони починають освоювати з народження.

У дипломній роботі буде вирішуватись задача категоризації предметів побуту. Так як для великих поставок категоризація товару може забирати багато часу у працівників, автоматизація цього процесу значно спростить їхню роботу. Також це може бути корисно, якщо на сайті не дуже очевидна категоризація (з категоріями і підкатегоріями) то користувачу, який захоче продати/купити товар буде легше розмістити оголошення/купити товар по фото.

Поставлена мета вимагає вирішення наступних наукових задач:

- дослідження, очищення і обробка набору даних;
- підбір та проектування архітектури нейронної мережі;
- вибір метрики оцінки алгоритму, вибір способу валідації;
- дослідження впливу компонентів нейронної мережі на точність класифікації;
- дослідження впливу гіперпараметрів на точність класифікації.

Метою дипломної роботи є розробка такої моделі, що дозволила б визначати категорію товарів предметів для дому, а саме таких категорій буде 69. Важливо проаналізувати підходи до вирішення таких задач, щоб вибрати такий, який найефективніше б підійшов для конкретної задачі категоризації на 69 класів. Не менш істотним у даному контексті є глибокий аналіз структури та підходів до здобуття, аналізу та обробки даних.

Об’єктом дослідження є аналіз способів і засобів використання нейронних мереж у прикладних системах та процес такої мережі, яка б відповідала поставленим цілям роботи. Предметом дослідження є архітектура нейронної мережі.

Досягнення поставленої мети реалізовано з використанням мови програмування Python, фреймворк Keras для роботи безпосередньо з архітектурами нейронних мереж та імплементацією різних методів та бібліотек numPy, scikit-learn, HTML5, CSS3, фреймворк Flask та мова JavaScript для створення сайту для відображення результатів.

Наукова новизна дипломної роботи полягає в тому, що було створено систему, яка класифікує товари з точністю 92.5% і за розмірами є досить мала і швидка, таким чином модель можна використовувати на смартфоні, результат передбачень для нового забраження можна отримати за 10 мс. На даний момент не має аналогів такій систем з відкритим вихідним кодом.

Потенційні застосування та практична цінність результатів дипломної роботи:

1. Модель може використовуватись на будь-якому сайті чи мобільному додатку, де потрібна автоматична категоризація товарів предметів побуту;
2. Сформульовано основні концепти, на які потрібно звернути увагу при проектуванні архітектури нейронної мережі, описано метод та підходи, які дозволяють майже без втрат точності зменшити розміри моделі та швидкість отримання передбачень;
3. Розширивши набір даних та дотренувавши систему, можна використовувати для більшої кількості категорій, також можна взяти інший набір даних і спробувати застосувати ці ж методи та архітектури, можливо, трохи змінити параметри мережі і отримати іншу готову систему для класифікації зображень.

1. ДОСЛІДЖЕННЯ ІСНУЮЧИХ МЕТОДІВ ТА ЗАСОБІВ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ

1.1 Класифікація зображень за допомогою методів комп’ютерного зору

Ознаки Хаара - ознаки цифрового зображення, використовувані в розпізнаванні образів. Своєю назвою вони зобов'язані інтуїтивним схожістю з вейвлетами Хаара. Ознаки Хаара використовувалися в першому детекторі осіб, що працює в реальному часі.

Історично склалося так, що алгоритми, що працюють тільки з інтенсивністю зображення (наприклад значення RGB в кожному пікселі), мають велику обчислювальну складність. В роботі Папагеоргіу [1], була розглянута робота з безліччю ознак, заснованих на вейвлет Хаара. Віола і Джонс [2] адаптували ідею використання вейвлетів Хаара та розробили те, що було названо ознаками Хаара. Ознака Хаара складається з суміжних прямокутних областей. Вони позиціонуються на зображенні, далі сумуються інтенсивності пікселів в областях, після чого обчислюється різниця між сумами. Ця різниця і буде значенням певної ознаки, визначеного розміру, певним чином позиційований на зображенні.

Для прикладу розглянемо базу даних з людськими обличчями. Загальним для всіх зображень є те, що область в районі очей темніше, ніж область в районі щік. Отже загальним ознакою Хаара для осіб є 2 суміжних прямокутних регіону, що лежать на очах і щоках.

На етапі виявлення в методі Віоли - Джонса вікно встановленого розміру рухається по зображенню, і для кожної області зображення, над якою проходить вікно, розраховується ознака Хаара. Наявність або відсутність предмета в вікні

визначається різницею між значенням ознаки і учнем порогом. Оскільки ознаки Хаара мало підходять для навчання або класифікації (якість трохи вище ніж у випадкової нормально розподіленої величини), для опису об'єкта з достатньою точністю необхідна більша кількість ознак. Тому в методі Віоли - Джонса ознаки Хаара організовані в каскадний класифікатор.

Ключовою особливістю ознак Хаара є найбільша, в порівнянні з іншими ознаками, швидкість. При використанні інтегрального представлення зображення, ознаки Хаара можуть обчислюватися за постійний час (приблизно 60 процесорних інструкцій на ознака з двох областей).

1.2 Класифікація зображень за допомогою методів машинного навчання

Машинне навчання успішно використовується в задачах комп'ютерного зору, роботи з текстами, медицини, безпілотного керування дронів та ін. Ядром для багатьох з цих додатків є методи та засоби штучного інтелекту, такі як класифікація, локалізація і виявлення. За декілька попередніх років нейронні мережі дуже добре себе показали у задачах, де дані представлені у вигляді текстів чи зображень, відео, тому зараз їх використовують досить часто для таких задач. Для задач класифікації можна використовувати класичні методи машинного навчання, такі як: наївний Баєсовий класифікатор, метод опорних векторів, логістичну регресію, випадкові ліси чи метод стимулювання градієнта.

MNIST — об'ємна база даних зразків рукописного написання цифр. База даних є стандартом, запропонованим Національним інститутом стандартів і технологій США з метою калібрації і зіставлення методів розпізнавання зображень за допомогою машинного навчання в першу чергу на основі нейронних мереж [1]. Дані складаються з заздалегідь підготовлених прикладів зображень, на

основі яких проводиться навчання та тестування систем. База даних MNIST містить 60000 зображень для навчання і 10000 зображень для тестування. На цьому наборі даних метод наївного байєса працює досить точно, а саме за джерелом [5] вдалось отримати 85% точності. Так як наївний баєсовий класифікатор — модель, що базується на теоремі Баєса для визначення ймовірності приналежності екземпляра до одного з класів C за умови того, що залежні змінні приймають задані значення. Це означає що, якщо на основі відомих змінних можна точно визначити, до якого класу належить екземпляр, баєсовий класифікатор підсумує, що ймовірність приналежності до даного класу рівна 1. У інших випадках, коли екземпляр може з різною ймовірністю належати до різних класів, результатом роботи класифікатора буде набір ймовірностей приналежності до певного класу [12]. Виходячи з визначення і самого набору даних, можна пояснити чому ця модель дала прийнятну точність, а саме, що зазвичай люди пишуть схоже і так як зображення на вхід ми подаємо у вигляді пікселів, то основна частина буде розміщатись у одного типу цифр в тих же місцях.

Але, на реальних даних, зокрема і в цій роботі, краща точність була отримана за допомогою згорткових нейронних мереж. Детальніше про ці алгоритми у наступному розділі. Але варто зазначити, що на наборі даних ImageNet - набір даних, за станом на 2017 рік там було 14 197 122 зображення, розбитих на 21 841 категорію, за допомогою різних архітектур нейронних мереж та підходів було отримано на 2017 рік точніть більше 95%.

Отже, класичні методи машинного навчання, такі як - наївний Баєсовий класифікатор, метод опорних векторів, логістичну регресію, випадкові ліси чи метод стимулювання градієнта, а також методи глибинного навчання, дають хороші результати для класифікації зображень. Звісно, метод потрібно вибирати виходячи зі специфіки даних та інших характеристик, таких як - задовільна

точність алгоритму, час передбачення і тд. Зазвичай на реальних даних надають перевагу згортковим нейронним мережам, так як класичні методи не здатні виявити всі необхідні закономірності.

1.3 Комбінація методів комп'ютерного зору та нейронних мереж

Класифікація цифрових зображень, що зберігаються в базах даних, з використанням традиційних алгоритмів машинного навчання характеризується високою трудомісткістю, що обумовлено великою кількістю зображень і деталей, якими описуються зображення. Зазначені алгоритми характеризуються невисокою стабільністю при класифікації зображення з великих баз даних. Крім того, класифікація з використанням цих алгоритмів вимагає великих тимчасових витрат. Існуючі системи зберігання зображень, такі як QBIC [5] и VisualSEEK [4], які обмежують методи класифікації способами опису зображень, заснованими на формі, текстурі і колірній інформації [6].

Одним з методів, що використовуються для розпізнавання, класифікації та відновлення зображень, є метод, заснований на нейронних мережах. Для того щоб зменшити число вхідних нейронів мережі, система класифікації зображень зазвичай розташовується на кроці предобробки. Одним з кроків предобробки цифрових зображень є вейвлет-перетворення. В даний час вейвлет-перетворення є широко відомим методом, застосовуваним для аналізу зображень та отримання таких характеристик зображення, як форма і текстура.

Алгоритм, заснований на комбінації вейвлет-перетворення Хаара і нейронної мережі для класифікації цифрових зображень з бази даних. Кольорове зображення ділиться на три RGB-компоненти. Моменти кольору першого порядку і

коефіцієнти розкладання вейвлет перетворення Хаара [7] трьох RGB-компонентів зображень є вхідним вектором багатошарової нейронної мережі, навченої алгоритмом зворотного поширення помилки.

Подання змісту цифрових зображень. Зміст і контури цифрових зображень зазвичай використовуються в класифікації зображень. У алгоритмі моменти кольору і вейвлет-коефіцієнти розкладання використовуються для подання змісту цифрових зображень.

Моменти кольору. Моменти кольору використовуються в багатьох системах відновлення кольорового зображення [8], особливо в тих випадках, коли зображення містить тільки один об'єкт. Моменти кольору першого, другого і третього порядків є ефективними для подання розподілу кольору зображень.

Вейвлет-перетворення зазвичай використовується в системах відновлення змісту зображень. На кожному рівні вейвлет-перетворення сигнал розкладається на чотири піддіапазони частот (квадранта) LLn, LHn, HLn, HHn (рис 1.1) , де L - низька частота; H - висока частота; n - рівень розкладання. На рис. 1.1 представлені стандартні позначення квадрантів перетвореного зображення: LL, LH, HL, HH. Квадрант LLn являє собою зображення з низьким дозволом (cAn), HLn - вертикальні деталі зображення (cVn), ЛНн - горизонтальні деталі зображення (cHn), HHn - діагональну інформацію зображення (cDn). У алгоритмі використовується вейвлет-перетворення Хаара [8] по шести рівням розкладання. Отримані вейвлет-коефіцієнти подаються на входи нейронної мережі.

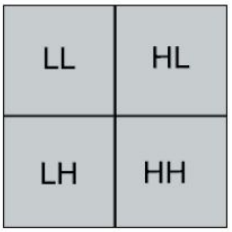


Рисунок 1.1 - Одноразове застосування двовимірного вейвлет-перетворення до квадратному зображенню [5]

У роботі [10] зазначалось, що максимальна отримана точність класифікації була 88%.

1.4 Огляд існуючих рішень

За останнє десятиліття було опубліковано значну кількість статей і наукових робіт про алгоритми класифікації зображень у різних областях застосування, від класифікації предметів побуту, тварин до класифікації ракових пухлин чи знімків із космосу. Крім того, було запропоновано наступні підходи: класифікація зображень, де на виході ми використовуємо лише мітку класу або ж використання ймовірності того, що на даному зображені клас з певною міткою. Останній підхід буде частіше використовуватись у системах, де помилки є не припустимими, наприклад, при класифікації ракових клітин. У статті[10] було описано, що класичні підходи до вирішення таких задач на наборі даних, який вони використовували дали 85% точності. У роботі[11], де потрібно було класифікувати 53 класи на їхньому наборі даних використовували специфічну обробку даних за рахунок чого їм і вдалось отримати 87% точності. Також була розглянуто рішення[12], де було поєднано два вище зазначених підходи, що дозволило покращити результати. Звісно були і статті, результати яких відтворити не вдалось, так як деякі з них використовували закритий набір даних чи дуже

багато обчислювальних потужностей.

1.5 Висновок

У розділі було проаналізовано підходи та методи виконання задач багатокласової класифікації зображень. Було описано як класичні методи комп’ютерного зору так і методи машинного навчиння. Як ми бачимо, на даний момент найкращі результати класифікації зображень було отримано за допомогою використання нейронних мереж, а саме згорткових нейронних мереж. Що і буде далі досліджуватись в даній роботі.

Також було виконано огляд існуючих рішень та статей, де вирішувались проблеми класифікації зображень. На даний момент немає статей чи рішень, які б демонстрували свої результати на відкритому наборі даних, а саме наборі даних товарів побуту. Але є рішення класифікації зображень, коли класифікували від 2 до 200 класів, деякі із розглянутих рішень[15] було взято за основу. У наступних розділах буде описано вхідні дані, які були проблеми, як вони вирішувались та аналіз вибору і реалізація методів, які дали в кінцевому рахунку 88.7% точності класифікації.

2. РОЗРОБКА АЛГОРИТМІВ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ ЗА ДОПОМОГОЮ НЕЙРОННИХ МЕРЕЖ НА GPU

2.1 Нейронні мережі

2.2.1 Перцептрон

“Перцептрон - це одна з перших моделей нейронних мереж. Незважаючи на свою простоту, ця модель здатна навчатися і вирішувати досить складні завдання. Основна математична задача, з якою він справляється, - це лінійне розділення будь-яких нелінійних множин, так зване забезпечення лінійної сепарабельності [12]”. Типова архітектура багатошарового перцептрону схематично зображена на рис. 2.1.

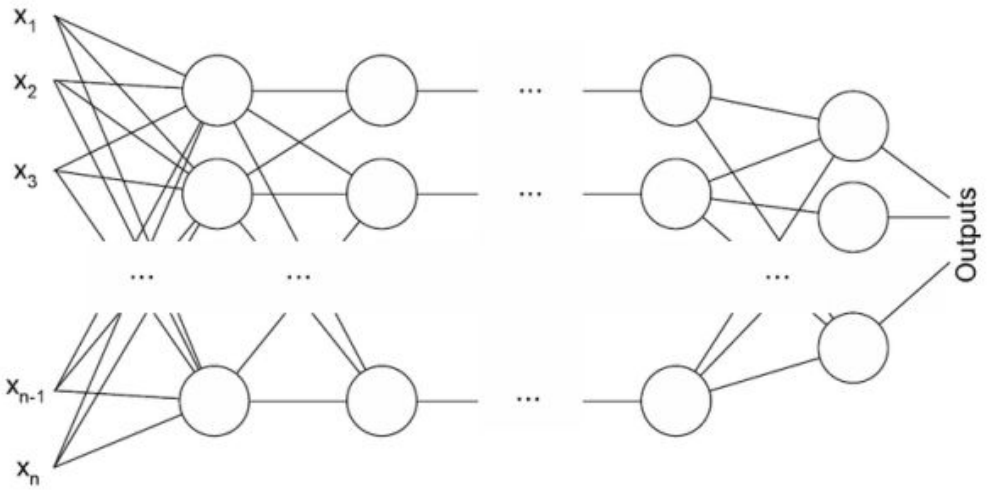


Рисунок 2.1 – Архітектура багатошарового перцептрону [5]

Перцептрон складається з трьох типів елементів, а саме: сигнали, що надходять від давачів, передаються до асоціативних елементів, а відтак до реагуючих. Таким чином, перцептрони дозволяють створити набір «асоціацій»

між вхідними стимулами та необхідною реакцією на виході. В біологічному плані це відповідає перетворенню, наприклад, зорової інформації у фізіологічну відповідь рухових нейронів. Перцептрон застосовується для вирішення класичних задач машинного (класифікація, регресія) навчання як окрема модель, так і в складі більш складних моделей.[2]

2.2.2. Згорткові нейронні мережі

“Згорткові нейронні мережі - одні з найвпливовіших інновацій в області комп'ютерного зору. Вперше нейронні мережі привернули загальну увагу в 2012 році, коли Алекс Крижевський завдяки їм виграв конкурс ImageNet (грубо кажучи, це щорічна олімпіада по машинному зору), знизивши рекорд помилок класифікації з 26% до 15%, що тоді стало проривом [16]”. Сьогодні глибинне навчання лежить в основі послуг багатьох компаній: Facebook використовує нейронні мережі для алгоритмів автоматичного створення тегів, Google - для пошуку серед фотографій користувача, Amazon - для генерації рекомендацій товарів, Pinterest - для персоналізації домашньої сторінки користувача, а Instagram - для пошукової інфраструктури.

Завдання класифікації зображень - це прийом початкового зображення і виведення його класу (кішка, собака і т.д.) або групи ймовірних класів, яка найкраще характеризує зображення. Для людей це один з перших навичок, який вони починають освоювати з народження. На рис. 2.2 показано як сприймає картинки комп’ютер.

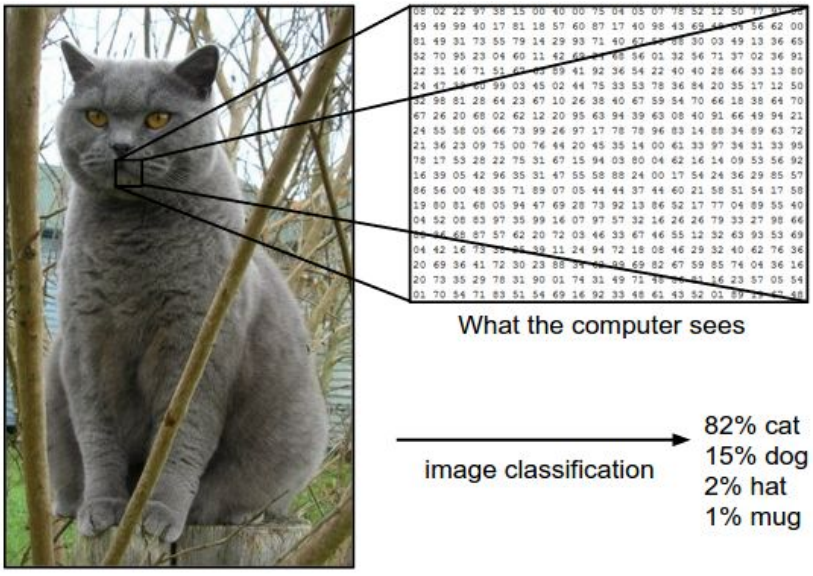


Рисунок 2.2 - Представлення зображення у вигляді матриці пікселів [5]

Коли комп'ютер бачить зображення (приймає дані на вхід), він бачить масив пікселів. Залежно від дозволу і розміру зображення, наприклад, розмір масиву може бути 32x32x3 (де 3 - це значення каналів RGB). Щоб було зрозуміліше, давайте уявимо, у нас є кольорове зображення у форматі JPG, і його розмір 480x480. Відповідний масив буде 480x480x3. Кожному з цих чисел присвоюється значення від 0 до 255, яке описує інтенсивність пікселя в цій точці. Ці цифри, залишаючись безглуздими для нас, коли ми визначаємо що на зображенні, є єдиними вступними даними, доступними комп'ютера.

Ідея нейронних мереж конструкцій запозичені з біології: нейронні мережі імітують процес обробки нейронами головного мозку сприймаються з навколишнього середовища образів і участь цих нейронів в прийнятті рішень [10]. Принцип роботи окремо взятого штучного нейрона по суті дуже простий: він обчислює зважену суму всіх елементів вхідного вектора $\underline{x}$, використовуючи вектор ваг $\underline{w}$ (а також адитивну складову зсуву w_0), а потім до результату може застосовуватися функція активації σ.

“Класичними реалізаціями ЗНМ, що стали проривом в індустрії, є LeNet5 та AlexNet [6]”(рис. 2.3).

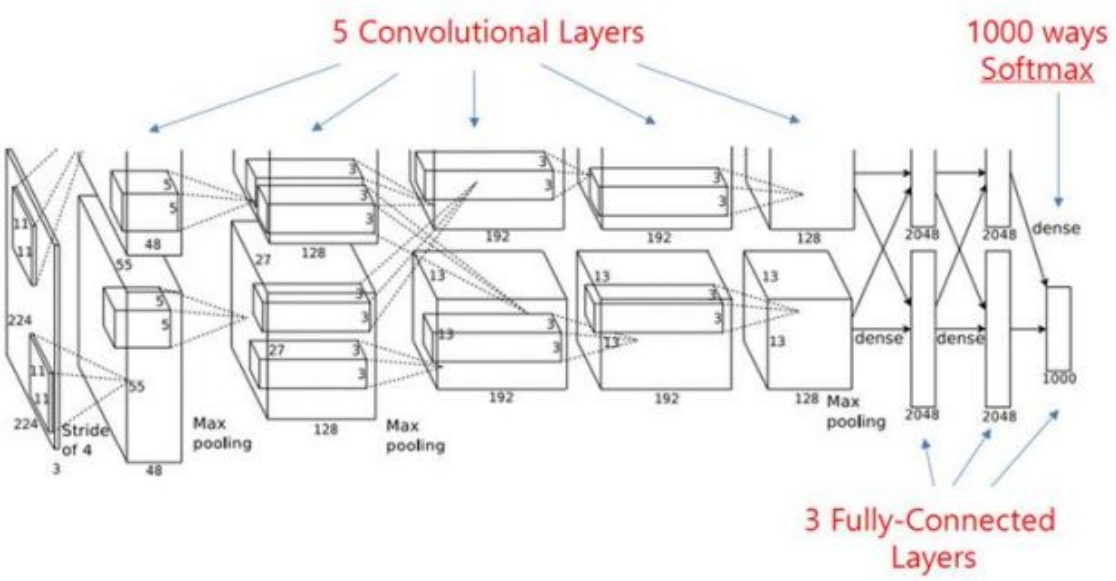


Рисунок 2.3 – Архітектура AlexNet [13]

2.2.4 Задачі, які вирішують нейронні мережі

Кажуть, що програма навчається на досвіді E щодо класу задач T в сенсі міри якості L, якщо при вирішенні задачі T якість, яка вимірюється мірою L, зростає при демонстрації нового досвіду E [5].

Рисунок 2.4 - Загальна класифікація задач машинного навчання [8]

Два основні класи задач машинного навчання - це задачі навчання з учителем (supervised learning) і навчання без вчителя (unsupervised learning).

При навчанні з учителем на вхід подається набір тренувальних прикладів, який зазвичай називають навчальним або тренувальним набором даних і завдання полягає в тому, щоб продовжити вже відомі відповіді на новий досвід, виражений зазвичай у вигляді тестового набору даних. Основне припущення тут в тому, що дані, доступні для навчання, будуть чимось схожі на дані, на яких потім доведеться застосовувати навчену модель, інакше ніяке узагальнення буде неможливо.

Завдання навчання з вчителем зазвичай діляться на завдання класифікації і регресії. У задачі класифікації потрібно поданий на вхід об'єкт визначити в один з (зазвичай кінцевого числа) класів. А в задачі регресії потрібно передбачити значення якоїсь функції, у якій зазвичай може бути нескінченно багато різних значень. Наприклад, по зростанню людини передбачити його вага, зробити прогноз завтрашньої погоди, і тд. Наприклад, в пошукових і рекомендаційних системах часто зустрічається завдання навчання ранжирування. Вона ставиться так: за наявними даними (в пошуковій системі це будуть тексти документів і минулу поведінку користувачів) отранжіровать, розставити наявні об'єкти в порядку убування цільової функції (в пошуковій системі вона називаєте я релевантність: наскільки даний документ підходить, що б видати його у відповідь на даний запит).

Якщо ж розміченого набору даних, відповідного конкретному завданні, немає, а є просто дані, в яких треба «знайти якийсь сенс», то виникають завдання навчання без учителя. Типовий приклад завдання навчання без вчителя - це кластеризація: потрібно розбити дані на заздалегідь невідомі класи по деякій мірі

схожості так, щоб точки, віднесені до одного і того ж кластеру, були якомога ближче один до одного, як можна більш схожі, а точки з різних кластерів були б якомога далі один від одного, як можна менш схожі.

Нейронні мережі, можуть використовуватись для всіх цих задач, але потрібно підібрати правильно функцію втрат.

Використання нейронних мереж має такі переваги:

1. Можуть знайти складніші закономірності в даних, в порівнянні із класичними методами машинного навчання, добре себе показали для задач з картинками і текстами;

2. Можна донавчати, коли з’являються нові дані;

3. Зазвичай добре модель, для якої вдалоо підібрані гіпер параметри та достатньо даних буде краще працювати, за класичні алгортми машинного навчання;

4. Нейронна мережа сама шукає закономірності в даних і певні признаки, по яким і навчається, робить передбачення, в той час як для класичних методів потрібно самому придумати всі ці ознаки, щоб модель краще навчилась.

Хоча використання нейронних мереж має і ряд недоліків:

1. Для того, щоб отримати хороші результати потрібно мати набагато більше даних, аніж для класичних методів;

2. Для тренування потрібно мати більше потужностей, наприклад одну або декілька відеокарт (GPU), в той час як для класичних методів достатньо виконувати обчислення на процесорі;

3. Час тренування виходячи з попередніх двох пунктів також забирає

більше часу, в порівнянні з класичними методами (хоча на це можна впливати використовуючи додаткові потужності, хмарні рішення);

4. Легше перенавчаються, важче впроваджувати в реальну систему, на меленьких пристроях, застосовувати в реальному часі.

2.3 Вибір архітектури для класифікації зображень

На даний момент існує досить багато архітектур згорткових нейронних мереж. Кожного року виходять нові статті, нові архітектури або модифікуються існуючі. В роботі було використано архітектуру DenseNet, архітектуру InceptionV3, Squeezenet та MobileNet. Дві перших архітектури досить об'ємні і навчання займало досить багато часу, але і точність була краща. Останні дві менші за розмірами так і навчались швидше, так як розраховані для використання в пристроях, де мало пам'яті і обчислювальних потужностей.

На рис 2.5 зображена архітектура DenseNet, як бачимо вона складається з трьох блоків, приклад одного такого блоку зображено на рис 2.6.

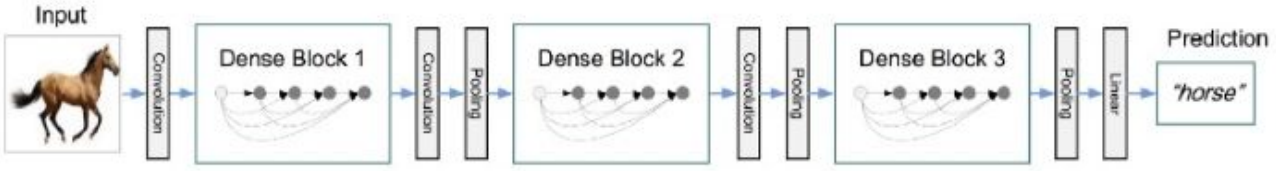


Рисунок 2.5 - Архітектура DenseNet [7]

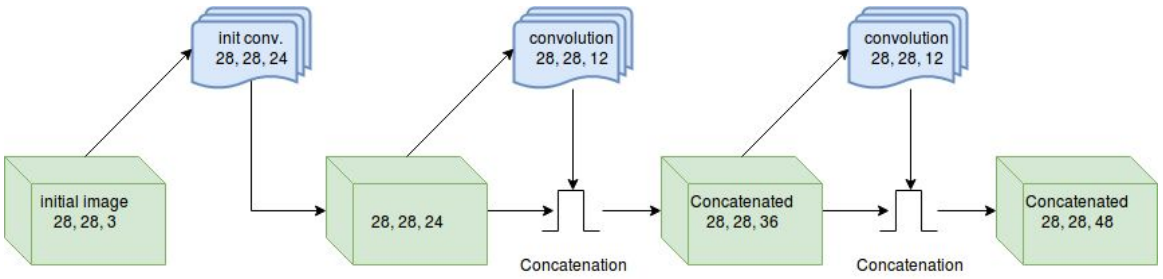


Рисунок 2.6 - Блок з архітектури DenseNet [7]

На даний момент ця архітектура показує одну з найнижчих похибок на відкритих наборах даних CIFAR/SVHN, як зображено в таблиці 2.1.

Таблиця 2.1 - Результати оцінки якості різних алгоритмів на CIFAR/SVHN наборах даних [10]

Method	Depth	Params	C10	C10+	C100	C100+	SVHN
Network in Network [22]	-	-	10.41	8.81	35.68	-	2.35
All-CNN [31]	-	-	9.08	7.25	-	33.71	-
Deeply Supervised Net [20]	-	-	9.69	7.97	-	34.57	1.92
Highway Network [33]	-	-	-	7.72	-	32.39	-
FractalNet [17]	21	38.6M	10.18	5.22	35.34	23.30	2.01
with Dropout/Drop-path	21	38.6M	7.33	4.60	28.20	23.73	1.87
ResNet [11]	110	1.7M	-	6.61	-	-	-
ResNet (reported by [13])	110	1.7M	13.63	6.41	44.74	27.22	2.01
ResNet with Stochastic Depth [13]	110	1.7M	11.66	5.23	37.80	24.58	1.75
	1202	10.2M	-	4.91	-	-	-
Wide ResNet [41]	16	11.0M	-	4.81	-	22.07	-
	28	36.5M	-	4.17	-	20.50	-
with Dropout	16	2.7M	-	-	-	-	1.64
ResNet (pre-activation) [12]	164	1.7M	11.26*	5.46	35.58*	24.33	-
	1001	10.2M	10.56*	4.62	33.47*	22.71	-
DenseNet ($k = 12$)	40	1.0M	**7.00**	5.24	**27.55**	24.42	1.79
DenseNet ($k = 12$)	100	7.0M	**5.77**	**4.10**	**23.79**	**20.20**	1.67
DenseNet ($k = 24$)	100	27.2M	**5.83**	**3.74**	**23.42**	**19.25**	**1.59**
DenseNet-BC ($k = 12$)	100	0.8M	**5.92**	4.51	**24.15**	22.27	1.76
DenseNet-BC ($k = 24$)	250	15.3M	**5.19**	**3.62**	**19.64**	**17.60**	1.74
DenseNet-BC ($k = 40$)	190	25.6M	-	**3.46**	-	**17.18**	-

На рис. 2.7 зображена архітектура InceptionV3, вона була розроблена компанією Google, перед DenseNet і вона також дає дуже хороші результати на тих же відкритих наборах даних CIFAR/SVHN. Як бачимо на малюнку, модель

складається із 11 модулів, вихід кожного з яких дається на вхід наступному.

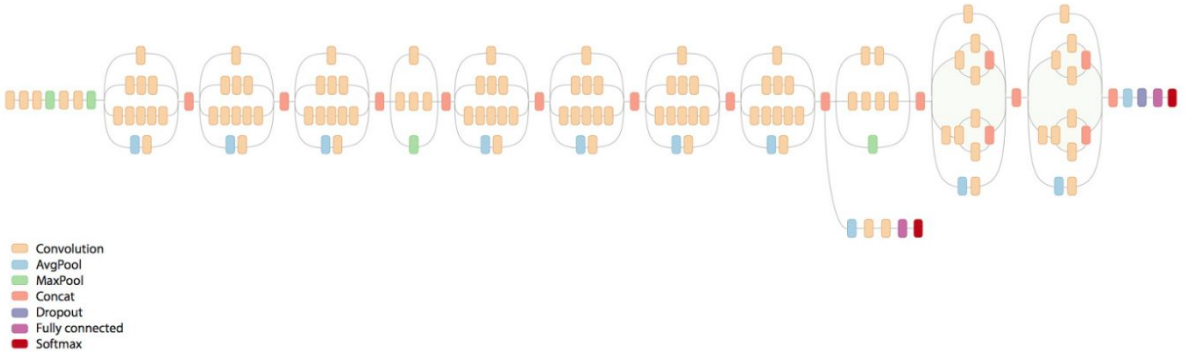


Рисунок 2.7 - Архітектура InceptionV3 [11]

На рис. 2.8 зображена архітектура Squzeenet. Робота SqueezeNet забезпечує розумну архітектуру, а також кількісний аналіз. Для такої ж точності AlexNet SqueezeNet може бути в 3 рази швидше і в 500 разів менше.

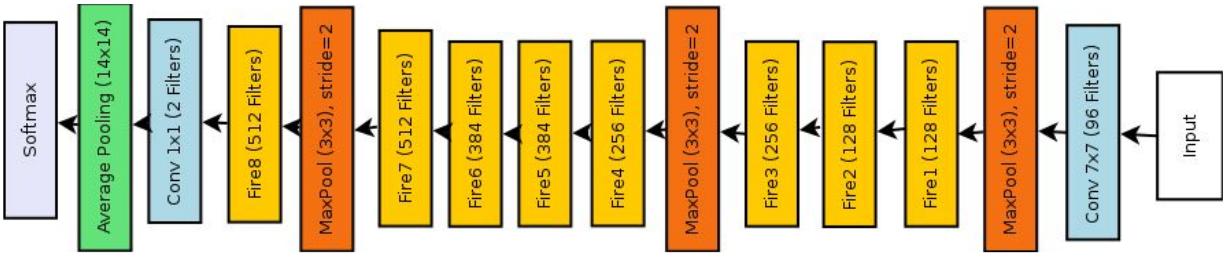


Рисунок 2.8 - Архітектура Squzeenet[18]

У таблиці 2.2 зображено переваги SqueezeNet над моделлю ALexNet(на наборі даних ImageNet), як бачимо вдалось досягти такої ж точності при тому, що модель SqeezeNet в 50 разів менша.

Таблиця 2.2 - Результати оцінки якості SqeezeNet та AlexNet [18]

CNN architecture	Compression Approach	Data Type	Original → Compressed Model Size	Reduction in Model Size vs. AlexNet	Top-1 ImageNet Accuracy	Top-5 ImageNet Accuracy
AlexNet	None (baseline)	32 bit	240MB	1x	57.2%	80.3%
AlexNet	SVD (Denton et al., 2014)	32 bit	240MB → 48MB	5x	56.0%	79.4%
AlexNet	Network Pruning (Han et al., 2015b)	32 bit	240MB → 27MB	9x	57.2%	80.3%
AlexNet	Deep Compression (Han et al., 2015a)	5-8 bit	240MB → 6.9MB	35x	57.2%	80.3%
SqueezeNet (ours)	None	32 bit	4.8MB	**50x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	8 bit	4.8MB → 0.66MB	**363x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	6 bit	4.8MB → 0.47MB	**510x**	57.5%	80.3%

На рис. 2.9 зображено архітектуру MobileNet, яка показує схожі результати до попередньої. Зазвичай ці архітектури використовують для мобільних пристроях чи інших пристроях з обмеженою пм’яттю.

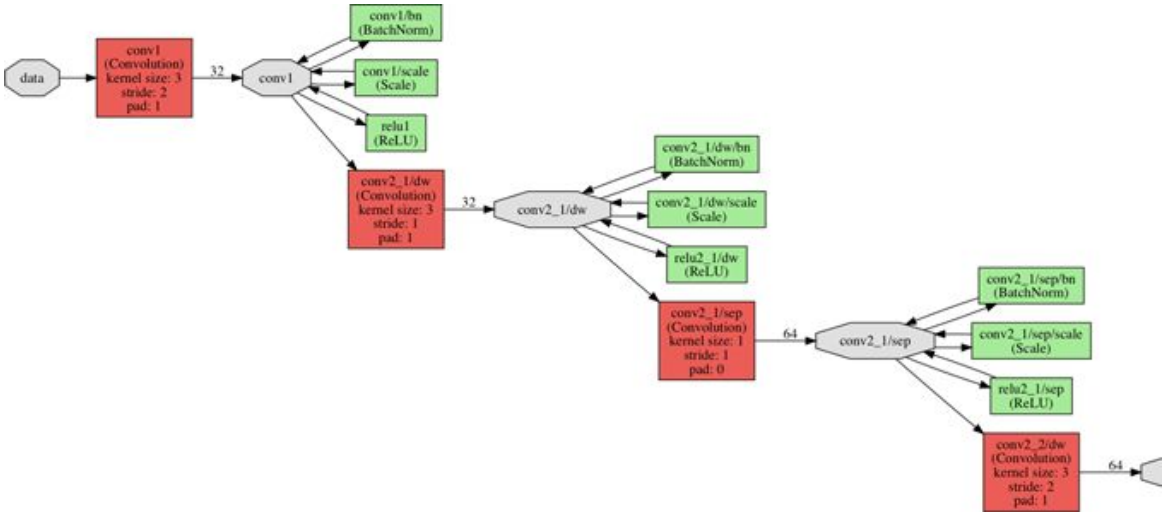


Рисунок 2.9 - Архітектура MobileNet [20]

Отже, проаналізувавши попередні архітектури можна зробити висновок, що більші архітектури такі як DenseNet, InceptionV3 навчаються набагато довше в порівнянні з двома іншими, але дають трохи більшу точність. На деяких наборах даних можна отримати точність, яка буде відрізнятись до 5% (якщо порівнювати

всі попередні архітектури), але для задач які вирішуються в реальному житті ця точність не критична, а останні дві моделі мають більший спектр застосування.

2.4 Функції активації у вибраних архітектурах

“Функція активації (активаційна функція, функція збудження) - функція, що обчислює вихідний сигнал штучного нейрона. Як аргумент приймає сигнал Y, одержуваний на виході вхідного суматора. Найбільш часто використовуються наступні функції активації [5]”.

2.4.1. Жорстка порогова функція активації

Ця функція являє собою просту кусочно-лінійну функцію. Якщо вхідне значення менше порогового, то значення функції активації дорівнює мінімальному допустимому, інакше - максимально допустимому. На рис. 2.10 зображено графік цієї функції.

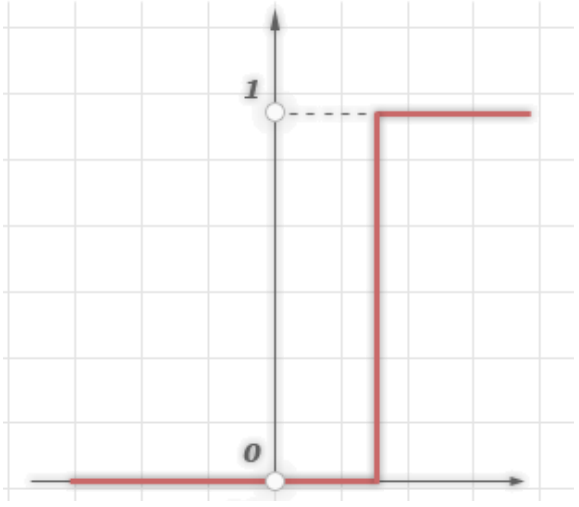


Рисунок 2.10 - Проста кусочно-лінійна функція [12]

2.4.2 Функція активації лінійний поріг

Ця функція являє собою нескладну кусочно-лінійнуа функцію. Має два лінійних ділянки, де функція активації тотожно дорівнює мінімально допустимому і максимально допустимого значення і є ділянка, на якому функція строго монотонно зростає. На рис. 2.11 зображено графік цієї функції.

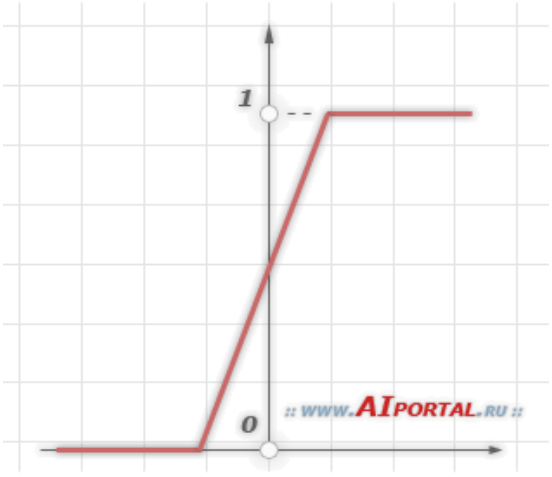


Рисунок 2.11 - Нескладна кусочно-лінійна функція [12]

2.4.3 Сигмовидна нелінійна функція

Сигмоїда - це гладка монотонна нелінійна функція, що має форму літери "S", яка часто застосовується для «згладжування» значень деякої величини. Математична формула - $\sigma(x) = 1 / (1 + e^{-x})$ і вона показана на рис. 2.12 (а). Вона приймає на вхід дійсне число і стискає його в діапазон від 0 до 1. Зокрема, великі негативні числа стають 0 і великі позитивні числа стають 1. Сигмовидна функція часто використовувалась протягом довгого часу, так як вона має хорошу інтерпретацію як збудження нейрона: від спокійного стану (0) до повністю насиченого збудження при максимальному значення (1). Зараз сигмовидна

нелінійна функція рідко використовується на практиці.

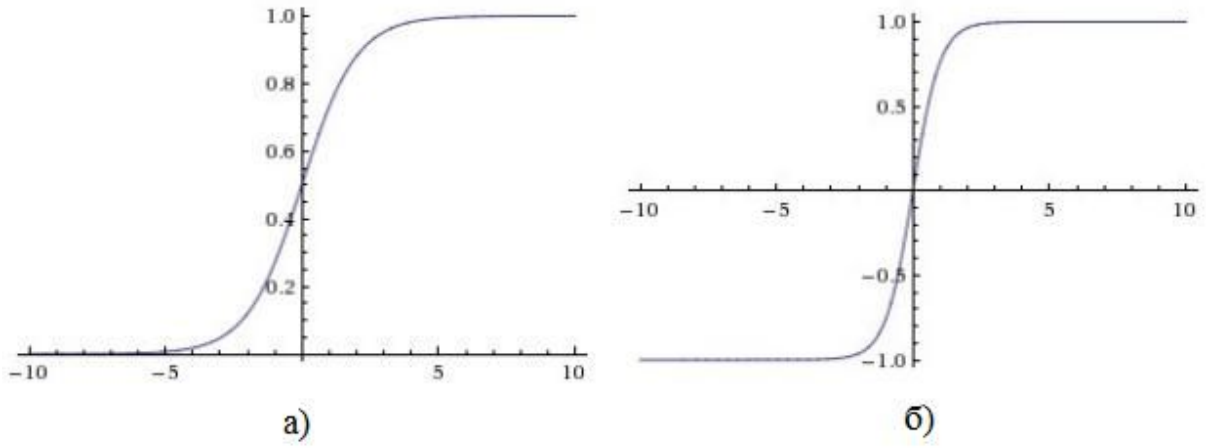


Рисунок 2.12 - Сигмовидна нелінійна функція (а) та тангенціальна функція (б) [9]

2.4.4 Тангенціальна функція активація

Функція, яка представляється гіперболічним тангенсом. На відміну від сигмоїдальної функції вихід тангенціальної дорівнює нулю в центрі. Тому на практиці TANH завжди краще сигмовидної нелінійності. На додаток зазначимо, що TANH функція просто масштабується сигмовидною, отже, можемо записати наступну рівність: $\tanh(x) = 2\sigma(2x) - 1$. Функція зображена на рис 2.12 б.

2.4.5 Функція активації прямокутний елемент

Активаційна функція, яка визначається:

$$F(X) = max(0, x), \quad (1)$$

де x - вхідний параметр [22].

Іншими словами, активація відбувається по нульовому порогу та зображена на рис. 2.13.

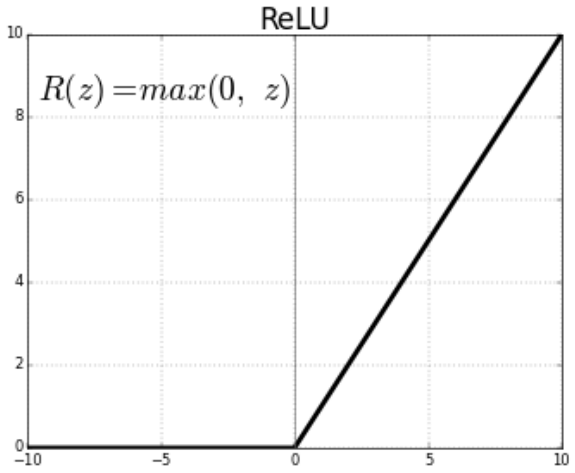


Рисунок 2.13 - Прямокутний лінійний елемент [16]

З’явилась відносно не давно, прийшла на зміну сигмоїді, має наступні переваги над попередниками:

- значне прискорення збіжності стохастичного градієнтного спуску в порівнянні з функціями TANH / сигмовидною.
- в порівнянні з функціями активації TANH / сигмовидною, які включають дорогі операції (експонент і т.д.), ReLU може бути реалізований за допомогою простої порогової матриці активацій в нулі.

Але є і недоліки, такі як: одиниці ReLU можуть бути крихкими під час тренування і можуть «померти». Наприклад, великий градієнт, що протікає через нейрон ReLU може привести ваги оновити таким чином, що нейрон ніколи не буде активувати на будь-якій точці даних знову. Якщо це станеться, то градієнт, що протікає через пристрій завжди буде дорівнює нулю з цього моменту. Тобто,

блоки ReLU можуть незворотньо померти під час тренування, так як вони можуть зіб'ють колектор даних. Наприклад, ви можете виявити, що майже 40% від вашої мережі може бути «мертвими» (тобто нейрони, які ніколи не активують по всьому навчальному набору даних), якщо швидкість навчання встановлена занадто високими. При правильному налаштуванні швидкості навчання це не є частою проблемою [12].

2.5 Вибір оптимізатора

Одним із найвідоміших варіантів навчання нейронної мережі є так званий алгоритм зворотного поширення помилки. Існують сучасні алгоритми другого порядку, такі як метод сполучених градієнтів і метод Левенберга-Маркара [8], які на багатьох завданнях працюють істотно швидше (іноді на порядок). Алгоритм зворотного поширення є досить простим для розуміння і має переваги у використанні. Також існують евристичні модифікації, які добре працюють для деяких класів задач, - швидке поширення (Fahlman, 1988) і Дельта-дельта з межею (Jacobs, 1988). В цьому алгоритмі обчислюється вектор градієнту поверхні помилок, який показує напрямок найкоротшого спуску по поверхні з даної точки, тому якщо просуватись по ньому, помилка буде зменшуватись і так до локального екстремуму.

Якщо спускатись великими кроками збіжність буде швидшою, але можна перескочити локальний мінімум, який буде кращим рішенням або піти в неправильному напрямку. На рис. 2.14 зображено пошук мінімума функції.

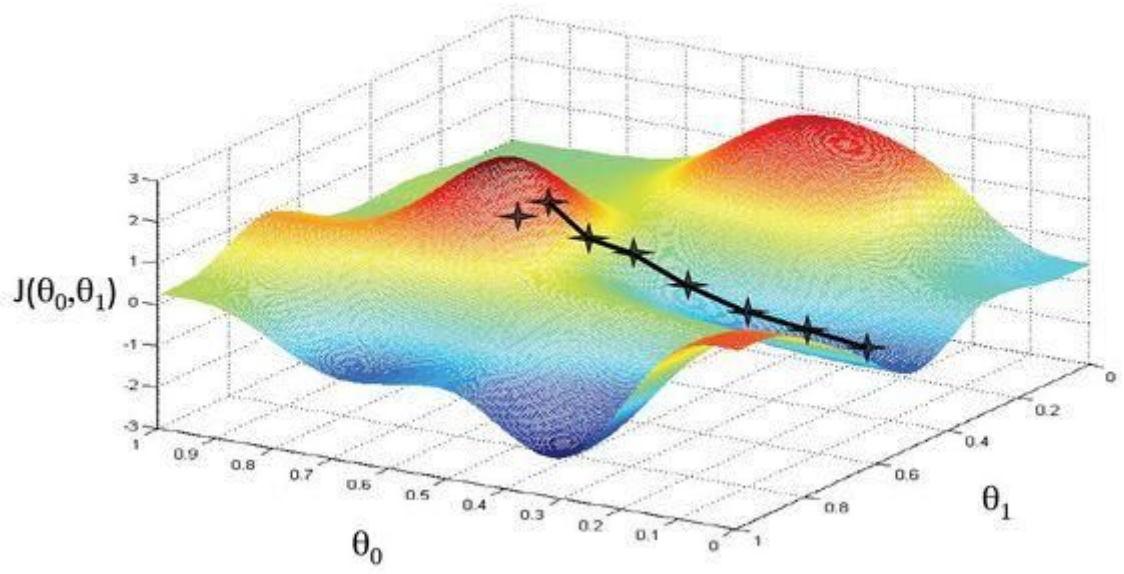


Рисунок 2.14 – Демонстрація роботи методу градієнтного спуску [16]

Зазвичай величину кроку вибирають пропорційну крутизні схилу з деякою константою, яка називається швидкістю навчання. Правильний вибір швидкості навчання залежить від конкретного завдання і зазвичай здійснюється досвідченим шляхом; ця константа може також залежати від часу, зменшуючись у міру просування алгоритму.

Отже, алгоритм працює ітеративно, а його кроки називаються епохами. Початкові ваги мережі зазвичай вибираються випадковим чином або наприклад з Гаусівського розподілу, процес навчання зупиняють після проходження заданої кількості епох, або тоді коли помилка коли помилка перестане зменшуватися деяку кількість епох.

Отже, враховуючи все вище зазначене, а саме проблеми з локальними мінімумами і вибором архітектури мережі, а відповідно і кількість навчальних

параметрів, призводять до того, що на практиці потрібно проводити різні експерименти з різними архітектурами мереж, підібрати гіперпараметри до декількох кращих, оптимізувати рішення і тд.

Розглянемо декілька оптимізаторів нейронних мереж. Почнемо з найпростішого, а саме градієнтного спуску.

Градієнтний спуск – це спосіб мінімізувати цільову функцію $C(\theta)$, де $\theta \in \mathbb{R}d$ − параметри моделі, шляхом оновлення параметрів у напрямі, протилежному градієнту цільової функції $\nabla\theta C(\theta)$ [3]. Параметр η – означає крок алгоритму, який виконується в напрямі (локального) мінімуму. Загалом, відбувається рух в напрямі схилу по поверхні цільової функції аж поки не буде досягнуто «долини».

На практиці використовують пакетний градієнтний спуск, так як він значно швидше рахується, відповідно модель швидше навчається. Ідея полягає в том, що ми рахуємо градієнт не на всіх даних, а пакетами, точність виходить трохи гірша, але якщо ми йдемо маленькими кроками і в нас багато таких пакеті ми отримуємо усереднення, яке дуже подібне до звичайного обчисленого градієнта, але набагато швидше. Є ще оптимізований варіант попереднього, а саме - стохастичний градієнтний спуск. Ідея якого полягає, а тому що для кожного екземляру ми лише раз будемо оновлювати параметри. Існує ще багато інших методів, але зараз ми на них детально зупинятись не будемо.

Отже, вибраний нами оптимізатор - стохастичний градієнтний спуск. Який для нашої задачі має якраз безліч переваг, які було вказано вище.

2.6 Дослідження різних метрик для оцінки та покращення моделей

Для задач машинного навчання існує дуже багато різних метрик, так як для кожної задачі є різні бізнес цілі та іноді буває не так важливо помилитись при класифікації в межах певного класу, натомість як помилка в іншому є дуже суттєва. Прикладом може бути класифікація зображень ракових клітин, якщо ми випадково здорову клітину позначимо як ракову, це звісно не приємно, але після обстеження людина швидко дізнається про помилку, натомість якщо ракову класифікувати як здорову, кінець може бути фатальним. Опишемо деякі з метрик, а тоді визначимо яка ж з них найкраще підходить для поставленої задачі.

2.6.1 Метрики precision, recall та F-міра

Автори робіт у сфері комп'ютерного зору використовують різні метрики для оцінки ефективності роботи програми. Є декілька основних метрик для інформаційно пошукових систем:

1) Точність (precision)

Визначається, як відношення кількості релевантних документів, знайдених у ІПС до кількості знайдених релевантних документів:

$$\text{Precision} = \frac{|D_{rel} \cap D_{retr}|}{|D_{retr}|}, \quad (2)$$

де D_{rel} – це множина релевантних документів у базі, а D_{retr} – це множина документів, знайдених системою.

2) Повнота (recall)

Визначається, як відношення кількості знайдених у ІПС релевантних

документів до загальної кількості релевантних документів у базі:

$$\text{Recall} = \frac{|D_{rel} \cap D_{retr}|}{|D_{rel}|}, \qquad (3)$$

де D_{rel} – це множина релевантних документів у базі, а D_{retr} – це множина документів, знайдених системою.

3)F-міра (F-measure)

Іноді буває корисно об'єднати точність та повноту у одній, усередненій величині. Для цієї цілі не підходить середнє арифметичне так як, пошуковій системі достатньо видати всі документи взагалі, щоб забезпечити повноту рівну одиниці при точності близькій до нуля. Тоді середнє арифметичне буде не менше 1/2. Середнє гармонійне не має цього недоліку, оскільки при великій різниці між усереднюваними значеннями наближається до найменшого з них.

Тому досить гарною мірою для сумісної оцінки точності та повноти є F-міра, яка визначається, як зважене середнє гармонійне точності P та повноти R:

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}}, \qquad \alpha \in [0,1]. \qquad (4)$$

Зазвичай F-міру записують у вигляді:

$$F = \frac{(\beta^2+1)PR}{\beta^2 P + R}, \qquad \beta^2 = \frac{(1-\alpha)}{\alpha}, \quad \beta^2 \in [0,\infty]. \qquad (5)$$

При $\alpha = 1 / 2$ або $\beta = 1$ F-міра надає однакову вагу точності та повноті та називається збалансованою F_1-мірою (в нижньому індексі прийнято вказувати величину β), вираз для неї скорочується:

$$F_1 = \frac{2PR}{P+R}. \quad (6)$$

Використання збалансованої F-міри не є обов'язковим: при 0<β<1 перевага надається точності, а при β>1 більша вага надається повноті.

При застосуванні цих метрик для оцінки ефективності алгоритмів трекінгу замість множини документів розглядалася множина пікселів.

2.6.2 Метрика точності

Точність моделі показує, скільки ми дали правильних відповідей, зі всіх. Ще можна сформулювати, що точність - зважене середнє арифметичне метрики precision і звотньої метрики precision, або зважене середнє recall та звортньої метрики recall. Плюси використання цієї метрики:

1. легко інтерпретується;
2. легко проваджується;
3. можна використовувати для багато класової класифікації, натомість як попередні метрики не можна, лише модифікувавши їх.

Але є і недоліки:

1. можна використовувати, коли класів майже порівну, або вирішити дизбалансу класів, так як якщо одного класу буде 99%, а іншого 1, модель може на всі елементи вказувати, що це перший клас і в даному випадку точність буде 99%. Хоча ми ж розуміємо, що одни клас ми просто не передбачуємо взагалі, що не завжди нам підходить;
2. не враховує, що передбачення деяких класів важливіші за інші, тобто її зручно використовувати, коли класи рівноцінні.

Отже, описавши деякі з метрик машинного навчання, ми бачимо що для нашої

задачі, а саме класифікація на 69 класів, де всі класи є рівноцінними. А також класи мають більш-менш однакову кількість екземплярів, а для тих що не мають буде застосовано метод “аугументації”, що означає, що ми видозмінимо існуючі зображення та додамо в набір даних. Можна зробити висновок, що для даної задачі чудово підходить метрика точності.

2.7 Опис тестових даних

У даній роботі використовувався відкритий набір даних [18], де було 69 класів по предметам побуту. Набір даних не збалансований. Проблему не збалансованості класів я вирішувала за допомогою методу аугументації(приклад зображено на рис. 2.15), що означає коли збільшують вхідний набір даних перетворюючи існуючі, наприклад, відзеркалення зображень, обрізання зображень, повернення на певний кут та інші модифікації зображень.

Рисунок 2.15 - Аугументація зображень [12]

Всього в наборі даних було 104461 зображень, для тесту використовувалось 3450 зображень, для навчання використовувалось - 101011. Відповідно на кожен клас було в середньому по 1400 екземлярів, найменший клас містив 719 зображень для тренування і найбільший - 2389. Для тесту використовувалось 50

картинок для кожного класу.

На рис. 2.16 зображено приклади зображень для навчання, на рис. 2.17 - для тесту. Як бачимо, якість фото хороша, картинки невеликого розміру, але всі різного. Також на деяких зображеннях є декілька різних об’єктів, одні картинки з фоном, інші - без.

Рисунок 2.16 - Приклади начального набору даних

Рисунок 2.17 - Приклади тестувального набору даних

Отже, як бачимо деякі класи легко класифікувати наочно, а деякі навіть людині не дуже. Хоча зображень, на яких було не дуже зрозуміло, який саме об’єкт був зображений відносно не так багато і через це моделі давали хорошу точність.

Для того, щоб використовувати ці дані для навчання нейронних мереж зображення всі були приведені до однакових розмірів, далі було перетворено їх у матриці чисел і ці матриці віднормовано. Також для кожного класу використовувалось 2389 зображень, для класів в яких було менше використовувався вище описаний метод, таким чином, що обиралось випадкове число зображень і до кожного здійснювались випадкові модифікації. Таким чином мережа не могла перенавчитись під якийсь певний клас чи просто вивчити зображення.

2.8 Процес оцінки алгоритмів

Для оцінки якості узагальнюючої можливості моделей машинного навчання використовують 2 загальних методи. Перший полягає в тому, що ми ділимо всі дані на три частини: тренувальну, валідаційну та тестувальну вибірки, як зображено на рис. 2.18. На навчальній та валідаційній вибірках моделі навчаються, відбирається краща модель та виконується підбір гіперпараметрів алгоритм, а на тестовій зазвичай просто перевіряють якісь отриманої моделі. Звісно потрібно враховувати, що природа всіх трьох вибірок має бути однакова, також вони не мають перетинатись, тоді на тестовій вибірці ми зможемо оцінити як буде працювати наша модель на нових даних, яких ще не бачила.

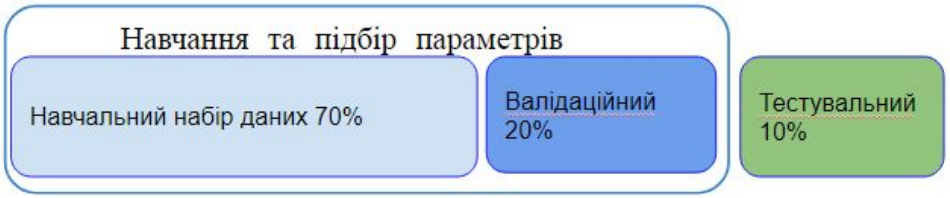


Рисунок 2.18 - Метод відкладеної перевірки

Використовують цей метод якщо багато даних і перехресну перевірку робити довго, а також, за рахунок, що даних багато, то і якість моделей має бути репрезентативною.

Другий метод - перехресна перевірка. Метою даного методу є визначення набору даних для оцінки моделі на етапі навчання, щоб усунути чи зменшити вплив перенавчання, а також дати уявлення про те, як модель буде узагальнюватись до незалежного набору даних (набору даних, якого не бачила модель).

Різновиди перехресної перевірки:

1. Вичерпна
    a. Залишок N зовні
    b. Залишок 1 зовні
2. Невичерпна
    a. K-кратна
    b. 2-кратна [10]

Опишемо, як працює K-кратна перехресна перевірка. На рис. 2.19 зображено як працює цей метод. Отже, початкову вибірку випадковим чином розбиють на K підвиборок однакового розміру. І з цих K підвиборок, вибирається одна в якості тестувального набору даних, а всі інші підвибірки використовуються в якості навчального набору даних. Цей алгоритм перехресної перевірки повторюється K раз, де кожен раз береться наступна підвибірка. В кінці отримані K оцінок

усереднюються, хоча можна використовувати й інший варіант узагальнення оцінок.

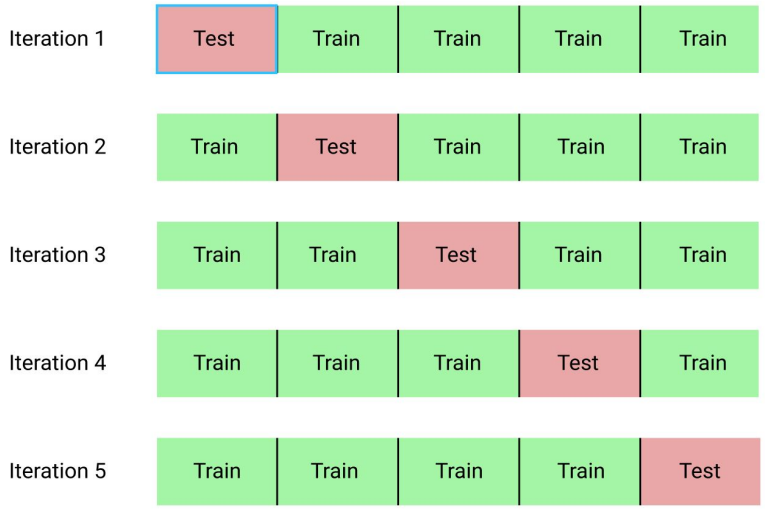


Рисунок 2.19 - Алгоритм роботи 5-кратної перевірки [19]

Перевагами цього підходу є те, що ми отримаємо точнішу якість нашої моделі, так як ми використовуємо всі спостереження як для тренування, так і для перевірки. Недоліками ж цього підходу, є те що перевірка таким чином забирає більше часу, так як нам потрібно навчити K моделей, а отже, по часу це довше в K разів.

Для моєї задачі, було використано перший метод - метод відкладеної перевірки, так як даних достатньо, щоб отримати якісну оцінку моделей, а також навчати K моделей було б дуже затратно по часу.

2.9 Опис програмної реалізації

Для програмної реалізації описаних методів та алгоритмів було обрано мову програмування Python версії 3.3. Python підтримує кілька парадигм програмування, в тому числі об'єктно-орієнтованого, імперативного та

функціонального програмування або процедурних стилів. Вона має динамічну систему типів і автоматичне керування пам'яттю, а також має велику кількість бібліотек різного призначення[20]. Його філософія дизайну підкреслює читаність коду, а його синтаксис дозволяє програмістам висловлювати поняття в меншій кількості рядків коду, ніж це можливо в таких мовах, як C ++ або Java. Python інтерпретатори доступні для багатьох операційних систем, дозволяючи запускати код написаний на Python на найрізноманітніших системах. За допомогою сторонніх інструментів Python код може бути упакований в автономні виконувані програми для деяких з найбільш популярних операційних систем, так що програмне забезпечення Python може бути поширеним і використовуватись у різних середовищах, навіть там де інтерпретатор Python не встановлений[16]. Замість того, щоб вимагати всіх функцій у ядрі мови, Python був розроблений, щоб бути максимально розширюваним.

В роботі було використано фреймворк Keras - відкрита нейромережева бібліотека, написана на мові Python. Вона являє собою надбудову над фреймворками Deeplearning4j, TensorFlow і Theano [4]. Націлена на оперативну роботу з мережами глибинного навчання, при цьому спроектована так, щоб бути компактною, модульної та розширюється. Загалом в цьому фреймворку є реалізація деяких архітектур, з тих що використовувались в роботі, також є можливість імплементувати свої моделі, також є основні активаційні функції, оптимізатори, функції втрат і тд. Лістинг до основних моментів в реалізованої програми, відображений у додатку А.

2.9 Висновок

У даному розділі було описано вибір методів, архітектури, функцій активацій, оптимізатора, метрик та метод оцінки якості моделей. Також було

проаналізовано набір даних, на яких проводилось навчання, описано як вирішувались проблеми з даними і описано, що було використано для програмної реалізації.

Враховуючи завдання, яке поставлене в роботі, а саме - класифікація зображень за допомогою нейронних мереж, на прикладі класифікації товарів побуту на 69 класів, було вибрано згорткові нейронні мережі. Для порівняння було вибрано 4 архітектури: InceptionV3, DenseNet, SquzeeNet та MobileNet. В цих архітектурах переважно використовувались активаційні функції, такі як Relu та функцію нормованих експоненційних втрат. Для оцінки узагальнючої здатності моделей використовувався метод відкладеної вибірки. Метрикою. оцінки якостей моделей було вибрано - точність, а проблему незбалансованості класів було вирішено за допомогою аугументації даних, в результаті чого для навчання моделей використовувалось 2386 зображень на один клас.

3. ОГЛЯД РЕЗУЛЬТАТІВ ТЕСТУВАННЯ РОЗРОБЛЕНИХ МОДЕЛЕЙ

3.1 Порівняння результатів вибраних моделей

У розділі 2 було вибрано 4 архітектури для порівняння результатів, а саме DenseNet, InceptionV3, SquzeeNet та MobileNet. Як було зазначено вище, дві перших ахітектури об’ємніші з більшою кількістю параметрів, навчаються довше, але дають кращу точність зазвичай, як зазначається [5]. А дві останніх менші за розмірами, а відповідно і навчаються швидше, можна використовувати на пристроях таких як смартфони.

У таблиці 3.1 наведено результати використаних архітектур. Варто зазначити, що всі архітектри були натреновані на ImageNet наборі даних, а в кінці кожної архітектури додано два повнозв’язаних шари, які донавчались на нашому наборі даних (предметів побуту).

Таблиця 3.1 - Результати оцінки архітектур на тестовому наборі даних

Модель	Точність, %			Кількість параметрів
	Без аугументації, з вагами	З аугументацією і з вагами	З аугументацією	
DenseNet	67.5	**80.32**	73.5	7 216 256
InceptionV3	64.9	79.53	69.62	11 307 840
SqeezeNet	62.4	72.69	65.4	853 824
MobileNet	61.9	71.05	65.5	659 362

Як бачимо, найкращі результати дає архітектура DenseNet, якщо робити аугументацію даних 20% і використовувати ваги для класів, щоб класи були збалансовані і чинили однаковий вплив на метрику. Також бачимо, що моделі SqeezeNet та MobileNet дають трохи гіршу якість, але при тому, що у них набагато менше кількість параметрів. На рис. 3.1 бачимо гістограму отриманих результатів.

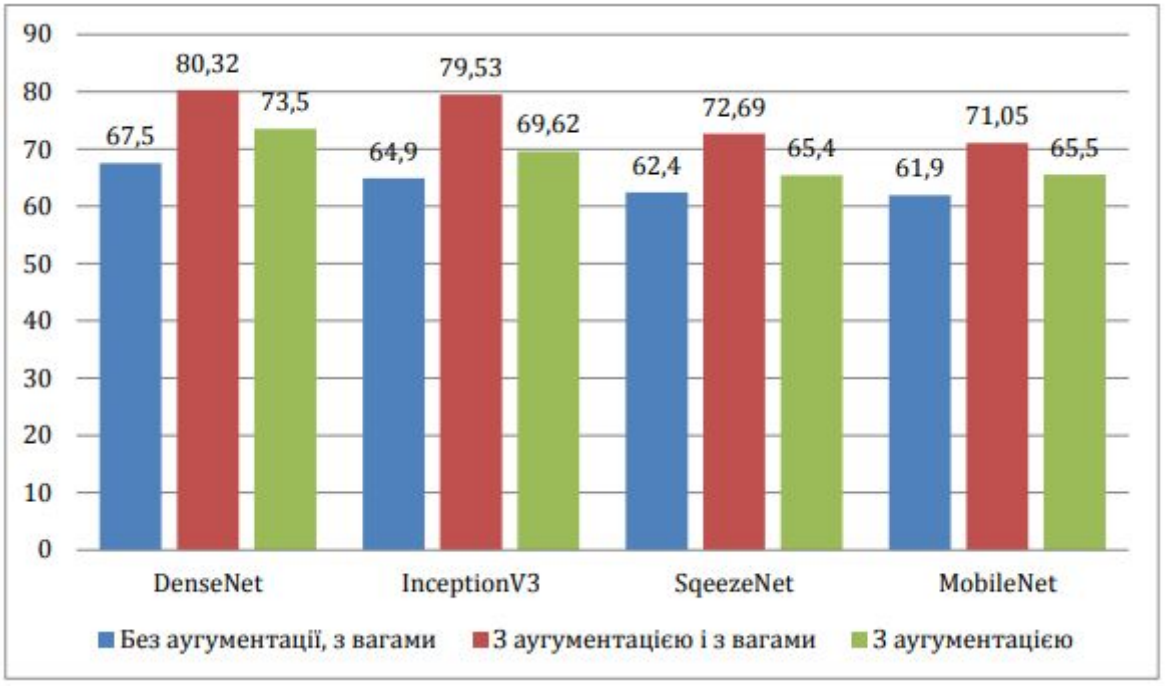


Рисунок 3.1 - Діаграма оцінки результатів моделей

Результати першої колонки (“Без аугументації, з вагами”) можна використовувати як точку опори, так як вони моделі навчались зі стандартними параметрами, оптимізаторами і тд. Всі моделі навчались до моменту, коли за останні 5 епох не було покращень результатів.

3.2 Метод підбору гіперпараметрів

Для того, щоб вибрані моделі машинного навчання давали кращу точність потрібно підібрати для них оптимальні гіерпараметри - фактори, які впливають на безпосередній процес навчання. Варто зазначити, що підбір гіперпараметрів впливає саме на процес навчання, тобто контролює поведінку моделей при навчанні, що є окремим завданням оптимізації. На рис. 3.2 зображено різницю між підбором гіперпараметрів та начанням моделі.

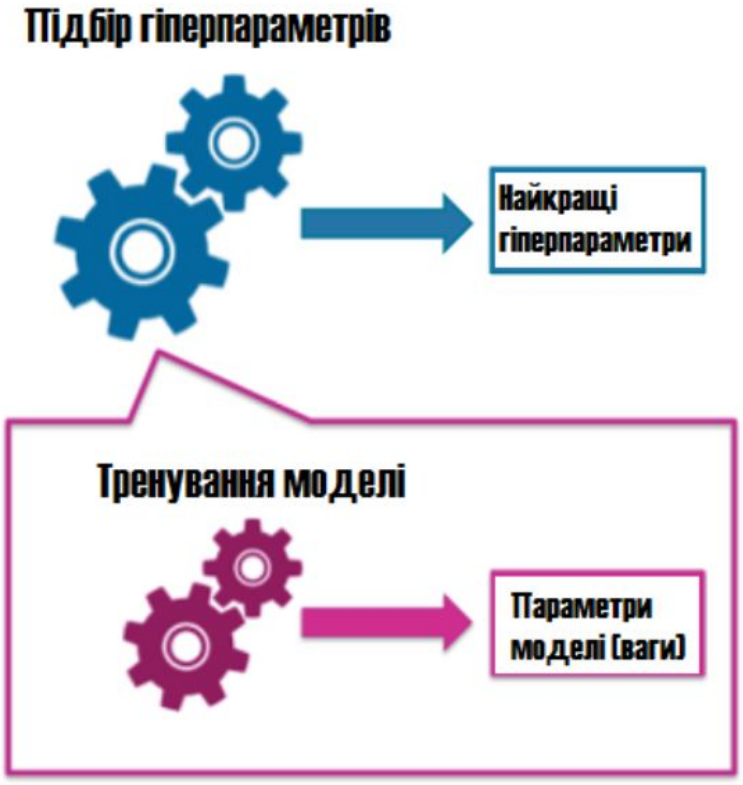


Рисунок 3.2 - Підбір гіперпараметрів та тренуванням моделі [8]

Є декілька відомих способів підбору параметрів:

1. сітковий пошук;

2. випадковий пошук.

Звісно існують ще й інші методи, якщо задачі досить специфічні, але зазвичай користуються вище зазначеними. На рис. 3.3 зображено сітковий та випадковий пошуки параметрів.

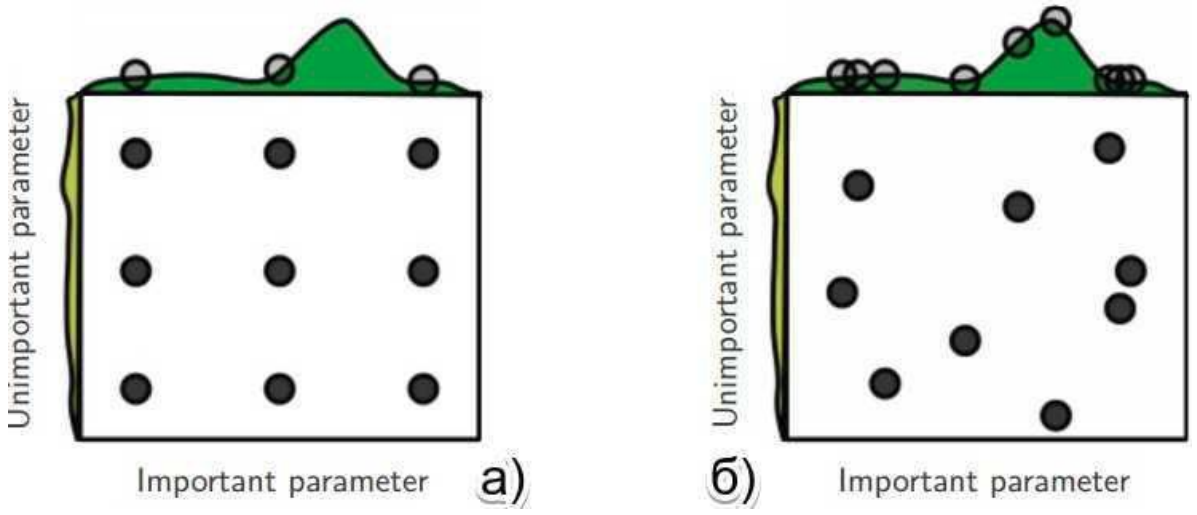


Рисунок 3.3 - Підбір гіперпараметрів (а - сітковий, б - випадковий)

Як бачимо з малюнка, сітковий пошук заключається в простому вичерпному пошуці через певну підмножину простору гіперпараметрів. Оскільки простір пошуку зазвичай містить дійсні значення, то його потрібно дискретизувати перед початком підбору [19]. Натомість як випадковим пошуком ми попадаємо у випадкові параметри і перевіряємо оцінку моделі, якщо вони нас не задовільняють вибираємо іншу випадкову точку, але в протилежному напрямку. Зазвичай якщо використовувати другий метод він швидший, але потірбно взяти достатню кількість точок, щоб максимально приблизитись до кращого результату. Натомість як сітковий пошук дає змогу перебрати всі необхідні параметри і підібрати такі, які вцілому дають найкращий результат.

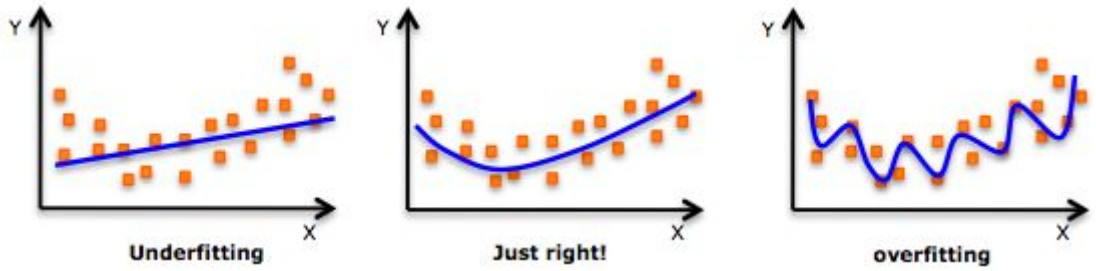
Отже, в даній роботі було вибрано сітковий пошук для підбору гіперпараметрів. Зрозуміло, що простір гіперпараметрів для різних алгоритмів машинного навчання відрізняється. Параметри для нейронних мереж буде описано далі.

3.2.1 Налаштування гіперпараметрів та базова модель

Зазвичай процес розробки нейронної мережі починається з розробки будь-якої простої мережі, або безпосередньо застосовуючи ті архітектури, які вже успішно застосовувалися для вирішення подібних завдань, або використовуючи ті гіперпараметри, які раніше вже давали непогані результати. В кінцевому підсумку досягається такого рівня продуктивності, який послужить гарною відправною точкою, після якої ми можемо спробувати змінювати усі зафіксовані параметри і витягти з мережі максимальну продуктивність. Цей процес зазвичай називають налаштуванням гіперпараметров, тому що він включає зміну компонентів мережі, які повинні бути встановлені до початку навчання.

Найчастіше змінюють такі параметри: швидкість навчання, розмір батчу, оптимізатор. Також можна частково змінювати саму архітектуру додаючи регуляризію, для того щоб мережа не перенавчалась.

Перенавчання - це дуже точна відповідність нейронної мережі конкретному набору навчальних прикладів, при якому мережу втрачає здатність до узагальнення. Іншими словами, наша модель могла вивчити навчальну множину (разом з шумом, який в ньому присутній), але вона не змогла розпізнати приховані процеси, які це безліч породили. Як приклад розглянемо задачу апроксимації синусоїди з адитивним шумом (рис. 3.4).



а) б) в)

Рисунок 3.4 - Недонавчання (а), нормальне навчання (б), перенавчання (в) [10]

У глибоких згортальних нейронних мереж маса різноманітних параметрів, особливо це стосується повнозв'язних шарів. Перенавчання може проявити себе в такій формі: якщо у нас недостатньо навчальних прикладів, маленька група нейронів може стати відповідальною за більшість обчислень, а решта нейрони стануть надлишкові; або навпаки, деякі нейрони можуть нанести шкоду продуктивності, при цьому інші нейрони з їх шару не будуть займатися нічим, окрім виправлення їх помилок.

Щоб допомогти нашій мережі не втратити здатності до узагальнення в цих обставинах, ми вводимо прийоми регуляризації: замість скорочення кількості параметрів, ми накладаємо обмеження на параметри моделі під час навчання, не дозволяючи нейронам вивчати шум навчальних даних.

Регуляризація - метод додавання певної додаткової інформації до умови з метою вирішення некоректно поставленої задачі або запобігання перенавчанню. Для нейронних мереж використовують таку регуляризацію: нормалізація по батчам, викидання частини нейронів, максимальне об’єднання.

Зокрема, викидання частини нейронів з параметром p за одну ітерацію навчання проходить по всіх нейронах певного шару і з ймовірністю p повністю

виключає їх з мережі на час ітерації. Це змусить мережу обробляти помилки і не покладатися на існування певного нейрона (або групи нейронів), а покладатися на "єдину думку" нейронів всередині одного шару. Це досить простий метод, який ефективно бореться з проблемою перенавчання сам, без необхідності вводити інші регуляризатора. На рис. 3.5 зображена схема нижче ілюструє даний метод.

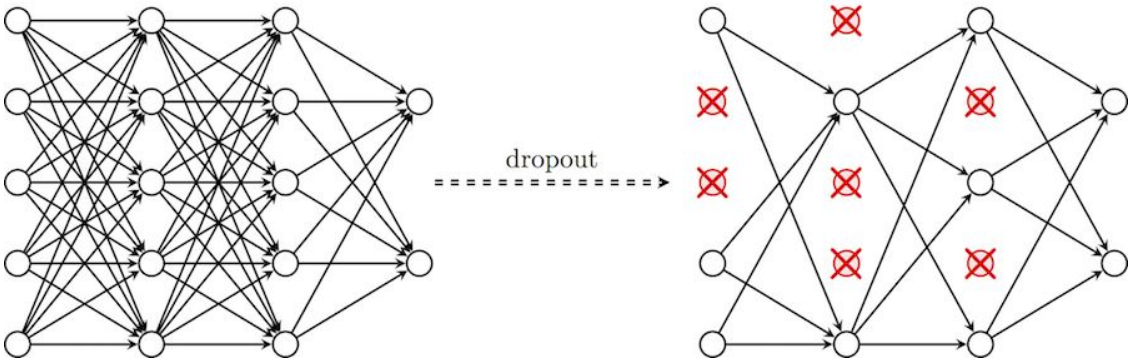


Рисунок 3.5 - Процес викидання частини нейронів [8]

Максимальне об’єднання - зменшення зображення, таким чином, що вибирається максимальне значення із сусідніх (на рис. 3.6 застосовується ядро із розмірами 2x2, тому вибирається максимум із 4 значень).

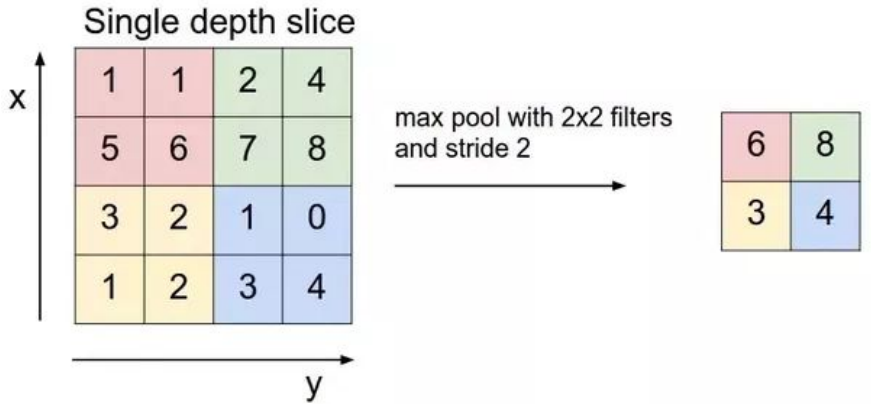


Рисунок 3.6 - Алгоритм максимального об’єднання [15]

Отже, було розглянуто основноі способи поращення якості роботи моделей, в наступному розділі застосуємо деякі з них та перевіримо як зміняться оцінки наших моделей.

3.3 Порівняння налаштованих моделей

Було проведено сітковий пошук параметрів в результаті якого, бачимо, що зменшення швидкості навчання дало кращу точність для всіх моделей. Також бачимо, що оптимізатор Адам дав кращу точність для MobileNet та DenseNet, а інші моделі дали кращі результати при використанні оптимізатора стохастичного градієнтного спуску. Також було помічено, що збільшення розміру батча з 2 до 32 дало краще точність для всіх моделей. І при використанні картинок більшого розміру 64x64 дало найкращі результати для SqeezeNet та MobileNet, в той час як 128x128 для DenseNet та InceptionV3.

У таблиці 3.2 наведено які параметри підбирались та у таблиці 3.3 наведено як змінювалась точність при зміні параметрів. Порівняння оцінки моделей до та після підбору параметрів наведено в таблиці 3.4. Загалом як бачимо, після підбору гіперпараметрів вдалось подолати бар’єр 80% і навіть отримати 88.7%, що є досить хорошим результатом, враховуючи особливості набору даних які використовувались для завдання.

Таблиця 3.2 - Простір гіперпараметрів

Модель	Швидкість навчання	Розмір батчу	Розмір зображень	Оптимізатор	Регуляризація
DenseNet	0.1; 0.01; 0.001; 0.0001;	2;8;16;32	64x64; 128x128; 256x256	Адам, стохастичний градієнтний спуск	максимальне об’єднання, з ядром 2; викидання нейронів (25%, 50%)
InceptionV3					
SqeezeNet					
MobileNet					

Таблиця 3.3 - Оптимальні значення гіперпараметрів

Модель	Швидкість навчання	Розмір батчу	Розмір зображень	Оптимізатор	Регуляризація
DenseNet	0.0001;	32	128x128;	Адам	викидання нейронів (25%)
InceptionV3	0.001	32	128x128;	стохастичний градієнтний спуск	викидання нейронів (50%)
SqeezeNet	0.001	32	64x64;	стохастичний градієнтний спуск	-
MobileNet	0.0001	32	64x64;	Адам	-

Таблиця 3.4 - Вплив підбору гіперпараметрів

Модель	До підбору			Після підбору		
	Без аугумен-тації, з вагами	З аугумен-тацією і з вагами	З аугумента-цією	Без аугумента-ції, з вагами	З аугумента-цією і з вагами	З аугумента-цією
DenseNet	67.5	**80.32**	73.5	71.35	**88.7**	80.5
InceptionV3	64.9	79.53	69.62	69.9	86.9	79.9
SqeezeNet	62.4	72.69	65.4	67.4	79.8	74.6
MobileNet	61.9	71.05	65.5	66.9	77.5	70.3

Можна було б ще спробувати ансаблі з усіх цих архітектур і отримати ще кращу точність на 2-3%, але на жаль в мене не було достатніх обчислювальних потужностей. Також на рис. 3.7-3.10 зображено діаграмb оцінки результатів для кожної моделі.

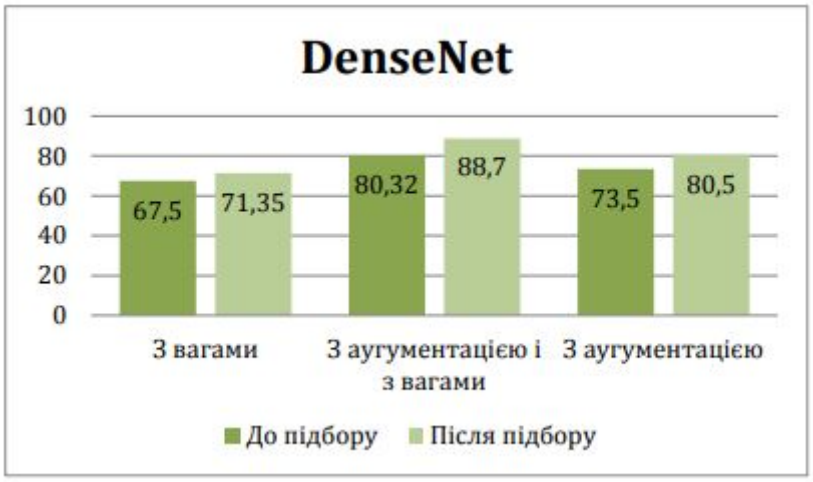


Рисунок 3.7 - Діаграма оцінки результатів DenseNet

Рисунок 3.8 - Діаграма оцінки результатів InceptionV3

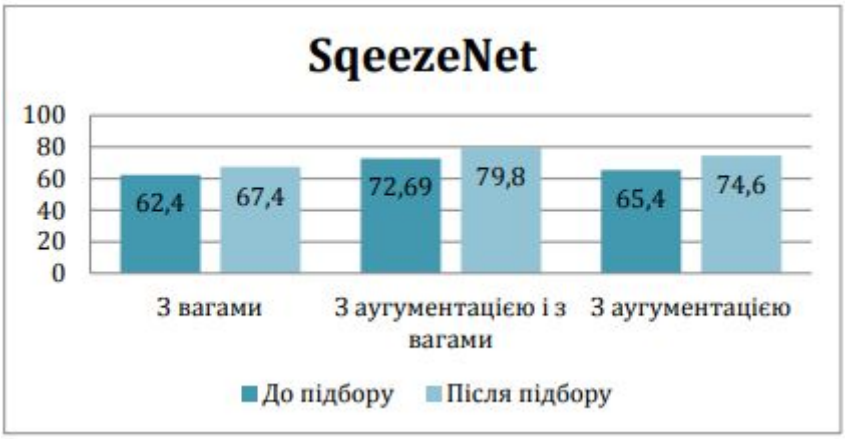


Рисунок 3.9 - Діаграма оцінки результатів SqeezeNet

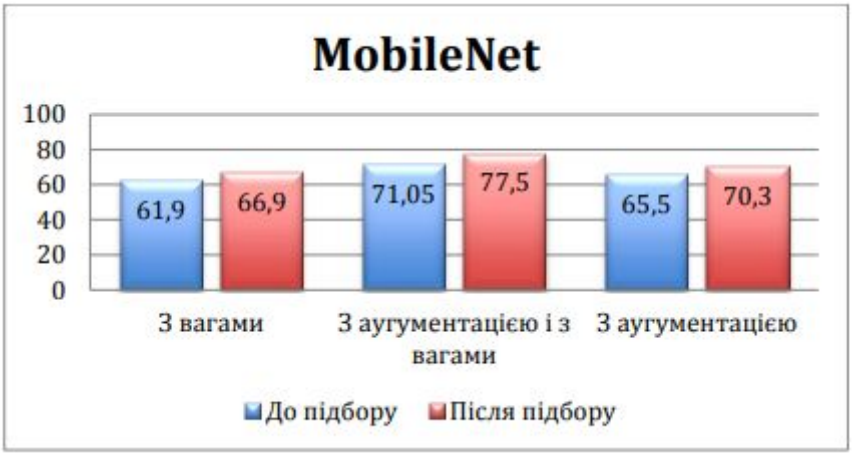


Рисунок 3.10 - Діаграма оцінки результатів MobileNet

Отже, як бачимо з результатів DenseNe та InceptionV3 дали кращу точність в порівнянні із MobileNet та SqeezeNet, але розрив у всіх моделей не такий великий і для реальних сиситем іноді жертвують точністю задля використання менших моделей. Вцілому, найкращі результати дала модель DenseNet - 88.7%, далі за нею InceptionV3 - 86.9%, SqeezeNet - 79.8% і найгірші результати були отримані MobileNet - 77.5%. Хоча якщо враховувати, що задача була класифікувати 69

класів, то навіть найгірша модель дала досить не погані результати.

3.5 Висновок

Отже, було вибрано моделі, протестовано різні підходи роботи з даними, а також підібрано гіперпараметри. Як бачимо, якщо використовувати більше даних за допомогою аугументації, то ми отримуємо більшу точність на 10-15%, також після підбору гіперпараметрів точність моделей була збільшена на 6-8%. Також найкращою моделлю виявилась модель із використанням DenseNet архітектур, яка має меншу кількість парметрів від InceptionV3, {оча і більше на відміну від SqeezeNet чи MobileNet. Архітектури MobileNet та SqeezeNet дали теж непогані результати, якщо враховувати показник точність на кількість навчальних параметрів. В подальшому можна було б спробувати різні методи, які дозволяють передати знання великих мереж меншим, таким чином можна використовувати невелику нейронну мережу SqeezeNet і отримувати точність близьку до більшої (в нашому випадку DenseNet).

Для підбору використовувався сітковий метод пошуку гіперпараметрів, який надійним і допомагає знайти потрібні найкращі гіперпараметри. Також з параметрів які підбирались, набільш ефективними виявились такі як: швидкість навчання та розмір батчу. Для всіх мереж ці показники дали найбільший приріст точності моделі.

4. РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ “КЛАСИФІКАЦІЯ ЗОБРАЖЕНЬ”

4.1 Опис ідеї проекту

У даному розділі описано економічне обґрунтування реалізації стартап-проекту на тему “Класифікація зображень”.

Метою розділу є формування інноваційного мислення, підприємницького духу та формування здатностей щодо оцінювання ринкових перспектив і можливостей комерціалізації основних науково-технічних розробок, сформованих у попередній частині магістерської дисертації у вигляді розроблення концепції стартап-проекту “ Класифікація зображень” в умовах висококонкурентної ринкової економіки глобалізаційних процесів.

Опис стартап-проекту “Класифікація зображень” наведено у Таблиці 4.1.

Таблиця 4.1 - Опис ідеї стартап-проекту

Зміст ідеї	*Напрямки застосування*	*Вигоди для користувача*
Ідея полягає в тому, щоб створити сервіс, який можна було б використовувати для категоризації товарів предмету побуту	1. Для працівників на складі	При поступанні нових товарів, процес категоризації буде автоматичним, так як працівнику потрібно лише зробити фото товару
	2. При розміщенні товарів на складі	При завантаженні товару на сайт, категорія буде визначатись автоматично

Отже, проект може бути використаним як для працівників складу так і для завантаження нових товарів на сайт для автоматичного визначення категорій на сайті.

Таблиця 4.2 - Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	*Техніко-економічні характеристики ідеї*	*(Потенційні) товари/концепції конкурентів*				*W (слабка сторона)*	*N (нейтральна сторона)*	*S (сильна сторона)*
		Мій проект	*Конкурент1*	*Конкурент2*	*Конкурент3*			
1.	Форма виконання	Сер-віс	До- даток	Веб-сервіс	Веб-сервіс			+
2.	Собівартість	Ни-зька	Висо-ка	Сере-дня	Сере-дня			+
3.	Наявність адміністратора для налаштуван-ня	Не треба	Не треба	Пот-рібно	Потрібно			+
4.	Наявність інтернету	Не треба	Треба	Треба	Не треба		+	
5.	Крос-платформеність	Ні	Так	Ні	Ні	+		

Визначений перелік слабких, сильних та нейтральних характеристик та властивостей ідеї потенційного товару є підґрунтям для формування його конкурентоспроможності.

Сильними сторонами є форма виконання, собівартість та наявність дміністратора для налаштування, а слабкою – крос-платформеність.

4.2 Технологічний аудит ідеї проекту

В межах даного підрозділу необхідно провести аудит технології, за допомогою якої можна реалізувати ідею проекту (технології створення товару).

Таблиця 4.3 - Технологічна здійсненність ідеї проекту

№ п/п	*Ідея проекту*	*Технології її реалізації*	*Наявність технології*	*Доступність технології*
1.	Створення сервісу	Python, Flask	Наявна	Безкоштовна, доступна
		Flash	Наявна	Платна, доступна
Обрана технологія реалізації ідеї проекту: для створення сервісу обрана технологія Python (Flask), яка є безкоштовною та доступною.				

4.3 Аналіз ринкових можливостей

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням

стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

Спочатку проводимо аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку.

Таблиця 4.4 - Попередня характеристика потенційного ринку стартап-проекту

№ п/п	*Показники стану ринку (найменування)*	*Характеристика*
1.	Кількість головних гравців, од	3
2.	Загальний обсяг продаж, грн/ум.од	20000 грн./ум.од
3.	Динаміка ринку (якісна оцінка)	Зростає/спадає/стагнує
4.	Наявність обмежень для входу (вказати характер обмежень)	Немає
5.	Специфічні вимоги до стандартизації та сертифікації	Немає
6.	Середня норма рентабельності в галузі (або по ринку), %	R = (3000000 * 100) / (1000000 * 12) = 25%

Отже, було проаналізовано наявність попиту, обсяг, динаміку розвитку ринку. Обмеження для входу на ринок відсутні, динаміка ринку зростає, галузь є рентабельною.

Надалі визначаються потенційні групи клієнтів, їх характеристики, та формується орієнтовний перелік вимог до товару для кожної групи (табл. 4.5).

Таблиця 4.5 - Характеристика потенційних клієнтів стартап-проекту

№ п/п	*Потреба, що формує ринок*	*Цільова аудиторія (цільові сегменти ринку)*	*Відмінності у поведінці різних потенційних цільових груп клієнтів*	*Вимоги споживачів до товару*
1.	Сервіс для автоматичної категоризації товарів	Будь-які підприємства, які займаються виробництвом/розповсюдженням побутових предметів	Цільова група працівники на підприємствах, відмінностей між групами не має, всі можуть використовувати сервіс	Рішення повинне бути зручним у користуванні, надійним, швидким

Згідно проведеної характеристики потенційних клієнтів стартап-проекту випливає, що на ринку є затребуваним програмне забезпечення, а саме сервіс для автоматичної категоризації товарів.

Після визначення потенційних груп клієнтів проводиться аналіз ринкового середовища: складаються таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають (табл. № 6-7). Фактори в таблиці подавати в порядку зменшення значущості.

Таблиця 4.6 - Фактори загроз

№ п/п	*Фактор*	*Зміст загрози*	*Можлива реакція компанії*
1.	Конкуренція	Вихід на ринок великої компанії	• вихід з ринку; • запропонувати великій компанії поглинути себе; • передбачити додаткові переваги власного сервісу для того, щоб повідомити про них саме після виходу міжнародної компанії на ринок
2.	Зміна потреб користувачів	Користувачам необхідний сервіс з іншим функціоналом	Передбачити можливість додавання нового функціоналу до створюваного додатку

Отже, було проаналізовано фактори загроз ринкового впровадження проекту, серед яких: конкуренція, уповільнення росту ринку, зміна потреб користувачів, зміна тарифів провайдера хмарного розгортання на платні та надходження на ринок альтернативних продуктів. Було також запропоновано можливі реакції компанії.

Таблиця 4.7 - Фактори можливостей

№ п/п	*Фактор*	*Зміст можливості*	*Можлива реакція компанії*
1.	Зростання можливостей потенційних покупців	Зростання фінансування у підприємств, які займаються предметами побуту	Запропонувати їм свої послуги
2.	Зниження довіри до конкурента 1	У додатку конкурента 1 нещодавно стався збій і протягом тижня система не правильно функціонувала	При виході на ринок звертати увагу покупців на надійність нашого сервісу

У таблиці 4.7 наведено фактори можливостей ринкового впровадження проекту, серед яких: зростання можливостей потенційних покупців та зниження довіри до конкурента.

Надалі проводиться аналіз пропозиції: визначаються загальні риси конкуренції на ринку.

Таблиця 4.8 - Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	*В чому проявляється дана характеристика*	*Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)*
1. Вказати тип конкуренції: - досконала	Існує 3 фірми-конкуренти на ринку	Врахувати ціни конкурентних компаній на початкових етапах створення бізнесу,

		реклама (вказати на конкретні переваги перед конкурентами)
2. За рівнем конкурентної боротьби: - міжнародний	Одна з компаній – з іншої країни, дві – з України	Додати можливість вибору мови ПЗ, щоб легше було у майбутньому вийти на міжнародний ринок
3. За галузевою ознакою: - внутрішньогалузева	Конкуренти мають ПЗ, який використовується лише всередині даної галузі	Створити основу ПЗ таким чином, щоб можна було легко його переробити для використання у інших галузях
4. Конкуренція за видами товарів: - товарно-видова	Види товарів є однаковими, а саме - програмне забезпечення	Створити ПЗ, враховуючи недоліки конкурентів
5. За характером конкурентних переваг: - нецінова	Вдосконалення технології створення ПЗ, щоб собівартість була нижчою	Використання менш дорогих технологій для розробки, ніж використовують конкуренти
6. За інтенсивністю: - не марочна	Бренди відсутні	-

У таблиці 4.8 наведено ступеневий аналіз конкуренції на ринку, де було визначено особливості конкурентного середовища та їх вплив а діяльність підприємства. Однією з найбільш важливих дій компанії для досягнення конкурентоспроможності є необхідність створити основу ПЗ таким чином, щоб можна було легко переробити дане ПЗ для використання у інших галузях

Після аналізу конкуренції проводиться більш детальний аналіз умов

конкуренції в галузі (табл. 4. 9).

Таблиця 4.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	*Прямі конкуренти в галузі*	*Потенційні конкуренти*	*Постачальники*	*Клієнти*	*Товари-замінники*
	Навести перелік прямих конкурентів	*Визначити бар'єри входження в ринок*	*Визначити фактори сили постачальників*	*Визначити фактори сили споживачів*	*Фактори загроз з боку замінників*
Висновки	Існує 3 конкуренти на ринку. Найбільш схожим за виконанням є конкурент 2, так як його рішення також представлене у вигляді Веб сервісу.	Так, можливості для входу на ринок є, бо наше рішення спрощує та пришвидшує роботу спеціаліста.	Постачальники відсутні	Важливим для користувача є зручність у користуванні	Товари-замінники можуть використат и більш дешеву технологію створення ПЗ та зменшити собівартість товару

На основі аналізу конкуренції із урахуванням характеристик ідеї проекту, вимог споживачів до товару та факторів маркетингового середовища визначається та обґрунтовується перелік факторів конкурентоспроможності (табл. 4.10).

За результатами аналізу таблиці робиться висновок щодо принципової можливості роботи на ринку з огляду на конкурентну ситуацію. Також робиться висновок щодо характеристик (сильних сторін), які повинен мати проект, щоб

бути конкурентоспроможним на ринку. Другий висновок враховується при формулюванні переліку факторів конкурентоспромож- ності у п. 3.6. 3.6) На основі аналізу конкуренції, проведеного в п. 3.5 (табл. 4.9), а та- кож із урахуванням характеристик ідеї проекту (табл. 4.2), вимог споживачів до товару (табл. 5) та факторів маркетингового середовища (табл 4.6-4.7) визначається та обґрунтовується перелік факторів конкурентоспроможності. Аналіз оформлюється за табл. 10

Таблиця 4.10 - Обґрунтування факторів конкурентоспроможності

№ п/п	*Фактор конкурентноспроможності*	*Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)*
1.	Використання ПЗ у вигляді веб-сервісу	Дозволяє наочно побачити роботу ПЗ і правильність роботи.
2.	Простота інтерфейсу користувача	Користувач має лише завантажити фото.

За визначеними факторами конкурентоспроможності (табл. 4.10) проводиться аналіз сильних та слабких сторін стартап-проекту (табл. 4.11). Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін.

Таблиця 4.11 - Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	*Фактор конкурентноспроможності*	*Бали 1-20*	*Рейтинг товарів-конкурентів у порівнянні з нашим підприємством*						
			-3	*-2*	*-1*	*0*	*+1*	*+2*	*+3*
1.	Використання ПЗ у вигляді веб-сервісу	15			+				
2.	Простота інтерфейсу користувача	20	+						

Перелік ринкових загроз та ринкових можливостей складається на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Ринкові загрози та ринкові можливості є наслідками (прогнозованими результатами) впливу факторів, і, на відміну від них, ще не є реалізованими на ринку та мають певну ймовірність здійснення.

Таблиця 4.12 – SWOT-аналіз стартап-проекту

Сильні сторони: простий користувацький інтерфейс, використання технологій	Слабкі сторони: потрібно мати обладнання для тренування моделі, яка буде класифікувати зображення
Можливості: у конкурента 1 виявлена проблема із надійністю ПЗ, додаткове фінансування для розповсюдження даної технології	Загрози: конкуренція, зміна потреб користувачів

На основі SWOT-аналізу розробляються альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок. Визначені альтернативи аналізуються з точки зору строків та ймовірності отримання ресурсів.

Таблиця 4.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	*Альтернатива (орієнтовний комплекс заходів) ринкової поведінки*	*Ймовірність отримання ресурсів*	*Строки реалізації*
1.	Створення ПЗ використовуючи нейронні мережі	80%	6 місяців
2.	Створення ПЗ на основі класичних методів машинного навчання	30%	12 місяців

З означених альтернатив обирається та, для якої: а) отримання ресурсів є більш простим та ймовірним; б) строки реалізації – більш стислими.

4.4 Розробка ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів.

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів.

Таблиця 4.14 - Вибір цільових груп потенційних споживачів

№ п/п	*Опис профілю цільової групи потенційних клієнтів*	*Готовність споживачів сприйняти продукт*	*Орієнтовний попит в межах цільової групи (сегменту)*	*Інтенсивність конкуренції в сегменті*	*Простота входу у сегмент*
1.	Магазини	Можливість автоматичної категоризації, що зменшує роботу людей	Великий	Існує 3 конкуренти, які надають схожі, але менш швидкі рішення.	Швидкодія, зручний користува-цький інтерфейс, точність класифіка-ції
2.	Підприємства	Можливість автоматичної категоризації, що зменшує роботу людей	Великий		Швидкодія, зручний користува-цький інтерфейс, точність класифіка-ції
Які цільові групи обрано: обираємо магазини та підприємства					

За результатами аналізу потенційних груп споживачів (сегментів) автори ідеї обирають цільові групи, для яких вони пропонуватимуть свій товар, та визначають стратегію охоплення ринку. Для роботи в обраних сегментах ринку необхідно сформувати базову стратегію розвитку. За М. Портером, існують три базові стратегії розвитку, що відрізняються за ступенем охоплення цільового ринку та типом конкурентної переваги, що має бути реалізована на ринку (за витратами або визначними якостями товару).

Отже, проілюструвати базову стратегію розвитку можна у виляді таблиці 4.15.

Таблиця 4.15 – Визначення базової стратегії розвитку

№ п/п	*Обрана альтернатива розвитку проекту*	*Стратегія охоплення ринку*	*Ключові конкурентоспроможні позиції відповідно до обраної альтернативи*	*Базова стратегія розвитку*
1.	Створення ПЗ використовуючи нейронні мережі	Ринкове позиціонування	Швидкодія, простота у користуванні, точність результатів	Диференціація

Було обрано таку альтернативу розвитку проекту: створення ПЗ використовуючи нейронні мережі.

Таблиця 4.16 - Визначення базової стратегії конкурентної поведінки

№ п/п	*Чи є проект «першопрохідцем» на ринку?*	*Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?*	*Чи буде компанія копіювати основні характеристики товару конкурента, і які?*	*Стратегія конкурентної поведінки*

1.	Ні	Так	Буде, а саме: основною задачею є розробка ПЗ з використанням нейронних мереж (конкуренти 1, 2, 3), простий інтерфейс користувача (конкурент 2)	Зайняття конкурентної ніші

Отже, було визначено базову стратегію конкурентної поведінки як зайняття конкурентної ніші.

Визначимо стратегію позиціонування у таблиці 4.17, що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект. На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту, а також в залежності від обраної базової стратегії розвитку та стратегії конкурентної поведінки розробляється стратегія позиціонування, що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект.

Таблиця 4.17 - Визначення стратегії позиціонування

№ п/п	*Вимоги до товару цільової аудиторії*	*Базова стратегія розвитку*	*Ключові конкурентоспроможні позиції власного стартап-проекту*	*Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)*

1.	Простота інтерфейсу, швидкодія, точність результатів	Диферен- ціація	Простота користувацького інтерфейсу дозволить отримувати необхідні дані і відслідковувати події в режимі реального часу	Швидкодія, безпека, простота, точність результатів

Отже, було вибрано такі асоціації, які мають сформувати комплексну позицію власного проекту.

4.5 Розробка маркетингової програми

Першим кроком є формування маркетингової концепції товару, який отримає споживач. Для цього у табл. 4.18 потрібно підсумувати результати попереднього аналізу конкурентоспроможності товару.

Таблиця 4.18 - Визначення ключових переваг концепції потенційного товару

№ п/п	*Потреба*	*Вигода, яку пропонує товар*	*Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)*
1.	Швидкодія	ПЗ працює досить швидко, результат можна отримати до 10 мс	Перевага у швидкості
2.	Простота користувацького інтерфейсу	Простота роботи додатку	Користувачі мають зручний інтерфейс для взаємодії з ПЗ

Отже бачимо, що проект має ключові переваги перед конкурентами, які

повністю відповідають потребам цільової аудиторії. Першим кроком є формування маркетингової концепції товару, який отримає споживач. Для цього у табл. 4.18 потрібно підсумувати результати попереднього аналізу конкурентоспроможності товару.

Далі у Таблиці 4.19 проілюстрована трирівнева маркетингова модель товару: уточнюється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання.

Таблиця 4.19 - Опис трьох рівнів моделі товару

Рівні товару	*Сутність та складові*		
I. Товар за задумом	ПЗ допомагає виконувати автоматичну класифікацію товарів		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Зручність та простота користувацького інтерфейсу 2. Швидкість роботи 3. Безпека згідно до світових стандартів	1.Нм 2.Нм 3.Нм	1.Технологічна 2.Технологічна 3.Технологічна
	Якість: згідно до стандарту ISO 4444 буде проведено тестування		
	Маркування відсутнє		

	Моя компанія: “Furniture classification”
III. Товар із підкріплюнням	1-місячна пробна безкоштовна версія
	Постійна підтримка для користувачів
За рахунок чого потенційний товар буде захищено від копіювання: патент	

Було описано три рівні моделі товару, з чого можна зробити висновок, що основні властивості товару у реальному виконанні є нематеріальними та технологічними. Також було надано сутність та складові товару у задумці та товару з підкріпленням

Після формування маркетингової моделі товару слід особливо відмітити – чим саме проект буде захищено від копіювання. Захист може бути організовано за рахунок захисту ідеї товару (захист інтелектуальної власності), або ноу-хау, чи комплексне поєднання властивостей і характеристик, закладене на другому та третьому рівнях товару.

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар (остаточне визначення ціни відбувається під час фінансово-економічного аналізу проекту), яке передбачає аналіз ціни на товари-аналоги або товари субститути, а також аналіз рівня доходів цільової групи споживачів (табл. 4.20). Аналіз проводиться експертним методом.

Таблиця 4.20 - Визначення меж встановлення ціни

№ п/п	*Рівень цін на товари-замінники, грн.*	*Рівень цін на товари-аналоги, грн.*	*Рівень доходів цільової групи споживачів, грн.*	*Верхня та нижня межі встановлення ціни на товар/послугу, грн.*
1.	25000	30000	200000	20000

Наступним кроком є визначення оптимальної системи збуту, в межах якого приймається рішення (табл. 4.21).

Таблиця 4.21 - Формування системи збуту

№ п/п	*Специфіка закупівельної поведінки цільових клієнтів*	*Функції збуту, які має виконувати постачальник товару*	*Глибина каналу збуту*	*Оптимальна система збуту*
1.	Купують підписку та роблять щорічні внески для подовження ліцензії	Продаж	0(напряму), 1(через одного посередника)	Власна та через посередників

Отже, система приноситиме прибуток завдяки щомісячним внескам для подовження ліцензії та придбанням підписок, продаж будк виконуватись напряму або через одного посередника.

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (табл. 4.22).

Таблиця 4.22 - Концепція маркетингових комунікацій

№ п/п	*Специфіка поведінки цільових клієнтів*	*Канали комунікацій, якими користуютьс я цільові клієнти*	*Ключові позиції, обрані для позиціонування*	*Завдання рекламного повідомлення*	*Концепція рекламного звернення*
1.	Використання за допомогою сайту	Інтернет	Швидкодія, простота у використанні, безпека	Показати переваги сервісу, у тому числі і перед конкурента- ми	Демо-ролик із використання

Отже, в Таблиці 4.22 наведено концепцію маркетингових комунікацій, було визначено, що придбання ліцензії на користування буде здійснюватись в мережі Інтернет, необхідним буде щомісячне її продовження, користування сервісом можливе у хмарі або ж на власних серверах

4.6 Висновки

Згідно до проведених досліджень існує можливість ринкової комерціалізації проекту. Також, варто відмітити, що існують перспективи впровадження з огляду на потенційні групи клієнтів, бар‘єри входження не є високими, а проект має дві

значні переваги перед конкурентами. Для успішного виконання проекту необхідно реалізувати програму із використанням засобів Python, Keras.

Для успішного виконання проекту необхідно реалізувати сервіс із використанням нейронних мереж для класифікації зображень. В рамках даного дослідження були розраховані основні фінансово-економічні показники проекту, а також проведений менеджмент потенційних ризиків. Проаналізувавши отримані результати, можна зробити висновок, що подальша імплементація є доцільною.

В рамках даного дослідження були розраховані основні фінансово-економічні показники проекту, а також проведений менеджмент потенційних ризиків. Проаналізувавши отримані результати, можна зробити висновок, що подальша імплементація є доцільною.

Було визначено такі сильні сторони: форма виконання, собівартість та наявність дміністратора для налаштування. Натомість як слабкою є - крос-платформеність.

Наявні такі фактори загроз: конкуренція, зміна потреб користувачів, зміна тарифів провайдера, надходження на ринок альтернативних продуктів, уповільнення росту ринку.

ВИСНОВКИ

В роботі були розглянуті та досліджені методи та підходи в задачах класифікації зображень за допомогою нейронних мереж. Було описано вибір моделей, метрик якості, методів оцінки моделей та підбір гіперпараметрів, виконано порівняння різних моделей та відповідних підходів.

Вхідними даними для даної задачі були відкриті дані - зображення предметів побуту, яких було 69 класів. Набір даних не збалансований. Проблему дизбалансу класів було вирішено за допомогою методу аугументації. Всього в наборі даних було 104461 зображень, для тесту використовувалось 3450 зображень, для навчання використовувалось - 101011. Відповідно на кожен клас було в середньому по 1400 екземлярів, найменший клас містив 719 зображень для тренування і найбільший - 2389. Для тесту використовувалось 50 картинок для кожного класу.

Було проведено порівняння таких архітектур як:

1. DenseNet;
2. InceptionV3;
3. SqeezeNet;
4. MobileNet.

В результаті чого найкращу точність показала модель DenseNet, яка в процесі покращила точність початкової моделі більш, ніж на 20%. Якщо ж готове рішення потрібно використовувати в системі з обмеженими ресурсами, то можна використати модель SqeezeNet, яка дає трохи гіршу точність, зате об’єм моделі менший в 8.5 разів, а точність гірша лише на 8.9%.

Було запропоновано методи покращення моделей. Модифіковані моделі було

протестовано, в результаті чого і було отримано точність 88.7%.

Також у роботі описана теоретична частина по алгоритмам, які використовуються для задач класифікації та безпосередньо використовувані моделі. Також було викладено суть методів обробки вхідних даних, вибір архітектур, функцій активації, метрик оцінки якості та методу оцінки моделі. Після чого було проведено підбір гіперпараметрів і проаналізовано отримані результати. Також було виконано порівняння чотирьох різних моделей, які були в процесі розроблено для конкретної задачі.

Якщо на одному зображені присутні два предмети, погана якість зображена або сам предмет зливається з фоном модель буде давати не правильні передбачення.

У розділі розробки стартап-проекту було визначено слабкі та сильні характеристики потенційного товару, що є підґрунтям для формування його конкурентоспроможності. Також в цьому розділі було проаналізовано вибір фреймворку, мови написання і тд з аналогів. Було розроблено стратегії, на різні випадки розвитку ринку, описано вихід на ринок та залучення інвестицій.

Описаний продукт є корисним для підприємств, які потребують швидкої автоматичної класифікації товарів побуту.

Отже, в роботі розрита науково-прикладна проблема: розробка методу автоматичної багатокласової категоризації зображень за допомогою систем штучного інтелекту.

Відповідно до поставленої мети в роботі було:

1. досліджено, очищено та оброблено вхідний набір даних;
2. підібрано та спроектовано архітектури нейронних мереж;

3. вибрано метрики оцінки алгоритму, вибір способу валідації;

4. досліджено вплив компонентів нейронної мережі на точність класифікації;

5. досліджено пливу гіперпараметрів на точність класифікації.

Отримані результати готові до впровадження в реальну систему, нижче наведено потенційні застосування та практична цінність результатів дипломної роботи:

1. Модель може використовуватись на будь-якому сайті чи мобільному додатку, де потрібна автоматична категоризація товарів предметів побуту;

2. Сформульовано основні концепти, на які потрібно звернути увагу при проектуванні архітектури нейронної мережі, описано метод та підходи, які дозволяють майже без втрат точності зменшити розміри моделі та швидкість отримання передбачень;

3. Розширивши набір даних та дотренувавши систему, можна використовувати для більшої кількості категорій, також можна взяти інший набір даних і спробувати застосувати ці ж методи та архітектури, можливо, трохи змінити параметри мережі і отримати іншу готову систему для класифікації зображень.

Отже, дана робота є актуальною і містить наукову новизну, в подальшому її можна вдосконалити таким чином, щоб її можна було використовувати в реальному часі, та спробувати зробити її більш універсальною для різних наборів даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Darken, C. Learning rate schedules for faster stochastic gradient search / Darken, C., Chang, J. Moody, J. // Neural Networks for Signal Processing II Proceedings of the 1992 IEEE Workshop, September 1–11, 1992.
2. Dauphin, Y. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization / Dauphin, Y., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., Bengio, Y. – Режим доступу: http://arxiv.org/abs/1406.2572.
3. Densely Connected Convolutional Networks – Режим доступу: https://arxiv.org/pdf/1608.06993v3.pdf - Дата доступу – 01.05.2018
4. Ensemble learning - Режим доступу: https://en.wikipedia.org/wiki/Ensemble_learning - Дата доступу – 07.04.2018.
5. Fausett L. V. Fundamentals of neural networks: architectures, algorithms, and applications. Upper Saddle River (USA): Prentic Hall, 1994. 476 p.
6. Flickner M., Sawhney H., NIBLACK W., ET AL. Query by image and video content: The QBIC system // IEEE Comput. 1995. V. 28, N 9. P. 23–32.
7. Gonzalez A. C., Sossa J. H., Felipe E. M. Wavelet transforms and neural networks applied to image retrieval // Proc. of the 18th Intern. conf. on pattern recognition. Hong Kong (China), 20–24 Aug. 2006. P. 909–912.
8. Goodfellow I. Deep Learning / Goodfellow I., Bengio Y. and Courville A.; Cambridge MA : MIT Press [2017] – 777 pages.
9. Hackeling Gavin. Mastering Machine Learning with scikit-learn / Hackeling Gavin. - Packt Publishing Ltd. - 2014. – С. 1-32.
10. Kingma, D. P. Adam: a Method for Stochastic Optimization / Kingma, D. P., Ba, J.L. // International Conference on Learning Representations, May 7-9, 2015, San-Diego, USA.

11. Lofti M., Solimani A., Dargazany A., et al. Combining wavelet transforms and neural networks for image classification // Proc. of the 41st Southeastern symp. on system theory. Tennessee (USA), Mar. 15-17, 2009. IEEE SSST, 2009. P. 44–48.
12. Niblack W., Barber R., Equitz W., et al. The QBIC project: querying images by content using color, texture, and shape // Proc. of the Intern. conf. on storage and retrieval for image and video databases. Bellingham, Washington, USA, Febr. 1993. IS&T/SPIE, SPIE, 1993. P. 173–187.
13. Nielsen M. Neural Networks And Deep Learning. – Режим доступу: http://neuralnetworksanddeeplearning.com/.
14. Park S. B., Lee J. W., Kim S. K. Content based image classification using a neural network // Pattern Recognition Lett. 2004. V. 25. N 3. P. 287–300.
15. Petzold J. Augsburg Indoor Location Tracking Benchmarks / Petzold J. // Technical Report, Institute of Computer Science, University of Augsburg, Germany. – 2004. – C. 1-10.
16. Robinds H. A stochastic approximation method // Annals of Mathematical Statistics / Robinds H. and Monro S. – 1951 – vol. 22 – pp. 400–407.
17. Ruder S. (2016) An overview of gradient descent optimisation algorithms. – Режим доступу: https://arxiv.org/abs/1609.04747.
18. Sermanet, P. Traffic Sign Recognition with Multi-Scale Convolutional Networks / Sermanet, P., LeCun, Y. // The 2011 International Joint Conference on Neural Networks, September 2011.
19. Smith j. B. Chang S. F. Tools and techniques for color image retrieval // Proc. of the Intern. conf. on symp. on electronic imaging: science and technology storage and retrieval for image and video databases. San Jose (USA), Feb. 1996. IS&T/SPIE, SPIE, 1996. P. 426–437.

20. Springenberg, J. T. Striving for Simplicity: The All Convolutional Net /Springenberg, J. T. Dosovitskiy, A.; Brox, T. & Riedmiller, M. – Режим доступа: https://arxiv.org/abs/1412.6806.
21. Sutton, R. S. Two problems with backpropagation and other steepest-descent learning procedures for networks. Proc. 8th Annual Conf. Cognitive Science Society – 1986.
22. Swain M. J., Ballard D. H. Color indexing // Intern. J. Comput. Vision. 1991. V. 7, N 1. P. 11–32.
23. Буй Тхи Тху Чанг, Спицын В. Г. Разложение цифровых изображений с помощью двумерного дискретного вейвлет-преобразования и быстрого преобразования // Изв. Том. политехн. ун-та. 2011. Т. 318. № 5. С. 73–76.
24. Наївний баєсів класифікатор - Режим доступу: https://uk.wikipedia.org/wiki/Наївний_баєсів_класифікатор - Дата доступу – 03.04.2018.
25. Штучні нейронні мережі (НС) – Режим доступу: https://knhelp.wordpress.com/2012/04/19/л8-штучні-нейронні-мережі-нс/ - Дата доступу – 01.04.2018

ДОДАТКИ

Наведемо програмну реалізацію для SqeezeNet:

```
import numpy as np
import keras
from keras.models import Model
from keras.layers import Input, Convolution2D, MaxPooling2D
from keras.layers import Activation, Dropout, GlobalAveragePooling2D, concatenate


mean = np.array([0.485, 0.456, 0.406], dtype='float32')
std = np.array([0.229, 0.224, 0.225], dtype='float32')


def preprocess_input(x):
    x /= 255.0
    x -= mean
    x /= std
    return x



        # a building block of the SqueezeNet architecture
        def fire_module(number, x, squeeze, expand, weight_decay=None, trainable=False):

            module_name = 'fire' + number

            if trainable and weight_decay is not None:
                kernel_regularizer = keras.regularizers.l2(weight_decay)
            else:
                kernel_regularizer = None

            x = Convolution2D(
                squeeze, (1, 1),
                name=module_name + '/' + 'squeeze',
                trainable=trainable,
                kernel_regularizer=kernel_regularizer
            )(x)
            x = Activation('relu')(x)

            a = Convolution2D(
                expand, (1, 1),
                name=module_name + '/' + 'expand1x1',
                trainable=trainable,
                kernel_regularizer=kernel_regularizer
            )(x)
```

```
    a = Activation('relu')(a)

    b = Convolution2D(
        expand, (3, 3), padding='same',
        name=module_name + '/' + 'expand3x3',
        trainable=trainable,
        kernel_regularizer=kernel_regularizer
    )(x)
    b = Activation('relu')(b)

    return concatenate([a, b])


def SqueezeNet(weight_decay, image_size=224):

    image = Input(shape=(image_size, image_size, 3))

    x = Convolution2D(
        64, (3, 3), strides=(2, 2), name='conv1',
        trainable=False
    )(image) # 111, 111, 64

    x = Activation('relu')(x)
    x = MaxPooling2D(pool_size=(3, 3), strides=(2, 2))(x) # 55, 55, 64

    x = fire_module('2', x, squeeze=16, expand=64) # 55, 55, 128
    x = fire_module('3', x, squeeze=16, expand=64) # 55, 55, 128
    x = MaxPooling2D(pool_size=(3, 3), strides=(2, 2))(x) # 27, 27, 128

    x = fire_module('4', x, squeeze=32, expand=128) # 27, 27, 256
    x = fire_module('5', x, squeeze=32, expand=128) # 27, 27, 256
    x = MaxPooling2D(pool_size=(3, 3), strides=(2, 2))(x) # 13, 13, 256

    x = fire_module('6', x, squeeze=48, expand=192) # 13, 13, 384
    x = fire_module('7', x, squeeze=48, expand=192) # 13, 13, 384
    x = fire_module('8', x, squeeze=64, expand=256) # 13, 13, 512
    x = fire_module('9', x, squeeze=64, expand=256) # 13, 13, 512

    x = Dropout(0.5)(x)
    x = Convolution2D(
        256, (1, 1), name='conv10',
        kernel_initializer=keras.initializers.RandomNormal(stddev=0.01),
        kernel_regularizer=keras.regularizers.l2(weight_decay)
    )(x) # 13, 13, 256

    x = Activation('relu')(x)
    logits = GlobalAveragePooling2D()(x) # 256
    probabilities = Activation('softmax')(logits)

    model = Model(image, probabilities)
    model.load_weights('squeezenet_weights.hdf5', by_name=True)

    return model
```

Наведемо також код для MobileNet:

```
import keras
from keras.applications.mobilenet import MobileNet
from keras.models import Model
from keras.layers import Activation, GlobalAveragePooling2D, Dropout, Dense, Input


def get_mobilenet(input_size, alpha, weight_decay, dropout):
    input_shape = (input_size, input_size, 3)
    base_model = MobileNet(
        include_top=False, weights='imagenet',
        input_shape=input_shape, alpha=alpha
    )
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dropout(dropout)(x)
    logits = Dense(256, kernel_regularizer=keras.regularizers.l2(weight_decay))(x)
    probabilities = Activation('softmax')(logits)
    model = Model(base_model.input, probabilities)

    for layer in model.layers[:-2]:
        layer.trainable = False

    return model
```

Частина коду, де створюється модель DenseNet:

```
class DenseNet(nn.Module):
    r"""Densenet-BC model class, based on
    `"Densely Connected Convolutional Networks" <https://arxiv.org/pdf/1608.06993.pdf>`

    Args:
        growth_rate (int) - how many filters to add each layer (`k` in paper)
        block_config (list of 4 ints) - how many layers in each pooling block
        num_init_features (int) - the number of filters to learn in the first convolution layer
        bn_size (int) - multiplicative factor for number of bottle neck layers
          (i.e. bn_size * k features in the bottleneck layer)
        drop_rate (float) - dropout rate after each dense layer
        num_classes (int) - number of classification classes
    """
    def __init__(self, growth_rate=32, block_config=(6, 12, 24, 16),
                 num_init_features=64, bn_size=4, drop_rate=0.2, final_drop_rate=0.2, num_classes=1000):

        super(DenseNet, self).__init__()

        # First convolution
        self.features = nn.Sequential(OrderedDict([
            ('conv0', nn.Conv2d(3, num_init_features, kernel_size=7, stride=2, padding=3, bias=False)),
            ('norm0', nn.BatchNorm2d(num_init_features)),
            ('relu0', nn.ReLU(inplace=True)),
            ('pool0', nn.MaxPool2d(kernel_size=3, stride=2, padding=1)),
        ]))

        # Each denseblock
        num_features = num_init_features

    for i, num_layers in enumerate(block_config):

        # add dropout to the last dense block
        if i == len(block_config) - 1:
            dropout = drop_rate
        else:
            dropout = 0

        block = _DenseBlock(
            num_layers=num_layers, num_input_features=num_features,
            bn_size=bn_size, growth_rate=growth_rate, drop_rate=dropout
        )
        self.features.add_module('denseblock%d' % (i + 1), block)
        num_features = num_features + num_layers*growth_rate
        if i != len(block_config) - 1:
            trans = _Transition(
                num_input_features=num_features,
                num_output_features=num_features // 2
            )
            self.features.add_module('transition%d' % (i + 1), trans)
            num_features = num_features // 2

    # Final batch norm
    self.features.add_module('norm5', nn.BatchNorm2d(num_features))
```

```
# Linear layer
self.dropout = nn.Dropout(p=final_drop_rate)
self.classifier = nn.Linear(num_features, num_classes)


def forward(self, x):
    features = self.features(x)
    out = F.relu(features, inplace=False)
    # I changed 7 -> 9 because 224 -> 299
    out = F.avg_pool2d(out, kernel_size=9).view(features.size(0), -1)
    out = self.dropout(out)
    out = self.classifier(out)
    return out
```