Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Методи розпізнавання та ідентифікації об'єктів у доповненої реальності і переклади їх назви на іноземну мову
(тема)

Виконав:
студент 2 курсу, групи СШІм-19-2
Зареченський О.А.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)

Керівник к.т.н. Шевченко О. Ю.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Зареченському Олексію Альбертовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Методи розпізнавання та ідентифікації об'єктів у доповненої реальності і переклади їх назви на іноземну мову

затверджена наказом університету від 29 березня 20 21 р. № 390Ст

2. Термін подання студентом роботи до екзаменаційної комісії _____ _____ 20 ___ р.

3. Вихідні дані до роботи Розробка мобільного додатку під платформу iOS, що містить методи розпізнавання та ідентифікації об'єктів, а також використання доповненої реальності, що відображає обрану мову користувача та її переклад на обрану користувачем мову.

4. Перелік питань, що потрібно опрацювати в роботі мета роботи; аналіз предметної області; аналіз нейронних мереж; розробка специфікації програмного додатку; кодування програми; тестування і налагодження програми;

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Рисунок 4.1 – Діаграма прецедентів (Use Case); Рисунок 4.2 – Діаграма послідовності; Рисунок 4.3 – Клієнт-серверна архітектура; Рисунок 4.4 – Діаграма розгортання; Рисунок 4.5 – Діаграма компонентів; Рисунок 4.6 – Діаграма класів;

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	к.т.н. Шевченко О. Ю.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	01.03.2021	виконано
2	Аналіз предметної області	05.03.2021	виконано
3	Аналіз нейронних мереж	07.03.2021	виконано
4	Розробка специфікації програмного	20.03.2021	виконано
5	Кодування програми	27.03.2021	виконано
6	Тестування і налагодження програми	01.04.2021	виконано
7	Підготовка пояснювальної записки	05.04.2021	виконано
8	Оформлення пояснювальної записки	10.04.2021	виконано
9	Нормоконтроль	05.05.2021	виконано
10	Попередній захист	15.05.2021	виконано
11	Захист перед ЕК	20.05.2021	

Дата видачі завдання _____ _____ 20 21 р.

Студент _____
(підпис)

Керівник роботи _____ _____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Записка пояснювальна: 114 с., 50 рис., 2 дод. 24 джерел

АРХІТЕКТУРА, МОБІЛЬНИЙ ДОДАТОК, ALAMOFIRE, AR, ARKit, AUGMENTED REALITY, CNN, CORE ML, MACHINE LEARNING, MOBILE, HTTP, OBJECTIVE-C, REST, SWIFT, VISION.

Метою роботи є поєднання методів розпізнавання та ідентифікації об'єктів, доповненої реальності та переклад їх назв на іноземну мову.

Методом вирішення задачі є розробка мобільного додатку під платформу iOS, що містить методи розпізнавання та ідентифікації об’єктів, а також використання доповненої реальності, що відображає обрану мову користувача та її переклад на обрану користувачем мову. Мобільний додаток був розроблений за допомогою мов програмування Swift та Objective-C, модель натренованої мережі була реалізована за допомогою вбудованого фреймворку Core ML [1] та Vision, доповнена реальність на основі фреймворку ARKit, переклад на основі Google Translate API, для розробки клієнтської частини Cocoa Touch [2], операційної системи MacOS Big Sur, протоколу передачі даних HTTP, RESTful архітектури та фреймворку для передачі даних Alamofire [3].

Об’єктом дослідження є існуючі методи та інструментарій доповненої реальності, використання нейронних мереж у мобільних додатках на платформі IOS та процес розпізнавання об’єктів при використанні камери.

РЕФЕРАТ

Записка пояснительная: 114 с., 50 рис., 2 прил., 24 источника

АРХІТЕКТУРА, МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, ALAMOFIRE, AR, ARKit, AUGMENTED REALITY, CNN, CORE ML, MACHINE LEARNING, MOBILE, HTTP, OBJECTIVE-C, REST, SWIFT, VISION.

Целью работы является сочетание методов распознавания и идентификации объектов, дополненной реальности и перевод их названий на иностранный язык.

Методом решения задачи является разработка мобильного приложения под платформу iOS, содержащий методы распознавания и идентификации объектов, а также использования дополненной реальности, отражает выбранную язык пользователя и его перевод на выбранную пользователем язык. Мобильное приложение был разработан с помощью языков программирования Swift и Objective-C, модель натренированной сети была реализована с помощью встроенного фреймворка Core ML [1] и Vision, дополненная реальность на основе фреймворка ARKit, перевод на основе Google Translate API для разработки клиентской части Cocoa Touch [2], операционной системы MacOS Big Sur, протокола передачи данных HTTP, RESTful архитектуры и фреймворка для передачи данных Alamofire [3].

Объектом исследования являются существующие методы и инструментарий дополненной реальности, использовасние нейронных сетей в мобильных приложениях на платформе IOS и процесс распознавания объектов при использовании камеры.

ABSTRACT

Explanatory note: 114 p., 50 fig., 2 ann., 24 sources.

ALAMOFIRE, AR, ARCHITECTURE, ARKit, AUGMENTED REALITY, CNN, CORE ML, MACHINE LEARNING, MOBILE APPLICATION, HTTP, OBJECTIVE-C, RECOGNIZING SURROUNDING OBJECTS, REST, SWIFT, VISION.

The aim of the work is to combine methods of object's recognition and identification, augmented reality and translation of its names into a foreign language.

The method of solving the problem is to develop a mobile application for the iOS platform, which contains methods for objects recognizing and identifying, also the use of augmented reality, which reflects the user's chosen language and its translation into the user's chosen language. The mobile application was developed using Swift and Objective-C programming languages, the trained network model was implemented using the built-in Core ML [1] and Vision framework, augmented reality is based on the ARKit framework, translation is based on Google Translate API, to develop the client part is used Cocoa Touch [2], MacOS Big Sur operating system, HTTP data transfer protocol, RESTful architecture and Alamofire data transfer framework [3].

The object of research is the existing methods and tools of augmented reality, the use of neural networks in mobile applications on the IOS platform and the process of object recognition when using the camera.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ ТА ТЕРМІНІВ

МБ – мобільний додаток;

ОС – операційна система;

ПЗ – програмне забезпечення;

ШНМ – Штучна нейронна мережа;

API – Application Programming interface – програмний інтерфейс сервісу;

AR – Augmented reality – доповнена реальність;

CNN – Convolutional Neural Network – згорткова нейронна мережа;

Cocoa Touch – фреймворк для розробки мобільних додатків під платформу iOS;

CORE ML – Machine Learning framework – фреймворк для машинного навчання;

CPU – Central Processing Unit – центральний процесор;

DNN – Deep Neural Network – глибинна нейронна мережа;

GPU – Graphic Processing Unit – графічний процесор;

HTTP – HyperText Transfer Protocol – протокол передачі гіпертекста;

MDP – Markov Decision Process – Процеси прийняття рішень Маркова;

MVC – Model-View-Controller – патерн проектування;

REST – Representation State Transfer – передача репрезентативного стану;

UML – Unified Modeling Language – уніфікована мова моделювання.

ВСТУП

У двадцять першому столітті все більше уваги приділяється інформаційним технологіям та їх використанню за допомогою мобільних пристроїв. Люди настільки звикли до технологічних благ, що навіть не можуть уявити життя без них. Саме тому більшість сфер життя переходить на новий рівень – рівень інформаційних технологій. Люди більше не мають потреби йти до таксофону, щоб подзвонити та домовитись про зустріч, до банку щоб сплатити комунальні послуги або отримувати інформацію від людей чи друкованих матеріалів, але сфера інформаційних технологій постійно змінюється, таким чином з'являються нові галузі та науки. У тому числі, ще в 1956 році, зародився такий розділ як штучний інтелект. Він включає в себе комп'ютерну лінгвістику та інформатику, що сприяє подальшому розвитку технологій за рахунок знань та, зокрема, застосувань новітніх алгоритмів.

Мобільні додатки – програмне забезпечення, призначене для роботи на смартфонах та інших видах компактних пристроїв, що включає в себе ряд окремих технічних засобів і прийомів роботи з ними. Таке програмне забезпечення має певні переваги, а саме: доступність за рахунок мережі Internet, швидкість розповсюдження, компактність та зручність використання, менші вимоги до апаратних засобів, а також великий спектр послуг. Для даної роботи ці переваги мають виняткове значення, тому що ринок постійно зростає.

На сьогоднішній день існує велика кількість різноманітних додатків і сервісів для вирішення побутових проблем, спілкуванню з людьми, розваг та самоосвіти. Зокрема, у сфері мобільних додатків, самоосвіта не є дуже розвиненою темою, адже не кожен готовий платити за нові знання чи знайти час для них.

Актуальність даної роботи зумовлена проблемою вивчення іноземних мов та інтересом до цього процесу, адже мотивація може згаснути ще з початку

навчання, до того ж постійно зростає попит на методи навчання і тренування людей таким способом, щоб виключити ризики і небезпеку реальних ситуацій. Вирішенням цього питання є технология доповненої реальності.

Існує величезна кількість різноманітних моделей машинного навчання, що містять різні реалізації та алгоритми для вирішення задач класифікації, кластерного аналізу, регресії та інші.

Спеціалісти стикаються з проблемами використання машинного навчання у мобільних додатків, а саме у режимі офлайн, тому що моделі можуть бути великих розмірів, що є перешкодою для використання усіма охочими. Використання різноманітних сервісів, наприклад: AWS Cloud, Machine Learning від Microsoft Azure, Cloud Machine Learning Engine [4] та інші, за умовою, що треба використовувати Internet та витрачати гроші на їх послуги. Допомогти впоратися з цими труднощами може компактна модель машинного навчання, в основі якої лежить згорткові мережі. За допомогою спеціального фреймворку Core ML, маємо можливість довчити модель та використовувати для своїх цілей. В XXI столітті люди постійно користуються мобільними додатками, тому такі моделі, у майбутньому, можуть спростити нам життя та надати можливість використовувати їх скрізь, а фреймворк ARKit

Метою даної роботи є розробка ПЗ, що базується на інструментах розробки додатків під iOS, що надає можливим використання машинного навчання та перекладу на інші мови у режимі реально часу за допомогою доповненої реальності. В даній роботі використовується об’єктно-орієнтований підхід до розробки програмної системи.

1 АНАЛІЗ ПРОБЛЕМНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Діяльність, спрямована на виявлення реальних потреб користувачів, а також на з'ясування сенсу висловлених вимог, називається аналізом предметної області. Аналіз предметної області – це перший крок етапу системного аналізу, з якого починається розробка програмної системи.

Інтерес до задачі розпізнаванню образів за допомогою методів класифікації в останні роки неухильно зростає. Одним з головних факторів є стрімке збільшення зображень в Internet та необхідність автоматичної анотації для задач пошуку. Зокрема, зберігання великого обсягу зображень дає змогу використовувати як для аналітики, так і для збору конкретних даних з метою поставленої бізнес-задачі. На даний момент збільшується розвиток технологій машинного навчання і штучного інтелекту, а це, в свою чергу, дозволяє багатьом організаціям зберігати величезні бази даних, які необхідно обробляти, класифікувати та аналізувати, як, наприклад, сервера Amazon Web Services. Ці бази даних настільки великі, що експерти у цій галузі не завжди можуть впоратися і саме це вплинуло на створення методів аналізу та автоматичного дослідження цих даних.

Багатьом відомо, що найпопулярнішим видом даних є зображення: котиків, різних тварин, навколишнього середовища, селфі та інші. Однак, маючи можливість фотографувати, не можна так швидко навчити дитину всім базовим словам або користувача з обмеженими можливостями, тому ми маємо змогу все обробляти в режимі реального часу і в ту саму секунду відображати на екрані об’єкти що знаходяться перед тобою.

В ході дослідження предметної області було обрано тему «Методи розпізнавання та ідентифікації об'єктів у доповненої реальності і переклади їх назви на іноземну мову». Основною метою стало створення зручного додатку, за допомогою якого людина має змогу в інтерактивному режимі досліджувати навколишнє середовище, дізнаватися більше та вивчати нові слова. Додаток повинен вміти взаємодіяти з навколишнім середовищем за допомогою камери, розпізнавати об’єкти, класифікувати зображення та перекладати на обрану іноземну мову, а також відображати це за допомогою доповненої реальності. На початковому рівні цього буде достатньо для дослідження предметів, що оточують користувача та збільшення словникового запасу через переклад нових слів.

Щоб розширити діапазон користувачів, був обраний мінімалістичний та зрозумілий дизайн проекту з точки зору User Experience. Найбільша кількість людей знаходиться у Китаї, Саудівській Аравії, Індії, Європі та Америці, тому потенційна аудиторія, в першу чергу, є англомовною, а коло користувачів звужується через особливості перекладу на такі мови як: Китайська, Індійська та Арабська. З цього можна зробити висновок, що основна аудиторія – користувачі з Європи та Америки.

1.2 Виявлення проблем та актуалізація рішень

Зараз, у відповідь на основну проблему, існує безліч способів упаковки та доставки вмісту до мобільних додатків, і шляхи досягнення динамічної доставки вмісту є значні.

Одним із ключових питань створення досвіду доповненої реальності є правильна інтеграція віртуального вмісту в реальний світ. Необхідний унікальний процес отримання технології для відстеження динаміки поверхні

реальних об'єктів і правильного проектування цільової моделі на ці поверхні, що вимагає належної підготовки нейронної мережі, яка може класифікувати та визначити бажаний об'єкт як опорну точку проекції. Проблема полягає в тому, що для правильного використання моделі нейронної мережі сморід повинен бути включений в пакет мобільних додатків [5], що значно обмежує гнучкість такої розробки системи. Тому було вирішено вивчити цю проблему та запропонувати її рішення шляхом динамічної доставки добре навченої моделі кінцевому користувачеві, яка може бути надалі використана в додатку, встановленому на пристрої користувача.

Інша головна проблема полягає в тому, що модель нейронної мережі з доповненою реальністю є більш динамічною. Зазвичай ці підготовлені моделі вже є в SDK, наприклад: обличчя, QR-коди, коти, які важко змінити одразу. Якщо спробувати додати власну нейронну мережу є великий ризик зустрітися з проблемою сумісності, надмірним навантаженням системи, до того ж розмір цих моделей занадто великий, тому крім динамічних проблем, існують також проблеми з оптимізацією [6].

1.3 Поняття віртуальність континуум, змішана, доповнена та віртуальна реальність

Віртуальність континуум представляє собою безперервну шкалу в діапазоні між повністю віртуальним, у віртуальності, досконало реальною та реальністю. Таким чином, континуум реальності – віртуальність включає в себе всі можливі варіації та композиції реальних та віртуальних об'єктів. Це було описано як концепцію нових медіа та інформатики, але насправді це можна розглянути як питання антропології. Концепція була вперше представлена Полом Мілграмом. Область між двомісними краями, де змішане реальне та віртуальне,

називається змішаною реальністю. Він, у свою чергу, є як з повної реальності, де віртуальне поповнення реального, так і з повної віртуальності, де реальне поповнення віртуального.

У травні 2007 року у звіті «Physical Review E» було описано інтерактивну систему, яка включає фактичний фізичний маятник, прикріплений до маятника, який існує лише у віртуальній реальності. Рівень переходу від реальності до самої віртуальності показано на рисунку 1.1.

Рисунок 1.1 – Континуум реальності – віртуальність

Цей континуум був розширений до двовимірної площини віртуальності і медіальності, що можна побачити на рисунку 1.2. Таксономія реальності, віртуальності, медіальності. Походження R позначає незмінну реальність. Континуум по осі віртуальності, V, включає реальність, доповнену графікою (доповнена реальність), а також графіку, доповнену реальністю (доповнена віртуальність). Однак таксономія також включає модифікацію реальності або віртуальності або будь-яку їх комбінацію. Ось медіальної позначає зміни. Модифікація позначається переміщенням вгору по осі медіальної. Далі по цій осі, наприклад, ми можемо знайти опосередковану реальність, опосередковану віртуальність або будь-яку їх комбінацію. Вище і правіше у нас є віртуальні світи, які реагують на сильно змінену версію реальності. Доповнена реальність і змішана реальність тепер іноді використовуються як синоніми.

Континуум віртуальності виріс і пройшов повз таких ярликів, як інформатика і нові медіа. Оскільки ця концепція багато в чому пов'язана з тим, як люди продовжують змінювати спосіб спілкування; спосіб формування ідентичностей і спосіб їх взаємодії зі світом і всередині нього; його більш точно описати як предмет антропології. Зміни у ставленні до технологій і засобів масової інформації та збільшення їх доступності змінили і розширили способи їх використання. Один до одного (SMS), один до багатьох (електронна пошта) і багато до багатьох (чати) міцно увійшли в суспільство. Використання таких предметів зробило колись чіткі відмінності, такі як онлайн і офлайн, застарілими, а відмінності між реальністю і віртуальністю стали розмитими, оскільки люди включаються і в значній мірі покладаються на віртуальність в свої повсякденні особисті реальності.

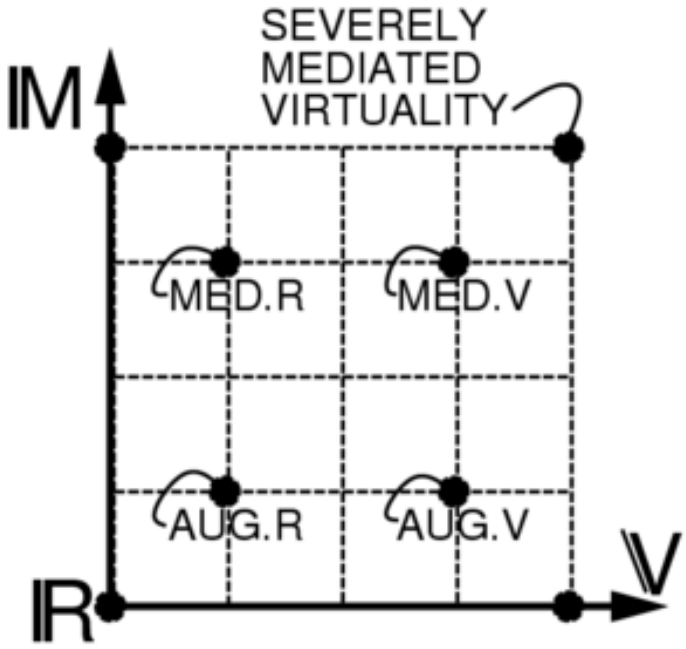


Рисунок 1.2 – Континуум опосередкованої реальності, який показує чотири точки: розширена реальність, розширена віртуальність, опосередкована реальність і опосередкована віртуальність на осях віртуальності і медіальної

Як найвідоміший термін, почнемо з Віртуальної реальності. Слово «віртуальний» змінювалося протягом століть. Випливаючи з латинського слова virtus, потім virtualis, віртуальний у XV столітті означало «бути чимось по суті чи ефекту, хоча насправді чи насправді», то в 1959 році значення загальновживаного змінилося на комп'ютерне значення «не фізично існуючого, а з'являються за допомогою програмного забезпечення».

Спроба захопити «Віртуальну реальність» набагато довше, ніж просто останні п'ять-десять років. У 1950-х були популярні однорангові іграшки та закриті імітатори польотів, дебютовані в 1960-х, але ідея VR сягає ще далі.

Ще в 30-х роках письменники наукової фантастики, винахідники та майстрині мріяли про середовище, де можна врятуватися від реальності за допомогою мистецтва та машин. Ми зважували питання про віртуальну реальність проти доповненої реальності проти змішаної реальності задовго до того, як у нас з’явилася технологія, щоб зробити їх можливими. Технологія наздогнала фантастику, і дослідники ринку прогнозують швидке зростання галузі VR.

Віртуальна реальність (VR) – це технологія, яка використовує програмне забезпечення та гарнітурні пристрої, щоб замінити погляд з реального світу на цифрову сцену. Використання гарнітур із повним покриттям повністю блокує ваше оточення та вимикає фізичний світ під час використання. РК-екрани або OLED-панелі всередині лінз цих пристроїв гарнітури відображають віртуальне середовище, створене комп’ютером, і ваш світогляд замінюється. Зазвичай пристрої підключаються до ПК, консолі або смартфона, що забезпечує віртуальне бачення. Ці бачення можуть бути копіями реального місця або місця з цілком уявного світу. VR дозволяє людям отримати повний захоплюючий досвід у цих віртуальних місцях. Це обманює ваші почуття, дозволяючи побачити лише те, що відображають лінзи ваших очей. Крім того, завдяки реалістичним звукам,

візуальним зображенням на 360 градусів та механізмам зйомки руху він може імітувати ваші дії, дозволяти інтерактивні зустрічі з віртуальними предметами та змушувати вас відчувати себе, ніби ви насправді знаходитесь у цьому змодельованому місці.

VR є найпопулярнішою з усіх цих трьох технологій, здебільшого тому, що він йде далеко від обох інших двох, і це вже дуже відома та використовувана високотехнологічна функція у багатьох галузях. Ігрові та розважальні компанії були ранніми адаптерами VR, як для AR. І так само, зараз він використовується в багатьох інших галузях, таких як архітектура та будівництво, подорожі, нерухомість, освіта, військова сфера тощо.

Змішана реальність, у певному сенсі, є спектром між VR та реальним світом. Як висловилися Мілграм і Кішино, змішана реальність знаходиться десь між крайностями віртуального континууму. Це включає AR та AV. Подібно до доповненої, змішана реальність – це злиття реального та віртуального світів. Однак у змішаній реальності віртуальний світ може інтегруватися з реальним світом, щоб забезпечити взаємодію між ними. Щоб пояснити, змішана реальність відрізняється від AR тим, що MP має оклюзію та інтеграцію з AV. Оклюзія дозволяє віртуальні об'єкти затемнюватись реальними об'єктами. Це змушує користувача відчувати, ніби віртуальне середовище занурене в реальний світ, а не відчуває себе зануреним у такий віртуальний світ, як VR.

Змішана реальність має багато практичних застосувань, одним з яких є інтерактивне управління вмістом продуктів (IPCM). IPCM використовується, щоб дозволити клієнтам переглядати меблі, фарби та інші товари у своїх будинках або на робочому просторі практично перед тим, як вони купують. Інші способи використання MP включають художню виразність, дослідження в галузі штучного інтелекту, оцифровану людську свідомість і, звичайно, військову та медичну підготовку. Іншим основним використанням є підгалузь MR, розширена

віртуальність (AV). AV інтегрує реальні об'єкти у віртуальний простір. Зелений екран та фони відеочату – чудові приклади AV. Ця технологія була особливо важливою у 2020 році, оскільки підтримувала та вдосконалювала дистанційну роботу під час пандемії COVID-19.

1.4 Постановка задачі

Мобільний додаток буде націлений на самоосвіту, головною метою якої є вивчення іноземних мов, що включає в себе такий інструментарій: моделі машинного навчання на основі фреймворку Core ML [7], переклад та озвучування слів іноземною мовою. У майбутньому є намір створити інтерактивну гру з інтервальним повторенням, що допоможе не тільки вивчати навколишнє середовище, а й запам’ятовувати нові слова. Застосунок буде конкурентоспроможним за допомогою зручного та простого інтерфейсу, а також матиме корисний функціонал. Даний продукт буде вирішувати наступні проблеми, які присутні в популярних сервісах:

– відсутність нав’язливої монетизації;

– використання багатопотоковості для збереження енергії у додатку.

Продукт буде відрізнятися наступним функціоналом:

– присутність офлайн режиму, який надає змогу використовувати додаток без підключення до мережі Internet;

– зручний та зрозумілий інтерфейс;

– можливість прослухати слова на іноземній мові;

– можливість сканувати предмети та перекладати в реального часу.

Головними та практично унікальними функціями будуть:

– можливість використовувати як онлайн режим, так і офлайн зі зберіганням якості результатів;

– можливість вивчати мову за допомогою інтервального повторення;

– можливість розпізнавати об’єкти навколишнього середовища та перекладати їх [8];

– створення списку об’єктів батьками, які підліток має знайти та відсканувати.

Головною метою даної роботи є дослідження існуючих методів доповненої реальності, а також засобів та їх інструментів. Процес генерації, відображення та зчитування контента повинен бути цілком в реальному часі у мобільному додатоку, використовуючи лише сервера Google для перекладу на іноземну мову. Для досягення мети за допомогою доповненої реальності, треба врахувати, що ця технологія будет оперувати просторовими моделями, тому спочатку вона буде проеціювати віртуальний об'єкт, що враховує просторову геометрію будування та розташування об'єктів.

2 ЗАСОБИ ТА ТЕХНОЛОГІЇ ДОПОВНЕНОЇ РЕАЛЬНОСТІ

2.1 Технології доповненої реальності

Доповнена реальність (augmented reality, AR) – результат введення в поле сприйняття будь-яких сенсорних даних з метою доповнення відомостей про оточення і поліпшення сприйняття інформації. Її специфіка полягає в тому, що вона програмним чином візуально поєднує два спочатку незалежних простору: світ реальних об'єктів навколо нас і віртуальний світ, відтворений на комп'ютері.

Нове віртуальне середовище утворюється шляхом накладення запрограмованих віртуальних об'єктів поверх відеосигналу з камери, і стає інтерактивною шляхом використання спеціальних маркерів.

Доповнена реальність вже багато років використовується в медицині, в рекламній галузі, у військових технологіях, в іграх, для моніторингу об'єктів і в мобільних пристроях.

Основа технології доповненої реальності – це система оптичного трекінгу. Це означає, що «очима» системи стає камера, а «руками» – маркери. Камера розпізнає маркери в реальному світі, «переносить» їх у віртуальне середовище, накладає один шар реальності на інший і таким чином створює світ доповненої реальності. Існують три основних напрямки в розвитку цієї технології:

- маркерна;
- безмаркерна;
- просторова;
- на основі накладання.

2.1.1 Маркерна доповнена реальність

AR технологія на базі маркерів вимагає статичного зображення, яке також називається тригерною фотографією, яку людина може сканувати за допомогою свого мобільного пристрою через додаток доповненої реальності. Мобільне сканування спричинить додатковий вміст (відео, анімація, 3D чи інший), підготовлений заздалегідь, щоб з’явитись у верхній частині маркера як показано на рисунку 2.1.

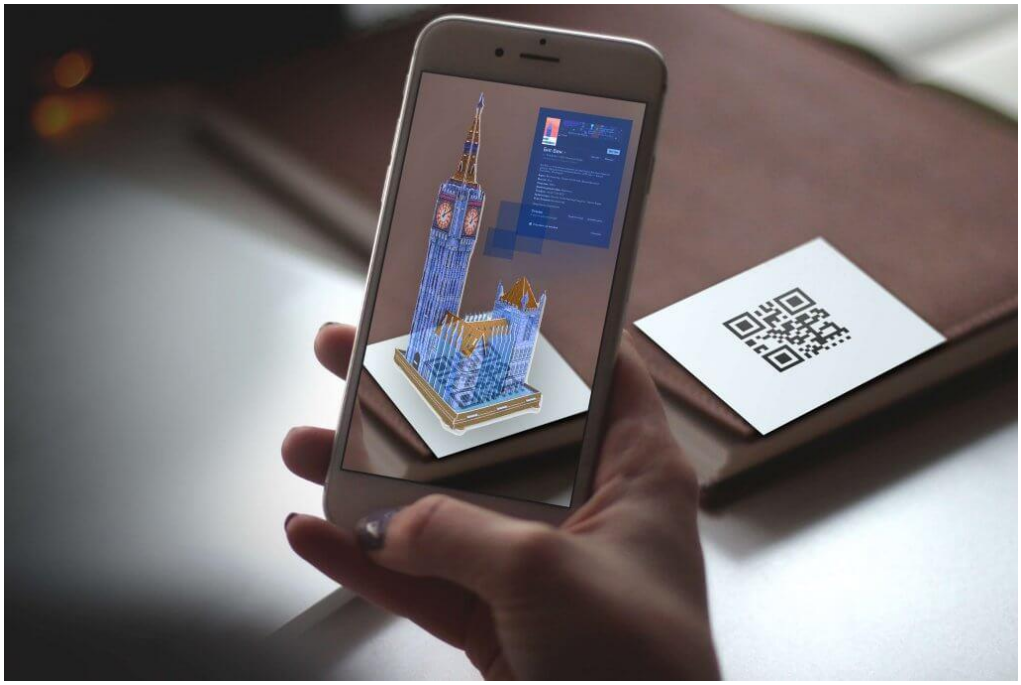


Рисунок 2.1 – Доповнена реальність на основі маркера

Розпізнавання маркерів може бути локальним або хмарним, це означає, що бази даних маркерів можуть зберігатися на пристрої, а розпізнавання також відбувається на пристрої. Бази даних також можна зберігати в хмарі, а розпізнавання відбувається на сервері, телефон надсилає на сервер лише хмари точок. Розпізнавання на основі пристрою може відбутися негайно, але якщо

використовується розпізнавання у хмарі, завантаження вмісту з сервера займе трохи більше часу. Зазвичай потрібно кілька секунд, перш ніж користувач може побачити будь-який досвід доповненої реальності. Така технологія набагато надійніше «безмаркерной» і працює практично без збоїв.

Переваги:

– якщо зображення маркера підготовлено правильно, вміст AR на основі маркера забезпечує якісний досвід, а відстеження дуже стабільне, вміст AR не тремтить;

– проста у використанні, детальна інструкція не потрібна людям, які користуються нею вперше.

Недоліки:

– коли мобільну камеру відсувають від маркера, досвід AR зникає, і фотографію тригера потрібно сканувати ще раз. Можна використовувати розширене відстеження, але в більшості випадків розширене відстеження погіршує ситуацію;

– сканування не спрацює, якщо маркери відбивають світло в певних ситуаціях (може бути складним завданням для широкоформатних банерів OD у постійно мінливих погодних умовах);

– маркер повинен мати чіткі межі / контраст між чорним і білим кольорами, щоб зробити відстеження більш стабільним. Плавний перехід кольору унеможливить розпізнавання.

2.1.2 Безмаркерна доповнена реальність

Безмаркерний AR об'єднує цифрові дані з входами з реальних входів реального світу, зареєстрованих у фізичному просторі. Технологія поєднує програмне забезпечення, аудіо та відео графіку з камерами смартфона або

гарнітури, гіроскопом, акселерометром, гаптичними датчиками та службами локації для реєстрації 3D графіки в реальному світі. Наведений нижче приклад на рисунку 2.2 показує, що AR без маркерів не потребує жодних фізичних маркерів для розміщення об’єктів у реальному просторі.

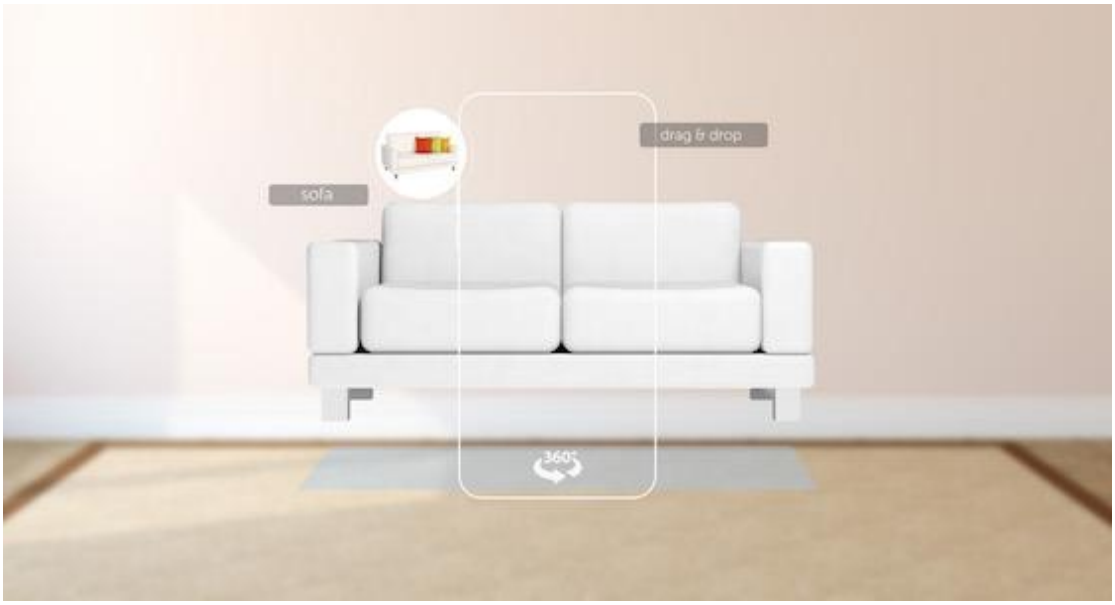


Рисунок 2.2 – Безмаркерна доповнена реальність

Безмаркерний AR виявляє предмети або характерні точки сцени без будь-якого попереднього знання про навколишнє середовище, наприклад стіни або точки перетину. Ця технологія часто пов’язана з візуальним ефектом, який поєднує комп’ютерну графіку та реальні образи. Перші безмаркерні системи використовували служби локації пристрою та апаратне забезпечення для взаємодії з доступними ресурсами AR та визначали його розташування та орієнтацію в просторі. Розробка технології одночасної локалізації та картографування (SLAM) покращила точність аналізу зображень AR без маркерів. Безмаркетове відстеження зображень SLAM сканує навколишнє середовище та створює карти місця розміщення віртуальних 3D-об’єктів. Навіть

якщо об'єкти не знаходяться в полі зору користувача, вони не рухаються, коли користувач рухається, і користувачеві не потрібно сканувати нові зображення.

Переваги:

– після розміщення вмісту в кімнаті він стає більш гнучким, ніж альтернативи, засновані на маркерах.

Недоліки:

– вміст доповненої реальності може не мати сенсу в певному контексті;

– для кращого досвіду потрібно, щоб поверхня мала текстуру для комп'ютерного зору, щоб її розпізнати.

2.1.3 Просторова доповнена реальність

AR на основі проекції описується як техніка відеопроекції, яка може розширити та посилити візуальні дані, кидаючи зображення на поверхню тривимірних об'єктів або простору; це належить просторовій доповненій реальності в широкому розумінні та зображен на рисунку 2.3. Використовуючи AR на основі проекції, легко реалізувати графічне зображення, яке звичайні методи освітлення не можуть виразити. На відміну від загальної техніки освітлення, ця техніка може проектувати зображення або відео високої чіткості та візуально змінювати форму об’єкта з плином часу. Тому він може динамічно відображати візуальні зображення. Ця комбінація зображень та реальних об’єктів дозволяє глядачам розпізнати візуально розширений простір. Крім того, на відміну від звичайних високозалежних AR, які обмежують тіло аудиторії, ця техніка може стати перевагою для поліпшення їх занурення. Ці переваги є вимогами до кращого досвіду глядачів у виконанні. У галузі медіа-мистецтва AR на основі проекції називається Projection Mapping, яке охоплює менше поле, ніж AR на основі проекції.

Проекційне відображення – це техніка, яка викликає оптичну ілюзію, аналізуючи тривимірний об'єкт, проектуючи зображення, а потім точно вирівнюючи їх. Ця техніка широко використовується в різних галузях, таких як фасад будівлі, пластикові об'єкти, а також у сфері виконавського мистецтва. У цих випадках він зазвичай проектує зображення на нерухомі об’єкти, використовуючи ручне вирівнювання між об’єктами та проектованими зображеннями.

Рисунок 2.3 – Просторова доповнена реальність

З іншого боку, було проведено багато останніх досліджень, які намагалися виконати проекцію на динамічні об'єкти з автоматичним вирівнюванням, але ці дослідження застосовні лише до обмежених форм та руху об'єктів, що вимагає величезних обчислень для вирівнювання зображень. Це спричиняє затримку, поки зображення не буде вирівняно з 3D-об’єктами. Чим більше латентність, тим

повільніше рух проектованого зображення після руху об'єкта, отже, це генерує візуальну помилку, що зменшує занурення аудиторій.

Виявлення взаємодії користувача здійснюється шляхом розрізнення очікуваних (або відомих) проекцій та модифікованих проекцій (викликаних взаємодією користувача). Проекційний метод побудови AR використовує фізичні об’єкти, такі як стіни, книги, гіпсові прикраси та будь-який комп’ютер, на який вміст може оптично проектуватися. Тобто проекція дозволяє використовувати реальний об’єкт як дисплей. Ми в основному зосереджуємося на зйомці та використанні тривимірної форми поверхні об’єкта. При поєднанні фізичних об’єктів та візуальних об’єктів інформація може змусити систему AR / MR враховувати візуальну узгодженість. Дані 3D-об’єкта можна використовувати для компенсації спотворень, спричинених різницею між положенням проектора та спостерігача. Нажаль, цей метод не є можливим для використання за допомогою мобільних пристроїв, тому його можна вважати неактуальним.

2.2 Існуючі інструменти доповненої реальності

Щороку ми отримуємо все більше і більше ознак того, наскільки великим буде ринок доповненої реальності протягом найближчих кількох років. Незалежно від того, чи є ці ознаки найманням персоналу від Apple AR, зростаючими потребами у сценаріях технічного обслуговування чи невизначеними натяками засновників Facebook, ми надихаємося на них.

За словами таких основних гравців, як одна з найпопулярніших інтерактивних платформ для розробки, Unity3D, дохід від AR перевищує доходи від VR. Враховуючи, що кожен сучасний смартфон за замовчуванням є AR-пристроєм, ці основні гравці, мабуть, мають рацію.

Індустрія розвитку AR є висококонкурентоспроможною з основними технологічними компаніями, що вкладають кошти у власні комплекти для розробки програмного забезпечення AR (SDK). Apple випустила свій ARKit ще в 2017 році, і лише через рік Google представив ARCore. Сьогодні існує чотири основних AR SDK: ARkit, ARCore, Vuforia та AR Foundation.

Принципи розвитку доповненої реальності в більшості випадків подібні до розвитку віртуальної реальності. Більшість принципів залишається незмінним, однак при AR потрібно брати до уваги рівняння того, що користувач бачить реальний світ разом із нав'язаними візуальними даними.

Як правило, ринок можна розділити на дві області розвитку AR-інструментів:

– фірмові інструменти, найкращими є Apple ARKit та Google ARCore;
– інструменти з відкритим кодом, особливо система OpenCV.

ARKit (Apple ARKit) – це платформа для розширеної реальності від Apple для мобільних пристроїв iOS. ARKit дозволяє розробникам створювати деталізовані AR-можливості для iPad та iPhone. Середовища, захоплені пристроєм, можуть додавати до них анімований віртуальний текст, предмети та символи. AR-сцени, зроблені однією особою, є постійними, і їх можуть побачити інші, хто відвідує місце згодом.

ARKit був представлений разом з iOS 11. Оскільки ARKit призначений для роботи на Core A9 і вище пристроях iOS, досвід AR може мати більш детальну інформацію та підтримувати кращу екологічну обізнаність. За допомогою iPhone X ARKit може виконувати сканування обличчя в режимі реального часу та використовувати ці дані для формування міміки 3D-персонажів.

Використовуючи камеру пристрою iOS, акселерометри, гіроскоп та усвідомлення контексту, ARKit виконує картографування середовища під час переміщення пристрою, приклад чого можна побачити на рисунку 2.4. Злиття

датчиків даних інерційного датчика з даними камери забезпечує дуже точне усвідомлення місцезнаходження та відображення. Програмне забезпечення виділяє візуальні особливості навколишнього середовища, такі як площини та відстежує рух у поєднанні з інформацією від інерційних датчиків. Камера також використовується для визначення джерел світла, за допомогою яких освітлюються об'єкти AR. Рішення Apple для збільшення деталізації, а отже, використання пам’яті – це розсувна карта, на якій старі дані зникають за новими. Користувачі можуть розміщувати якорі, щоб позначити творіння, які вони хочуть зберегти.

Рисунок 2.4 – Apple ARKit

ARCore: Google також розробив платформу для впровадження технології доповненої реальності для Android та iOS. Використовуючи API цього фреймворку, ARCore дозволяє вашому телефону переглядати своє оточення та взаємодіяти з зовнішньою інформацією.

ARCore працює на Java / OpenGL, Unity і Unreal і фокусується на таких напрямках:

– відстеження руху. Використовуючи камеру телефону для відстеження опорних точок в кімнаті (п.п. ці точки визначають місце, де буде розташований віртуальний об'єкт) і даних гіроскопа, ARCore визначає положення і орієнтацію пристрою під час руху. При цьому віртуальні об'єкти залишаються саме там, де ви їх розташували;

– розпізнавання навколишнього середовища. Зазвичай об'єкти доповненої реальності розміщуються на підлозі або столі. ARCore може розпізнавати горизонтальні поверхні, використовуючи ті ж опорні точки, що і при відстежуванні руху;

– оцінка освітлення. ARCore визначає рівень освітленості навколишнього середовища і дає можливість розробникам висвітлювати віртуальні об'єкти відповідно до обстановки навколо. Завдяки цьому вони виглядають ще більш реалістично.

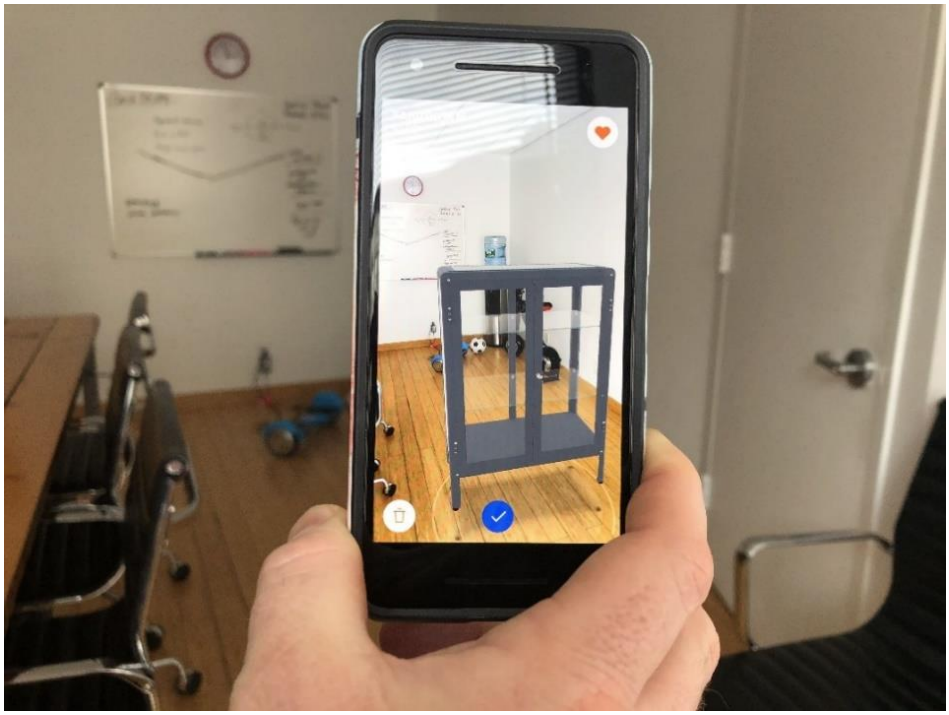


Рисунок 2.5 – Фреймворк від Google – ARKit

Коли ви переміщаєте свій телефон, ARCore запам'ятовує оточення і будує власний світ, в якому він може розміщувати віртуальні об'єкти. Він також використовує технологію відстеження руху для визначення того, як деякі об'єкти рухаються, враховуючи руху вашої камери.

Vuforia підтримує Android, iOS, UWP та Unity Editor. Цей набір інструментів для розробки AR від PTC – ще один хороший варіант для розробників додатків AR для відстеження заздалегідь визначених зображень, моделей, об’єктів або 3D-сканування. Vuforia може працювати як на iOS, так і на Android і навіть на старих моделях iPhone, з якими ARKit не сумісний. Крім того, Vuforia використовує ARKit або ARCore, коли апаратне забезпечення, на якому працює, підтримує це, інакше він може використовувати власну платформу. Приклад роботи фреймворку зображен на рисунку 2.6.

Ще однією суттєвою можливістю, яку пропонує Vuforia, є VuMark, яка є поєднанням зображення та QR-коду. Розробники можуть сканувати та створювати об’єктні цілі за допомогою Vuforia Object Scanner.

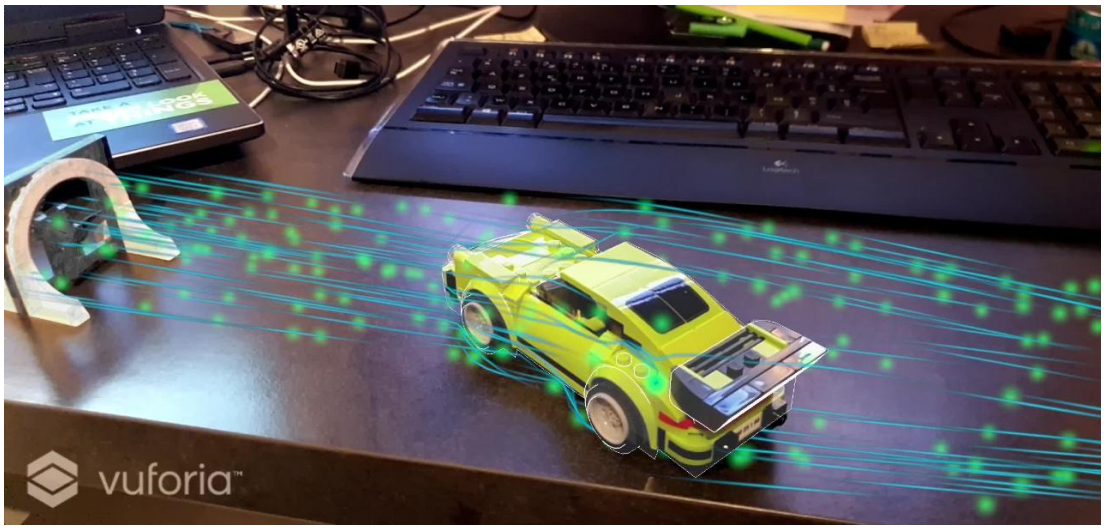


Рисунок 2.6 – Приклад доповненої реальності на основі фреймворку Vuforia

Хоча Vuforia має обмежені можливості в порівнянні з ARKit та ARCore, він пропонує велику перевагу над ARKit та ARCore, що значною мірою врівноважує обмеження Vuforia. Він був розроблений для використання переваг як ARCore, так і ARKit через Vuforia Engine. Оскільки Vuforia Engine використовує ARCore та ARKit (або навіть AR Foundation) для розширення можливостей в SDK, завдяки 30 Vuforia Engine 7.2 (або пізнішої версії), розробники можуть автоматично використовувати переваги як ARKit, так і ARCore у своєму проекті через бібліотеку Vuforia Engine.

Цей інструмент допомагає розпізнавати різні типи об’єктів, такі як коробки, циліндри та площини. Він також забезпечує розпізнавання тексту та середовища.

Основними можливостями Vuforia є відстеження об’єктів та зображень для розробки AR. Платформа Vuforia пропонує розпізнавання об’єктів із передбачених цілей для розпізнавання, що дозволяє розробникам завантажувати моделі, зображення, сканування об’єктів та інші типи цільової підтримки для виявлення.

3 МОДЕЛІ ТА АЛГОРИТМИ НЕЙРОННИХ МЕРЕЖ

3.1 Штучні нейронні мережі

Штучна нейронна мережа (Neural Network) – математична модель, що складається з сенсорів, асоціативних та реагуючих елементів. Вона ґрунтується на сукупності з'єднаних вузлів, що називають штучними нейронами. Кожне з'єднання (аналогічне синапсові) між штучними нейронами може передавати сигнал.

Нейронні мережі [9] представляють один із напрямів обчислювального інтелекту – потужний інструмент для вирішення широкого спектру завдань інтелектуального аналізу даних. Важливою особливістю нейронних мереж є їх здатність до навчання чи самонавчання. Під процесом навчання розуміють адаптацію параметрів та архітектури мережі для розв’язання поставленої задачі шляхом оптимізації обраного критерію якості. У наш час існує безліч типів нейронних мереж, що дозволяють постійно «донавчатися» на даних, обробляти великі об’єми даних у потоковому режимі та робити налаштування синаптичних ваг за короткі проміжки часу.

Нейронні мережі використовуються для вирішення складних задач, що вимагають аналітичних обчислень подібних тим, що робить людський мозок та зводяться до апроксимації багатовимірних функцій, тобто побудови відображення $F : x \rightarrow y$. Вони допомагають групувати непомічені (unlabeled) дані відповідно до схожості між прикладами вхідних даних, а у випадку застосування порогових активаційних функцій вихідні сигнали ШНМ мають дискретний характер, а задачі, що при цьому розв’язуються – є задачами класифікації. Нейронні мережі можуть також витягувати особливості, які надходять до інших алгоритмів кластеризації та класифікації. Тому ви можете розглядати глибокі

нейронні мережі, як компоненти великих машино-навчальних додатків, що використовують алгоритми навчання, класифікації та регресії. Одним із законів машинного навчання є: чим більше даних тренувалося алгоритмом, тим точніше він буде. Таким чином, навчання без нагляду має потенціал для створення високоточних моделей.

Найпоширеніші задачі навчання по прецедентах є: класифікація, кластерний аналіз та навчання з підкріпленням, розглянемо їх детальніше.

Класифікація [10] – тип навчання з вчителем, коли дані діляться за мірою схожості та подібності, що з технічної точки зору є розподілом вектора на заздалегідь відому кількість класів. Вона відповідає за розподіл даних по параметрах. Всі задачі залежать від позначених наборів даних (labeled datasets), тобто люди повинні передати свої знання набору даних для того, щоб нейронна мережа дізналася кореляцію між мітками та даними. Передбачення має можливість прогнозувати наступний крок. Наприклад, зростання або падіння акцій, спираючись на ситуацію фондового ринку. Розпізнавання – найпоширініше застосування нейронних мереж у наш час, адже коли ви перекладаєте на іншу мову якесь речення у Google Translate чи шукаєте схожу фотографію через Google Search, ви переглядаєте результат роботи натренованих моделей.

Основні задачі, що вирішує класифікація:

– виявлення обличчя, ідентифікація людей у зображеннях, розпізнавання міміку (сердитий, радісний);

– визначення об'єкти на зображеннях (знаки зупинки, пішоходи, маркери доріжок та ін.);

– виявлення голоси, ідентифікація говорючого, перетворення звукового запису до тексту, розпізнавання почуття голосами;

– класифікація тексту як спаму (електронною поштою).

Кластеризація (або кластерний аналіз) – це тип методу навчання без вчителя, де існує міра подібності та відмінності у даних датасету, внаслідок чого можна визначити, до якого кластеру відноситься об’єкт з вибірки. Усередині кожного кластеру повинні знаходитися «схожі» об’єкти, а об’єкти різних груп повинні бути якомога більш відмінні. Головною відмінністю кластеризації від класифікації є те, що перелік груп чітко не заданий і визначається у процесі роботи алгоритмів

Під час пошуку, було виділено дві основні класифікації алгоритмів кластеризації.

Ієрархічні та плоскі. Ієрархічні алгоритми (також відомі як алгоритми таксономії) будують не одне розбиття вибірки на кластери, що не перетинаються, а систему вкладеності кластерів один в одного, та будується на основі відстані між вузлами, тобто на виході можна отримати дерево кластерів, коренем якого є вся вибірка, а листям – найбільш дрібні кластери (формула ієрархічного алгоритму 3.1). Плоскі алгоритми (формула 3.2) будують одне розбиття об'єктів на кластери [11];

$$X^k(W) = \{W_1^{(I)}, \dots, W_{k^{(I)}}^{(I)}\}_{I=1}^{L}$$

$$W_1, W_2, \dots, W_{i-1}, W_i, \dots, W_{j-1}W_j, \dots, W_{I-1}, W_I, \dots, W_{s-1}, W_s, \dots, W_{N-1}, W_N$$

$$W_1^{(1)}, \dots, W_i^{(1)}, \dots, W_j^{(1)}, \dots, W_s^{(1)}, \dots, W_{k^{(1)}}^{(1)}$$

$$W_1^{(2)}, \dots, W_i^{(2)}, \dots, W_j^{(2)}, \dots, W_{k^{(2)}}^{(2)} \qquad (3.1)$$

$$\dots$$

$$W_1^{(1)}$$

$$N = k^{(0)} > k^{(1)} > \dots > k^{(L)} = 1$$

$$\bigcup_{i=1}^{k^{(I)}} W_1^{(I)} = \ W \ \ \forall 1 \leq I \leq L$$

$$X^k\ (W)\ =\ \{W_1\ ,\ldots,W_k\ \} \tag{3.2}$$

Чіткі і нечіткі. Чіткі (або непересічні, формула 3.3) алгоритми кожному об'єкту вибірки ставлять у відповідність номер кластера, тобто кожен об'єкт належить тільки одному кластеру. Нечіткі (або пересічні, формула 3.4) алгоритми кожному об'єкту ставлять у відповідність набір речових значень, що показують ступінь відношення об'єкта до кластерів, тобто кожен об'єкт належить до кожного кластеру з певною ймовірністю [12].

$$\tau_{\chi^k}(w)\ =\ i\ :\ w\ \in\ W_i \tag{3.3}$$

$$\tau_{fuzzy\chi^k}(w) =\ f\ \in R^k\colon f_i\ \geq 0, \sum_i\ f_i \tag{3.4}$$

Навчання з підкріпленням належить до області машинного навчання, що дозволяє агенту вчитися в інтерактивному середовищі методом проб і помилок, використовуючи зворотний зв'язок від власних дій і досвіду. Це дає гарне поступове навчання і може спростити навчання агента в задачах, де ви не можете визначити правильне значення помилки. Наприклад: боту дається завдання грати в Space Invaders, він намагається навчитися грати, взаємодіючи з грою і отримуючи нагороду за бали, які він набрав у кінці гри. Більше нагороди, більші його шанси зробити подібний геймплей. Таким чином, він дізнається, як грати в гру і діяти найкращим чином. У промисловості робот використовує глибоке навчання підкріплення, щоб вибрати пристрій з однієї коробки і покласти його в контейнер. Незалежно від того, чи це вдасться або не вдасться, він запам'ятовує

об'єкт і отримує знання, а також тренує себе, щоб робити цю роботу з великою швидкістю і точністю. Самостійне навчання є своєрідним підкріплюючим навчанням за умови, що навчання знаходиться в позитивному вимірі. Не буде зайвим згадати про процеси прийняття рішень Маркова, що являють собою математичні рамки для опису середовища в навчанні підкріплення, і майже всі проблеми цього типу навчання можуть бути формалізовані за допомогою MDP. Складаються з множини станів S кінцевих середовищ, набору можливих дій A(s) в кожному стані, функції нагородження R (s) і моделі переходу P (s ', s | a).

Навчання з підкріпленням разом з його фундаментальними поняттями необхідно розглянути практично. На рисунку 3.1 представлена основна ідея та елементи, залучені до цієї моделі [13].

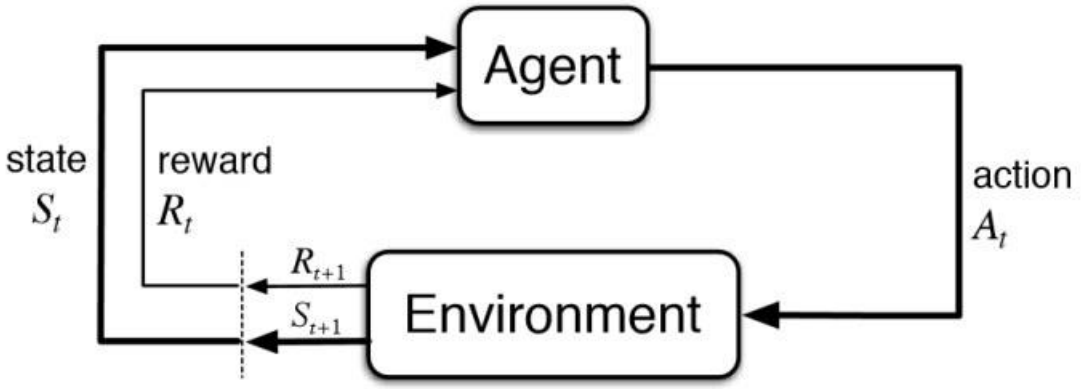


Рисунок 3.1 – Основна ідея навчання з підкріпленням

В даному типі навчання використовуються наступні алгоритми.

Q-навчання (Q-Learning) – є найбільш використовуваним алгоритмом навчання підкріплення [14]. Застосовуючи цей алгоритм, агент дізнається якість (значення Q) кожної дії, виходячи з того, з якою нагородою повертається у

середовище. Він використовує таблицю для зберігання значення стану кожного середовища разом із значенням Q;

SARSA (State-Action-Reward-State-Action) – нагадує Q-навчання, але єдиною відмінністю між ними полягає у тому, що SARSA дізнається значення Q на основі дії, що виконується поточною політикою у порівнянні з способом використання Q-навчання з використаними жадної політики. Розглянувши приклад собаки, візьмемо за приклад власника собаки і «собаку» (агент). Тепер, коли власник собаки присутній в саду з собакою, він викидає м'яч. Викидання м'яча – «стан» для агента, і тепер собака буде бігати за м'ячем, який буде «дією». Результатом буде вдячність або харчування для собаки від власника, який буде «винагородою» в результаті дії, і якщо собака не піде після м'яча черговою альтернативною дією, то вона може отримати деяке «покарання». Таким чином, це саме те, про що йдеться про навчання з підкріпленням.

S (State) – означає стан. Держава – це безпосередня ситуація, в якій агенти опиняються по відношенню до інших важливих речей в оточенні, таких як інструменти, перешкоди, вороги і призи чи нагороди.

A (Action) – агент має набір дій A, з яких він вибирає, яку дію виконувати. Так само, як і собака, що вирішила піти після м'яча, просто подивіться на м'яч або стрибайте на позицію.

R (Reward) – винагорода – це результат, що отримує агент у відповідь на дії агента. Наприклад, собака отримує корм для собак як нагороду , якщо собака (агент) повертає м'яч, інакше отримує лайку як покарання, якщо вона не бажає цього робити.

Тепер, щоб зрозуміти, як же працюють нейронні мережі, давайте поглянемо на їх складові та параметри.

Нейрон – це обчислювальна одиниця, яка отримує інформацію, виробляє над нею прості обчислення і передає її далі [15]. Вони діляться на три основних

типи: вхідний (синій), прихований (червоний) і вихідний (зелений) та зображені на рисунку 3.2. У тому випадку, коли нейронна мережа складається з великої кількості нейронів, вводять наступний термін шар. Відповідно, є вхідний шар, який отримує інформацію, n-прихованих шарів (зазвичай їх не більше 3), які її обробляють і вихідний шар, який виводить результат. У кожного з нейронів є 2 основні параметри: вхідні дані (input data) і вихідні дані (output data). У разі вхідного нейрона: input = output. В інших, в поле input потрапляє сумарна інформація всіх нейронів з попереднього шару, після чого, вона нормалізується, за допомогою функції.

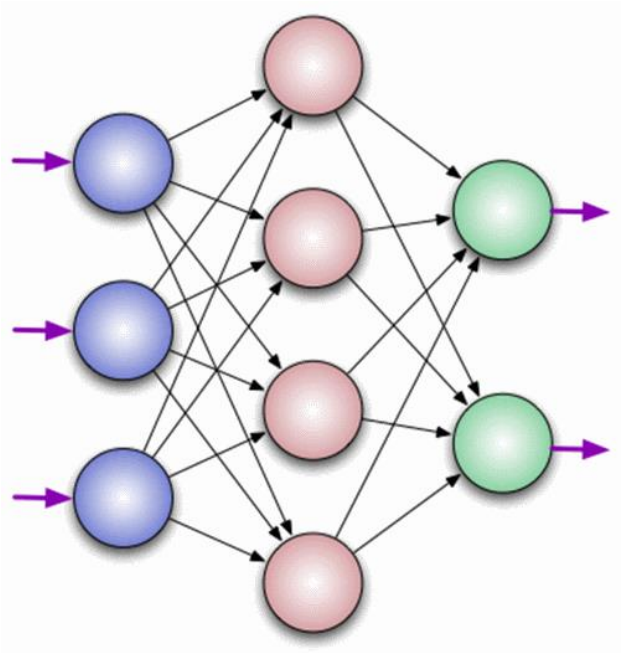


Рисунок 3.2 – Логічна схема елементарного перцептрону

Тепер поговоримо детальніше про прості типи нейронів, з яких складається перцептрон [16]:

– простим S-елементом (синій нейрон) являється чутливим, або, як ще його називають – сенсорним. Він виробляє сигнал від будь-якого з видів енергії

(тиск, тепло, світло, звук). Якщо вхідний сигнал перевищує певний поріг θ, то на виході елемента отримуємо +1, в усіх інших випадках – 0;

– простим А-елементом (червоний нейрон) є елемент (асоціативний), який дає на вихідний сигнал +1, коли алгебраїчна сума його вхідних сигналів перевищує або дорівнює деяку граничну величину θ, в інших випадках – вихід є 0;

– простим R-елементом (зелений нейрон) називається реагуючим елементом та видає сигнал +1, якщо сума його вхідних сигналів є суворо додатною, а також –1, якщо сума його вхідних сигналів являється виключно від'ємною. Вихід вважається рівним нулю, або невизначеним, якщо сума вхідних сигналів дорівнює нулю.

Також, якщо в будь-якому елементі отримати 0 на виході, тоді елемент або збуджений, або активний.

Важливо пам'ятати, що нейрони оперують числами в діапазоні [0,1] або [-1,1]. На даному етапі ділять 1 на числа, що виходять з даного діапазону. Цей процес називається нормалізацією, і він дуже часто використовується в нейронних мережах.

Синапс являє собою зв'язок між двома нейронами (рисунок 3.3). У синапсів є 1 параметр – вага. Завдяки йому, вхідна інформація змінюється, коли передається від одного нейрона до іншого. Припустимо, є 3 нейрона, які передають інформацію з наступних підстав. Тоді у нас є 3 ваги, відповідні кожному з цих нейронів. У того нейрона, у якого вага буде більше – та інформація і буде домінуючою в наступному нейроні (приклад – змішання кольорів). Насправді, сукупність ваг нейронної мережі або матриця ваг – це своєрідний мозок всієї системи. Саме завдяки цим вагам, вхідна інформація обробляється і перетворюється в результат. Також важливо пам'ятати, що під час ініціалізації нейронної мережі, ваги розставляються в випадковому порядку.

У прикладі (рисунок 3.4) зображена частина нейронної мережі, де буквами I позначені вхідні нейрони, буквою H – прихований нейрон, а буквою w – ваги. З формули видно, що вхідна інформація – це сума всіх вхідних даних, помножених на відповідні їм ваги. Тоді дамо на вхід 1 і 0. Нехай w1 = 0.4 і w2 = 0.7 Вхідні дані нейрона H1 будуть наступними: 1 * 0.4 0 * 0.7 = 0.4. Отже, коли у нас є вхідні дані, можна отримати вихідні дані, підставивши вхідні значення в функцію активації.

Тепер, коли у нас є вихідні дані, їх треба передати далі. І так, це повторюється для всіх верств, поки не дійдемо до вихідного нейрона. Запустивши таку мережу в перший раз можна побачити, що відповідь далека від правильної, тому що мережа не натренована. Щоб поліпшити результати, її треба тренувати.

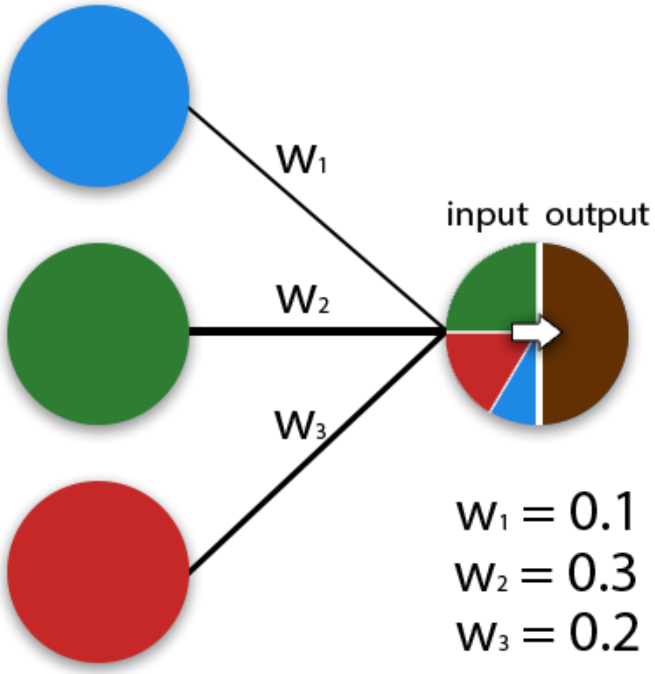
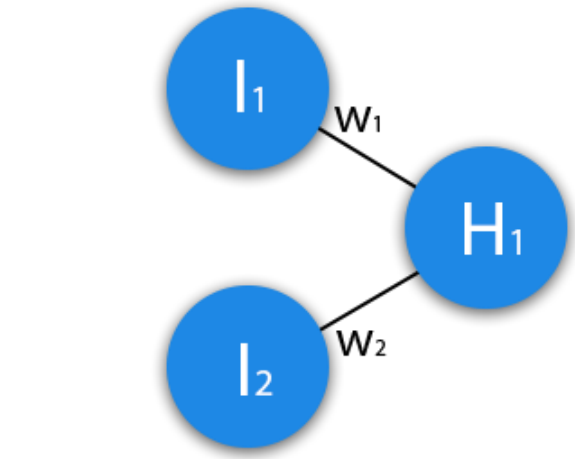


Рисунок 3.3 – Логічна схема синапсу зі зв’язками та вагою кожного нейрону



$$1)H_{1_{input}} = (I_1 * w_1) + (I_2 * w_2)$$

$$2)H_{1_{output}} = f_{activation}(H_{1_{input}})$$

Рисунок 3.4 – Схема частини нейронної мережі з вхідним та прихованим нейроном

Одним з найважливіших аспектів глибокої нейронної мережі є функція активації (activation function), яка привносить в мережу нелінійність. Тобто, якщо на вході буде велике число, пропустивши його через функцію активації, можна отримати вихід у потрібному діапазоні. Далі будуть розглянуті найпоширеніші функції активації: лінійна, сигмоїд та гіперболічний тангенс. Головні їх відмінності – це діапазон значень.

Лінійна функція (рисунок 3.5) – застосовується для тих моделей мереж, де не потрібне послідовне з'єднання шарів нейронів один за одним. Ця функція майже ніколи не використовується, за винятком випадків, коли потрібно протестувати нейронну мережу або передати значення без перетворень [17].

$$f(x) = x$$

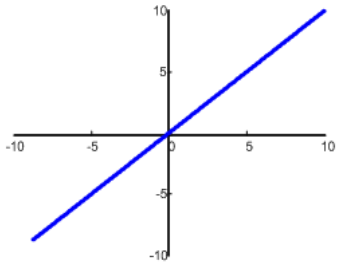


Рисунок 3.5 – Лінійна функція активації

Сигмоїдальна функція активації (рисунок 3.6) – використовується для багатошарових перцептронів та інших мереж з безупинними сигналами, а її діапазон значень варіюється від нуля до одного. Гладкість, безперервність функції – важливі позитивні якості. Безперервність першої похідної дозволяє навчати мережу градієнтними методами (наприклад, метод зворотнього поширення помилки).

$$f(x) = \frac{1}{1+e^{-x}}$$

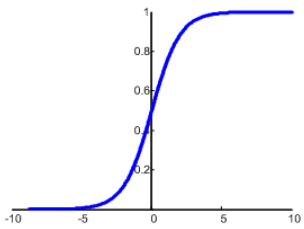


Рисунок 3.6 – Сигмоїдальна функція активації

Гіперболічний тангенс (рисунок 3.7) – функція, що приймає на вході довільне дійсне число, а на виході дає дійсне число в інтервалі від -1 до 1. Подібно сигмоид, гіперболічний тангенс може насичуватися. Використовувати цю функцію тільки з позитивними значеннями не має сенсу, так як це значно погіршить результати вашої нейромережі.

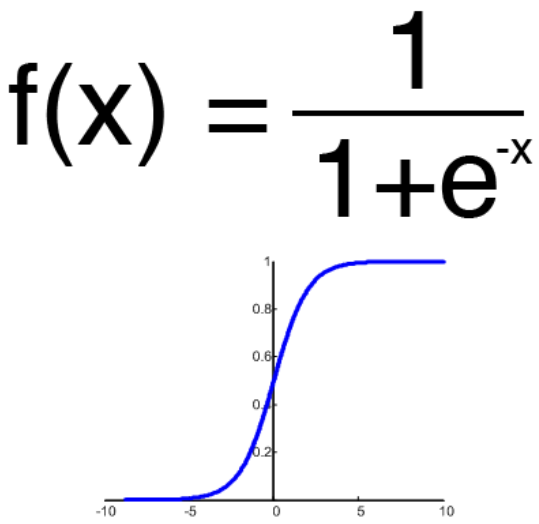


Рисунок 3.7 – Функція гіперболічного тангенсу

ReLu (рисунок 3.8) – випрямлена функція активації, що має область допустимих значень [0, inf) та виражається за такою формулою (формула 3.5).

$$f(s) = \max(0, s) \tag{3.5}$$

Подібно сигмоїдальної або тангенційної функції, що являються нелінійними, призводять до проблем з затуханням або збільшенням градієнтів. Це означає, що майже всі активації повинні бути оброблені для опису виходу мережі.

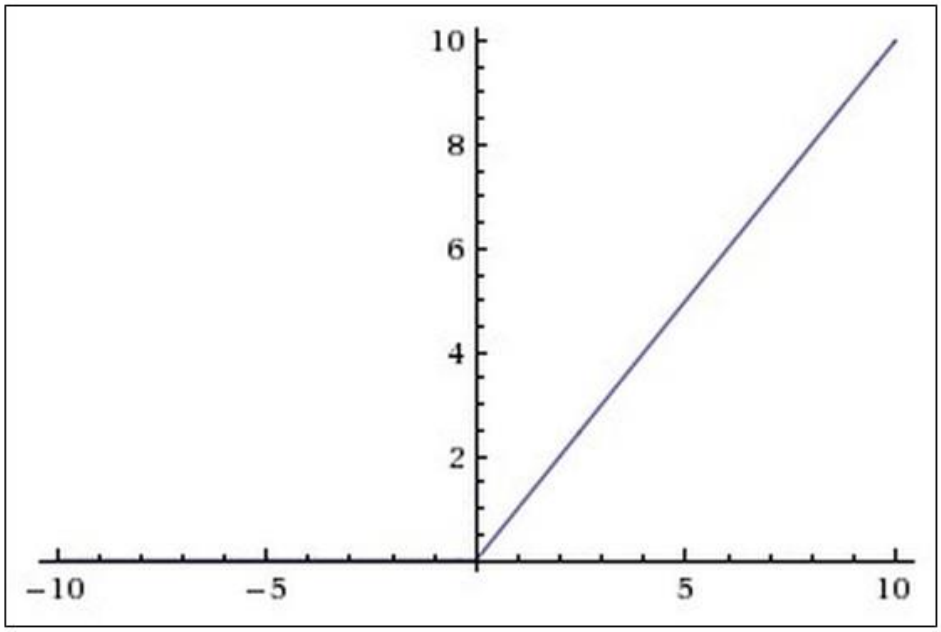


Рисунок 3.8 – Функція активації ReLu

Переваги використання ReLu:

– похідна у функції активаціїї ReLu дорівнює або одиниці, або нулю, і тому не може статися розростання або загасання градієнтів, тому що помноживши одиницю на дельту помилки буде отримано дельту помилки. Якщо була використана інша функція, наприклад, гіперболічний тангенс, то дельта помилки могла, або зменшитися, або зрости, або залишитися такою ж, тобто, похідна гіперболічного тангенса повертає число з різним знаком і величиною, що можна сильно вплинути на загасання або розростання градієнта. Більш того, використання даної функції приводить до проріджування ваг;

– обчислення сигмоїдної та гіперболічного тангенса функцій вимагає виконання ресурсномістких операцій, таких як спорудження до рівня, в той час як ReLu може бути реалізований за допомогою простого порогового перетворення матриці активацій в нулі;

– відсікає непотрібні деталі в каналі при негативному виході.

При побудові моделі та підготовці нейронної мережі, вибір функцій активації є критичним. Експеримент з різними функціями активації для різних завдань дозволить досягти найкращих результатів, а для цього потрібно постійно переходити з однієї на іншу.

3.2 Згорткові нейронні мережі

Згорткові нейронні мережі (CNN, Convolutional Neural Networks) – це алгоритм глибокого навчання, який може приймати вхідне зображення, призначати важливість (навчальні ваги і упередження) різним аспектам / об'єктам зображення і вміти диференціювати один з іншого [18]. Якщо розглянути з математичної точки зору, то згортка – це інтеграл, який виміряє наскільки дві функції перекривають одна одну, коли одна з них проходить над іншою. У CNN значно менше необхідна попередня обробка, порівняно з іншими алгоритмами класифікації. Вони використовуються, в першу чергу, для класифікації зображень (наприклад, ім'я того, що вони бачать), сортування їх за подібністю (пошук фотографій), а також для розпізнавання об'єктів у сценах. Це алгоритми, які дозволяють ідентифікувати обличчя, індивідуумів, знаки вулиці, пухлини, кабачки і багато інших аспектів візуальних даних.

Convolutional Neural Networks виконують оптичне розпізнавання символів для оцифровування тексту і роблять можливу обробку природних мов на аналогових і рукописних документах, де зображення є символами, які необхідно переписати. Вони також можуть бути застосовані до звуку, коли воно візуально представлено як спектрограма. Ефективність цих мереж у розпізнаванні образів є однією з головних причин, чому світ приділяє пильну увагу до ефективності глибокого навчання. Вони забезпечують великі досягнення у комп'ютерному зорі (Computer Vision), який має зрозумілі програми для самостійного водіння

автомобілів, робототехніки, безпілотних літальних апаратів, безпеки, медичних діагнозів і методів лікування для людей з вадами зору.

Перший шар завжди згортковий. Головною роллю є об’єднання зображення у форму – матрицю пікселів, що є окремим кроком від згортки.

Якщо зображення забарвлене, то вважається, що воно має ще один вимір кольору RGB. З цієї причини 2D згортки зазвичай використовуються для чорно-білих, у той час як 3D – для кольорових зображень.

Зображення зі стандартної цифрової камери буде мати три канали – червоний, зелений і синій. Ви можете уявити їх, як три 2d-матриці, розташовані одна над одній (по одному для кожного кольору), в кожній з яких піксель змінює своє значення у діапазоні від 0 до 255. Нуль вказує на чорний, а 255 на білий. Особливістю детектора – є матриця, зазвичай 3x3 (також вона може бути 7x7). Детектор ознак також називається ядром або фільтром, а області, над якими виконується дія – називаються рецептивних полем (полем сприйняття)

Кожен фільтр можна розглядати як ідентифікатор властивості, що можуть бути прямими кордону, прості кольори чи криві. Перший фільтр 7 x 7 x 3, і він буде детектором кривих [19].. У фільтра піксельна структура, в якій чисельні значення вище уздовж області, яка визначає форму кривої (рисунок 3.9).

Об'єднання використовується для зменшення розміру зображення ширини і висоти. Зауважимо, що глибина визначається кількістю каналів. Як випливає з назви, все це робить вибір максимального значення в певному розмірі вікна, яку набагато легше обробити та знайти, які з цих чисел є важливими сигналами, не втрачаючи особливості, що є необхідною умовою для отримання точного прогнозу.

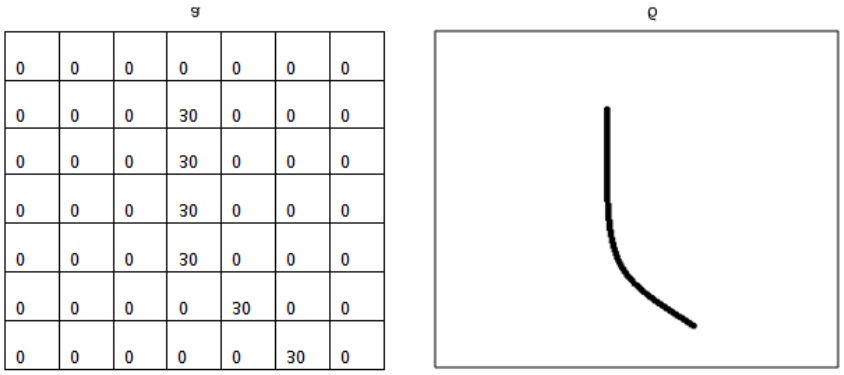


Рисунок 3.9 – Представлення фільтрів: а) піксельне представлення фільтра; б) візуальна форма кривої на зображенні

Наступний шар у згортковій мережі має дві назви – максимальне (max pooling) та середнє (average pooling) об’єднання:

Максимальне об’єднання (max-pooling) використовується для зменшення розміру зображення шляхом зіставлення розміру даного вікна в один результат, приймаючи максимальне значення елементів у матриці (рисунок 3.10).

Середнє об’єднання (average-pooling) – має той самий підхід, що і максимальне, за винятком того, що даний шар усереднює вікна, а не вибирає максимальне значення (рисунок 3.11).

Рисунок 3.10 – Схема зменшення розміру зображення за максимальним значенням

Рисунок 3.11 – Схема зменшення розміру зображення за середнім значенням

На рисунку відображається операція на просторі пікселів. Як альтернативу – ми можемо зробити подібну операцію на деякому іншому математичному просторі. Крім того, можна змінити операцію взяття «max-pooling» на щось інше, скажімо, прийнявши «average-pooling» [20].

Зображення максимального об'єднання здійснюється для областей, що не перекриваються. Це іноді призводить до гіпотези, що максимальне об'єднання об'єктів зазвичай виконується без перекриттів. Майже у всіх відомих архітектурах CNN максимальне об'єднання об'єктів було виконано з регіонами, що не перекриваються: AlexNet, GoogleNet, VGG16, MobileNet, ResNet50, InceptionV3 та SqueezeNet. Деякі з них буде розглянуто пізніше при використанні моделей у проекті. Таким чином, було показано на рисунку всі максимальні варіанти об'єднання у відомих архітектурах CNN.

Треба зауважити, що об'єднання є окремим кроком від згортки. Об'єднання використовується для зменшення розміру ширини і висоти. Глибина визначається кількістю каналів. Як випливає з назви, все це робить вибір максимального значення в певному розмірі вікна. Хоча це зазвичай застосовується просторово для зменшення x- та y- координат розмірів зображення.

Ця особливість надає можливість створити архітектуру, що не тільки володіє можливостями навчання на високому рівні, а й є масштабованою для великих наборів даних. У цьому процесі слід виділити три важливих пункти: вхідне зображення, тобто виявлене, детектор об'єктів і мапа характеристик. Кожен шар називається каналом, який використовується для позначення певних особливостей зображення, наприклад, лінія або крива, що створюють карти. Воно являє собою двовимірну матрицю з шириною і висотою.

Матричне подання вхідного зображення множиться за допомогою детектора ознак, для створення карти об'єктів, також відомої як згорнута функція або карта активації. Метою цього кроку є зменшення розміру зображення, швидкість та легкість обробки. На цьому кроці втрачаються деякі його особливості. Проте, зберігаються основні особливості, які є важливими для його виявлення. Ці особливості є унікальними для ідентифікації конкретного об'єкта. Наприклад, кожна тварина має унікальні особливості, які дозволяють нам ідентифікувати її. Спосіб запобігання втрати інформації полягає в тому, що існує багато карт. Кожна карта особливостей виявляє розташування певних функцій у зображенні (рисунок 3.12).

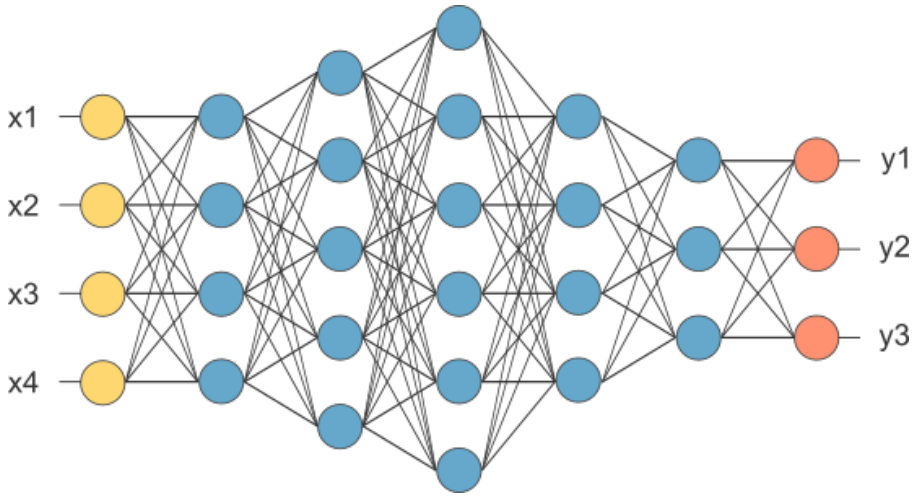


Рисунок 3.12 – Схема повністю пов’язаного шару

Повністю пов’язаний шар – шар, який вирівнює нашу матрицю у вектор і подається в нього як нейронна мережа. На наведеній схемі матриця карти об'єктів буде перетворена у вектор (x1, x2, x3,…). Завдяки повністю підключеним шарам, він об'єднує ці функції разом для створення моделі. В результаті маємо функцію активації для класифікації виходів таких об’єктів як: кішки, собаки, автомобіля, вантажівки і та інших. Для багатьох застосувань тренувальних даних доступно мало, а згорткові нейронні мережі, зазвичай, вимагають великої кількості тренувальних даних, щоби запобігати перенавчанню. Поширеною методикою є тренувати мережу на ширшому наборі даних з пов'язаної області визначення.

3.3 Фреймворк Core ML та його особливості

Для розробників iOS, найпростішим способом реалізації програми з функцією виявлення об'єктів є використання Core ML 2 – фреймворк машинного навчання. Він використовує TensorFlow модель, яка в свою чергу конвертується у розишерння .mlmodel (рисунок 3.13).

Core ML – це фреймворк від Apple, що може бути використаний для інтеграції моделей машинного навчання у мобільний додаток чи програму. Apple представила пакет з відкритим вихідним кодом Python, що називається coremltools, для перетворення навчених мереж у формат файлу моделі ML. Пакет підтримує Keras, Torch, Caffe, scikit-learn та Create ML за замовчуванням.

Core ML є основою для конкретних областей та функціональних можливостей, він оптимізований для роботи на пристрої, що мінімізує кількість пам'яті та енергоспоживання. Виконання суворої роботи на пристрої забезпечує конфіденційність даних користувача і гарантує, що програма залишиться функціональною та чутливою, коли підключення до мережі недоступне. Даний

фреймворк підтримує Vision API для аналізу зображень, Natural Language для обробки природних мов і GameplayKit для оцінки отриманих рішень.

Додатковою перевагою TensorFlow є велика бібліотека вільних і відкритих шаблонів моделей машинного навчання і API. У порівнянні з Core ML, який є скоріше непрозорим, TensorFlow не приховує інформацію про використані моделі.

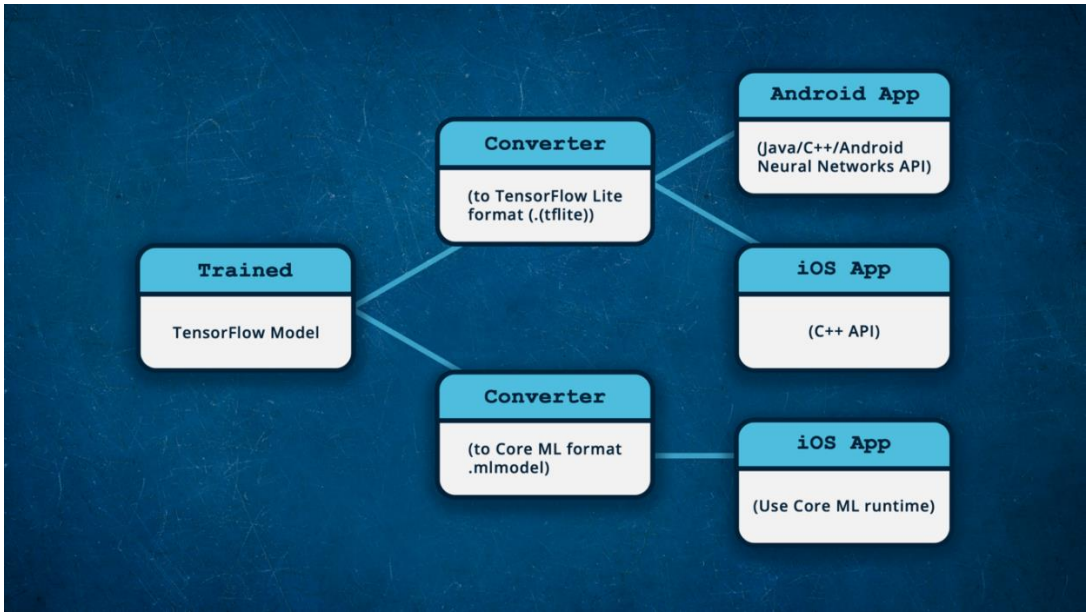


Рисунок 3.13 – Схема сумісності моделей TensorFlow

TensorFlow Lite – це крос-платформенний фреймворк та має підтримуку Android. Це означає, що обраний шаблон або спеціально створена попередньо підготовлена модель може бути інтегрована в додаток для обох платформ. Це дуже важливо, тому що зазвичай паралельно для клієнта розробляються мобільні програми для Android і iOS.

Підготовлена модель є результатом застосування алгоритму глибинного машинного навчання до набору навчальних даних. Для розпізнавання зображень використовується клас глибинних згорткових нейронних мереж. Модель робить

прогнози на основі нових вхідних даних. Наприклад, модель, яка пройшла навчання з історичних цін на житло в регіоні, може спрогнозувати ціну будинку, враховуючи кількість спалень і ванних кімнат.

Фреймворк побудований на базі примітивів низького рівня: MPSCNN та BNNS.

BNNS – Basic Neural Network Subroutines, є частиною фреймворку Accelerate, – це колекція математичних функцій, які повністю використовують швидкі векторні інструкції процесора. BNNS приймає вказівку (pointer) на буфер значень з плаваючою комою (рисунок 3.14). Він має такі особливості:

– обробка виконується за допомогою CPU;
– API на основі мови програмування C;
– має швидку роботу.

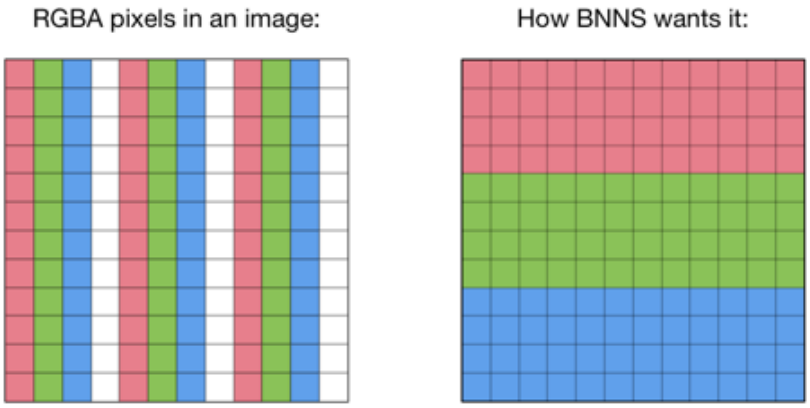


Рисунок 3.14 – Схема розташування зображення з каналами RGB

MPSCNN – це частина Metal Performance Shaders (MPS), що являє собою бібліотеку оптимізованих обчислювальних ядер, який працює на GPU, а не на CPU. Metal Performance Shaders являє собою набір класів, які дозволяють розкрити потужність фреймворку Metal без необхідності писати будь-який код для GPU, або багато знати про Metal. Можна створювати нейронні мережі за

допомогою цього API на основі графів. Нові шари додаються до нейронної мережі, а потім MPS робить все інше. MPSCNN вимагає, щоб усі дані розміщувалися в спеціальному об'єкті MPSImage, який є колекцією 2D текстур (рисунок 3.15). Його головними рисами є:

– обробка виконується за допомогою GPU;
– адаптивний для мови програмування Swift;
– дозволяє створити власну функцію активації.

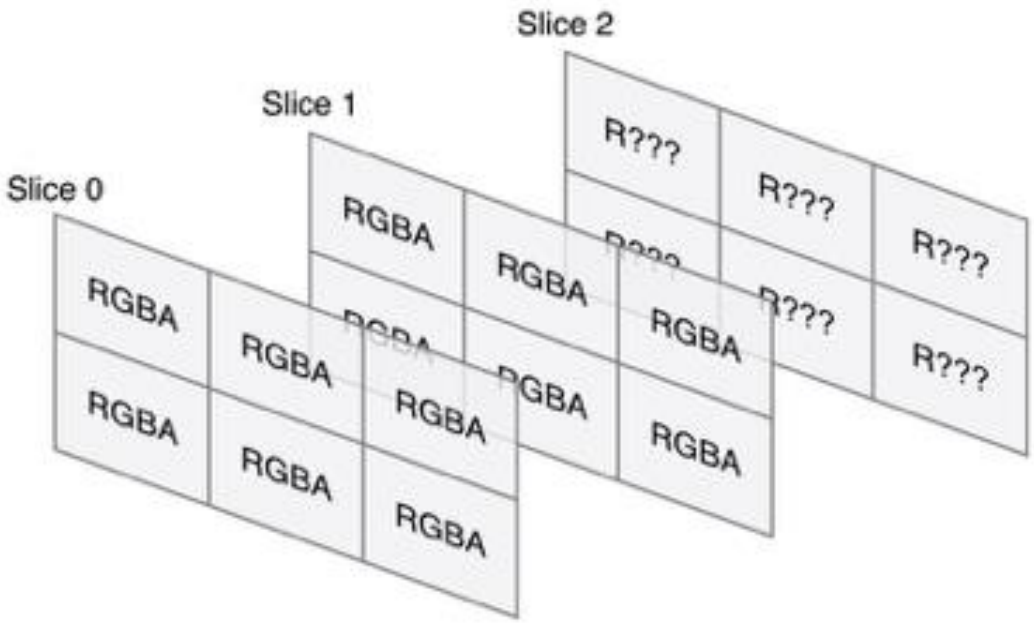


Рисунок 3.15 – Схема розташування RGBA зображення CNN з шириною 3 і висотою 2

Плюси API графу є:

– якщо програма намалює зображення на основі вихідних даних штучної моделі машинного навчання, то можна легко інтегрувати машинне навчання з вашою комунікаційною лінією візуалізації Metal і запустити все це на GPU;

– коли Core ML запускає моделі на GPU, вона фактично використовує MPS, але приховує це. Безпосередньо використовуючи MPS, з’являється більше контролю над тим, що відбувається, ніж з Core ML;

З недоліків маємо:

– потрібно знати трохи про метал, перш ніж цей API буде мати сенс для розробника. Наприклад, входи та виходи для вашої моделі повинні бути об'єктами MPSImage або MPSMatrix, і розробник повинен розуміти, як конвертувати дані в ці об'єкти. Не потрібно бути експертом з програмування GPU, але й не потрібно боятися цієї ідеї;

– не має можливості створювати власні типи вузлів для додавання до графу.

Тому, якщо є бажання підтримувати тип шару, якого в MPS не має, використовувати API граф MPS не можна. Через це, API не є значною мірою оновленням над Core ML – якщо фреймворк не підтримує готову навчену модель, то й MPS також.

Ідея API графу MPS гарна, але на практиці це не відчувається. Світ глибокого навчання рухається швидко. Використовуючи ці API, розробники обмежуються операціями, які компанія Apple вирішила реалізувати. Якщо на наступному тижні новий документ вийде з новим типом шару або новою функцією активації, то змоги до реалізації не буде за допомогою MPS. І навіть, якщо Apple намагатиметься підтримувати нові типи шарів у майбутньому, завжди доведеться чекати до наступного оновлення ОС, щоб отримати їх.

Оскільки моделі стають більш просунутими, вони можуть стати великими і зайняти значні місця для зберігання. Для моделі, на основі нейронної мережі, буде розглянута можливість зменшення її розміру – квантування – використовуючи більш низьке представлення точності для його вагових (рисунок 3.16).

Інструменти Core ML Tools забезпечують функцію перетворення для конвертації ваг з плаваючою точкою моделі нейронної мережі з точністю до значень напівточності (зменшення кількості бітів, що використовуються на вході від 32 до 16), таким чином можна досягати однобітного квантування моделей. Нажаль, на етапі перетворення з 8-бітних до 4-х бітних, вихідне зображення втрачає свої особливості, а якість розпізнавання знижується. Цей тип перетворення може значно зменшити розмір мережі, більшість з яких часто походить від ваги підключення в мережі, без втрати точності розпізнавання об’єктів.

Рисунок 3.16 – Схема використання квантування зображення при різних параметрах

Create ML – одна з найцікавіших особливостей Core ML, що дозволяє тренувати моделі для виконання завдань, таких як: розпізнавання зображень,

вилучення сенсу з тексту або пошук взаємозв'язків між числовими значеннями (рисунок 3.17).

Навчання моделі відбувається за допомогою передачі на вхід навчальних датасетів, які вже позначені мітками на яких вона тренується розпізнавати потрібні зображення і все це робиться з використанням згорткових мереж. Наприклад (рисунок 3.18), можна навчити класифікатор зображень, щоб розпізнати тварин сафарі, показуючи йому різноманітні фотографії слонів, жирафів, левів і так далі. Щоб це зробити, необхідно створити два набори даних, що містять навчальний та тестовий датасет. Треба переконатися, що будь-яке зображення з’являється тільки в одному з цих двох наборів. У кожній папці треба створити підпапки, використовуючи мітки як імена даних для кожного типу (наприклад: тека з котами повинна містить виключно котів), після цього можна починати тренувати модель.

Рисунок 3.17 – Схема роботи Create ML

Точні рядки міток не є важливими, але якщо є бажання створити модель, яка розпізнає різні породи котів, то їх підпапки повинні містити різні породи киць з відповідними мітками. Наприклад, можна використовувати етикетку Cat для всіх зображень котів, також не потрібно називати файли зображень певним чином або додавати до них метадані. Потрібно лише помістити їх у папку з правильною міткою.

Рекомендацією є використовування принаймні 10 зображень на етикетку для навчального набору, але більше завжди краще. Також необхідно балансувати з кількістю зображень для кожної мітки. Наприклад, поганим прикладом є використання 10 зображень для Cat та 1000 зображень для Elephant.

Рисунок 3.18 – Схема тренування моделі

Зображення можуть бути у будь-якому форматі, де єдиний ідентифікатор типу якого відповідає public.image. Це включає в себе поширені формати, такі як JPEG і PNG. Зображення не повинні бути однакового розміру, мати певний розмір, хоча найкраще використовувати зображення розміром не менше 299x299 пікселів. Якщо це можливо, рекомендується тренувати на зображеннях, зібраних таким чином, як і зображення для прогнозування.

Не буде зайвим надати зображенням різноманітності. Наприклад зображення, які показують тварин з різних ракурсів і в різних умовах освітлення. Класифікатор, підготовлений на майже ідентичних зображеннях для даної мітки, прагне мати більш низьку продуктивність, ніж ті, що тренуються на більш різноманітному наборі зображень.

На виході отримується готова модель, яку бажано перевірити за допомогою тестового датасету. Вона впливає на навчання, але по-різному – повідомляє, наскільки добре підготовлена модель класифікує зображення з відповідних множин. Оскільки модель навчена на цих зображеннях, то, як правило, робить хорошу роботу класифікації. Поділ виконується випадковим чином, тобто можна отримати різний результат кожного разу, коли модель навчається.

Якщо точність навчання вашої моделі низька, це свідчить про те, що поточна конфігурація моделі не може врахувати складність ваших даних. Вирішенням проблеми є налаштування параметрів навчання – збільшити максимальну кількість ітерацій (рисунок 3.19).

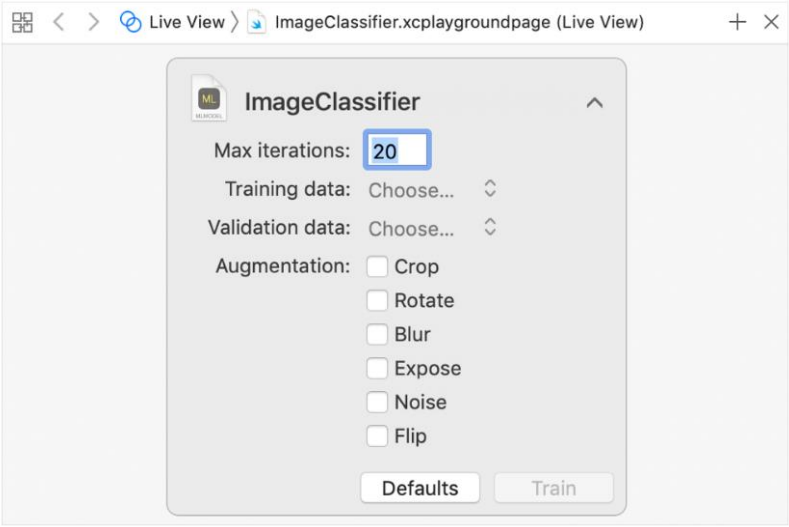


Рисунок 3.19 – Інтерфейс для збільшення ітерацій навчання

Також можна мати багато даних і точність перевірки, що все ще значно нижче, ніж точність навчання. У цьому випадку модель перенасичується, а це означає, що вона вивчає занадто багато конкретних деталей навчального набору, які зазвичай не застосовуються до інших прикладів. У цьому випадку потрібно зменшити кількість ітерацій навчання, щоб запобігти надмірному вивченню моделі навчальних даних.

Коли навчання завершується, перегляд у реальному часі показує точність навчання та перевірки. Якщо модель працює згідно вимог задачі – можна інтегрувати та використовувати її у мобільному додатку.

Однією з унікальних особливостей цього фреймворку є те, що моделі машинного навчання можна довчити. Це робиться дуже легко – досить додати нову групу зображень та повторити процес навчання за допомогою Create ML – донавчання займе лічені хвилини! Робочий процес додавання нового шару передбачення такий самий, як і для моделі без спеціальних шарів. Матриці згортки на краях зображення будуть мати непропорційне число нульових значень і спотворюватимуть результати передачі стилю для цих пікселів. Цю проблему можна усунути за допомогою віддзеркалення, щоб створити візуально узгоджений кордон навколо вхідного зображення, використовуючи обрізання в кінці, щоб зберегти початковий розмір послідовним. В багатьох глибоких навчальних пакетах, включаючи Keras, функції активації фактично розглядаються як окремі шари. Нажаль, без недоліків тут не вийшло – Core ML підтримує тільки фіксований набір функцій активації – ReLu та сигмовидні. Параметри для мережі – це словник, що зберігається в полі нового шару. Коли клас реалізації ініціалізований, ці параметри будуть передані до реалізації класу для налаштування ініціалізації. Щоб перенести вагові коефіцієнти, треба створити новий масив ваг у новому шарі, а потім скопіювати ваги з початкового вихідного шару у новий масив.

Незважаючи на те, що зараз моделі вбудовуються в додатки простіше, ніж коли-небудь, але щоб включити їх у свої системи, все одно потрібно знати щось про машинне навчання. Це особливо стосується власних моделей та даних, які розробники хочуть використовувати, тому що доведеться навчитися їх будувати та тренувати власноруч.

Найбільш простим способом додати машинне навчання до своїх додатків є використання хмарних сервісів, таких як Clarifai, Google Cloud Vision, IBM Watson, Microsoft Azure Cognitive Services, Amazon Rekognition та багато інших. З такими типами хмарних сервісів, машинне навчання – це просто питання виклику веб-API – будь-який розробник iOS знатиме, як це зробити. Також можна розмістити власні моделі у хмарі.

Головні переваги хмарних технологій:

– якість розпізнавання вище;

– змога довчати моделі у реальному часу.

З недоліків маємо:

– дорого. Потрібно буде платити постачальнику послуг у хмарі за послуги;

– повільно. Кожен запит, який відправляється на сервер, повинен перейти по мережі. Не можна використовувати хмарні сервіси для глибокого вивчення відео в реальному часі;

– можливі проблеми конфіденційності. Обов’язково потрібно надіслати дані користувача до хмарної служби;

– блок постачальника. Успіх програми тепер залежить від стороннього учасника, якого розробник майдже не контролює.

Якщо використовується Core ML або будь-яке інше рішення у додатку, слід розробити модель, яка найкраще підходить для програми. Моделі, які Apple робить доступними для завантаження, здатні виконувати тільки обмежену

кількість завдань. Для більшості програм доведеться тренувати власні моделі, які потребують спеціальних знань. Фахівці з машинного навчання сьогодні користуються великим попитом, тому вони не дешеві.

Плюси вбудованого фреймворку:

– дійсно легко додати до свого додатка;

– використовується для глибокого навчання та і логістична регресія, дерева рішень та інші «класичні» моделі машинного навчання.;

– поставляється із зручним інструментом конвертера, який підтримує кілька різних навчальних пакетів (Keras, Caffe, scikit-learn та інші)..

Мінусами є:

– фреймворк підтримує лише обмежену кількість типів моделей. Якщо модель навчена та робить щось, що не підтримує Core ML, то змоги використовувати її не має можливості;

– фреймворк не підтримує користувацькі та різноманітні функції активації, за винятком сигмоїдної та ReLu;

– інструменти перетворення наразі підтримують лише декілька навчальних пакетів. Помітним недоліком є TensorFlow, що є найпопулярнішим інструментом для машинного навчання. Можна написати власні конвертери, але це не робота для новачків. Причина, за якою TensorFlow не підтримується, полягає в тому, що це пакет низького рівня для створення загальних обчислювальних графів, в той час як Core ML працює на набагато більш високому рівні абстракції;

– немає гнучкості, мало контролю. Core ML API дозволяє завантажувати модель і запускати її. Немає способу додати власний код до своїх моделей;

– може буди вбудована в додатки, мобільною операційною системою яких є iOS 11 та вище.

3.4 Порівняння моделей машинного навчання

Добре зроблені згорткові штучні нейронні мережі є звірами з мільйонами параметрів та багатьма прихованими шарами. Насправді, погане правило полягає в тому, що «чим вища кількість прихованих шарів, тим краще мережа». VGG, MobileNet, ResNet та Inception – це деякі з популярних мереж. Розробка архітектури мережі є складним процесом, який займе деякий час, щоб вивчити та, навіть більше, щоб самостійно експериментувати з нею.

Вилучення ознак передбачає вилучення більш високого рівня інформації з сирих значень пікселів, які можуть враховувати відмінність між задіяними категоріями. Це вилучення ознак здійснюється без нагляду, де класи зображення не мають нічого спільного з інформацією, витягнутою з пікселів. Деякі з традиційних і широко використовуваних функцій GIST, HOG, SIFT чи LBP Після вилучення цієї функції навчається модуль класифікації з зображеннями та пов'язаними з ними мітками. Кілька прикладів цього модуля – це SVM, логістична регресія, випадковий ліс, дерева рішень, тощо. Проблема методу вилучення ознак полягає в тому, що вона повністю відрізняється від того, як люди навчаються розпізнавати речі. Відразу після народження дитина не здатна сприймати своє оточення, але коли вона прогресує і обробляє дані, вона вчиться ідентифікувати речі. Це філософія глибокого навчання, в якій не вбудоване жорстке вилучення функцій. Воно поєднує модулі вилучення та класифікації в одну інтегровану систему і навчається витягувати, розрізняючи зображення та класифікувати їх на основі контрольованих даних.

3.4.1 Модель згорткової нейронної мережі VGG

VGGNet (Visual Geometry Group) – модель згорткової нейронної мережі (група візуальної геометрії), винайдена в університеті Оксфорд [21]. Великою перевагою стала знахідка, що кілька згорток 3x3, об'єднаних в послідовність, можуть емулювати більші рецептивні поля, наприклад, 5x5 або 7x7. Ці ідеї пізніше будуть використані в архітектурі Inception та ResNet.

Мережі VGG для представлення складних властивостей використовують численні згортові шари 3x3. Зверніть увагу на блоки 3, 4 і 5 в VGG-E, зображених на рисунку 3.20. Конфігурація має обчислювальну конструкцію, представлену в архітектурі і відрізняється глибиною: від 11 шарів до з вагої в мережі A (8 згорткових і 3 повнозв’язних шари) до 19 (16 згорткових і 3 повнозв’язних шари). Ширина згорткових шарів (кількість каналів) відносно невелика: від 64 в першому шару до 512 в останньому зі збільшенням кількості каналів в 2 рази після кожного максимального об'єднання. Для отримання більш складних властивостей та їх комбінування застосовуються послідовності 256 × 256 і 512 × 512 фільтрів 3 × 3. Це рівносильно великим згортковим класифікаторам 512x512 з трьома шарами. Це дає величезну кількість параметрів та чудові здібності до навчання, але вчити такі мережі було складно, доводилося розбивати їх на більш дрібні, додаючи шари один за іншим. Причина полягала у відсутності ефективних способів уніфікації моделей або якихось методів обмеження великого простору пошуку, якому сприяє безліч параметрів. Можна бачити, що в D-блоці існують модулі з однаковим розміром фільтра, що застосовуються кілька разів для вилучення більш складних і репрезентативних функцій.

Ця концепція блоків модулів стала загальною темою в мережах після VGG. Згорткові шари VGG супроводжуються трьома повністю сполученими шарами. Ширина мережі починається з малого значення 64 і збільшується в 2 рази після

кожного шару об'єднання. Використовуючи два шари 3 × 3 фільтрів, мережа, фактично, охоплює 5 × 5 області, як на рисунку вище. Використовуючи 3 шари 3 × 3 фільтрів, охоплює 7 × 7 ефективної області (рисунок 3.21).

A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Рисунок 3.20 – Конфігурація згорткової мережі VGGNet

Архітектура даної мережі представлена на рисунку 3.22. На вхід першого шару подається RGB зображення розміру 224x224, далі зображення проходять через стек згорткових шарів, в яких використовуються фільтри з дуже маленьким розміром 3x3, що є найменшим розміром для вилучення інформації про те, де знаходяться сторони зображення (право / ліво, верх / низ та центр). В одній з

конфігурацій використовується згортковий фільтр розміром 1x1, який може бути представлений як лінійна трансформація вхідних каналів (з наступною нелінійністю). Згортковий шаг фіксується на значенні 1 піксель.

Просторове доповнення входу до верхнього шару вибирається таким чином, щоб простірове розширення зберегалося після згортки, тобто доповнення дорівнювало 1 для 3x3 згорткових шарів. Посторовий пулінг здійснюється за допомогою п'яти шарів максимальних об'єднань, які слідують за одним із згорткових шарів.

Операція max-pooling виконується у вікні, розміром 2x2 пікселів з шагом 2. Всі приховані шари забезпечені ReLu.

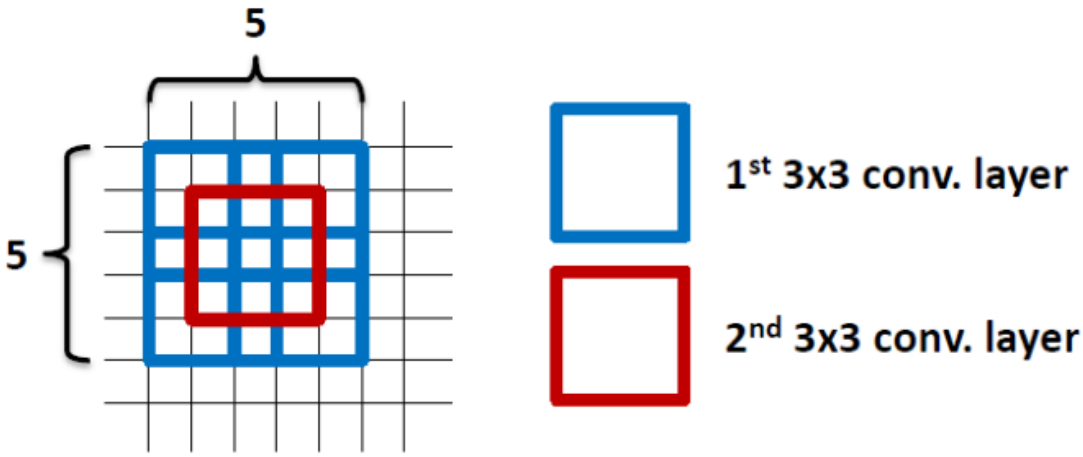


Рисунок 3.21 – Схема охоплення двох шарів 3x3 фільтрів, що охоплюють область 5x5

Відзначимо також, що мережі (за винятком однієї) не містять шару нормалізації (Local Response Normalisation), так як нормалізація не покращує результату на датасеті, а веде до збільшення споживання пам'яті та часу виконання коду.

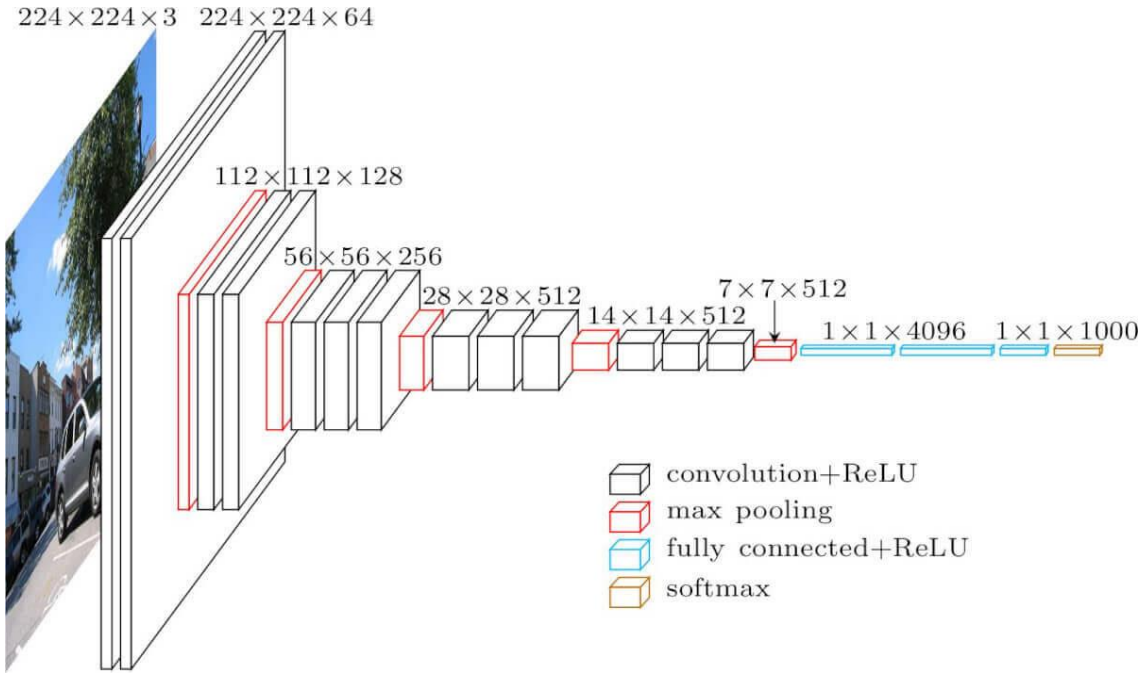


Рисунок 3.22 – Архітектура згорткової нейронної мережі VGG

Нажаль, мережа має два серйозних недоліка:

– дуже повільна швидкість навчання;

– архітектура мережі займає багато дискового простору через глибину та кількість повнозв’язних вузлів (більше 533 мегабайт).

3.4.2 Модель згорткової нейронної мережі Inception

Inception є широко використовуваною моделлю розпізнавання зображень, яка, як було показано, досягає понад 78,1% точності набору даних ImageNet. Модель є кульмінацією багатьох ідей, розроблених дослідниками протягом декількох років [22]. Розробники переосмислили початкову архітектуру (рисунок 3.23), таким чином, вона може використовуватися там, де обмежена пам'ять або обчислювальна потужність, а також має змогу зменшити кількість параметрів – у моделі може бути досягнута 42-шарова мережа глибокого навчання, що має таку ж складність, як VGGNet.

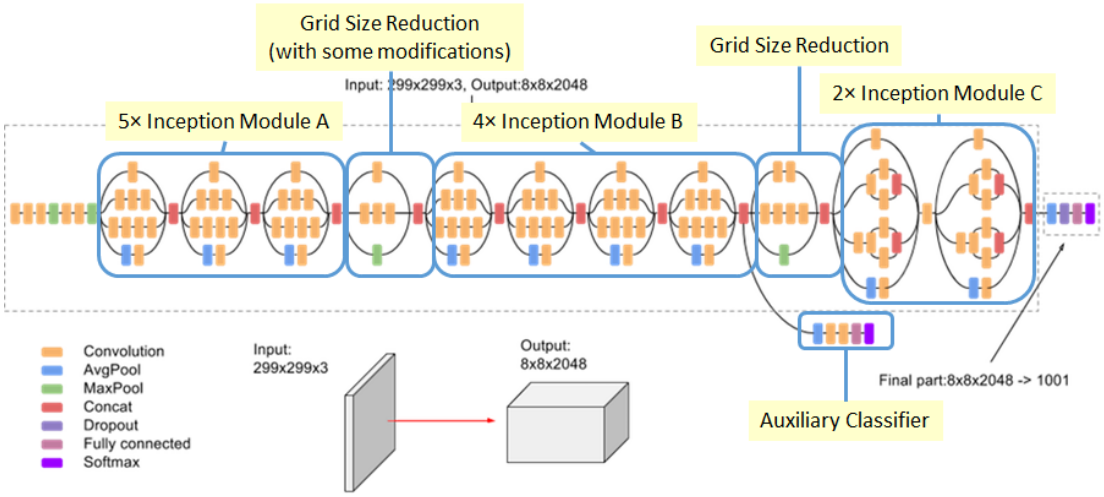


Рисунок 3.23 – Архітектура Inception

Сама модель складається з симетричних і асиметричних блоків, включаючи згортки, середнє та максимальне об'єднання. Пакетна нормалізація широко використовується в моделі і застосовується до входів активації.

Основні ідеї моделі:

– максимізація потоку інформації в мережі за рахунок акуратного балансу між її глибиною і шириною. Перед кожним pooling-гом карти властивостей збільшуються;

– зі збільшенням глибини також систематично збільшується кількість властивостей або ширина шару;

– ширина шару збільшується зі збільшенням комбінації властивостей перед наступним шаром.;

– міру можливості використовуються тільки згортки 3x3. З огляду на те, що фільтри 5x5 і 7x7 можна декомпозувати за допомогою декількох 3x3.

Метою декомпозиції мережі є зменшення кількості з'єднань (параметрів) без зменшення ефективності мережі. Дві 3x3 згортки замінює одну згортку 5x5 наступним чином (рисунок 3.24).

За допомогою одного шару фільтра 5x5 кількість параметрів буде дорівнювати 25. За допомогою двох шарів 3x3 фільтрів кількість параметрів дорівнюватиме 18. Кількість параметрів зменшено на 28%. Подібна методика вже згадувалася у VGGNet.

За допомогою цієї техніки (модуль A), нові модулі моделі стають такими (рисунок 3.25).

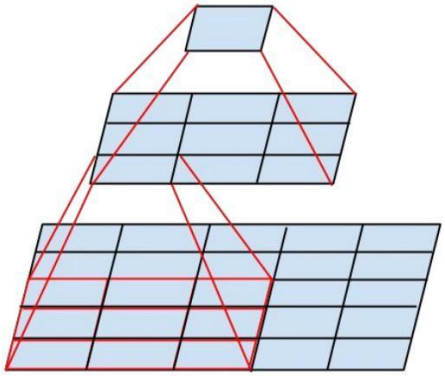


Рисунок 3.24 – Дві 3x3 згортки замінюють одну згортку 5x5

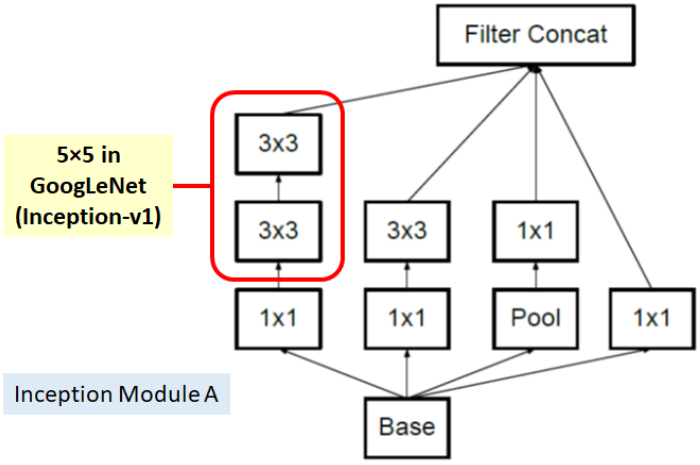


Рисунок 3.25 – Модуль A. Застосування симетричної декомпозиції

Декомпозиція в асиметричних згортках. Одна згортка 3x1, за якою слідує одна згортка 1x3, замінює одну згортку 3x3 наступним чином (рисунок 3.26).

За допомогою фільтра 3x3 кількість параметрів дорівнює дев’ятьом. За допомогою фільтра 3x1 та 1x3 фільтрів кількість параметрів дорівнює шістьом. Кількість параметрів зменшено на 33%.

Якщо використовувати два 2x2 фільтри, то кількість параметрів буде дорівнювати 8, а кількість параметрів буде зменшено лише на 11%

При декомпозиції, число параметрів зменшується для всієї мережі, що зменшує ймовірність перенавчання, отже мережа може йти глибше! За допомогою техніки декомпозиції в асиметричних згортках (модуль B), нові модулі моделі стають такими (рисунок 3.27).

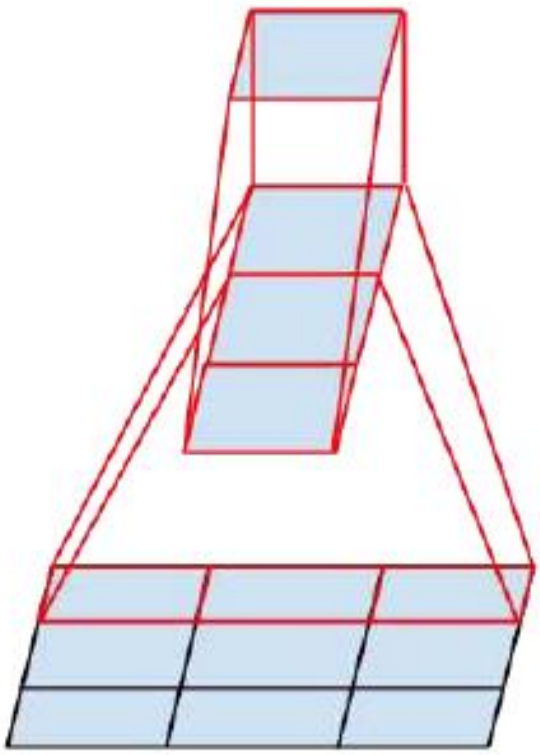


Рисунок 3.26 – Одна згортка 3x1, за якою слідує одна згортка 1x3, замінює одну згортку 3x3

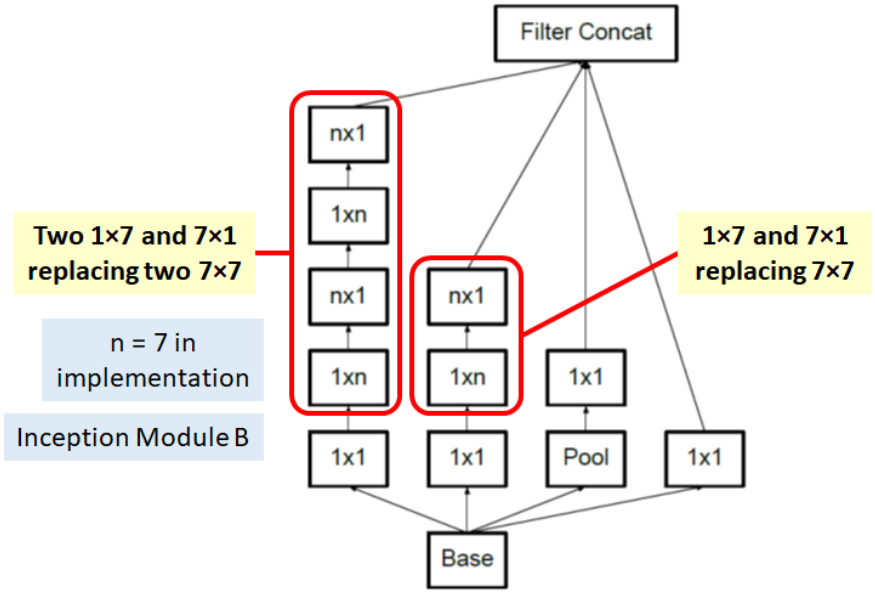


Рисунок 3.27 – Модуль B. Декомпозиція в асиметричних згортках

Запропоновано також додатковий модуль C, для просування зображень з високою розмірністю (рисунок 3.28).

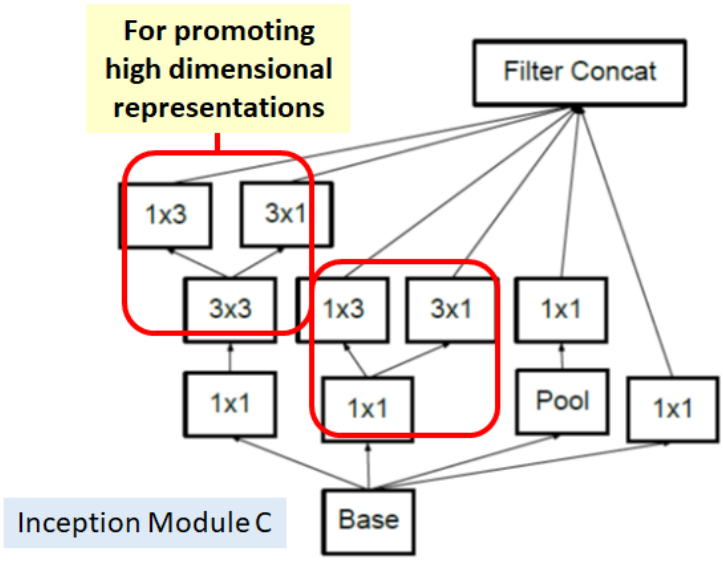


Рисунок 3.28 – Модуль C. Декомпозиція в асиметричних згортках для високо розмірних зображень

Лише один допоміжний класифікатор використовується на вершині останнього 17x17 шару, замість використання двох допоміжних класифікаторів (рисунок 2.29). (Загальна архітектура буде показана пізніше).

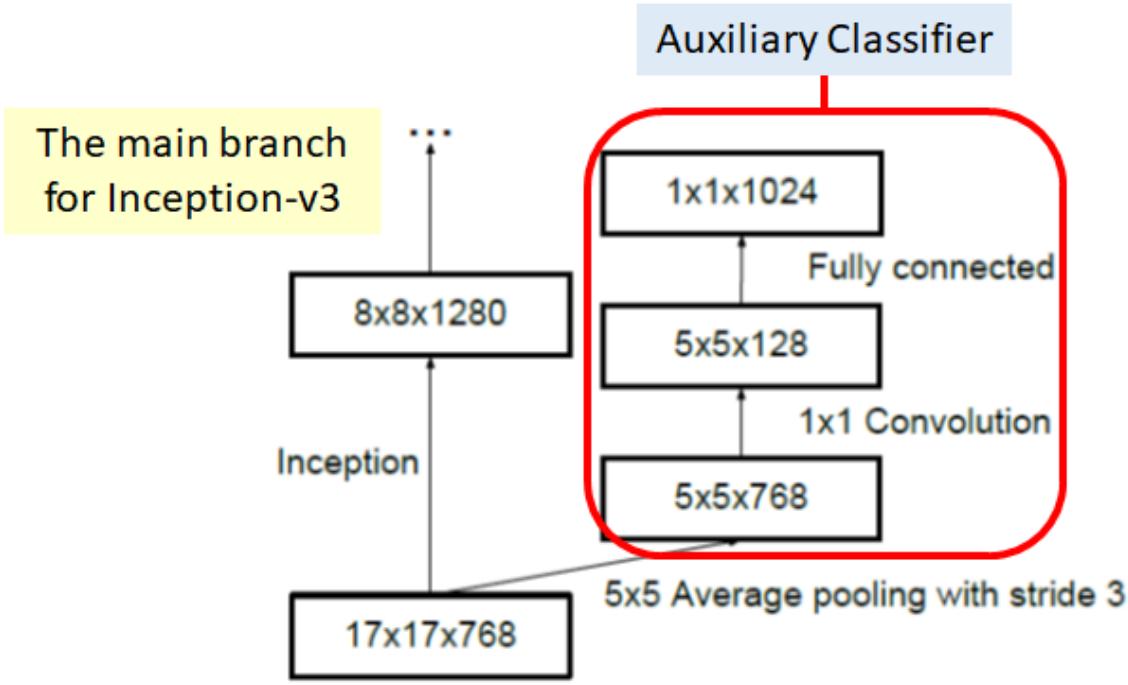


Рисунок 3.29 – Допоміжний класифікатор зі згорткою 17x17

В Inception, допоміжні класифікатори використовуються для більш глибокої мережі та в якості. Так, власне, в глибокому навчанні модулі все ще досить інтуїтивні. Пакетна нормалізація, запропонована для цієї моделі, також використовується в допоміжному класифікаторі.

3.4.3 Модель згорткової нейронної мережі ResNet

Архітектура ResNet (Residual Network) – залишкова мережа, що вміщує до 152 шарів, включаючи згорткові, об'єднані та повністю пов'язані шари [23].

Оскільки ми побачили раніше, що більш глибока модель, безумовно, дає кращі результати, ми можемо сміливо сказати, що при правильній кількості навчальних даних, результат, отриманий від моделі, буде ближче до точної, ніж з будь-якою іншою наявною моделлю. Наприклад, в той час, як була представлена архітектура Inception, відбулася революція – опубліковали ResNet. У ній закладені прості ідеї: подаються вихідні дані двох успішних згортків та обходиться вхідні дані наступного шару. В даному випадку розробники обходять два шари і застосовують підхід у великих масштабах. Обхід одного шару не дає особливої вигоди, а обхід двох – ключова знахідка. Це можна розглядати як маленький класифікатор чи як мережа в мережі.

Збільшення глибини повинно підвищити точність мережі, до тих пір, поки не відбудеться переналаштування, але проблема з підвищеною глибиною полягає в тому, що сигнал, необхідний для зміни ваг, що виникає з кінця мережі шляхом порівняння основної істини і прогнозування, стає дуже малим на більш ранніх шарах, через збільшення глибини. Це по суті означає, що попередні шари практично незначні, а входи дорівнють виходам. Це називається зникаючим градієнтом. Друга проблема з навчанням більш глибоких мереж полягає в оптимізації величезного простору параметрів і тому наївне додавання шарів, що призводять до вищої помилки навчання.

Якщо відображення ідентичності є оптимальним, ми можемо легко підштовхнути залишки до нуля $(F\,(x)\,=\,0)$, ніж підібрати ідентифікаційне відображення (x, input = output) стеком нелінійних шарів. У простій мові дуже легко придумати рішення, як $F\,(x)\,=\,0$, а не $F\,(x)\,=\,x$, використовуючи стек нелінійних шарів як функцію. Отже, ця функція F (x) є те, що називають Residual function або залишковим блоком (рисунок 3.31)

При проектуванні мережі, що дуже схожа на VGGNet, використовувались, в основному, 3x3 фільтри. Вибірка з використанням шарів згорткових мереж з

кроком 2. У кінцевому підсумку, глобальний середній шар об'єднання і 1000 – шляховий повністю пов'язаний шар з softmax (рисунок 3.30).

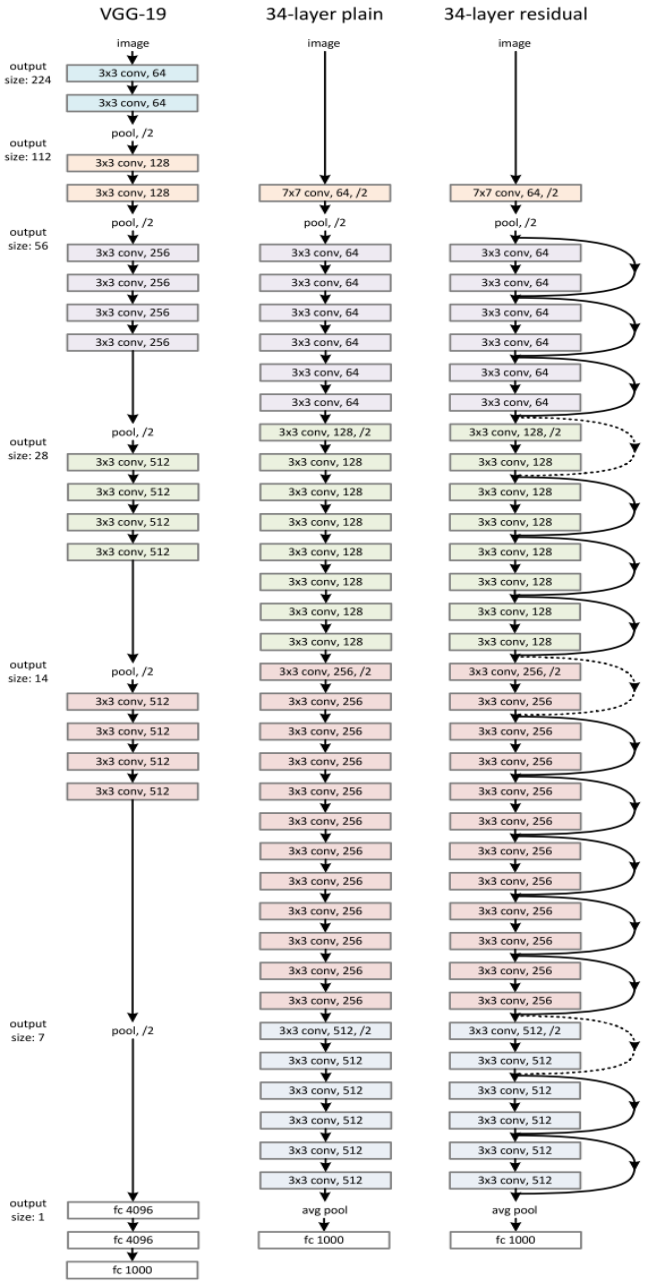


Рисунок 3.30 – Архітектура ResNet. Звичайна VGG (ліворуч) та VGG з залишковими блоками (праворуч)

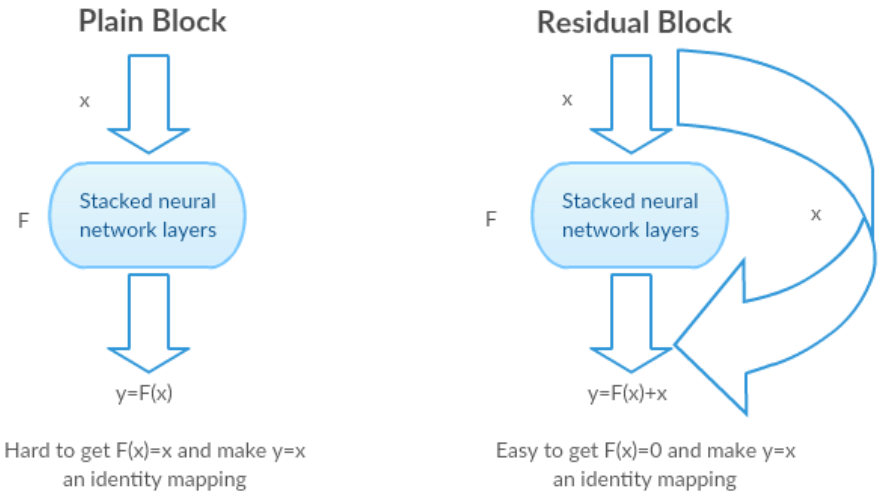


Рисунок 3.31 – Блоки згорткових нейронних мереж: ліворуч відображений простий блок, праворуч – залишковий блок

Штрихи ідентифікації (x) можна використовувати безпосередньо, коли вхідні та вихідні дані мають однакові розміри (формула 3.6).

$$y = F(x, \{W_i\}) + x \tag{3.6}$$

Коли розміри змінюються, ярлик все ще виконує відображення ідентичності, за додаткових нульових записів, доданими до збільшеного розміру та скорочення проекції використовується для відповідності розмірності (зроблено 1 * 1 угод) за допомогою наступної формули 3.7.

$$y = F(x, \{W_i\}) + W_s x \tag{3.7}$$

Залишкові блоки, в основному, є особливим випадком highway-мереж без будь-яких воріт у їхніх пропусках. Загалом, залишкові блоки дозволяють потоку пам'яті (або інформації) рухатись від початкових до останніх шарів нейронної

мережі. Незважаючи на відсутність виходу в їхніх пропусках, залишкові мережі на практиці працюють так само добре, як і будь-яка інша highway-мережа.

Додавання залишкових блоків чітко визначає складність функції. Можна навчити ефективну глибоку нейронну мережу, маючи залишкові блоки, що проходять через канали передачі даних між шарами. ResNet вплинув на розробку подальших глибоких нейронних мереж, як для згорткового, так і для послідовного характеру.

3.4.4 Модель згорткової нейронної мережі MobileNet

MobileNet – глибока, роздільна згортка використовується для зменшення розміру і складності моделі, що вийшла у 2017 році. За рахунок меншої кількості параметрів, кількість множин та додатків, вона має менший розмір моделі та складність. Модель використовує глибоко відокремлювані згортки, які в основному означає, що він виконує одну згортку на кожному кольоровому каналі, а не об'єднує всі три і згладжує його. Вона має ефект фільтрації вхідних каналів.

MobileNet також надає два параметри, що дозволяють додатково зменшити кількість операцій: множник ширини (від 0 до 1) зменшує кількість каналів. На кожному шарі замість того, щоб виробляти N каналів, він виробляє alpha * N. Цей множник може бути використаний для обробки компромісу між бажаною затримкою і продуктивністю. Також існує інший множник – множник дозволу. Він масштабує вхідний розмір зображення, від 224 до 128. Оскільки MobileNet використовує глобальне середнє об'єднання, а не згладжування, модель має можливість вчитися на зображеннях 224x224, а потім використовувати його на зображеннях 128x128. Дійсно, при глобальному об'єднанні, повністю пов'язаний класифікатор, в кінці мережі залежить тільки від кількості каналів, а не від мапи просторового розміру.

Глибока згортка (Depthwise convolution) – це канальна мутація DK × DK просторової згортки. Припустимо (рисунок 3.32), маємо 5 каналів, тоді будемо мати 5 DK × DK просторової згортки. Точкової згортки (Pointwise convolution) насправді є 1 × 1 згортки для зміни розмірності.

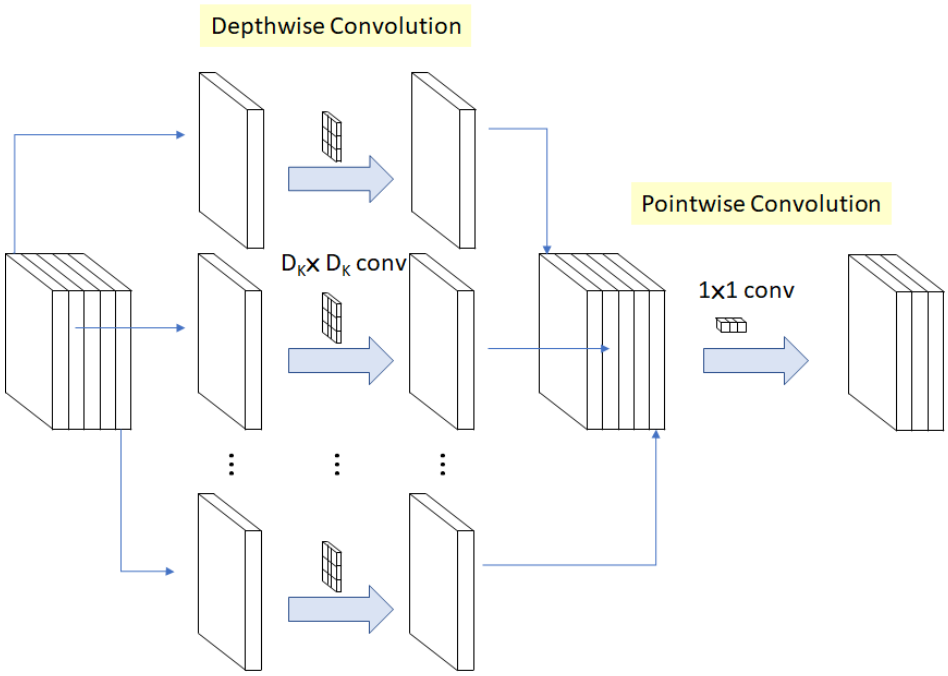


Рисунок 3.32 – Глибока відокремлювана згортка: ліворуч відображена глибока згортка, праворуч – точкова згортка

Таким чином, загальна архітектура Mobilenet (рисунок 3.33) полягає в наступному:

– згортковий шар з кроком 2;
– глибинний шар;
– точковий шар, що подвоює кількість каналів;
– глибинний шар з кроком 2;
– точковий шар, що подвоює кількість каналів.

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Рисунок 3.33 – Архітектура Mobilenet

Для MobileNet, поглиблена згортка застосовує один фільтр до кожного вхідного каналу. Точкова згортка застосовує 1 × 1 згортки, щоб потім об'єднати її виходи. Стандартна згортка обох фільтрів об'єднує входи в новий набір виходів за один крок. Глибоко роздільна згортка розбивається на два шари, один з яких є окремим шаром для фільтрації і окремим шаром для об'єднання. Ця декомпозиція має ефект різкого зменшення обчислень та розміру моделі.

Це також дуже низьке технічне виконання, тому досить добре працює з високою швидкістю. Існує безліч різновидів попередньо навчених моделей з розміром мережі в пам'яті і на диску, пропорційним кількості використовуваних параметрів.

Зазначається, що після кожної згортки застосовуються пакетна нормалізація та функцією активації ReLu (рисунок 3.34).

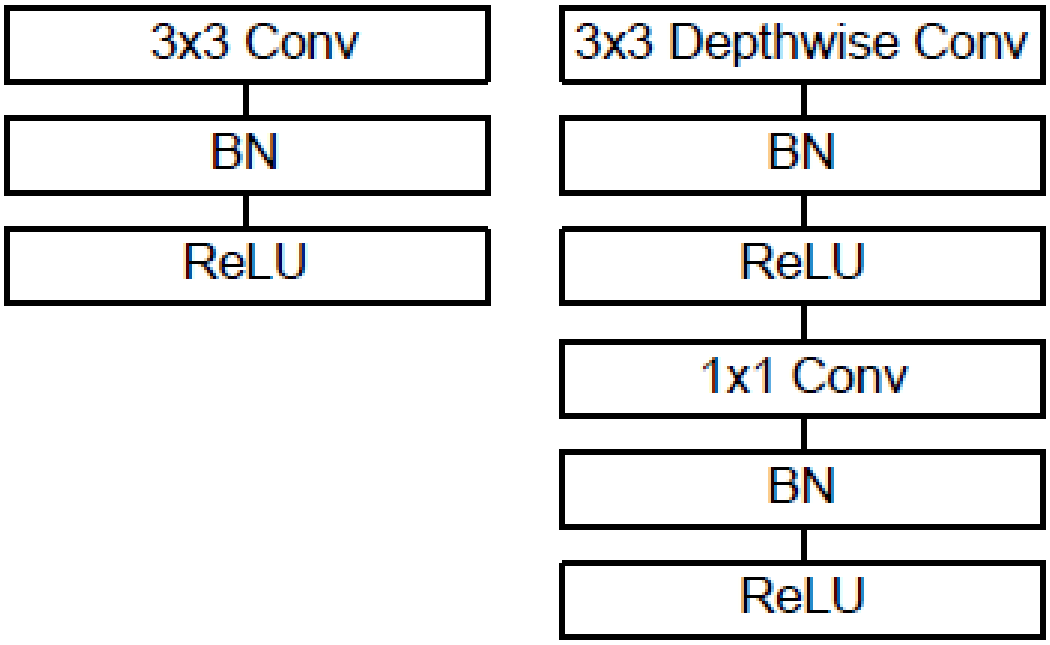


Рисунок 3.34 – Стандартна згортка (ліворуч) та глибинно відокремлена згортка (праворуч) з пакетною нормалізацією та функцією активації ReLu

3.4.5 Аналіз ефективності вищезазначених нейронних мереж

Мета цього розділу полягає в тому, щоб забезпечити більш всебічний та повний аналіз існуючих моделей для розпізнавання зображень. Робочим обладнанням буде виступати графічний процесор NVIDIA Titan X Pascal (часто називають Titan Xp). З цією метою був проведен аналіз та порівняння більш ніж

40 сучасних архітектур глибинних нейронних мереж з точки зору обчислювальної вартості та точності [24]. Зокрема, в експерименті вимірюється: ступінь точності, складність моделі, використання пам'яті, обчислювальну складність і час виведення.

Найбільш важливими висновками є:

– точність розпізнавання не зростає з ростом кількості операцій;

– не існує лінійної залежності між складністю та точністю моделі;

– бажана пропускна спроможність ставить верхню межу до досяжної точності;

– не всі моделі Deep Neural Network використовують свої параметри з однаковим рівнем ефективності;

– практично всі моделі здатні до високопродуктивних графічних;

– процесорів у реальному часі, тоді як деякі з них можуть гарантувати її на вбудованій системі;

– навіть моделі з дуже низьким рівнем складності мають мінімальний розмір пам'яті графічного процесора близько 0.6GB.

Була реалізована базова структура для порівняння глибинних нейронних мереж у Python. Пакет PyTorch використовується для обробки нейронних мереж за допомогою cuDNN-v5.1 і CUDA-v9.0 як back-end.

Усі експерименти виконуються на робочій станції та вбудованій системі: робоча станція оснащена процесором Intel Core i7-7700 @ 3.60GHZ, 16GB DDR4 RAM 2400 МГц, графічним процесором NVIDIA Titan X Pascal 3840 з ядрами CUDA. Операційна система Ubuntu 16.04.

Була проаналізована складність моделі, підраховуючи загальну кількість параметрів, які можна вивчати. Зокрема, розмір файлу параметрів з точки зору МБ для розглянутих моделей. Ця інформація дуже корисна для розуміння мінімальної кількості пам'яті GPU, необхідної для кожної моделі.

Графік кульки (рисунок 3.35), що відображає точність проти обчислювальної складності.

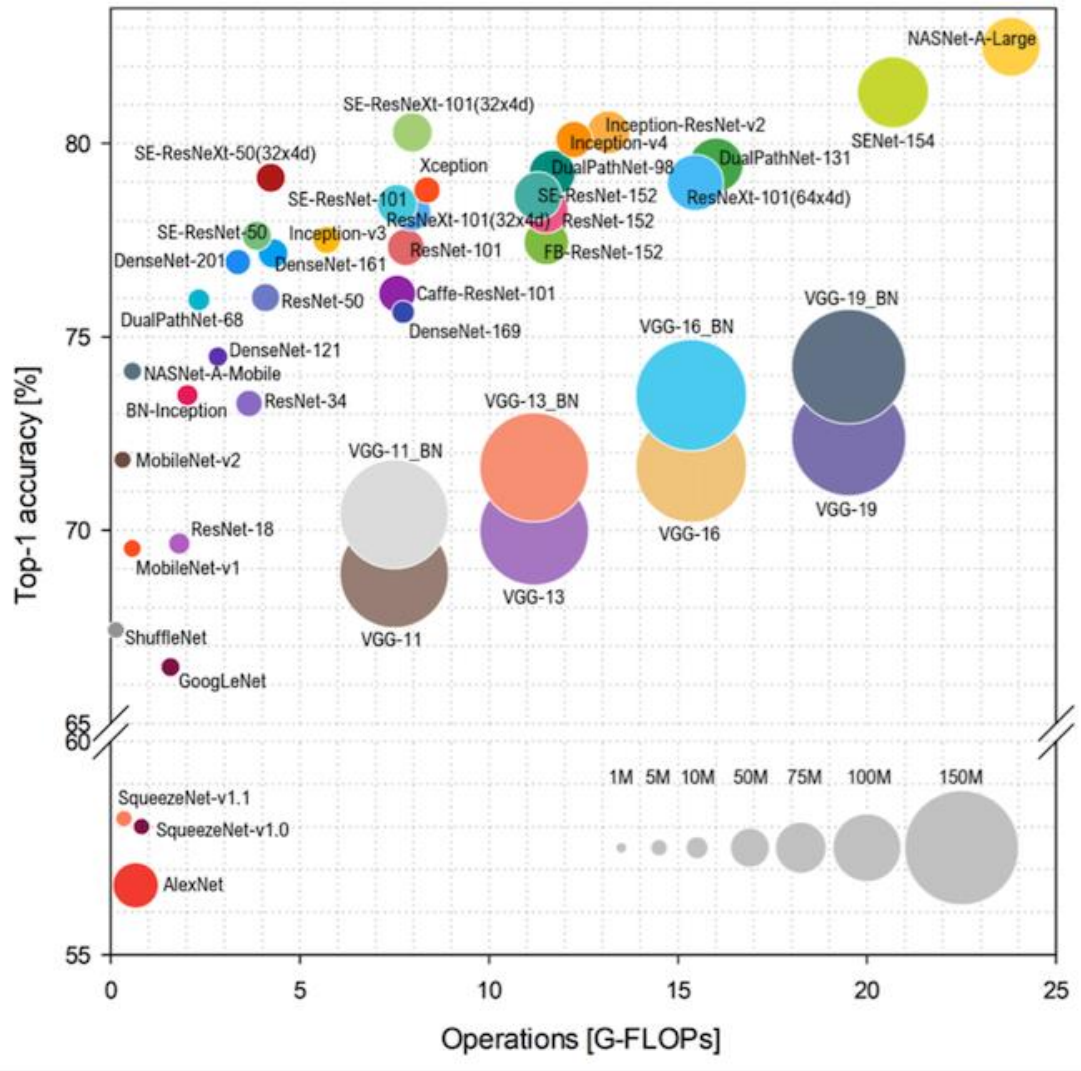


Рисунок 3.35 – Рисунок відображає точність моделі проти обчислюваної складності

Точність, що використовує тільки центральні операції проти операцій з плаваючою комою (FLOP), необхідна для одного проходу вперед. Розмір кожного кульки відповідає складності моделі.

З рисунку 3.34, видно, що модель NASNet-A-Large досягає найвищої точності, а також також має найвищу обчислювальну складність, але якщо розглянути моделі, про які було сказано вище – найоптимальнішою за точністю модель є Inception-v4 (четверта версія), а за обчислюваною складністю – VGG-19 (версія 19).

В цілому, здається, що не існує взаємозв'язку між обчислювальною складністю і точністю розпізнавання, наприклад, SENet-154 потребує приблизно трьохзначної кількості операцій, які потрібні SE-ResNeXt-101, маючи майже таку ж точність. Більше того, здається, що не існує взаємозв'язку між складністю моделі та точністю розпізнавання, наприклад: VGG-13 має значно вищий рівень складності моделі (розмір кульки), ніж ResNet-18, маючи майже однакову точність.

Відомо, що моделі глибинного навчання є неефективними при використанні їх повної навчальної потужності (виміряної як кількість параметрів по відношенню до ступенів свободи). Хоча існує багато документів, які використовують цю функцію для створення стислих моделей з однаковою точністю оригінальних, тому тут було виміряно, наскільки ефективно кожна модель використовує свої параметри.

Слідуючи за графіком (рисунок 3.35) і вимірюючи його як точність, розділену на кількість параметрів. Чим вище це значення, тим вище ефективність. На рисунку 3.35 можна бачити, що моделі, які використовують їхні параметри найбільш ефективно, є SqueezeNets, ShuffleNet, MobileNets і NASNet-A-Mobile. Щоб зосередитися на інформації про щільність, була побудована точність по відношенню до щільності точності (рисунок 2.36), що дозволяє легше знайти потрібний компроміс. Таким чином можна легко побачити, що серед найбільш ефективних моделей є NASNet-A-Mobile та MobileNet-v2 моделі, що забезпечують набагато вищу точність. Серед моделей, що мають найвищу

точність, тобто вище 80%, можна спостерігати, як моделі, що використовують їх параметри більш ефективно, є Inception-v4 і SE-ResNeXt-101 (32x4d). Середній час виведення зображення за 10 прогонів для всіх розглянутих моделей DNN наведено на рисунках 3.36 та 3.37 для пакетного розміру, рівного 1, 2, 4, 8, 16, 32 і 64 на графічному процесорі титан Xp.

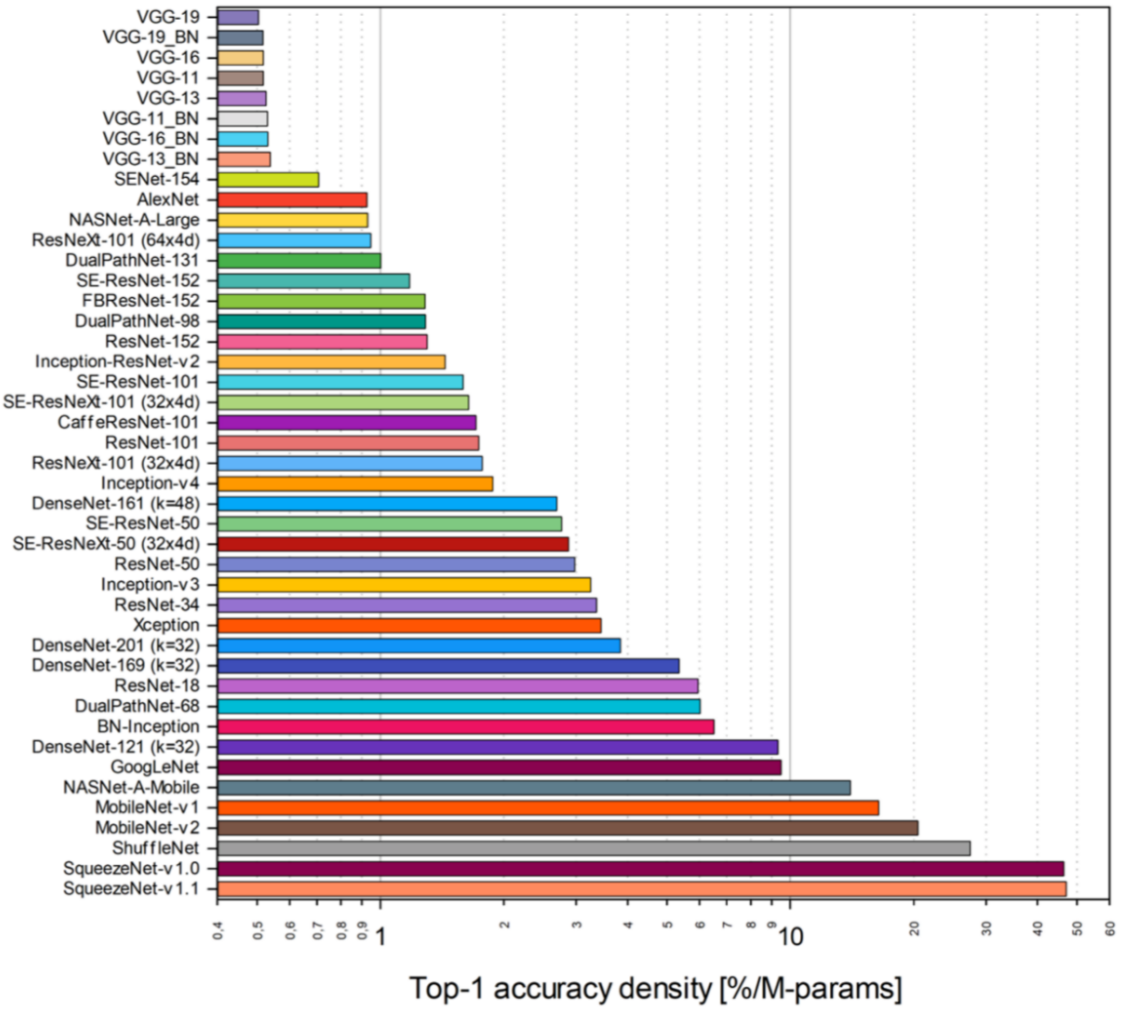


Рисунок 3.36 – Рисунок відображає точність, розділену на кількість параметрів

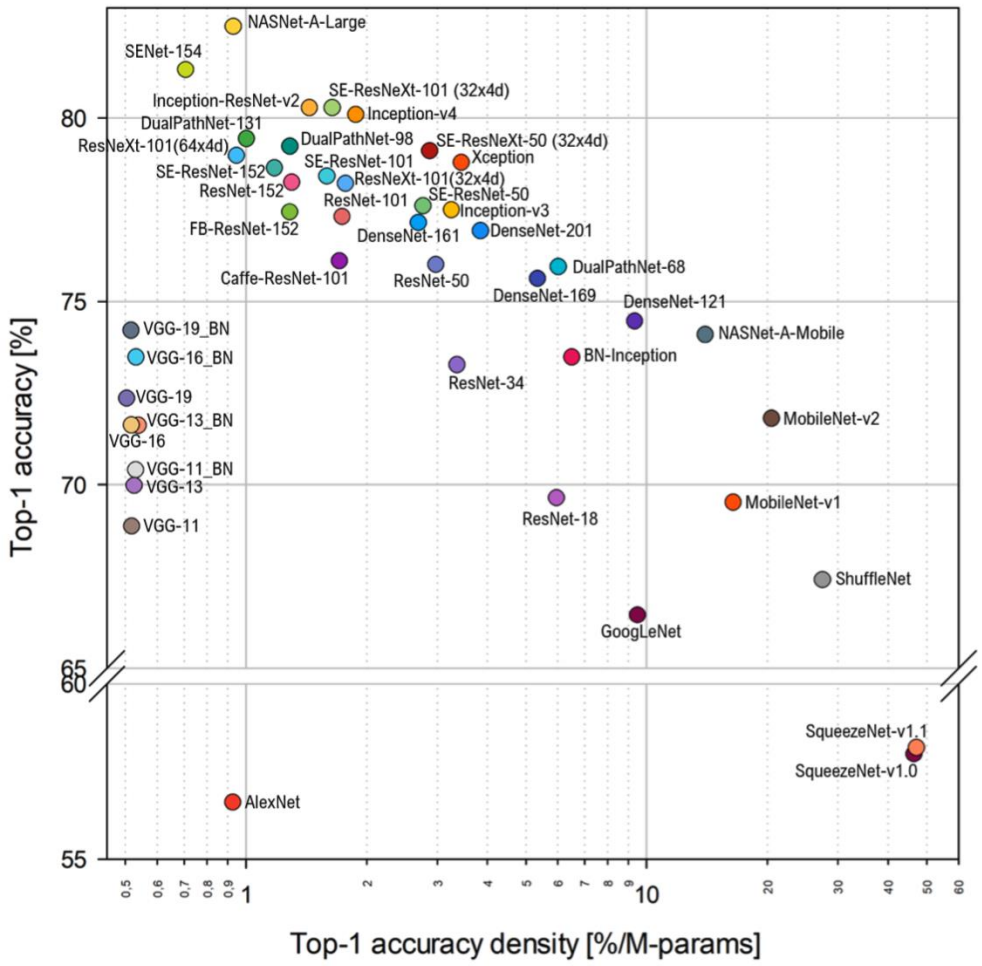


Рисунок 3.37 – Рисунок відображає точність по відношенню до щільності точності

З рисунку можна побачити, що всі розглянуті моделі DNN здатні досягати у режимі реального часу гарних результатів на Титані Xp, за винятком SENet-154, коли розглядається пакетний розмір 1.

На наступному рисунку (рисунок 3.38) наведено найкращі архітектури DNN з точки зору точності розпізнавання, коли конкретні апаратні ресурси надаються як обчислювальні обмеження. Цей аналіз робиться за допомогою Titan Xp.

Titan Xp, з обмеженим використанням пам'яті, досягає точності розпізнавання не більше 75.95%, використовуючи мережу DPN-68 незалежно від часу обчислень. Завдяки більшій кількості ресурсів, наприклад, середньому та високому використанню пам'яті, Titan Xp досягає точності розпізнавання не більше 79,11%, використовуючи SE-ResNeXt-50 (32x4d) з суперпропускною здатністю в реальному часі.

Titan Xp					
$\leq$ 0.7GB, @15FPS	Acc.	$\leq$ 0.7GB, @30FPS	Acc.	$\leq$ 0.7GB, @60FPS	Acc.
DPN-68	75.95	DPN-68	75.95	DPN-68	75.95
DenseNet-121 (k=32)	74.47	DenseNet-121 (k=32)	74.47	DenseNet-121 (k=32)	74.47
NASNet-A-Mobile	74.10	NASNet-A-Mobile	74.10	BN-Inception	73.48
BN-Inception	73.48	BN-Inception	73.48	MobileNet-v2	71.81
MobileNet-v2	71.81	MobileNet-v2	71.81	ResNet-18	69.64
$\leq$ 1.0GB, @15FPS	Acc.	$\leq$ 1.0GB, @30FPS	Acc.	$\leq$ 1.0GB, @60FPS	Acc.
Inception-ResNet-v2	80.28	Inception-ResNet-v2	80.28	SE-ResNeXt-50(32x4d)	79.11
Inception-v4	80.10	Inception-v4	80.10	ResNet-152	78.25
DPN-131	79.44	DPN-131	79.44	Inception-v3	77.50
DPN-98	79.23	DPN-98	79.23	FBResNet-152	77.44
SE-ResNeXt-50(32x4d)	79.11	SE-ResNeXt-50(32x4d)	79.11	ResNet-101	77.31
$\leq$ 1.4GB, @15FPS	Acc.	$\leq$ 1.4GB, @30FPS	Acc.	$\leq$ 1.4GB, @60FPS	Acc.
NASNet-A-Large	82.50	NASNet-A-Large	82.50	SE-ResNeXt-50(32x4d)	79.11
SENet-154	81.32	Inception-ResNet-v2	80.28	Xception	78.79
Inception-ResNet-v2	80.28	SE-ResNeXt-101(32x4d)	80.28	SE-ResNet-101	78.42
SE-ResNeXt-101(32x4d)	80.28	Inception-v4	80.10	ResNet-152	78.25
Inception-v4	80.10	DPN-131	79.44	SE-ResNet-50	77.61

Рисунок 3.38 – Топ 5 найпопулярніших моделей, відсортованих по зниженню точності

Підсумовуючи аналіз, можна зробити висновки, що найбільш ефективною моделлю навчання є ResNet-50, здатною гарантувати половину пропускної здатності в реальному часі, з точністю розпізнавання 76,01%, а не маючи вимог щодо використання пам’яті, MobileNet-v1 досягає точності розпізнавання не більше 69,52%.

4 АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Формування вимог до програмної системи

Даний програмний продукт повинен бути розроблений, як вже зазначалося вище, з метою вивчення іноземних мов та дослідження навколишнього середовища через інтерактивну систему з інтервальним повторенням. У якості моделі машинного навчання буде виступати ResNet-50, а за переклад слів буде відповідальний Google Translate. Мобільний додаток повинен розпізнавати зображення, що потрапляють до камери мобільного пристрою, ранжувати за ймовірністю того, який об’єкт відображений на фото, перекладати на іноземну мову та відображати всі вхідні дані на головному екрані. Основна функція повинна залучати користувачів до використання додатку, а також збільшити сесію використання застосунка до десяти хвилин, у порівнянні з сервісами, що були описані у попередньому розділі. Нажаль, через їх особливості, вони мають сесію, в середньому, у два рази меншу.

Отже, застосунок має можливість зберігати вхідні дані у базу даних, за допомогою якої користувач зможе у будь-який час відкрити застосунок та повторити вивчені слова, навіть якщо він не має змоги підключитися до інтернету, а також сканувати нові об’єкти.

База даних повинна зберігати списки всіх слів та їх переклад на іноземну мову. Для можливості працювати з основним функціоналом, сховище повинна зберігати інформацію кожного слова окремо.

У якості базового набору необхідно інтегрувати модель машинного навчання, зробити коректні запити до сервісу Google Translate та бази даних, що представлена у вигляді фреймворку Realm та мати чітку послідовність дій.

Додаток реалізує класичний та досить популярний архітектурний шаблон Model-View-Controller, який дозволяє представити роботу обробки запитів, сканування об’єктів та зберігання даних у вигляді окремих модулів, кожен з яких відповідає за окрему функцію та має свою мету. У якості View був обраний фреймворк для клієнтської взаємодії Cocoa Touch.

Додаток повинен мати зручний графічний інтерфейс для використання, звертатися в БД за допомогою CRUD-запитів та вилучати слова, що були додані користувачем, перекладати на іноземну мову.

У мобільному додатку необхідно мати наступні екрани:

– система повинна бути представлена у вигляді iOS-застосунку;
– сторінка з привітанням гостя та інструкцією з використання додатку;
– сторінка для сканування об’єктів навколишнього середовища;
– сторінка з налаштуванням;
– сторінка зі списком доданих слів;
– сторінка з використанням карток для інтервального повтору;
– застосунок повинен мати простий та зрозумілий інтерфейс.

Додаток повинен мати одну роль – роль користувача. Він має змогу пройти коротке ознайомлення з основним функціоналом застосунку, після якого може починати роботу з програмою та ознайомитись з інтерфейсом самостійно.

Система повинна безпосередньо працювати з камерою та галерею, адже без цього вона не зможе виконувати свої основні функції.

До даного програмного забезпечення висуваються наступні системні вимоги:

– для встановлення мобільного додатку необхідний iPhone з операційною системою iOS 11.3 і вище;
– доступ до камери та галереї;
– обсяг дискового простору повинен бути не менш ніж 150 Мб.

До даного продукту висуваються наступні функціональні вимоги: продукт повинен сканувати навколишні об’єкти в режимі real-time з встановленою швидкістю у п’ять секунд без можливості додавання слів, мати кнопку на головному екрані, натискання якої призводить до відображення сканованого об’єкту та варіанти перекладу слів, що обробила модель машинного навчання, а також мати змогу діставати фотографії з галереї користувача, працюючи за однаковим алгоритмом. Продукт повинен опрацьовувати й аналізувати вхідні дані для того, щоб представляти основні функції та можливості системи. Також продукт повинен гарантувати цілісність і надійність збереження користувальницьких даних та захищати їх авторське право. Також продукт повинен представляти можливість здійснення контролю за списком слів, які були додані в базу даних, але не повинен робити це автоматично. Продукт повинен дозволяти виконувати користувачам свої основні операції та функції, не викликаючи при цьому якісь проблеми, підозри або недовіри зі сторони користувачів.

Серед нефункціональних вимог до даного проекту варто віднести наступне: безпека системи, надійність, швидкодія, зручність у використанні та масштабування. Система повинна представляти та гарантувати безпеку персональних даних користувачів, коли вони використовують камеру, не дозволяючи при цьому використовувати та запрошувати доступ до вбудованих додатків чи даних, які можуть викликати підозру чи не відносяться до основного функціоналу.

Також система повинна мати досить високу швидкодію та мати невисокий термін відклику на якісь дії користувачів, не споживаючи багато енергії, аби телефон швидко не розряджався. Продукт повинен бути зручним у використанні, відповідати принципам «Usability» інтерфейсів та UX.

Система повинна мати високу швидкість, продуктивність, бути масштабованою та бути відновлюваною, тобто повинна мати можливість відновлюватися після якихось збоїв та перепадів.

4.2 UML проектування ПЗ

UML (Unified Modeling Language – уніфікована мова моделювання) – це мова візуального моделювання, яка розроблена для візуалізації, проектування та документування компонентів програмного забезпечення. UML дозволяє також досягти угоди в графічних позначеннях для подання загальних понять (таких як клас, компонент, узагальнення (generalization), об'єднання (agregation) і поведінка) і орієнтована на проектування та архітектуру програмного продукту.

UML є досить потужним інструментом моделювання, який може бути ефективно використаний у побудові концептуальних, логічних і графічних моделей складних систем різного цільового призначення.

Ця мова увібрала в себе найкращі якості і досвід методів програмної інженерії, які з успіхом використовувалися впродовж останніх років при моделюванні великих і складних систем.

Діаграма прецедентів описує основні дії користувача системи, що пов’язані зі скануванням навколишнього середовища, а також тренування та вивчення нових слів.

Користувач починає роботу у додатку з перегляду привітання та початкового інтро, аби ознайомитись з основними функціями додатку. Основний функціонал користувача пов’язан із роботою зі зовнішніми об’єктами, коли він сканує об’єкт в режимі реального часу, а на виході отримує слова на рідній та перекладеній мовах, які можуть встановлюватися ним самим.

Є можливість вибирати зображення з галереї та сканувати їх або зробити знімки, які система автоматично відсканує та видасть певний результат. Остання головна частина функціоналу пов’язана з інтервальним тренуванням слів за допомогою простих та зручних карток, що зберігаються у базі даних за бажанням користувача.

Також застосунок має додатковий функціонал. Перехід до налаштувань, де користувач може виконати такі дії як: вибір мови самого додатку, кнопки, що виконують публікації готових зображень з перекладом до таких соціальних мереж як Twitter чи Facebook.

Побудуємо діаграму прецедентів, виділивши при цьому основні дії користувача у додатку (рисунок 4.1).

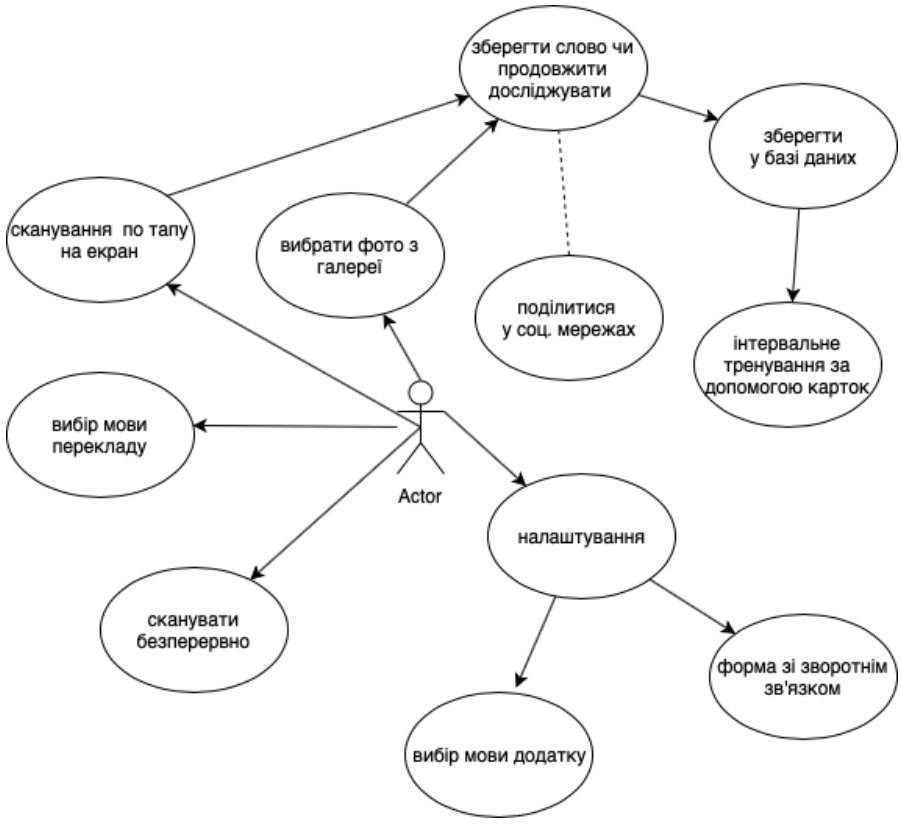


Рисунок 4.1 – Діаграма прецедентів (Use Case)

Діаграма послідовності відображає взаємодії впорядкованих за часом об’єктів. Побудуємо діаграму для сканування та тренування у системі, рисунку 4.2.

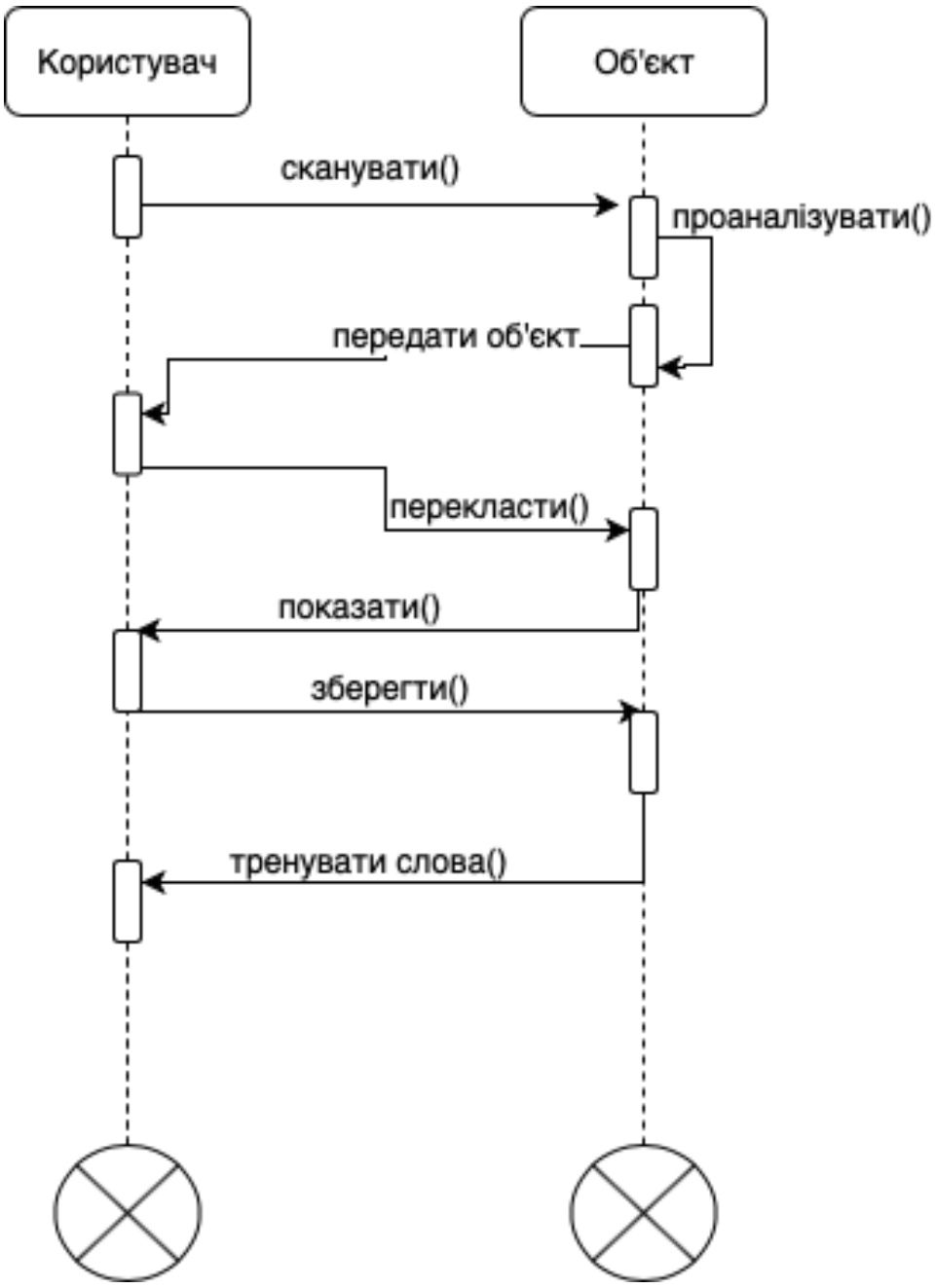


Рисунок 4.2 – Діаграма послідовності

Спочатку користувач починає сканувати об’єкт, спрямувавши камеру на нього. Після аналізу моделі машинного навчання, треба передати вихідні данні та перекласти їх на іноземну мову. Останнім кроком є відображення результату та надання можливості зберегти результат. Якщо користувач збереже достатньо слів, то він матиме змогу тренувати нові слова.

4.3 Проектування архітектури ПЗ

Архітектура програмного забезпечення – це структура програми або обчислювальної системи, яка включає в себе програмні компоненти, видимі зовні властивості цих компонентів, а також відносини між ними. Архітектура є безліччю структур, необхідних для міркування про програмній системі, і включає елементи системи, зв'язку між ними і властивості цих елементів та зв'язків.

Для задоволення проектованої системи різними атрибутами якості застосовуються різні архітектурні шаблони (патерни). Кожен патерн має свої завдання і свої недоліки.

Види архітектурних патернів:

– багаторівневий патерн;
– патерн посередника;
– патерн «клієнт-сервер».

Клієнт-серверна система передбачає розбиття програми на логічні рівні: клієнт і сервер (рисунок 4.3).

Архітектура «клієнт-сервер» визначає загальні принципи організації взаємодії в мережі, де є сервери, вузли-постачальники деяких специфічних функцій (сервісів) і клієнти, споживачі цих функцій.

Архітектура клієнт-сервер є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних застосувань і передбачає взаємодію та обмін даними між ними.

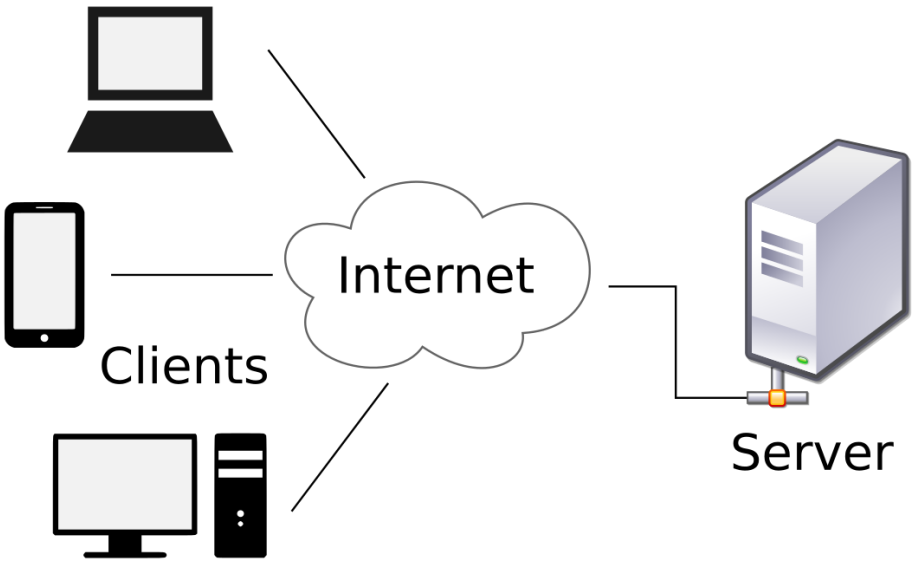


Рисунок 4.3 – Клієнт-серверна архітектура

Сервери є незалежними один від одного. Клієнти також функціонують паралельно і незалежно один від одного. Немає жорсткої прив'язки клієнтів до серверів.

Побудуємо діаграму розгортання (рисунок 4.4).

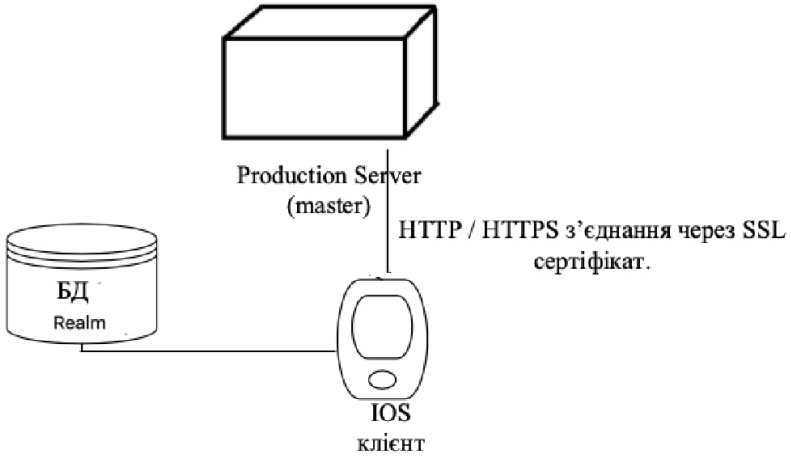


Рисунок 4.4 – Діаграма розгортання

Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів. Клієнти мають знати про доступні сервери, але можуть не мати жодного уявлення про існування інших клієнтів. Комп'ютер має деяку апаратну конфігурацію і працює під управлінням певної операційної системи. Корпоративні програми часто вимагають для своєї роботи деякої IT-інфраструктури, зберігають інформацію в базах даних, розташованих десь на серверах компанії, використовують загальні ресурси. Діаграма розгортання показує топологію системи і розподіл компонентів системи по її вузлів, а також сполуки – маршрути передачі інформації між апаратними вузлами.

За допомогою діаграми можна побачити, що користувач має змогу виконувати дії з мобільного пристрою iOS.

Далі відправляються дані до головного серверу через HTTP або HTTPS протоколи передачі даних.

Діаграма компонентів, що також розроблена з використанням UML нотації, відображує розбиття програмної системи на структурні компоненти та зв’язок між компонентами.

Побудуємо діаграму для нашою системи (рисунок 4.5). Модульні програми пропонують інший спосіб роботи над додатком. За допомогою створення невеликих компонентів багаторазового використання замість одного великого додатку досягається просте додавання або вилучання коду. Маючи набір компонентів, можна мати кілька додатків, спільне використання деяких функцій, але маючи свою специфіку

Розроблюваний сервіс також відповідає архітектурному стилю REST.

REST – підхід до архітектури мережевих протоколів, які забезпечують доступ до інформаційних ресурсів. Найвідомішою системою, побудованою переважно за архітектурою REST, є сучасна Всесвітня павутина. Для цього будет

використаний фреймворк Alamofire, щоб полегшити роботу з запитами в мережу інтернет.

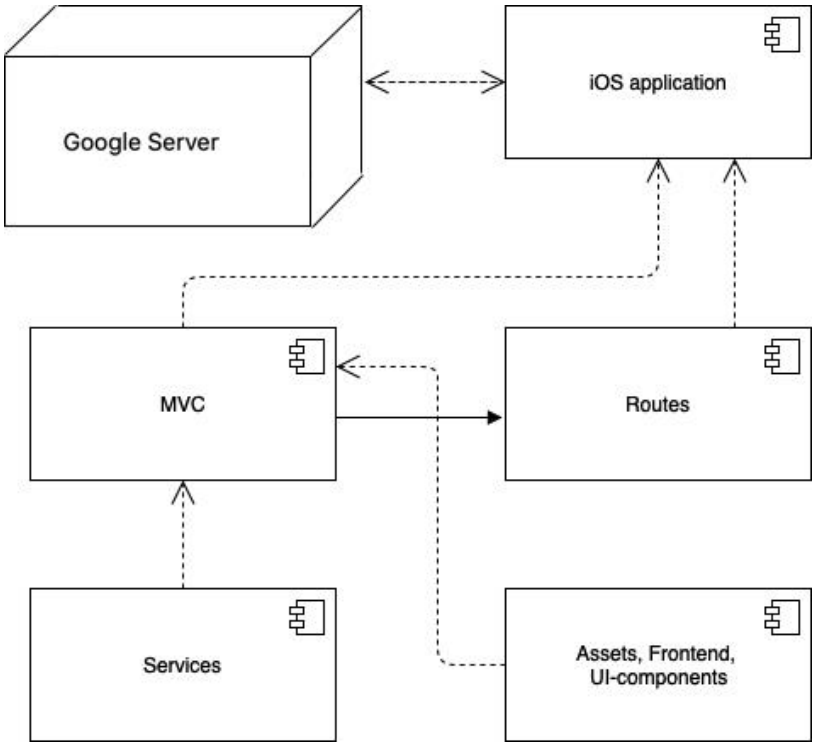


Рисунок 4.5 – Діаграма компонентів

Дані повинні передаватися у вигляді невеликої кількості стандартних форматів, наприклад JSON об’єктів. Мережевий протокол (як і HTTP) повинен підтримувати кешування, не повинен залежати від мережевого прошарку, не повинен зберігати інформацію про стан між парами «запит-відповідь». Стверджується, що такий підхід забезпечує масштабування системи і дозволяє їй еволюціонувати з новими вимогами.

4.4 Проектування бази даних

Бази даних NoSQL спеціально створені для певних моделей даних і мають гнучкі схеми, що дозволяє розробляти сучасні програми. Бази даних NoSQL

набули широкого поширення в зв'язку з простотою розробки, функціональністю і продуктивністю при будь-яких масштабах.

У базах даних NoSQL для доступу до даних і управління ними застосовуються різні моделі даних, в тому числі документна, графова, пошукова, з використанням пар «ключ-значення» і зберіганням даних в пам'яті. Бази даних таких типів оптимізовані для додатків, які працюють з великим обсягом даних, потребують низької затримки і гнучких моделях даних. Все це досягається шляхом пом'якшення жорстких вимог до несуперечності даних, характерних для інших типів БД.

У базі даних NoSQL запис про книгу зазвичай зберігається як документ JSON. Для кожного елемента, значення «Слово» ,«Переклад» та «Категорія», зберігаються в якості атрибутів в єдиному документі. У такій моделі дані оптимізовані для інтуїтивно зрозумілою розробки і горизонтальної масштабованості. NoSQL часто пропонують компроміс, пом'якшуючи жорсткі вимоги властивостей нормальних форм заради більш гнучкої моделі даних. Завдяки цьому, такий тип СУБД – відмінний вибір для прикладів використання з високою пропускною здатністю і низькою затримкою.

Об'єктно-орієнтовані API дозволяють розробникам додатків без праці здійснювати запис і витяг структур даних, розміщених в пам'яті. Завдяки використанню ключів секцій, додатки можуть вести пошук по параметрах «ключ-значення», наборам стовпців або частково структурованим документів, що містять серійні об'єкти і атрибути додатків.

Під час концептуального моделювання було виявлено основні об’єкти системи та їх атрибути.

Основними таблицями в розроблюваній системі є:

– словник;

– користувач;

– мова;

– слово;

– користувач та словник.

На основі проведеного аналізу побудовано схему основних таблиць бази даних (рисунок 4.6).

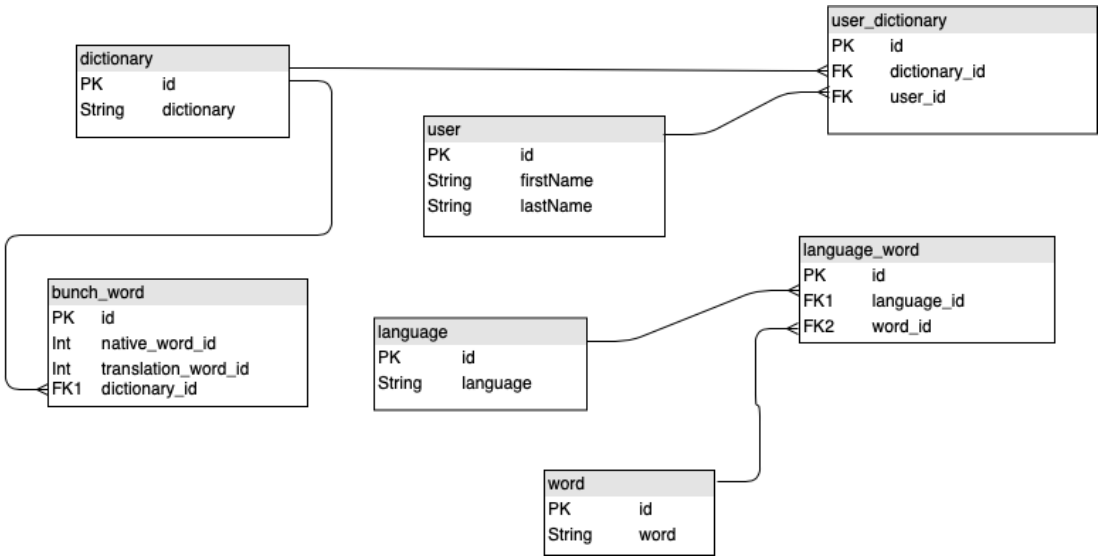


Рисунок 4.6 – Діаграма класів

В результаті концептуального моделювання було встановлено, що таблиця «користувач» має наступні атрибути:

– ідентифікатор користувача;

– ім’я;

– прізвище.

Сутність «користувач та словник» має наступні атрибути:

– ідентифікатор користувача та словника;

– ідентифікатор словника;

– ідентифікатор користувача.

Сутність «словник» має наступні атрибути:

– ідентифікатор словника;

– словник.

Сутність «зв’язка слів» має наступні атрибути:

– ідентифікатор зв’язки;

– слово рідної мови;

– іноземне слово;

– ідентифікатор словника.

Сутність «мова» має наступні атрибути:

– ідентифікатор мови;

– мову.

Сутність «слово» має наступні атрибути:

– ідентифікатор слова;

– слово.

Сутність «мова та слово» має наступні атрибути:

– ідентифікатор мови та слова;

4.5 Створення UI / UX або іншого дизайну системи

Для сучасного інтернет проекту UX-дизайн відіграє важливу роль і визначає успішність його розвитку. Враховувати його ключові моменти потрібно на всіх етапах розробки. UX дизайн (User eXperience Design) – застосування методів проектування, спрямованих на досягнення максимально ефективної взаємодії користувача з системою.

Розбролюваний додаток має наступні вимоги до UX-дизайну:

– користувач може легко перейти в режим розпізнавання в реальному часі;

– користувач має змогу легко знайти потрібний йому елемент в мобільному.

ВИСНОВКИ

Метою даної кваліфікаційної роботи було розробити мобільний додаток, який буде використовувати методи розпізнавання та ідентифікації об’єктів у доповненої реальності та перекладати назви об’єктів на іноземну мову.

В ході даної роботи проведено докладне дослідження предметної області, обґрунтовано доцільність розробки системи, проведено аналіз існуючих фреймворків доповненої реальності, описано принципи її роботи та виявлено основні функції.

Реалізація проекту відбувалася за допомогою мов програмування Swift та Objective-C, використовувалися фреймворки: Cocoa Touch – для клієнтської роботи, Core ML – для роботи з штучними нейронними мережами, ARKit – для роботи з доповненою реальністю, Alamofire – для роботи з Google API. Інтерфейс комунікації між сторонами бекенду та фронтеду побудовано за допомогою протоколу передачі JSON об’єктів. Сервер має REST-ful архітектуру. В якості NoSQL-СУБД виступає Realm.

Особливості технології розробки дозволяють при необхідності здійснити перенесення системи на платформи, відмінні від тієї, для якої велася розробка, зі збереженням функціонала.

Мобільний додаток може бути використаний у повсякденному житті: розпізнавання та переклад у реальному часі, закріплення нових вивчених слів за допомогою карток та функції інтервального повторення, а доповнена реальність робить досвід використання застосунків більш новим та цікавим.

В мобільний додаток впроваджено модель машинного навчання, здатну найточніше розпізнавати зображення серед усіх інших моделей та надавати їй величезну практичну користь.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Core ML. URL: https://developer.apple.com/documentation/coreml (дата звернення: 25.04.2024).

2. CocoaTouch. URL: https://developer.apple.com/library/archive/documentation/General/Conceptual/DevPedia-CocoaCore/Cocoa.html (дата звернення: 25.04.2021).

3. Alamofire. URL: https://github.com/Alamofire/Alamofire (дата звернення: 25.04.2021).

4. AWS vs. Azure vs. Google: Cloud. URL: https://www.datamation.com/cloud-computing/aws-vs-azure-vs-google-cloud-comparison.html (дата звернення: 25.04.2024).

5. Ian Goodfellow, Yoshua Bengio, Deep Learning (Adaptive Computation and Machine Learning series) // The MIT Press – p. 621-638, 2016.

6. Frank Millstein, Deep Learning: 2 Manuscripts – Deep Learning With Keras And Convolutional Neural Networks In Python // Paperback – p.117-129, 2018

7. Apple Developer Documentation: Creating Core ML. URL: https://developer.apple.com/documentation/coreml (Last accessed: 15.04.2021).

8. Apple Developer Documentation: Creating Core ML. URL: https://developer.apple.com/machine-learning/ (Last accessed: 15.04.2021).

9. Lawrence J. Introduction to neural networks: design, theory and applications. California Scientific Software, 1994.

10. Duda R. O, Hart P. E, D. G. Stork. Pattern classification. Wiley, 2001.

11. Котов А., Красильников Н. Кластеризация данных. 2006.

12. Информационно-аналитический ресурс, посвященный машинному обучению, распознаванию образов и интеллектуальному анализу данных. URL: www.machinelearning.ru (дата звернення: 03.04.2024).

13. Розенблатт Т. Принципы нейродинамики: Перцептроны и теория механизмов мозга. 1965. 480 с.

14. Machine Learning Proceedings 1991: Proceedings of the Eighth International Workshop (ML91). Elsevier Science. 2014. 364 с.

15. Нейронні мережі в матеріалознавстві // ISIJ International.1999. С. 966–979.

16. Cybenko G. V. Approximation by Superpositions of a Sigmoidal function. 2006. 314 с.

17. Функции активации нейросети. URL: https://neurohive.io/ru/osnovy-data-science/activation-functions (дата звернення: 29.03.2024).

18. Krizhevsky A., Sutskever I. Advances in Neural Information Processing Systems. 2012.1097 с.

19. Hubel D. H., Wiesel D.H. Brain and visual perception: the story of a 25-year collaboration. Oxford University, 2005. 106 с.

20. Meier U., Ciresan D. Multi-column deep neural networks for image classification. New York. 649 с.

21. Visual Geometry. URL: http://www.robots.ox.ac.uk/~vgg/research/very_deep/.

22. Inception-v3. URL: https://medium.com/@sh.tsang/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c (дата звернення: 15.04.2021).

23. Belongie S., Wilber M., Viet. A Residual Networks Behave Like Ensembles of Relatively Shallow Networks. 2016.

24. Benchmark Analysis of Representative Deep Neural Network Architectures. URL: https://arxiv.org/pdf/1810.00736.pdf (дата звернення: 15.04.2021).