Київський національний торговельно-економічний університет
Кафедра інженерії програмного забезпечення та кібербезпеки

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЕКТ

на тему:

«Використання технологій розпізнання образів для розробки додатку Smart Home»

Студента 2м курсу, 6м групи, спеціальності 121 «Інженерія програмного забезпечення» спеціалізації «Інженерія програмного забезпечення» _____ підпис студента — Ганах Олександр Ігорович

Науковий керівник кандидат економічних наук, доцент кафедри інженерії програмного забезпечення та кібербезпеки _____ підпис керівника — Палагута Катерина Олексіївна

Гарант освітньої програми доктор технічних наук, професор кафедри інженерії програмного забезпечення та кібербезпеки _____ підпис гаранта — Криворучко Олена Володимирівна

КИЇВ - 2020

Київський національний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення та кібербезпеки
Освітній ступінь магістр
Спеціальність 121 «Інженерія програмного забезпечення»

Затверджую
Зав. кафедри інженерії програмного забезпечення та кібербезпеки
Криворучко О. В.
8 жовтня 2019 р.

Завдання
на випускний кваліфікаційний проект студентові

Ганаха Олександра Ігоровича
(прізвище, ім'я, по батькові)

1. Тема випускного кваліфікаційного проекту «Використання технологій розпізнання образів для розробки додатку Smart Home»

Затверджена наказом ректора від "13" грудня 2019 р. № 4304

2. Строк здачі студентом закінченої проекту 01 грудня 2020

3. Цільова установка та вихідні дані до проекту

Мета проекту розробка та впровадження додатку розпізнавання образів для систем Smart Home

Об'єкт дослідження функціонування систем Smart Home

Предмет дослідження методи та алгоритми розпізнавання образів

4. Консультанти проекту із зазначенням розділів, які консультують:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв

5. Зміст випускного кваліфікаційного проекту (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ СИСТЕМ SMART HOME

1.1. Характеристика систем Smart Home

1.2. Аналіз ринку систем Smart Home

1.3. Безпека систем Smart Home

1.4. Висновки до розділу 1

РОЗДІЛ 2 ВИБІР ПРОГРАМНИХ ЗАСОБІВ ТА ПРОЕКТУВАННЯ ДОДАТКУ

2.1. Характеристика задачі розпізнавання образів

2.2. Мова програмування Python

2.3. Засоби розробки програм на Python

2.4. Проектування додатку розпізнавання образів

2.6. Висновки до розділу 2

РОЗДІЛ 3 РОЗРОБКА ДОДАТКУ РОЗПІЗНАВАННЯ ОБРАЗІВ

3.1. Основні бібліотеки Python для роботи з зображеннями

3.2. Характеристика механізму дії додатку розпізнавання образів

3.3. Опис реалізації додатку розпізнавання образів

3.6. Висновки до розділу 3

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ТЕХНІЧНЕ ЗАВДАННЯ

ТЕСТУВАННЯ ДОДАТКА

6. Календарний план виконання проекту

№ пор.	Назва етапів випускного кваліфікаційного проекту	Строк виконання етапів проекту	
		за планом	фактично
1	2	3	4
1.	*Вибір теми випускного кваліфікаційного проекта*	*20.09.2019*	*20.09.2019*
2.	*Розробка та затвердження завдання на проект магістра (Наказ №185 від 07.11.19)*	*08.11.2019*	*08.11.2019*
3.	*Вступ та перелік літературних джерел*	*24.01.2020*	*24.01.2020*
4.	*Наукова стаття*	*01.09.2020*	*01.09.2020*
5.	*Технічне завдання*	*28.02.2020*	*28.02.2020*
6.	*Розділ 1. Аналіз предметної області систем Smart Home*	*25.06.2020*	*25.06.2020*
7.	*Розділ 2. Вибір програмних засобів та проектування додатку*	*07.09.2020*	*07.09.2020*
8.	*Розділ 3. Розробка додатку розпізнавання образів*	*19.10.2020*	*19.10.2020*
9.	*Програма та методика тестування*	*21.10.2020*	*21.10.2020*
10.	*Керівництво користувача*	*23.10.2020*	*23.10.2020*
11.	*Висновки та пропозиції*	*30.10.2020*	*30.10.2020*
12.	*Здача випускного кваліфікаційного проекту на кафедру (перша перевірка)*	*05.11.2020*	*05.11.2020*
13.	*Підготовка автореферату та презентації доповіді*	*05.11.2020*	*05.11.2020*
14.	*Попередній захист випускного кваліфікаційного проекту*	*25.11.2020-27.11.2020*	*25.11.2020-27.11.2020*
15.	*Зовнішнє рецензування випускного кваліфікаційного проекту*	*01.12.2020*	*01.12.2020*
16.	*Підготовка до публічного захисту випускного кваліфікаційного проекту*	*10.12.2020-11.12.2020*	

7. Дата видачі завдання «8» листопада 2019 р.

8. Науковий керівник випускного кваліфікаційного проекту Палагута К.О.

(прізвище, ініціали, підпис)

9. Гарант освітньої програми Криворучко О.В.

(прізвище, ініціали, підпис)

10. Завдання прийняв до виконання студент Ганах О.І.

(прізвище, ініціали, підпис)

11. Відгук керівника випускного кваліфікаційного проекту

Науковий керівник випускного кваліфікаційного проекту
_____ Палагута К.О.
(підпис, дата)

Відмітка про попередній захист _____
(ПІБ, підпис, дата)

12. Висновок про випускний кваліфікаційний проект

Випускний кваліфікаційний проект студента _____Ганах О.І._____
(прізвище, ініціали)

може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми _____ Криворучко О. В.
(прізвище, ініціали, підпис)

Завідувач кафедри _____ Криворучко О. В.
(підпис, прізвище, ініціали)

«____»_____20 ___ р.

АНОТАЦІЯ

Відповідно до мети дослідження робота присвячена розробці додатку для розпізнавання образів системи Smart Home. Для реалізації додатку було побудовано згорткову нейронну мережу на мові програмування Python з використанням бібліотек OpenCV, TensorFlow, Keras, NumPy, h5py, SciPy, Pillow, Matplotlib, ImageAI, а також IDE PyCharm і хмарний сервіс Google Colaboratory. Розроблений додаток дозволяє полегшити процес ідентифікації осіб та транспортних засобів на зображеннях, знятих пристроями системи Smart Home, дає змогу обробити зображеннях, що містяться у файлах будь-якого з графічних форматів, а такж відеофайли.

Додаток розпізнавання образів системи Smart Home розроблено, протестовало, мона рекомендувати до експлуатації.

Ключові слова:система Smart Home, розпізнавання образів, згорткова нейронна мережа, мова програмування Python, безпека систем Smart Home.

ANNOTATION

In accordance with the purpose of the reseach, the work is devoted to the development of an application for image recognition of the Smart Home system. To implement the application, a convolutional neural network in the Python programming language was built using the OpenCV, TensorFlow, Keras, NumPy, h5py, SciPy, Pillow, Matplotlib, ImageAI libraries, as well as the PyCharm IDE and the Google Collaborative cloud service. The developed application makes it easier to identify people and vehicles on images taken by Smart Home devices, allows you to process images contained in files of any of the graphic formats, as well as video files.

The Smart Home image recognition application has been developed, tested and recommended for use.For the interaction of these parts, additional services and tools were used. The data is stored in a relational MySQL database.

The finished website has been successfully tested in accordance with the functional requirements.

Keywords: Smart Home system, pattern recognition, convolutional neural network, Python programming language, security of Smart Home systems.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (англ. Application Program Interface) – структура для створення служб HTTP

IDE (англ. Integrated Development Environment) – інтегроване середовище розробки програмного забезпечення

SMD (англ. Smart Home Device) – пристрій для системи Smart Home

БД – база даних

ПК – персональний комп'ютер

IDC (англ.International Data Corporation) – міжнародна дослідницька та консалтингова компанія, яка займається дослідженням всесвітнього ринку інформаційних технологій і телекомунікацій

ЗМІСТ

					КНТЕУ 121 06-05.МР			
Зм.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*				
Зав. каф.		Криворучко О.В.		*19.10.20*	Використання технологій розпізнання образів для розробки додатку Smart Home	*Стадія*	*Аркуш*	*Аркушів*
Керівник		Палагута К.О.		*19.10.20*		*Зміст*	*3*	*45*
Гарант		Криворучко О.В.		*19.10.20*		*Факультет інформаційних технологій 2м курс, 6 група*		
Розробив		Ганах О.І.		*19.10.20*	*Зміст*			

ВСТУП

Актуальність. У наш час системи Smart Home швидко розвиваються, використовуються для забезпечення комфорту, зручності, якості життя та безпеки для мешканців. Як свідчать данні аналізу попиту на пристрої для систем Smart Home популярність таких систем постійно зростає у всьому світі. У майбутньому системи розумного будинку повинні стати більш інтелектуальними, включати більш дружні сервіси, ніж просто управління приладами оселі. Реалізація цих задач неможлива без більш активного застосування технологій розпізнавання образів. Розпізнавання образів надає додаткові можливості для створення нових сервісів і послуг у системах Smart Home, наприклад, стає можливим у режимі реального часу обробляти інформацію, що надходить з зовнішньої відеокамери і відповідно до різних обставин діяти.

Мета дослідження: розробка додатку розпізнавання образів для системи Smart Home.

Об’єкт дослідження: функціонування систем Smart Home.

Предмет дослідження: методи, алгоритми, технологіїрозпізнавання образів.

У відповідності з метою дослідження поставлені наступні завдання:

• Дослідити як більш широке застосування технологій розпізнавання образів сприятиме підвищенню інтелектуалізації, ефективності, безпеці систем Smart Home.

- Дослідити сучасні методи і технології розпізнавання образів, обґрунтувати вибір методу розпізнавання образів для систем Smart Home.
- Проаналізувати існуючі засоби розробки програмного забезпечення, обґрунтувати вибір засобів для розробки додатку.
- Розробити додаток розпізнавання образів згідно з технічним завданням.
- Протестувати додаток.

Методи дослідження: аналіз, синтез, моделювання, спостереження, експеримент.

Наукова новизна дослідження полягає в удосконаленні систем Smart Home за допомогою застосування технологій розпізнавання образів, що ґрунтуються на згортковій нейронній мережі.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ СИСТЕМ SMART HOME

1.1.Характеристика систем Smart Home

Розумні побутові пристрої, об'єднані в єдину мережу і здатні виконувати рутинні дії без участі людини, покликані зробити життя більш комфортним та економічним. Системи Smart Home бувають декількох модифікацій, але по суті здійснюють однакові функції:

– включення і вимикання світла;

– регулювання вологості;

– контроль датчиків пожежної безпеки;

– аналіз інформації з датчиків руху;

– відкриття дверей;

– вибір музики;

– голосовий пошук в мережі;

– включення і вимикання кухонних приладів;

– автоматизація опалення:

– GSM-моніторинг (відстеження змін і передача команд за допомогою мобільного телефону).

Проведення автоматизації і диспетчеризація надає можливість системі Smart Home повністю контролювати процес експлуатації усіх приладів за

допомогою єдиної панелі управління, куди надходить уся інформація про робочий стан системи

Автоматизація будинку - це сукупність програмно-апаратного обладнання, здатного об'єднати і уніфікувати процес управління різними системами і приладами в будинку. Дана технологія отримала широку популярність в країнах Європи і США, а також в Україні.

Диспетчеризація будівель - це автоматизований процес збору інформації про робочі процеси системи в єдине сховище даних, звідки власник або система Smart Home може здійснювати контроль роботи всіх інженерних споруд і приладів.

Основною особливістю систем Smart Home вважається гнучкість, можливість індивідуальної настройки. Включати в мережу можна як кілька базових приладів і датчиків, так і безліч додаткових пристроїв: розумні кавоварки, пральні машини, колонки для голосового управління і багато іншого. Існує кілька типів розумної автоматизації, що відрізняються способом підключення і налаштування.

Встановити розумну систему в квартирі або будинку можна як самостійно, так і за допомогою фахівців. Деякі з систем вимагають ретельного планування ще на етапі ремонту приміщення, інші - підключаються в уже облаштованому житлі. Види розумних будинків:

- провідні;
- бездротові;
- з єдиним центром управління;
- децентралізовані;
- з відкритим протоколом;
- з закритим протоколом.

Провідні системи відрізняються стабільністю роботи, але вони складні в установці та налаштуванні. Планувати монтаж такої автоматизації розумного будинку потрібно починати ще на етапі будівництва.

Бездротові системи легше встановити, і вони коштують менше. Однак сигнал, що передається по радіоканалу, менш стабільний.

Централізовані автоматизовані системи управляються єдиним центром, дозволяють програмувати складні сценарії. Але, на жаль, вони занадто залежні від роботи самого контролера.

Децентралізовані системи більш надійні, оскільки пристрою не залежать один від одного. Кожне з них забезпечено мікропроцесором з власною пам'яттю.

Автоматизовані системи з відкритим протоколом (наприклад, KNX) дозволяють підключати пристрої різних виробників, що істотно розширює коло вибору. Закритий протокол передбачає використання продукції однієї фірми.

Сучасний розумний будинок переважно управляється голосом, датчиками руху та програмами для смартфонів. Голосові команди та програми для смартфонів безумовно панують у розумному домі 2020 року. Вони забезпечують швидкий доступ до управління пристроями, але іноді можуть бути незграбними, неінтуїтивними у використанні, якщо власник не пам’ятає точно потрібну фразу. Програми для смартфонів, такі як Google Home та Philips Hue, працюють досить добре, але за своєю суттю додають кілька кроків, щоб щось зробити. А саме, потрібно знайти телефон, розблокувати його, знайти та відкрити потрібну програму, а потім взяти під контроль свій розумний дім. Однак голос поступово стає більш особистим, завдяки тому, що Alexa і Google Assistant можуть розпізнавати голос власника. Таким чином, вони можуть виконувати дії, характерні для господаря, замість когось іншого у певному домогосподарстві.

Для розумного управління будинком активно використовуються різноманітні датчики. Вони бувають у декількох формах, причому найпоширеніші використовуються для попередження, якщо двері чи вікно відчинені, коли власник не вдома. Часто використовуючи магніти, це

прості пристрої, які мало що перешкоджають контексту, але принаймні можуть попередити власника про можливість потенційної крадіжки або сповістити, якщо він випадково залишите вхідні двері відчиненими. Інші поширені датчики розумного будинку призначені для виявлення руху. Ця технологія має декілька застосувань, наприклад, попереджає про можливий прорив, виявляючи рух у зачиненій кімнаті, та вмикаючи світло, коли проходиш повз. Це може бути прожектор для відлякування порушників на вулиці або м’яке світло передпокою, яке допоможе знайти дорогу до ванної кімнати вночі. Створюючи враження розумності, ці датчики нічого не знають про те, кого (або навіть що) бачили.

На теперішній час у системах Smart Home все більш активно використовуються засоби біометричної аутентифікації як складової технологій розпізнавання образів. Зараз вже доступні дверні замки із відбитками пальців. Samsung пропонує невеликий асортимент таких замків, але їм бракує розумної інтеграції зі Smart Home. За допомогою цих замків Alexa (або навіть голосовий асистент Samsung Bixby) не може прийняти господаря додому та автоматизувати його розумні домашні пристрої, коли він розблокуєте двері одним дотиком пальця.

Розпізнавання обличчя є в розумному будинку вже кілька років, а прихована камера охорони Netatmo Welcome пропонує технологію з 2015 року. Камера сповіщає господаря про кожне нове обличчя, яке бачить, і потім він можете ідентифікувати його як членів сім'ї. Як тільки система отримує цю інформацію, камера далі буде попереджати господаря коли помітить незнайомця. Розпізнавання обличчя - це також особливість камер безпеки від Arlo та Nest, включаючи Nest Hello, відеодзвінок, який не тільки повідомляє, що хтось біля дверей, але і хто це, перш ніж відповісти. У цих випадках розпізнавання обличчя використовується для інформування господаря про те, що або кого бачив ваш розумний дім, а не для того, щоб будинок працював, коли він впізнав власника.

Google починає експериментувати з цим через свій підрозділ Nest за допомогою нового Nest Hub Max, 10-дюймового сенсорного екрану та розумної колонки, що використовується для управління розумним будинком. За допомогою камери розумний дисплей розпізнає обличчя власника та пропонує особисту інформацію, подібно до того, як це робить звичайний домашній центр після розпізнавання голосу. Що важливо, Nest Hub Max робить це без запитання власника, відображає інформацію, що стосується власника. У світі, де вже використовується обличчя, щоб розблокувати смартфони та увійти в комп'ютери з Windows, це не здається занадто великим стрибком. Тільки тоді, коли Google Assistant привітає вас вголос і підкаже ваш улюблений список відпочинку після роботи, це стане відчутним кроком вперед для розумного будинку.

Apple також вивчає цей сектор розумного будинку. Патент, вперше поданий в 2017 році і оприлюднений на початку 2019 року, описує пристрій, який можна припустити як майбутній HomePod, з різними датчиками та камерами для «збору жестів руками та інших тривимірних входів жестами». У патенті описано, як «особистість осіб, які перебувають поблизу [пристрою], можна визначити за допомогою розпізнавання обличчя». Для цього пристрій може використовувати «ехолокацію, інфрачервоні камери, системи виявлення погляду, ємнісні датчики наближення, оптичні датчики наближення тощо».

Однією з перших практичних реалізацій розпізнавання обличчя у розумному будинку стала інтелектуальна сенсорна панель Elan, презентована у 2019 році. Сенсорна панель Elan призначена для розпізнавання господаря під час наближення, а потім відображає спеціальне меню опцій та керує різними розумними домашніми пристроями. Наприклад, її можна запрограмувати, щоб налаштувати освітлення та термостат на бажану настройку, опустити жалюзі на вікна та увімкнути улюблену музику, коли власник проходить через двері. Під час

запуску продукту Білл Хенслі з Nortek Security & Control, що виробляє сенсорну панель Elan, сказав: «Ми інтегруємо вдосконалену відеоаналітику та розпізнавання облич із нашими технологіями розумного управління будинком ... розумний дім стає все інтуїтивнішим та персоналізованішим» [1]. Подібна система, безсумнівно, виводить розумний дім вперед, наближаючи його до майже неминучої точки, коли більшість наших підключених пристроїв інтуїтивно реагують на нашу присутність і наші потреби. Безумовно, майбутнє розумного будинку - це те, яке налаштовується автоматично, з дуже незначним вкладом від самого користувача; розпізнавання обличчя - один із багатьох кроків, які дозволять досягти цієї мети.

1.2. Аналіз ринку систем Smart Home

Світовий ринок пристроїв розумного будинку (Smart Home Device, SMD) збереже позитивну динаміку, незважаючи на наслідки пандемії COVID-19, прогнозують аналітики IDC (International Data Corporation). Відповідно до їх оцінок, у 2020 році глобальні поставки SMD збільшяться на 4,1% і досягнуть 854 млн штук.

У майбутньому споживачі продовжать збільшувати витрати на техніку і сервіси, що дозволяють автоматизувати побут і підвищити комфорт, що забезпечить подальше зростання ринку. Постачання пристроїв розумного будинку будуть в середньому збільшуватися на 14% і в 2024 року перевищать 1,4 млрд штук. Найбільшою категорією на даному ринку залишаються пристрої відеорозваг, до яких відносяться розумні телевізори і розширювач. У 2020 році їх відвантаження будуть близькі до 354 млн штук, з урахуванням чого ринкова частка категорії складе 41,4%. Фахівці вважають, що середньорічне зростання в цьому сегменті в наступні п'ять років складе 6,3%, а поставки в 2024 року перевищать 451 млн одиниць. Близько 60% в сумарному обсязі займуть телевізори.

Серед чинників, які сприятимуть покупці нових пристроїв, аналітики назвали зниження цін на моделі з більш просунутими характеристиками, такими як підтримка 8K, HDR, висока частота оновлення, вбудований цифрові асистенти і ін. Крім того, нові сервіси, у тому числі ігрові, також зіграють позитивну роль з точки зору стимулювання попиту.

Другим за величиною сегментом будуть системи домашнього відеоспостереження і безпеки. У 2020 році їх поставки очікуються на рівні 166 млн штук, що відповідає частці 19,5%. З урахуванням середньорічного зростання більш ніж на 16% обсяг категорії до кінця 2024 року зросте до 303,5 млн пристроїв, а ринкова частка - до 21,1%.

У трійку найбільш затребуваних видів SMD увійдуть також розумні колонки. Поставки смарт-динаміків в найближчі п'ять років в середньому збільшуватися на 11,1% і в 2024 році може досягнуть майже 204 млн штук.

Динаміка зростання ринку пристроїв для систем Smart Home за даними Global Businnes Data Platform Statista [2] проілюстрована на рис.1.1.

За даними Statista, ринок нових технологій для будинків активно зріс у США - зафіксовано 23 577 мільйонів доларів, за ними йдуть 16 241 мільйон доларів у Китаї, 4127 мільйонів доларів у Великобританії, 4113 мільйонів доларів у Німеччині та 3814 мільйонів доларів у Японії. Однак найближчим часом дохід збільшиться і в інших країнах.

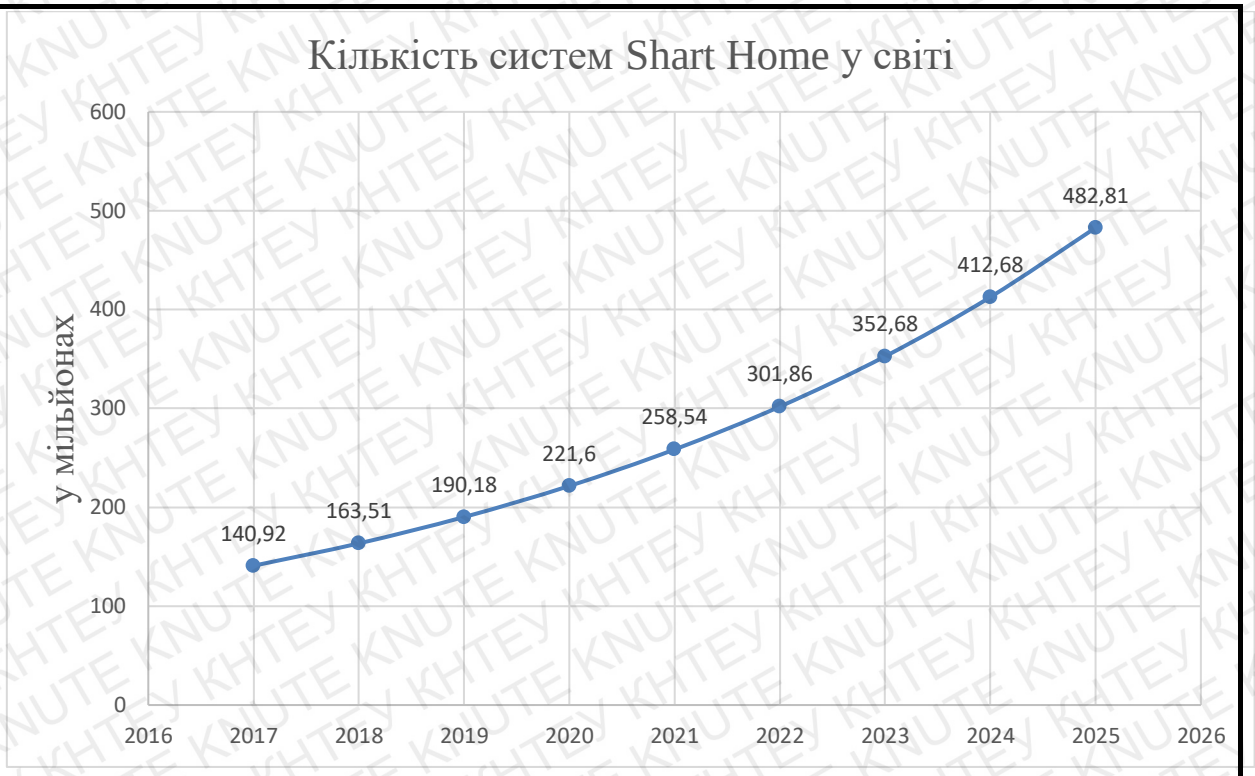

Рис.1.1. Кількість систем Shart Home у світі,
звітні та прогнозні дані за період з 2017 по 2025 рр.

** складено на основі даних Global Businnes Data Platform Statista [2]*

У січні-березні 2020 року в Європу надійшло трохи більше 22 млн домашніх смарт-пристроїв, включаючи розумні колонки, смарт-телевізори, інтелектуальні системи освітлення, відеоспостереження та розумні термостати. У порівнянні з тим же періодом 2019 р. відвантаження збільшилися на 3,8%. Характеризуючи динаміку, старший аналітик IDC по ринку Smart Home Західної Європи Антоніо Арантес (Antonio Arantes) зазначив, що в різних сегментах вона була неоднорідною. Зокрема, в першому кварталі відчутно збільшилися поставки розумних колонок. За оцінками IDC, в Європу надійшло 3,9 млн смарт-динаміків, що на 17,4% більше показників річної давності. Особливо вражаючою надбавка була в категорії смарт-дисплеїв - зростання тут вимірювався тризначними показниками.

Що стосується розстановки сил між вендорами, то найбільшим постачальником Smart Home систем була Google. За оцінками IDC, компанія відвантажила 3,7 млн пристроїв для розумного будинку або 16,8% від сумарного числа. На другому місці за цим показником Amazon з результатом 3,48 млн пристроїв або 15,8%. У п'ятірку лідерів також увійшли Samsung, LG Electronics і Sony, внесок яких склав 10,9%, 9% і 4,7% відповідно.

1.3. Безпека систем Smart Home

Розумна домашня система безпеки може підключатися до мережі Wi-Fi, щоб користувач міг контролювати свої захисні пристрої за допомогою смартфона та програмного забезпечення. Системи початкового рівня зазвичай включають деякі датчики дверей і вікон, детектор руху та концентратор, який взаємодіє з цими пристроями за допомогою одного або декількох бездротових протоколів, таких як Wi-Fi, Z-Wave, Zigbee або власна мережа. Можна також додати додаткові датчики дверей, руху та вікон, щоб забезпечити покриття всього будинку, і побудувати комплексну систему, яка включає дверні замки, дверні відкривання гаражних дверей, внутрішні та зовнішні камери спостереження, ліхтарі, сирени, детектори диму / CO, датчики води і більше.

У ідеальному світі всі компоненти домашньої безпеки використовують один і той же стандарт бездротового зв'язку для зв’язку з основним концентратором, але такі фактори, як вимоги до потужності, діапазон сигналів, ціна та розмір, унеможливлюють практично застосування одного протоколу. Наприклад, менші компоненти, такі як датчики дверей / вікон, зазвичай використовують технологію Z-Wave або Zigbee, оскільки вони не вимагають багато енергії і можуть живитися від менших батарейок. Вони також працюють у сітчастій топології та можуть допомогти розширити діапазон мережевих пристроїв. Однак жоден протокол не забезпечує

пропускну здатність, яку користувач отримує через Wi-Fi, саме тому він зазвичай використовується в камерах безпеки для забезпечення плавного потокового передавання відео та в інших пристроях. Більше того, пристрої Z-Wave та Zigbee підключаються та управляються за допомогою концентратора, тоді як пристрої Wi-Fi можна підключати безпосередньо до домашньої мережі та керувати ними за допомогою програми. Нарешті, пристрої Z-Wave та Zigbee використовують шифрування AES 128, і оскільки вони працюють у закритій системі із виділеним концентратором, вони забезпечують більший рівень безпеки, ніж пристрої Wi-Fi.

Практично будь-яка система безпеки Smart Home пропонує компоненти, які працюють разом у спільному середовищі та ними можна маніпулювати за допомогою індивідуальних правил. Наприклад, можна створити правила включення світла при виявленні руху, розблокування дверей, коли спрацьовує димова сигналізація, і зробити так, щоб камера починала записувати при спрацьовуванні датчика. Деякі системи зберігають записане відео локально на SD-карті або твердотільному накопичувачі, тоді як інші пропонують хмарне зберігання. Хмарне сховище дозволяє легко зберігати та отримувати доступ до записаного відео, але це може коштувати сотні доларів на рік. Деякі системи пропонують як хмарне, так і локальне сховище, а деякі - спеціальний накопичувач, який надає можливості відеореєстратора із тимчасовим записом, що полегшує пошук відеоподії, яка відбулася в певний момент часу.

Усі проаналізовані нами системи мають додаток, який дозволяє використовувати смартфон як командний центр для озброєння та зняття з охорони системи, створення правил, додавання та видалення компонентів та отримання сповіщень, коли спрацьовують тривоги. Більшість програм також дозволяють робити такі речі, як перегляд відео в прямому ефірі або записаного відео, блокування та розблокування дверей, зміна налаштувань

термостата та приглушення сигналізації. Деякі програми навіть використовують служби локації телефону для автоматичного включення та зняття з охорони системи відповідно до фізичного місцезнаходження власника. Дорожчі системи, як правило, оснащені настінною панеллю, що виконує роль комунікаційного центру, з сенсорним дисплеєм. Дисплей дозволяє спілкуватися з професійною службою моніторингу при спрацьовуванні сигналізації та переглядати відео з будь-якої з встановлених камер безпеки.

Майже всі найновіші домашні системи домашньої безпеки пропонують підтримку голосового управління через Amazon Alexa, Google Assistant, а в деяких випадках і Apple Siri, що дозволяє розблокувати двері, змінити налаштування термостата, відкрити гараж і підняти або обеззброїти систему за допомогою голосової команди на підключений пристрій, такий як розумний динамік. Багато з них також пропонують підтримку аплетів IFTTT (If This Then That), які використовують тригери, сумісні з IFTTT, веб-службами та пристроями для створення дії. Наприклад, можна створити аплет, який повідомляє, якщо відкрити гаражні ворота, щоб увімкнути прожектор.

Зовнішня камера ідеально підходить для спостереження за тим, що відбувається поза домом. Ці пристрої захищені від атмосферних впливів і, як правило, потребують сусідньої розетки GFCI (переривник ланцюга замикання на землю) для живлення, хоча є кілька моделей, що працюють від акумуляторів. Як і їх внутрішні аналоги, зовнішні камери підключаються до вашої мережі Wi-Fi і дозволяють переглядати відео в реальному часі з вашого телефону. Більшість зовнішніх камер, таких як Arlo Ultra, пропонують виявлення руху за допомогою push-повідомлень та сповіщень по електронній пошті, нічного бачення та хмарного сховища для відео, а деякі, як Ring Floodlight Cam, виконують подвійний обов'язок

як прожектори або ліхтарі. Деякі моделі можуть навіть відрізнити автомобіль, що проїжджає повз, тварину та людину.

Відео дзвінки пропонують простий спосіб побачити, хто біля дверей, не відкриваючи і навіть не наближаючись до дверей. Ці пристрої підключаються до мережі Wi-Fi і надсилатимуть сповіщення, коли хтось підійде до дверного отвору. Вони записуватимуть відео при натисканні на дверний дзвінок або при виявленні руху, і, як правило, пропонують двосторонній аудіозв'язок, що дозволяє розмовляти з відвідувачем з будь-якого місця за допомогою телефону. Більшість дзвіночків для відеодзвінків використовують існуючу проводку дверного дзвінка (два дроти низької напруги) і досить прості в монтажі, але є моделі на батарейках, які встановлюються за лічені хвилини. Деякі працюють з іншими розумними пристроями, такими як дверні замки та сирени, і підтримують голосові команди IFTTT та Alexa.

Розумний замок, як правило, є частиною надійної інтелектуальної системи захисту будинку, але для його використання не потрібно вкладати гроші в повноцінну систему. Якщо використовується концентратор домашньої автоматизації для управління такими речами, як освітлення та термостати, можна без особливих зусиль додати до системи розумний замок Z-Wave або Zigbee. Деякі моделі використовують існуючий апаратний ключ із циліндрами та засувками та кріпляться до внутрішньої сторони дверей, тоді як інші вимагають видалення існуючих внутрішніх та зовнішніх клавіш та заміни засувки та засувки. Розумні замки можна відкривати та закривати за допомогою мобільного додатка, і вони надсилатимуть сповіщення, коли хтось замикає або відмикає двері, і більшість з них дозволяють створювати графіки постійного та тимчасового доступу для членів сім'ї та друзів на основі конкретних годин дня та днів тиждень. Серед функцій, на які слід звернути увагу, є геозонування, яке використовує служби локації телефону для блокування та розблокування

дверей, голосова активація за допомогою голосових команд Siri (HomeKit), Google Assistant або Amazon Alexa, підтримка IFTTT та інтеграція з іншими розумними домашніми пристроями, такими як відеодзвінки, зовнішні камери, термостати, димові сигналізатори та підключене освітлення. На вибір є безліч моделей розумних замків, включаючи замки без дотику до клавіш, замки з сенсорним екраном, комбіновані замки та замки на сенсорній панелі, а також замки, які можна відкрити за допомогою біометричного зчитувача відбитків пальців.

Як і будь-який продукт, який підключається до Інтернету та використовує бездротові технології, розумні системи безпеки будинку вразливі до злому, особливо системи, які не мають шифрування. Хакери можуть сидіти біля будинку та використовувати ноутбук та програмне забезпечення для перехоплення бездротових сигналів, що надходять від системи Smart Home, що дозволяє їм придушувати тривоги та вимикати датчики. Інші пристрої дозволяють хакерам генерувати радіошум, який може перешкоджати зв'язку між датчиками та концентратором. Крім того, пристрої, які підключаються через Wi-Fi, такі як камери безпеки та розумні дверні замки, можна зламати, щоб отримати доступ до домашньої мережі. Потім кваліфікований хакер може використовувати пристрої Wi-Fi та інші мережеві ресурси для здійснення атак DDoS з розподіленою відмовою в обслуговуванні проти великих мереж. Можливо, ще більш тривожною є ідея про якісь незнайомі відеоспостереження з внутрішніх і зовнішніх камер охорони.

Існує кілька кроків, щоб переконатися, що система домашньої безпеки захищена від зловмисних кібер-зловмисників. Слід замінити системний пароль за замовчуванням на унікальний, що містить поєднання літер, цифр та символів. Якщо можливо, час від часу потрібно змінювати пароль. Потрібно переконатися, що домашня мережа захищена. Слід перевірити налаштування безпеки на власному бездротовому маршрутизаторі та

розглянути моделі, які додають додатковий рівень програмного захисту, наприклад Bitdefender Box 2. Деякі постачальники систем безпеки використовують технологію стрибка частоти для запобігання затухання сигналу, тоді як інші використовують вбудоване шифрування. Потрібно слідкувати за журналами камери, щоб побачити, коли вони були доступні. Слід переконатися, що системне програмне забезпечення та всі підключені пристрої оновлені. Оновлення мікропрограми часто вирішують проблеми безпеки і можуть допомогти захистити систему від проникнення.

1.4.Висновки до розділу 1

У розділі визначено види систем Smart Home, їх функціональне призначення, основні тенденції розвитку систем Smart Home, проведено аналіз ринку систем розумного будинку, визначені різноманітні аспекти безпеки систем.

У результаті можна констатувати, що основною тенденцією розвитку систем Smart Home є їх подальша інтелектуалізація та більш широке застосування методів і засобів обробки зображень з метою підвищення функціональних характеристик систем для користувача, надання конкурентних переваг і більш високого рівня безпеки.

РОЗДІЛ 2
ВИБІР ПРОГРАМНИХ ЗАСОБІВ ТА ПРОЕКТУВАННЯ ДОДАТКУ

2.1. Характеристика задачі розпізнавання образів

Здатність «розпізнавати» вважається основною властивістю біологічних істот, в той час як комп'ютерні системи цією властивістю в повній мірі не володіють. Зазвичай виділяють 3 групи методів розпізнавання образів:

– Порівняння із зразком - класифікація по найближчому середньому, класифікація по відстані до найближчого сусіда. У групу порівняння за зразком можна віднести структурні методи розпізнавання.

– Статистичні методи використовують деяку статистичну інформацію при вирішенні задачі розпізнавання. Метод визначає приналежність об'єкта до конкретного класу на основі ймовірності В ряді випадків це зводиться до визначення апостеріорної ймовірності приналежності об'єкта до певного класу, за умови, що ознаки цього об'єкта взяли відповідні значення. Прикладом служить метод на основі байєсівського вирішального правила.

– Нейронні мережі. Окремий клас методів розпізнавання. Відмінною особливістю від інших є здатність навчатися. Штучні нейромережі - це математична модель функціонування традиційних для живих організмів нейромереж, які представляють собою мережі нервових клітин. Як і в біологічному аналогу, у штучних мережах основним елементом виступають

нейрони, з'єднані між собою, що утворюють шари, число яких може бути різним у залежності від складності нейромережі і її призначення (вирішуваних завдань). Цей метод вимагає або великої кількості прикладів – завдання розпізнавання, або спеціальної структури нейронної мережі, яка враховує специфіку даного завдання. Недоліком і в той же час позитивною стороною є те, що таку систему потрібно навчати. Недоліком це є тому, що потрібно підготувати велику кількість навчальних прикладів образів, на яких система буде навчатися. Але в той же час, якщо навчальна база для системи вже є, система продовжує навчання самостійно.

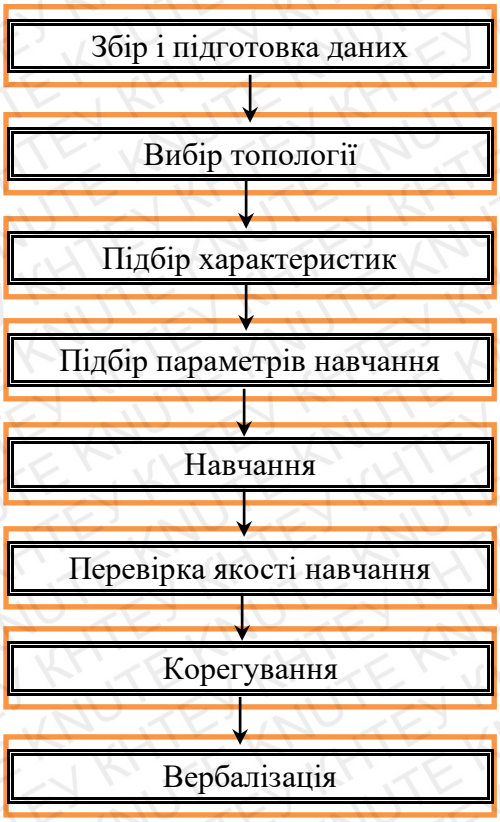

Рис.2.1. Етапи створення нейронної мережі для розпізнавання зображень

Досить часто в задачах розпізнавання та ідентифікації зображень використовуються класичні нейромережеві архітектури (багатошаровий персептрон, мережі з радіально базисної функцією та ін. [4]). Звичайною часто називають повнозв'язну нейронну мережу. У ній кожен вузол (крім вхідного і вихідного) виступає як входом, так і виходом, утворюючи

прихований шар нейронів, і кожен нейрон наступного шару з'єднаний з усіма нейронами попереднього. Входи подаються з вагами, які в процесі навчання налаштовуються і не змінюються надалі. При цьому у кожного нейрона є поріг активації, після проходження якого він приймає одне з двох можливих значень: -1 або 1, або 0 або 1. Однак, з аналізу даних робіт і експериментальних досліджень випливає, що застосування класичних нейромережевих архітектур у задачі розпізнавання образів досить неефективно за такими причин:

– зображення мають велику розмірність, відповідно зростає розмір нейронної мережі;

– велика кількість параметрів збільшує місткість системи і відповідно вимагає більшої тренувальної вибірки, збільшує час і обчислювальну складність процесу навчання;

– для підвищення ефективності роботи системи бажано застосовувати кілька нейронних мереж (навчені з різними початковими значеннями синаптичних коефіцієнтів і порядком пред'явлення образів), але це збільшує обчислювальну складність розв'язання задачі і час виконання;

– відсутня инваріантність до змін масштабу зображення, ракурсів зйомки камери і інших геометричних спотворень вхідного сигналу [5,6].

Для вирішення задачі розпізнавання образів, на наш погляд, більш ефективною є побудова згорткової нейронної мережі. Згорткові нейронні мережі мають спеціальну архітектуру, яка дозволяє їй максимально ефективно розпізнавати образи. Сама ідея згорткової нейронної мережі грунтується на чергуванні згорткових і субдіскретізірующіх шарів (pooling), а її структура є односпрямованою. Згорткова нейронна мережа отримала свою назву від операції згортки, яка передбачає, що кожен фрагмент зображення буде помножений на ядро згортки поелементно, при цьому отриманий результат повинен додаватися і записуватися у схожу позицію вихідного зображення. Така архітектура забезпечує інваріантність

розпізнавання щодо зсуву об'єкта, поступово збільшуючи «вікно», на яке «дивиться» згортка, виявляючи все більш і більш великі структури і патерни у зображенні.

2.2. Мова програмування Python

Реалізація програмного забезпечення обробки зображень і, зокрема розпізнавання образів, може здійснюватися за допомогою пакетів прикладних програм, бібліотек прикладних або системних програмних засобів. На даний момент існує безліч програмних пакетів, за допомогою яких можна проектувати і налаштовувати системи для вирішення різних завдань, наприклад, Neurooffice, Neuro Emulator, BrainMaker, Neural 10, Neural Planner, Mathlab і ін. Спеціально для обробки даних зображень розроблені сучасні програмні комплекси: ENVI (Environment for Visualizing Images), ER Mapper, ERDAS Imagine. Крім цього є бібліотеки для обробки і розпізнавання зображень, такі як OpenCV, LTI, VXL і ін. Однак перераховані нейропакети і програмні комплекси не розраховані на потокову обробку знімків і на реалізацію паралельних обчислювань.

Для реалізації задач розпізнавання образів можна використовувати різні мови програмування, у тому числі C++, Java та ін. Однак, найбільш популярною є мова програмування Python.

Python - об'єктно-орієнтована мова загального призначення, розроблена з метою підвищення продуктивності програміста. До основних переваг мови програмування Python можна віднести такі:

- Низький поріг входження. Синтаксис Python зрозумілий для новачка.
- Логічний, лаконічний і зрозумілий. У порівнянні з багатьма іншими мовами Python має чіткий зрозумілий синтаксис.
- Багатоплатформовий: підходить для різних платформ: Linux, Windows та ін.

• Є реалізація інтерпретаторів для мобільних пристроїв і непопулярних систем.

• Широке застосування. Використовується для розробки веб-додатків, ігор, зручний для автоматизації, математичних обчислень, машинного навчання, в області інтернету речей. Існує реалізація під назвою Micro Python, оптимізована для запуску на мікроконтролерах (можна писати інструкції, логіку взаємодії пристроїв, організовувати зв'язок, реалізовувати розумний будинок).

• Сильне community і багато конференцій.

• Потужна підтримка компаній-гігантів IT-індустрії.

• Висока затребуваність на ринку праці.

У світі Python розроблено багато якісних бібліотек, що спрощує процес розробки та підвищує якість коду. Для навчання є великі кількість літературних і Інтернет-джерел.

Python відрізняється суворою вимогою до написання коду (вимагає відступи), що є її перевагою. За численними оглядам і рейтингам мова займає високі позиції. Згідно DOU вона знаходиться на п'ятому місці і займає третю позицію у веб-технологіях.

Спільнота навколо Python одна з найсильніших у світі IT. Це складний добре організований механізм і він постійно розвивається. Крім сотні тисяч індивідуальних розробників і невеликих софтверних компаній, Python підтримують такі гіганти IT як: Google, Dropbox, Mozilla, Facebook, Yandex, Red Hat, Microsoft (з недавніх пір дуже активно, зокрема з Visual Studio), Intel (активно веде дослідницьку роботу в області паралельних обчислень на Python) і багато інших.

Що ж стосується великих і популярних проектів, написаних на Python то це такі монстри як:

YouTube (велика частина кодової бази повністю на Python)

Перша версія пошукового павука Google була написана на Python, а пізніше, через надзвичайно високого навантаження і вимог до швидкості, була переписана на C ++.

- Десктопний клієнт Dropbox
- Reddit
- Instagram (500M користувачів на Python)
- Bitbucket (Python 2.7 і Django 1.7.11)
- EVE Online MMOPG
- Quora
- Spotify
- Критичні сервіси PayPal, обробні до 2 мільярдів запитів на добу.
- Сервіси Mozilla
- Популярний сервіс ідей Pinterest
- Сервіс коментарів Disqus (використовую в цьому блозі, сервіс реалізований на Django)
- Внутрішні сервіси Facebook
- Система контролю версій Mercurial
- Сервіси Wargaming

Важливою перевагою мови програмування Python є наявність великої кількості бібліотек для наукових досліджень, зокрема Data Science, для штучного інтелекту, для розробки web-додатків.

До недоліків мови Python зазвичай відносять те, що Python - це мова з динамічною типізацією. Це обумовлює, що код простіше і швидше писати, але продуктивність додатку поступається таким компільованим мовам як, наприклад, C ++.

2.3. Засоби розробки програм на Python

Для створення додатків на мові програмування Python існує велика кількість різного ступеня потужності IDE (Integrated Development Environment). IDE допомагає автоматизувати завдання розробника, зменшуючи ручні зусилля та поєднуючи увесь проект у загальних рамках. Якщо IDE відсутній, розробник повинен вручну зробити вибір, інтеграцію та процес розгортання. IDE в першу чергу використовується для спрощення процесу розробки за рахунок зменшення кодування та уникнення помилок друку. До найбільш популярних IDE для створення проектів на Python можна віднести:

- Visual Studio дозволяє розробляти додатки для різних платформ і пропонує власний набір розширень. Python Tools for Visual Studio (PTVS) дозволяє писати на Python у Visual Studio та включає в себе Intellisense для Python, відладку та інші інструменти.
- Eclipse + PyDev. Дуже потужна IDE Eclipse є IDE із відкритим кодом для розробки на Java. Існує безліч розширень, які роблять Eclipse корисним для різного роду завдань. Зокрема розширення PyDev надає можливість застосовувати Eclipse для розробки додатків на Python.
- Spyder - IDE з відкритим кодом для Python, оптимізований для Data Science, добре взаємодіє з такими бібліотеками для Data Science, як SciPy, NumPy та Matplotlib. Spyder має функціональність, властиву IDE середньої потужності, зокрема редактор коду з підсвічуванням синтаксису, автодоповнення коду, вбудований оглядач документації.
- Thonny називають IDE для новачків. Написаний та підтримуваний Інститутом інформатики Тартуського університету в Естонії, Thonny доступний на всіх основних платформах.
- PyCharm від компанії JetBrains є однією з найкращих повнофункціональних IDE, призначених саме для Python, існує як безкоштовний відкритий (спільнота), так і платний (професійний) варіант

IDE. PyCharm доступний для Windows, Mac OS X та Linux. PyCharm підтримує розробку програм на Python у повному обсязі, крім того, в IDE є підтримка проектів та систем управління версіями.

Основні характеристики IDE PyCharm:

- Реалізована розумна навігація для переходу до будь-якого класу, файлу, символу, події IDE і вікна інструментів.
- Дає змогу здійснювати швидкий і безпечний рефакторинг коду.
- Велика колекція інструментів PyCharm включає в себе зокрема:
  - інтегрований відладчик і запуск тестування
  - профайлер Python;
  - вбудований термінал;
  - інтеграцію з великими VCS і вбудованими інструментами баз даних;
  - можливість віддаленої розробки з віддаленими інтерпретаторами;
  - інтегрований термінал ssh;
  - інтеграція з Docker і Vagrant.
- Надає широкі можливості для налагодження, тестування і профілювання, розгортання та дистанційної розробки
- Має інтерфейс для роботи з Git, SVN, Mercurial і іншими системами контролю версій.
- Включає інструменти для роботи з такими базами даних, як
  - Access Oracle
  - SQL Server
  - PostgreSQL
  - MySQL
- Надає підтримку різних фреймворків web-розробки від Python, окремих мов, JavaScript, CoffeeScript, TypeScript, HTML / CSS, AngularJS, Node.js

На основі аналізу засобів розробки проектів на мові Python обрано найпопулярнішу за оцінками користувачів та найбільш потужну IDE PyCharm. Дана IDE використовувалась для розробки додатку (рис.2.2).

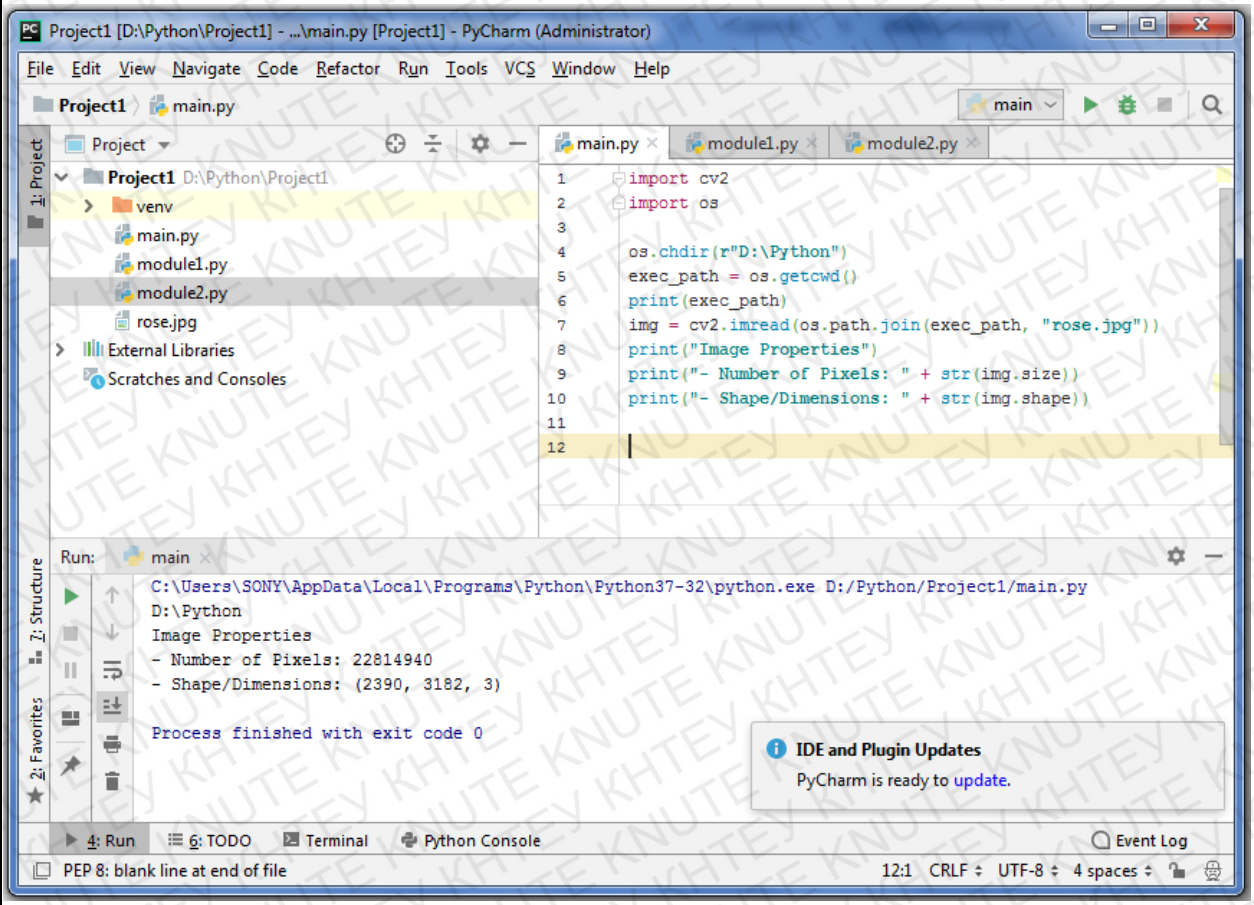

Рис.2.2. Вікно проекту у середовищі IDE PyCharm

Слід також зазначити, що на теперішній час існують досить потужні інтерактивні онлайн-засоби розробки. Частково під час розробки проекту використовувався хмарний сервіс Google Colaboratory (рис.2.3), спрямований на спрощення досліджень в області машинного та глибокого навчання. Google Colaboratory можна використовувати для написання та запуску коду, створення пов'язаної з ним документації та відображення графіки. Colaboratory дозволяє повністю взаємодіяти з файлами ноутбука, використовуючи GitHub або Google Disk як сховище. Основною перевагою застосування Colaboratory є те, що немає потреби встановлювати на комп'ютері всі необхідні бібліотеки.

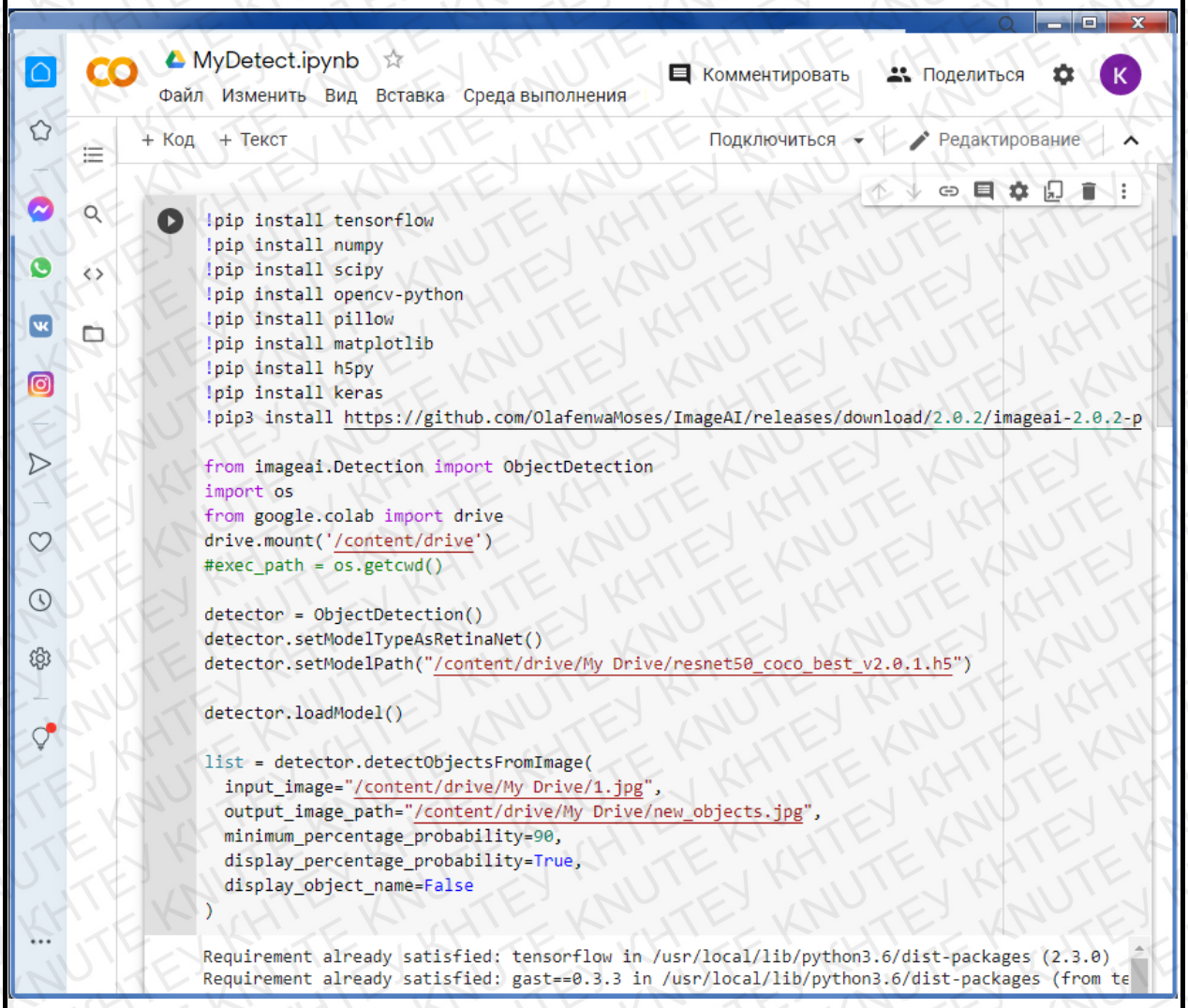

Рис.2.3. Вікно проекту у середовищі Google Colaboratory

2.4.Проектування додатку розпізнавання образів

Для реалізації додатку для системи Smart Home застосована технологія розпізнавання образів, що ґрунтується на нейронній мережі. Це науковий напрямок, пов'язаний з розробкою принципів і побудовою систем, призначених для визначення приналежності об'єкта до одного з класів об'єктів. Під об'єктами в розпізнаванні образів розуміють: різні предмети і явища, процеси і ситуації, сигнали і ін. Системи розпізнавання мають наступну типову функціональну схему: вхідні дані, що підлягають розпізнаванню, подаються на вхід системи і піддаються попередній

обробці з метою їх перетворення в необхідний для наступного етапу вид або для виділення з них необхідних характерних ознак. Далі на етапі прийняття рішення над опрацьованим масивом даних проводиться ряд обчислень і на основі їх результатів формується відповідь, що містить очікувані від системи відомості про вхідних даних. Зміст вхідних і вихідних даних визначається призначенням системи (рис.2.4).

. . . . . .

Рис.2.4. Алгоритм розпізнавання образів

Вхідними даними системи розпізнавання образів для системи Smart Home є зображення у будь-якому з графічних форматів. Система обробляє відеоінформацію по кадрово, тож принципової різниці між обробкою фото та відеоінформації немає. У системі використовувався алгоритм розпізнавання таких образів, як люди і транспортні засоби.

Для реалізації системи використовувалась мова програмування Python, IDE PyCharm, хмарний сервіс Google Colaboratory.

2.5. Висновки до розділу 2

У розділі проаналізовані існуючі методи розпізнавання образів, на основі аналізу визначено, що для реалізації додатку для системи Smart Home найбільш ефективною є побудова згорткової нейронні мережі.

З метою вибору мови програмування проаналізовані найбільш популярні мови, визначено, що для реалізації даної задачі доцільно використовувати мову програмування Python. Також проаналізовані сучасні засоби розробки проектів на мові програмування Python, обрано для реалізації IDE PyCharm, хмарний сервіс Google Colaboratory.

Згідно з технічними завданням визначені структура, алгоритм додатку розпізнавання образів для системи Smart Home.

РОЗДІЛ 3
РОЗРОБКА ДОДАТКУ РОЗПІЗНАВАННЯ ОБРАЗІВ
3.1. Основні бібліотеки Python для роботи з зображеннями

Для реалізації додатку використовувалися наступні бібліотеки Python:

- **OpenCV** - це бібліотека з відкритим кодом комп'ютерного зору, яка призначена для аналізу, класифікації та обробки зображень. У OpenCV входять понад 2500 алгоритмів, в яких є як класичні, так і сучасні алгоритми для комп'ютерного зору і машинного навчання. Ця бібліотека має інтерфейси на різних мовах, серед яких є Python, Java, C ++. Крім платформ і підтримки багатьох мов програмування, які дозволяють використовувати додатки на різних системах, бібліотека OpenCV вельми ефективна (в порівнянні з іншими схожими бібліотеками) з точки зору обчислень, так як майже всі функції і оператори в ній векторизіровані. Також у бібліотеці OpenCV реалізовані такі важливі і корисні методи опису, зберігання, розпізнавання, порівняння та пошуку графічних об'єктів як контурний аналіз.
- **TensorFlow** - це бібліотека Python для швидких числових обчислень, створена і випущена Google. Це базова бібліотека, яку можна використовувати для створення моделей глибокого навчання безпосередньо або за допомогою бібліотек-оболонок. TensorFlow - це бібліотека з відкритим вихідним кодом для швидких числових обчислень. На відміну від інших числових бібліотек, призначених для використання в

Deep Learning, таких як Theano, TensorFlow була розроблена для використання як в дослідженнях і розробках, так і в виробничих системах. Вона може працювати на однопроцесорних системах, графічних процесорах, а також на мобільних пристроях і в великих розподілених системах, що включають сотні машин. Tensorflow використовується для побудови нейронних мереж. На основі даної бібліотеки будуються більш високорівневі бібліотеки для роботи з нейронними мережами на рівні цілих шарів. Так, деякий час назад популярна бібліотека Keras стала використовувати Tensorflow як основний бекенд для обчислень замість аналогічної бібліотеки Theano. TensorFlow вже застосували до своїх сервісів такі бренди як Uber, Airbnb, Dropbox і ін. Разом з цим слід відмітити, що TensorFlow - це низькорівневий інструмент. Для її застосування потрібно ретельно продумувати архітектуру нейромережі, правильно оцінювати розмірність і обсяги вхідних і вихідних даних. Таким чином, робота з TensorFlow вимагає написання значної кількості програмного коду. TensorFlow оперує статичним обчислювальним графом. Тобто спочатку слід визначити граф, далі запустити обчислення і, якщо необхідно внести зміни в архітектуру, заново навчати модель. Такий підхід обраний заради ефективності, але багато сучасних нейромережевих інструментів вміють враховувати уточнення в процесі навчання без істотної втрати швидкості навчання.

- **Keras** - бібліотека для побудови нейронних мереж, що підтримує основні види шарів і структурні елементи. Keras підтримує як рекурентні, так і згорткові нейромережі, має в своєму складі реалізацію відомих архітектур нейромереж. Шари з даної бібліотеки стали доступними всередині бібліотеки Tensorflow. Існують готові функції для роботи з зображеннями і текстом. Дана бібліотека дозволяє на більш високому рівні працювати з нейронними мережами.

- **NumPy** - це розширення мови Python, що додає підтримку великих багатовимірних масивів і матриць, разом з великою бібліотекою високорівневих математичних функцій для операцій з цими масивами.
- **h5py** - це інтерфейс Pythonic для формату двійкових даних HDF5, дозволяє зберігати величезні обсяги числових даних та легко управляти цими даними з NumPy. Велику кількість наборів даних можна зберігати в одному файлі, класифікувати та помічати тегами як завгодно.
- **SciPy** - бібліотека для мови програмування Python з відкритим кодом, призначена для виконання наукових і інженерних розрахунків, такі як перетворення Фур'є, процедури лінійної алгебри, пакет n-мірних зображень, ортогональна регресія відстані, оптимізація, обробка сигналу, розріджені матриці, просторові структури даних і алгоритми, будь-які спеціальні математичні функції, статистика.
- **Pillow** - бібліотека зображень Python потрібна для обробки графіки в Python. Ця бібліотека забезпечує широку підтримку формату файлів, ефективне внутрішнє представлення та досить потужні можливості обробки зображень. Бібліотека зображень розроблена для швидкого доступу до даних, що зберігаються у декількох основних піксельних форматах. Це забезпечує міцну основу для загального інструменту обробки зображень.
- **Matplotlib** - бібліотека для створення статичної, анімованої та інтерактивної візуалізації в Python, призначена для розробки двовимірних графіків (включаючи 3D-образи).
- **ImageAI** - це бібліотека Python, створена для розширення можливостей розробників, дослідників та студентів для створення додатків та систем із автономними можливостями глибокого навчання та комп’ютерної візуалізації.

3.2. Характеристика механізму дії додатку розпізнавання образів

Для розпізнавання образів використовувалась згорткова нейронна мережа. Механізм дії згорткової нейронної мережі розглянуто на прикладі ідентифікації фрагмента зображення (рис.3.1).

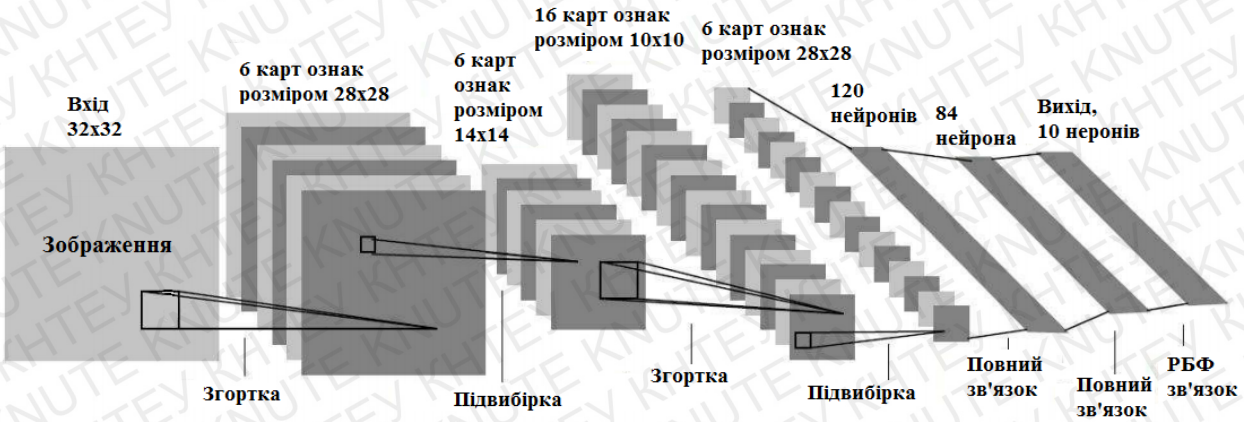

Рис.3.1. Механізм дії згорткової нейронної мережі

У вхідній шар надходить центроване зображення. Така операція робиться для того, щоб характерні ознаки малюнка (дуги, кінцеві точки) знаходилися в центрі рецептивного поля при добуванні ознак високого порядку. Перший прихований шар є шаром згортки. Він складається з 6 карт ознак розміром 28x28. Розглянемо процес формування цього шару, оскільки інші згорткові шари формуються подібним чином. Кожен елемент карти ознак з'єднаний з областю розміром 5x5 на вхідному зображенні. Відповідно, кожен елемент карти має 25 коефіцієнтів і зсув. Значення елемента карти обчислюється за формулою

$$X_i^{k,l} = f(\sum_{k=1}^{n_l-1} \sum_{j=-\infty}^{\infty} X_{i-j}^{k,l} W_{i-j,i}^{h,k,l} + B_i^{h,l}), \quad (1)$$

де $X_i^{k,l}$ - значення елемента i в карті ознак h шару l, n_l- кількість карт ознак е шарі l, $B_i^{h,l}$- значення зсуву для елемента i е карті ознак h шару l, $W_{i-j,i}^{h,k,l}$-, - синаптична вага зв'язку між елементом i в карті ознак h шару l і елементом i-j карти k шару l-1.

Рецептивні поля сусідніх елементів на карті ознак з'єднані з сусідніми областями попереднього шару, відповідно ці поля частково накладаються один на одного. 6 різних карт другого шару витягають 6 різних ознак для різних областей зображення. Значення всіх елементів карти ознак обчислюються шляхом послідовного проходу по вхідному прошарку і застосування формули (1) до тих областей, які є локальними рецептивними для даних елементів карти ознак. Особливістю згортальних шарів є той факт, що при зсуві вхідного зображення значення карт ознак будуть зрушені на ту ж саму величину. За рахунок цього згорткові мережі мають інваріантність до зрушень і спотворень вхідного сигналу.

Отже, перший прихований шар є шаром згортки, який містить 6 карт ознак розміром 28x28. Кожен його елемент з'єднаний з областю розміром 5x5 на вхідному зображенні. Перший прихований шар містить 122,304 зв'язків і 156 учнів параметрів. Така економія пам'яті і, що головне, обчислювальних витрат досягається за рахунок спільного використання ваг елементами однієї карти.

Другий прихований шар є шаром підвибірки. Він складається з 6 карт ознак розміром 14x14, причому кожен з елементів карт цього шару з'єднаний з областю 2x2 у відповідній карті ознак попереднього шару. Значення елементів верств згортки також обчислюються за формулою (1). Рецептивні поля для елементів шару підвибірки не перекриваються, відповідно карти ознак цього шару містять в 2 рази менше рядків і стовпців, ніж в попередньому шарі. Таким чином, другий прихований шар містить 5,880 зв'язків і 12 учнів параметрів. Третій прихований шар є згортальним шаром з 16 картами ознак розміром 10x10. Кожен елемент у кожній карті ознак пов'язаний з декількома областями розміром 5x5 певних карт попереднього шару. Такого незвичайного способу зв'язку цих двох шарів є ряд пояснень.

Перші 6 карт третього шару з'єднані з наборами з трьох послідовних карт другого шару. Наступні 6 карт пов'язані з наборами 4 послідовних карт. Карти 13-15 третього шару пов'язані з непослідовними наборами з 4 карт. І остання карта третього шару пов'язана з усіма картами попереднього шару. Загалом, третій шар містить 151,600 зв'язків і 1,516 учнів параметрів. Четвертий шар реалізує підвибірку і складається з 16 карт ознак розміром 5x5. Цей шар мало чим відрізняється від другого підвибірчого шару: кожен його елемент пов'язаний з областю 2x2 на відповідній карті попереднього шару.

Четвертий прихований шар містить 2,000 з'єднань і 32 учня параметра. П'ятий прихований шар є повнозв'язним згортальним шаром і містить 120 елементів. Кожен з них з'єднаний 5x5 областями з усіма 16 картами четвертого шару. П'ятий шар містить 48,120 учнів параметрів. Шостий шар містить 84 елемента-нейрона і також є повнозв'язним. Він містить 10,164 учнів параметрів.

На відміну від перших шести шарів, останній шар формує сигнал за допомогою так званих Радіально Базисних Функцій (РБФ) для кожного з класів, використовуючи 84 вхідних сигналу попереднього рівня:

$$f(a) = \sum_j (x_j - w_{ij})^2, \quad (2)$$

де x_j - значення сигналу j-го елемента попереднього шару, w_{ij} - значення вагового коефіцієнта для зв'язку даних елементів. Іншими словами, РБФ обчислює Евклідову відстань між вхідним і ваговим векторами. Чим більше відрізняються ці вектори - тим більше значення функції. Така сутність РБФ функції вкрай корисна для останнього шару нейромережі: фактично, вона показує наскільки відрізняються вхідний вектор і еталонний вектор даного класу. У цьому випадку функція втрат повинна бути налаштована таким чином, щоб вихідний вектор шостого шару був максимально наближений до параметричного векторе, що описує еталон даного класу.

Для навчання у нейронній мережі додатку розпізнавання образів для системи Smart Home використовувалось розширення бібліотеки Python ImageAI.

3.3. Опис реалізації додатку розпізнавання образів

Для реалізації додатку розпізнавання образів для системи Smart Home було розроблено програмне забезпечення на мові програмування Python з використанням бібліотек OpenCV, TensorFlow, Keras, NumPy, h5py, SciPy, Pillow, Matplotlib, ImageAI (код наведено в додатку А). При роботі додатку використовується згорткова нейронна мережа ідентифікації об'єктів. Реалізація проекту здійснювалась у середовищі IDE PyCharm.

Під час застосування додатку користувач повинен ввести ім'я файлу в будь-якому з графічних форматів або ім'я відеофайлу. У результаті виконання коду у поточній папці створюється новий файл, у якому до власно імені додається слово «new». У результатному файлі виділяються рамками ідентифіковані об'єкти – люди або транспортні засоби, а також виводяться їх координати у пікселях. За бажанням можна вивести лістинг з переліком ідентифікованих об'єктів та їх координатами. Робота додатку розпізнавання образів проілюстрована на рис.3.2.

Було проведено функціональне тестування додатку (functional testing), а також тестування продуктивноті системи (performance testing). Функціональне тестування полягало у перевірці додатку на відповідність технічному завданню. Було оброблено 120 зображень, правильність ідентифікації на них об'єктів складає 96%. Приклади результатів функціонального тестування наведено у розділі «Тестування додатку».

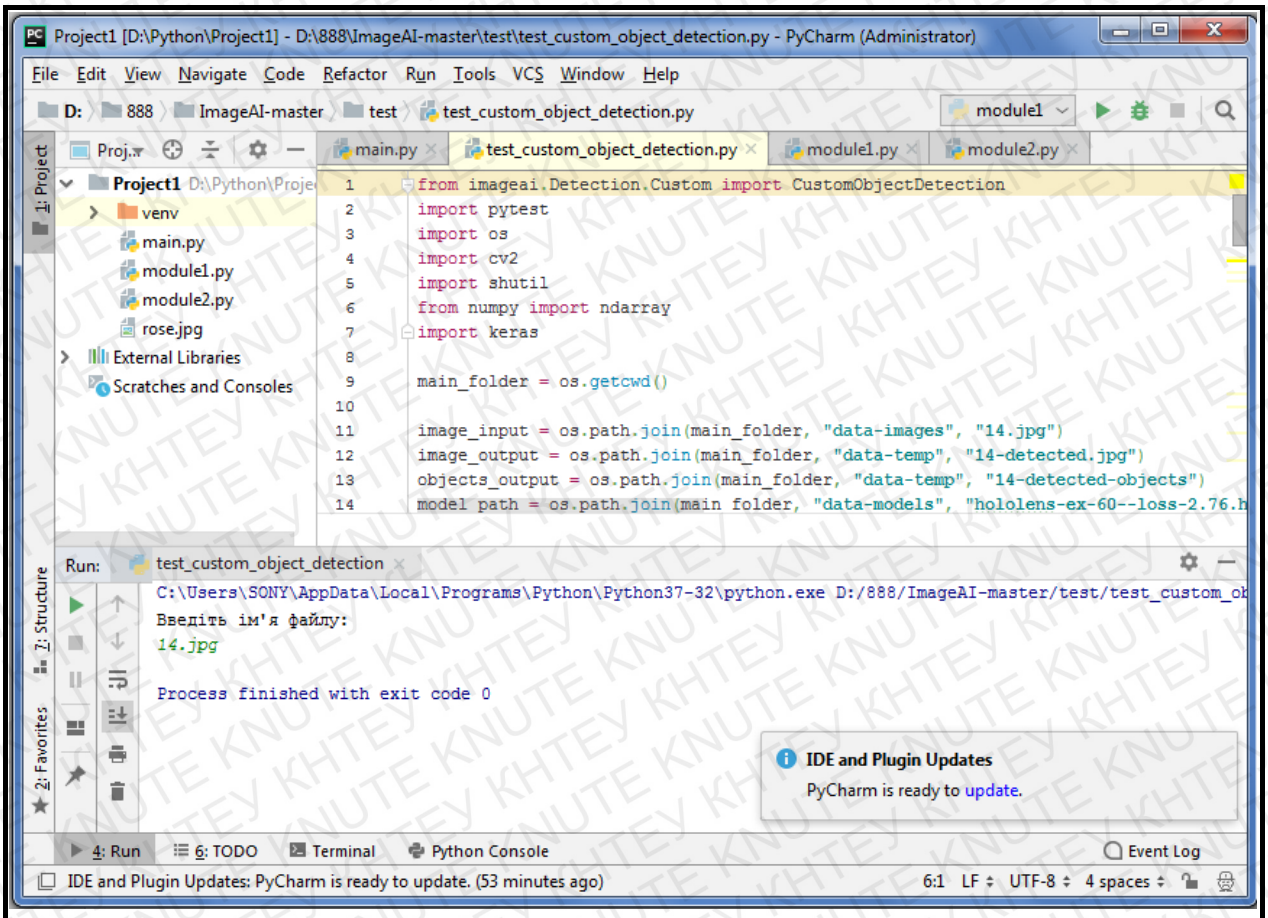

Рис.3.2. Робота додатку розпізнавання образів

Результати тестування продуктивності додатку показали, що для обробки графічного файла розміром 800-899 Гбайт витрачається в середньому 34 сек. Зростання розміру вхідних файлів призводить к зростанню часу обробки у середньому відповідно до моделі:

$$Y = 0{,}23 \cdot X + 22{,}56 (3)$$

деX – розмір вхідного файла,

Y – час на роботу додатку

Детальна інформація щодо тестування продуктивності додатку також наведена в розділі «Тестування додатку».

3.4. Висновки до розділу 3

У даному розділі, були описані бібліотеки Python, які використовувались під час розробки додатку, механізм роботи згорткової нейронної мережі, що був покладений в основу роботи додатку для ідентифікації зображень, інтерфейс системи, результати тестування системи.

Отже, розроблений додаток для обробки зображень відповідає технічному завданню, може використовуватися в системах Smart Home.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

З проведених досліджень можна зробити наступні висновки:

1. Проведено аналіз сучасних тенденцій розвитку систем Smart Home, аналіз ринку систем розумного будинку, визначені різноманітні аспекти безпеки систем. У результаті проведеного аналізу можна констатувати, що системи Smart Home постійно розвиваються, значними темпами зростає попит на устаткування для розумного будинку, основною тенденцією розвитку систем Smart Home є їх подальша інтелектуалізація та більш широке застосування методів і засобів обробки зображень з метою підвищення функціональних характеристик систем для користувача, надання конкурентних переваг і більш високого рівня безпеки.
2. У роботі проаналізовані існуючі методи розпізнавання образів, на основі аналізу визначено, що для реалізації додатку для системи Smart Home найбільш ефективною є побудова згорткової нейронні мережі.
3. З метою вибору мови програмування проаналізовані найбільш популярні мови, визначено, що для реалізації задачі розпізнавання образів для системи Smart Home доцільно використовувати мову програмування Python. Також проаналізовані сучасні засоби розробки проектів на мові програмування Python, обрано для реалізації IDE PyCharm, хмарний сервіс Google Colaboratory.

					КНТЕУ 121 06-05.МР		
Зм.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*			
Зав. каф.		Криворучко О.В.		30.10.20	Використання технологій розпізнання образів для розробки додатку Smart Home	*Стадія*	*Аркуш*	*Аркушів*
Керівник		Палагута К.О.		30.10.20		*ВП*	*41*	*45*
Гарант		Криворучко О.В.		30.10.20		Факультет інформаційних технологій 2м курс, 6 група		
Розробив		Ганах О.І.		30.10.20	*Висновки та пропозиції*			

4. Згідно з технічними завданням визначені структура, алгоритм, механізм дії додатку розпізнавання образів, обрані бібліотеки Python для використання під час розробки додатку.
5. Виконано розробку додатку, проведено функціональне тестування додатку і тестування продуктивноті системи.
6. Визначено, що розроблений додаток для обробки зображень відповідає технічному завданню, може бути рекомендований до впровадження в системах Smart Home.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Xiao Guo , Zhenjiang Shen, Yajing Zhang, Teng Wu, «Review on the Application of Artificial Intelligence in Smart Homes» Smart Cities, № 2(3), pp. 402-420, 2019.
2. Global Businnes Data Platform Statista - URL: https://www.statista.com /topics/2430/smart-homes/ (дата звернення: 25.11.2020)
3. M. Castrillón, . O. Déniz, . D. Hernández и J. Lorenzo, «A comparison of face and facial feature detectors based on the Viola–Jones general object detection framework,» International Journal of Computer Vision, № 22, pp. 481-494, 2011.
4. Y.-Q. Wang, «An Analysis of Viola-Jones Face Detection Algorithm,» IPOL Journal, 2013.
5. Khan, H. Abdullah и M. Shamian Bin Zainal, «Efficient eyes and mouth detection algorithm using combination of viola jones and skin color pixel detection» International Journal of Engineering and Applied Sciences, № Vol. 3 № 4, 2013.
6. Feraud R., Bernier O., Viallet J., Collobert M. A fast and accurate face detector based on neural networks // Transactions on pattern ana lysis and machine intelligence. – 2002. – V. 3. – № 23. – P. 42–53.
7. Craw I., Ellis H., Lishman J. Automatic extraction of face features // Pattern recognition letters. – 1987. – V. 5. – P. 183–187. 10. Yu N., Notkin B.S., Sedov V.A. Neuroiterative algorithm of tomographic reconstruction of the distributed physical fields in the fibre optic measuring systems // Computer optics. – 2009. – V. 33. – № 4. – P. 446–455.

					КНТЕУ 121 06-05.МР		
Зм.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*			
Зав. каф.		Криворучко О.В.		24.01.20	Використання технологій розпізнання образів для розробки додатку Smart Home	*Стадія*	*Аркуш*	*Аркушів*
Керівник		Палагута К.О.		24.01.20		*СВД*	*43*	*45*
Гарант		Криворучко О.В.		24.01.20		Факультет інформаційних технологій 2м курс, 6 група		
Розробив		Ганах О.І.		24.01.20	*Список використаних джерел*			

8. Lakhno, V. A., Hrabariev, A. V., Petrov, O. S., Ivanchenko, Y. V., & Beketova, G. S. (2016). Improving of information transport security under the conditions of destructive influence on the information-communication system. Journal of theoretical and applied information technology, 89(2), 352–361.

9. Methodology Forming for the Approaches to the Cyber Security of Information Systems Management / Adranova A., Yona L., Kryvoruchko J., Desiatko A., Palahuta K., Blozva A. // Journal of Theoretical and Applied Information Technology (ISSN: 1992-8645, E-ISSN: 1817-3195). 2020. P. 1993–2005.

10. Smart Home Market Reports 2020 - URL: https://www.reportlinker.com /smart-home/reports (дата звернення: 25.11.2020)

11. Best smart home systems 2020: Reviews and buying advice - URL: ttps://www.techhive.com /article/3206310/best-smart-home-system.html (дата звернення: 25.11.2020)

12. Haykin SO, editor. Neural networks and learning machines. 3rd ed. New Jersey: Prentice Hall, 2008.

13. Pal S.K., Srimani P.K., 1996. Neurocomputing: motivation, models, and hybridization. Computer March, 24–28.

14. Murphy, K. P. Machine Learning: A Probabilistic Perspective. Cambridge, Massachusetts: The MIT Press, 2012.

15. Quoc V Le et al. A tutorial on deep learning part 2: Autoencoders, convolutional neural networks and recurrent neural networks. Google Brain, pages 1–20, 2015.

16. Bhiksha Wang, HaohanandRaj. On the origin of deep learning. arXiv preprint arXiv:1702.07800, 2017.

17. Li Deng, Dong Yu, et al. Deep learning: methods and applications. Foundations and Trends R in Signal Processing, 7(3–4):197–387, 2014. doi: 10.1007/ 978-981-13-3459-7 3.

18. Chris Albon. Machine Learning with Python Cookbook: Practical Solutions from Preprocessing to Deep Learning, O'Reily, 2018.

19. Manohar Swamynathan. Mastering Machine Learning with Python in Six Steps: A Practical Implementation Guide to Predictive Data Analytics Using Python, Aptess, 2017.

ТЕХНІЧНЕ ЗАВДАННЯ

1. Загальні відомості

Назва проекту: ObjectsDetection.

Планові строки розробки: серпень 2019 – жовтень 2020.

Потенційні користувачі: користувачі стаціонарних ПК.

Призначення програмного рішення: ідентифікація об’єктів на зображенні або відеофайлі.

Мета створення програмного рішення: полегшити процес ідентифікації осіб та транспортних засобів на зображеннях, знятих пристроями системи Smart Home.

2. Структура проекту

Проект складається з таких частин:

- Програмне забезпечення ідентифікації об’єктів на зображеннях, що містяться у файлах будь-якого з графічних форматів.
- Програмне забезпечення ідентифікації об’єктів у відеофайлах з покадровою їх обробкою.

3. Функціональні вимоги до додатку

Додаток повинен обробляти файли у будь-якому з графічних форматів або відеофайли, на яких міститься зображення зовнішнього оточення будинка. Необхідно ідентифікувати людей і транспортні засоби на цих зображеннях.

					КНТЕУ 121 06-05.МР			
Зм.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*				
Зав. каф.		Криворучко О.В.		24.01.20	Використання технологій розпізнання образів для розробки додатку Smart Home	*Стадія*	*Аркуш*	*Аркушів*
Керівник		Палагута К.О.		24.01.20		*ТЗ*	*46*	*45*
Гарант		Криворучко О.В.		24.01.20		Факультет інформаційних технологій 2м курс, 6 група		
Розробив		Ганах О.І.		24.01.20	*Технічне завдання*			

ТЕСТУВАННЯ ДОДАТКА

З метою забезпечення якості розробленого додатку було проведено його функціональне тестування. Проведено 120 випробувань програмного продукту, які показали, що з ймовірністю 96% додаток правильно ідентифікує на зображенні – графічному файлі або кадрі відеофайла об’єкти – людей і транспортні засоби. На рис.Т.1.-Т.4. наведені приклади результатів тестування.

Рис.Т.1. Case 12 – вхідний файл

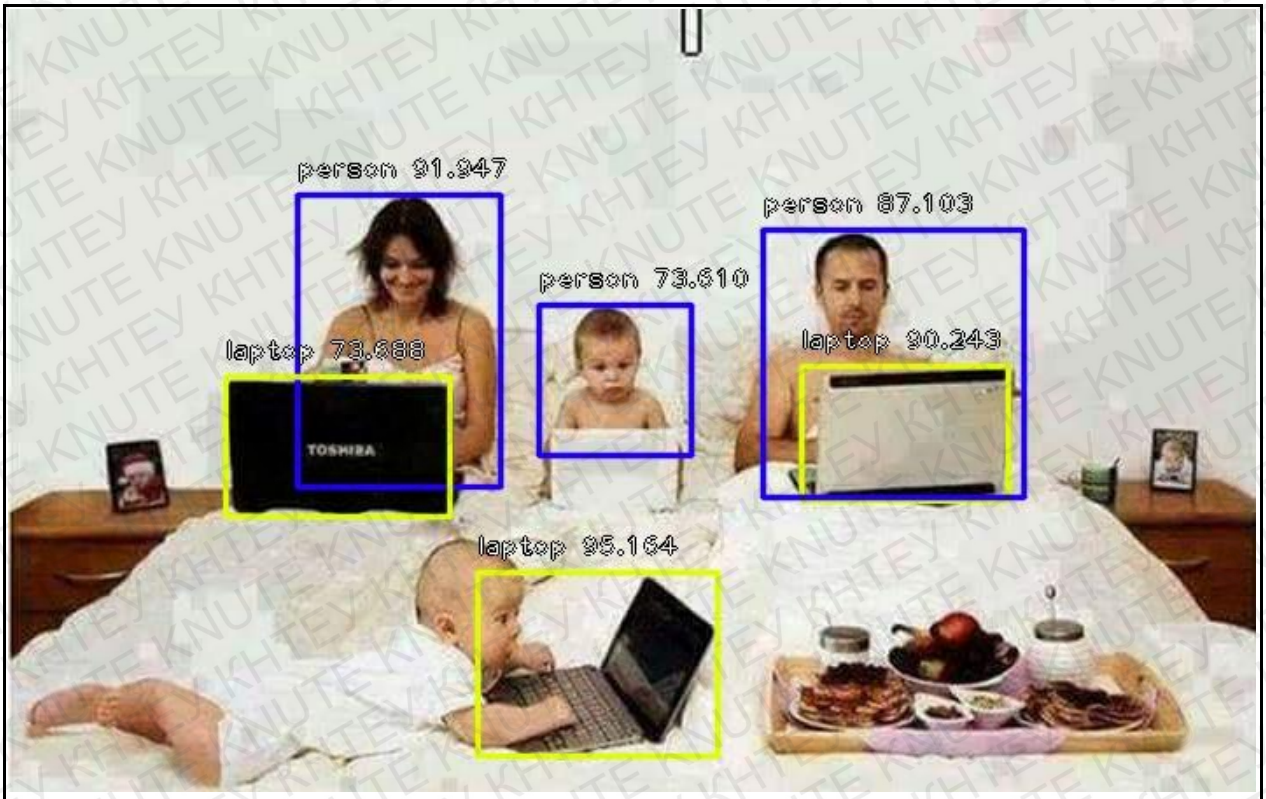

Рис.Т.2. Case 12 – вихідний файл

Рис.Т.3. Case 15 – вхідний файл

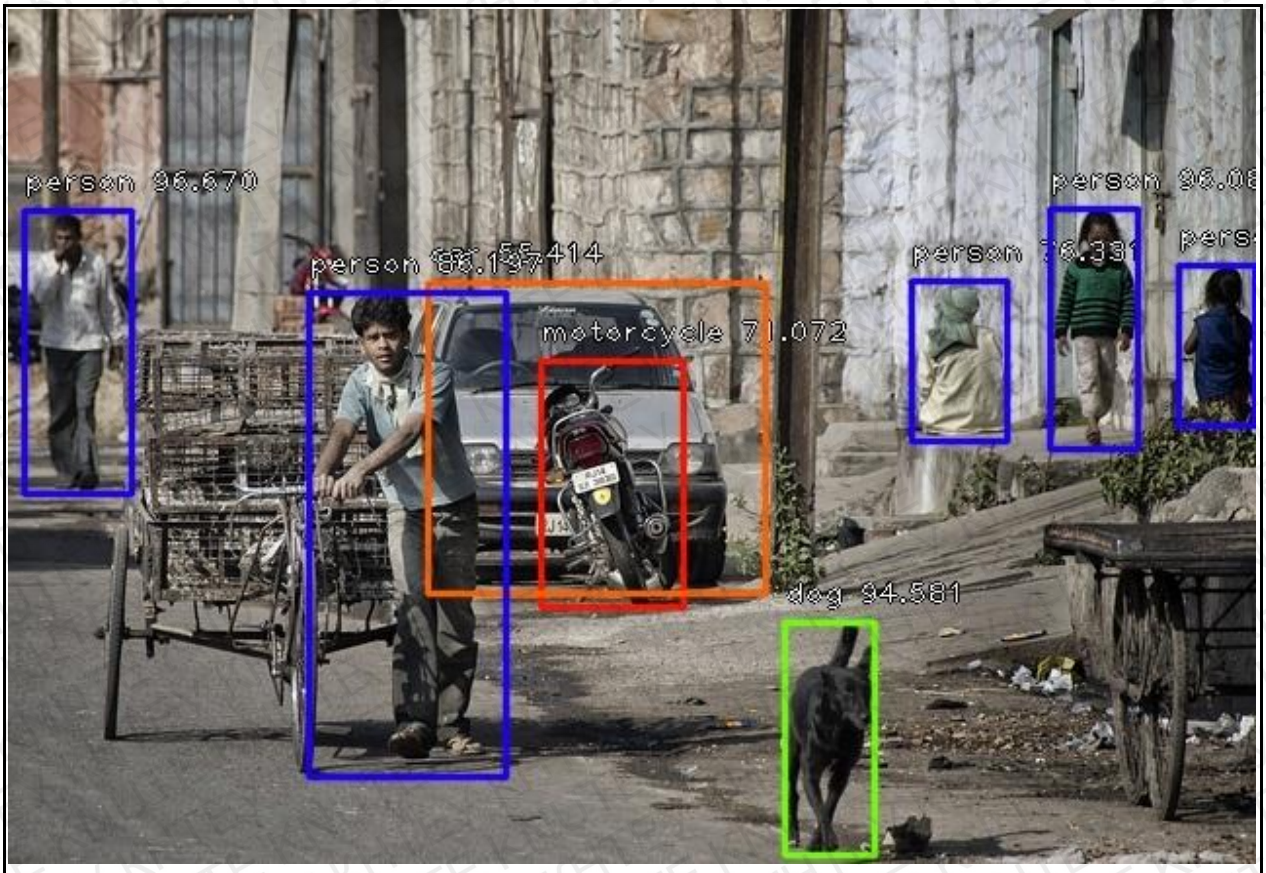

Рис.Т.4. Case 14 – вихідний файл

Під час тестування також оцінювалась продуктивність системи. Тестування проводилось на ПК з оперативною пам’яттю 8 Гбайт, 64-розрядним мікропроцесором, операційною системою MS Windows 10. Результати performance testing наведені в таблиці Т.1.

Таблиця Т.1

Результати тестування продуктивності додатку обробки зображень

Розмір графічного файлу, Мбайт	Кількість тестових файлів	Час обробки файлу
500-599	12	26
600-699	22	27
700-799	35	34
800-899	21	40
900-999	18	48
1000-1100	12	53

На основі даних, викладених у табл.Т.1, побудовано регресійну модель залежності розміру файлів від часу їх розробки з урахуванням кількості Case-тестів. Отримано модель: $Y = 0{,}23 \cdot X + 22{,}56$,

де X – розмір вхідного файла,

Y – час на роботу додатку

Коєфіцієнт детермінації моделі R^2=0,91.

Отримані данні свідчать про значну залежність часу обробки файлу від його розміру.

ДОДАТКИ

Додаток А

Файл custom_object_detection.py

```
from imageai.Detection.Custom import CustomObjectDetection
import pytest
import os
import cv2
import shutil
from numpy import ndarray
import keras

main_folder = os.getcwd()

image_input = os.path.join(main_folder, "data-images", "14.jpg")
image_output = os.path.join(main_folder, "data-temp", "14-detected.jpg")
objects_output = os.path.join(main_folder, "data-temp", "14-detected-objects")
model_path = os.path.join(main_folder, "data-models", "hololens-ex-60--loss-2.76.h5")
model_json = os.path.join(main_folder, "data-json", "detection_config.json")


@pytest.fixture
def clear_keras_session():
    try:
keras.backend.clear_session()
except:
None

@pytest.mark.detection
@pytest.mark.yolov3
@pytest.mark.custom_detection
def test_object_detection_yolov3(clear_keras_session):

detector = CustomObjectDetection()
detector.setModelTypeAsYOLOv3()
detector.setModelPath(model_path)
detector.setJsonPath(model_json)
detector.loadModel()
results = detector.detectObjectsFromImage(input_image=image_input,
output_image_path=image_output,
minimum_percentage_probability=40)

assert isinstance(results, list)
for result in results:
        assert isinstance(result["name"], str)
assert isinstance(result["percentage_probability"], float)
assert isinstance(result["box_points"], list)
assert os.path.exists(image_output)
os.remove(image_output)

results2, extracted_paths = detector.detectObjectsFromImage(input_image=image_input,
output_image_path=image_output,
minimum_percentage_probability=30,
extract_detected_objects=True)

assert isinstance(results2, list)
assert isinstance(extracted_paths, list)
assert os.path.isdir(objects_output)
assert len(os.listdir(objects_output)) == len(results2)
for result2 in results2:
        assert isinstance(result2["name"], str)
assert isinstance(result2["percentage_probability"], float)
assert isinstance(result2["box_points"], list)
```

```
for extracted_path in extracted_paths:
        assert os.path.exists(extracted_path)

shutil.rmtree(objects_output)


@pytest.mark.detection
@pytest.mark.yolov3
@pytest.mark.custom_detection
@pytest.mark.array_io
def test_object_detection_yolov3_array_io(clear_keras_session):

image_input_array = cv2.imread(image_input)

detector = CustomObjectDetection()
detector.setModelTypeAsYOLOv3()
detector.setModelPath(model_path)
detector.setJsonPath(model_json)
detector.loadModel()
detected_array, results =
detector.detectObjectsFromImage(input_image=image_input_array, input_type="array",
minimum_percentage_probability=40, output_type="array")

assert isinstance(detected_array, ndarray)
assert isinstance(results, list)
for result in results:
        assert isinstance(result["name"], str)
assert isinstance(result["percentage_probability"], float)
assert isinstance(result["box_points"], list)

detected_array, results2, extracted_arrays =
detector.detectObjectsFromImage(input_image=image_input,
output_image_path=image_output, minimum_percentage_probability=40,
extract_detected_objects=True, output_type="array")

assert isinstance(results2, list)
assert isinstance(extracted_arrays, list)
for result2 in results2:
        assert isinstance(result2["name"], str)
assert isinstance(result2["percentage_probability"], float)
assert isinstance(result2["box_points"], list)

for extracted_array in extracted_arrays:
        assert isinstance(extracted_array, ndarray)
```

Файл custom_recognition.py

```
from imageai.Prediction.Custom import CustomImagePrediction
import os
import pytest
from os.path import dirname
import keras


main_folder = os.getcwd()

all_images = os.listdir(os.path.join(main_folder, "data-images"))
all_images_array = []
```

```
def images_to_image_array():
    for image in all_images:
all_images_array.append(os.path.join(main_folder, "data-images", image))


@pytest.mark.recognition
@pytest.mark.resnet
@pytest.mark.recognition_custom
def test_custom_recognition_model_resnet():


predictor = CustomImagePrediction()
predictor.setModelTypeAsResNet()
predictor.setModelPath(os.path.join(main_folder, "data-models", "idenprof_resnet.h5"))
predictor.setJsonPath(model_json=os.path.join(main_folder, "data-json",
"idenprof.json"))
predictor.loadModel(num_objects=10)
predictions, probabilities =
predictor.predictImage(image_input=os.path.join(main_folder, main_folder, "data-
images", "9.jpg"))

assert isinstance(predictions, list)
assert isinstance(probabilities, list)
assert isinstance(predictions[0], str)
assert isinstance(probabilities[0], float)

@pytest.mark.recognition
@pytest.mark.resnet
@pytest.mark.recognition_custom
def test_custom_recognition_full_model_resnet():

predictor = CustomImagePrediction()
predictor.setModelPath(os.path.join(main_folder, "data-models",
"idenprof_full_resnet_ex-001_acc-0.119792.h5"))
predictor.setJsonPath(model_json=os.path.join(main_folder, "data-json",
"idenprof.json"))
predictor.loadFullModel(num_objects=10)
predictions, probabilities =
predictor.predictImage(image_input=os.path.join(main_folder, main_folder, "data-
images", "9.jpg"))

assert isinstance(predictions, list)
assert isinstance(probabilities, list)
assert isinstance(predictions[0], str)
assert isinstance(probabilities[0], float)

@pytest.mark.recognition
@pytest.mark.densenet
@pytest.mark.recognition_custom
def test_custom_recognition_model_densenet():

predictor = CustomImagePrediction()
predictor.setModelTypeAsDenseNet()
predictor.setModelPath(os.path.join(main_folder, "data-models", "idenprof_densenet-
0.763500.h5"))
predictor.setJsonPath(model_json=os.path.join(main_folder, "data-json",
"idenprof.json"))
predictor.loadModel(num_objects=10)
predictions, probabilities =
predictor.predictImage(image_input=os.path.join(main_folder, main_folder, "data-
images", "9.jpg"))
```

```
assert isinstance(predictions, list)
assert isinstance(probabilities, list)
assert isinstance(predictions[0], str)
assert isinstance(probabilities[0], float)
```

Файл custom_video_detection.py

```
from imageai.Detection.Custom import CustomVideoObjectDetection
import os
from numpy import ndarrayeo
import pytest
import keras


main_folder = os.getcwd()
video_file = os.path.join(main_folder, "data-videos", "holo-micro.mp4")
video_file_output = os.path.join(main_folder, "data-temp", "holo-micro-detected")

model_path = os.path.join(main_folder, "data-models", "hololens-ex-60--loss-2.76.h5")
model_json = os.path.join(main_folder, "data-json", "detection_config.json")


@pytest.fixture
def clear_keras_session():
    try:
keras.backend.clear_session()
except:
None


@pytest.mark.detection
@pytest.mark.custom_detection
@pytest.mark.video_detection
@pytest.mark.custom_video_detection
@pytest.mark.yolov3
def test_custom_video_detection_yolov3(clear_keras_session):


detector = CustomVideoObjectDetection()
detector.setModelTypeAsYOLOv3()
detector.setModelPath(model_path)
detector.setJsonPath(model_json)
detector.loadModel()
video_path = detector.detectObjectsFromVideo(input_file_path=video_file,
output_file_path=video_file_output, save_detected_video=True, frames_per_second=30,
log_progress=True)

assert os.path.exists(video_file_output + ".avi")
assert isinstance(video_path, str)
os.remove(video_file_output + ".avi")



@pytest.mark.detection
@pytest.mark.custom_detection
@pytest.mark.video_detection
@pytest.mark.custom_video_detection
@pytest.mark.yolov3
@pytest.mark.custom_video_detection_analysis
def test_custom_video_detection_yolov3_analysis(clear_keras_session):
```

```
detector = CustomVideoObjectDetection()
detector.setModelTypeAsYOLOv3()
detector.setModelPath(model_path)
detector.setJsonPath(model_json)
detector.loadModel()
video_path = detector.detectObjectsFromVideo(input_file_path=video_file,
output_file_path=video_file_output, save_detected_video=True, frames_per_second=30,
log_progress=True, per_frame_function=forFrame, per_second_function=forSecond,
return_detected_frame=True)

assert os.path.exists(video_file_output + ".avi")
assert isinstance(video_path, str)
os.remove(video_file_output + ".avi")


def forFrame(frame_number, output_array, output_count, detected_frame):
    assert isinstance(detected_frame, ndarray)
assert isinstance(frame_number, int)
assert isinstance(output_array, list)
assert isinstance(output_count, dict)


def forSecond(second_number, output_arrays, count_arrays, average_output_count,
detected_frame):
    assert isinstance(detected_frame, ndarray)
assert isinstance(second_number, int)
assert isinstance(output_arrays, list)
assert isinstance(count_arrays, list)
```

Файл detection_model_training.py

```
from imageai.Detection.Custom import DetectionModelTrainer
import os
import shutil
import keras
import pytest

"""

main_folder = os.getcwd()
sample_dataset = os.path.join(main_folder, "data-datasets", "hololens")
sample_dataset_json_folder = os.path.join(sample_dataset, "json")
sample_dataset_models_folder = os.path.join(sample_dataset, "models")
sample_dataset_cache_folder = os.path.join(sample_dataset, "cache")
pretrained_model = os.path.join(main_folder, "data-models", "pretrained-yolov3.h5")


@pytest.fixture
def clear_keras_session():
    try:
        keras.backend.clear_session()
    except:
        None


@pytest.mark.detection
@pytest.mark.training
@pytest.mark.detection_training
@pytest.mark.detection_transfer_learning
@pytest.mark.yolov3
def test_detection_training(clear_keras_session):
```

```
    trainer = DetectionModelTrainer()
    trainer.setModelTypeAsYOLOv3()
    trainer.setDataDirectory(data_directory=sample_dataset)
    trainer.setTrainConfig(object_names_array=["hololens"], batch_size=2,
num_experiments=1, train_from_pretrained_model=pretrained_model)
    trainer.trainModel()

    assert os.path.isdir(sample_dataset_json_folder)
    assert os.path.isdir(sample_dataset_models_folder)
    assert os.path.isdir(sample_dataset_cache_folder)
    assert os.path.isfile(os.path.join(sample_dataset_json_folder,
"detection_config.json"))
    shutil.rmtree(os.path.join(sample_dataset_json_folder))
    shutil.rmtree(os.path.join(sample_dataset_models_folder))
    shutil.rmtree(os.path.join(sample_dataset_cache_folder))
```

Файл image_recognition.py

```
from imageai.Prediction import ImagePrediction
import os
import cv2
import pytest
from os.path import dirname
import keras

main_folder = os.getcwd()



@pytest.mark.squeezenet
@pytest.mark.recognition
def test_recognition_model_squeezenet():


predictor = ImagePrediction()
predictor.setModelTypeAsSqueezeNet()
predictor.setModelPath(os.path.join(main_folder, "data-models",
"squeezenet_weights_tf_dim_ordering_tf_kernels.h5"))
predictor.loadModel()
predictions, probabilities =
predictor.predictImage(image_input=os.path.join(main_folder, main_folder, "data-
images", "1.jpg"))

assert isinstance(predictions, list)
assert isinstance(probabilities, list)
assert isinstance(predictions[0], str)
assert isinstance(probabilities[0], float)

@pytest.mark.resnet
@pytest.mark.recognition
def test_recognition_model_resnet():

predictor = ImagePrediction()
predictor.setModelTypeAsResNet()
predictor.setModelPath(os.path.join(main_folder, "data-models",
"resnet50_weights_tf_dim_ordering_tf_kernels.h5"))
predictor.loadModel()
predictions, probabilities =
predictor.predictImage(image_input=os.path.join(main_folder, main_folder, "data-
images", "1.jpg"))
```

```
assert isinstance(predictions, list)
assert isinstance(probabilities, list)
assert isinstance(predictions[0], str)
assert isinstance(probabilities[0], float)

@pytest.mark.inceptionv3
@pytest.mark.recognition
def test_recognition_model_inceptionv3():

predictor = ImagePrediction()
predictor.setModelTypeAsInceptionV3()
predictor.setModelPath(os.path.join(main_folder, "data-models",
"inception_v3_weights_tf_dim_ordering_tf_kernels.h5"))
predictor.loadModel()
predictions, probabilities =
predictor.predictImage(image_input=os.path.join(main_folder, main_folder, "data-
images", "1.jpg"))

assert isinstance(predictions, list)
assert isinstance(probabilities, list)
assert isinstance(predictions[0], str)
assert isinstance(probabilities[0], float)

@pytest.mark.densenet
@pytest.mark.recognition
def test_recognition_model_densenet():

predictor = ImagePrediction()
predictor.setModelTypeAsDenseNet()
predictor.setModelPath(os.path.join(main_folder, "data-models", "DenseNet-BC-121-
32.h5"))
predictor.loadModel()
predictions, probabilities =
predictor.predictImage(image_input=os.path.join(main_folder, main_folder, "data-
images", "1.jpg"))

assert isinstance(predictions, list)
assert isinstance(probabilities, list)
assert isinstance(predictions[0], str)
assert isinstance(probabilities[0], float)


@pytest.mark.squeezenet
@pytest.mark.recognition
def test_recognition_model_squeezenet_array_input():


predictor = ImagePrediction()
predictor.setModelTypeAsSqueezeNet()
predictor.setModelPath(os.path.join(main_folder, "data-models",
"squeezenet_weights_tf_dim_ordering_tf_kernels.h5"))
predictor.loadModel()
image_array = cv2.imread(os.path.join(main_folder, main_folder, "data-images",
"1.jpg"))
predictions, probabilities = predictor.predictImage(image_input=image_array,
input_type="array")

assert isinstance(predictions, list)
assert isinstance(probabilities, list)
assert isinstance(predictions[0], str)
assert isinstance(probabilities[0], float)
```

```
@pytest.mark.resnet
@pytest.mark.recognition
def test_recognition_model_resnet_array_input():

predictor = ImagePrediction()
predictor.setModelTypeAsResNet()
predictor.setModelPath(os.path.join(main_folder, "data-models",
"resnet50_weights_tf_dim_ordering_tf_kernels.h5"))
predictor.loadModel()
image_array = cv2.imread(os.path.join(main_folder, main_folder, "data-images",
"1.jpg"))
predictions, probabilities = predictor.predictImage(image_input=image_array,
input_type="array")

assert isinstance(predictions, list)
assert isinstance(probabilities, list)
assert isinstance(predictions[0], str)
assert isinstance(probabilities[0], float)

@pytest.mark.inceptionv3
@pytest.mark.recognition
def test_recognition_model_inceptionv3_array_input():

predictor = ImagePrediction()
predictor.setModelTypeAsInceptionV3()
predictor.setModelPath(os.path.join(main_folder, "data-models",
"inception_v3_weights_tf_dim_ordering_tf_kernels.h5"))
predictor.loadModel()
image_array = cv2.imread(os.path.join(main_folder, main_folder, "data-images",
"1.jpg"))
predictions, probabilities = predictor.predictImage(image_input=image_array,
input_type="array")

assert isinstance(predictions, list)
assert isinstance(probabilities, list)
assert isinstance(predictions[0], str)
assert isinstance(probabilities[0], float)

@pytest.mark.densenet
@pytest.mark.recognition
def test_recognition_model_densenet_array_input():

predictor = ImagePrediction()
predictor.setModelTypeAsDenseNet()
predictor.setModelPath(os.path.join(main_folder, "data-models", "DenseNet-BC-121-
32.h5"))
predictor.loadModel()
image_array = cv2.imread(os.path.join(main_folder, main_folder, "data-images",
"1.jpg"))
predictions, probabilities = predictor.predictImage(image_input=image_array,
input_type="array")

assert isinstance(predictions, list)
assert isinstance(probabilities, list)
assert isinstance(predictions[0], str)
assert isinstance(probabilities[0], float)
```