МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І.МЕЧНИКОВА

(повне найменування вищого навчального закладу)

Факультет математики, фізики та інформаційних технологій

(повне найменування інституту, назва факультету (відділення))

Кафедра математичного забезпечення комп'ютерних систем

(повна назва кафедри (предметної, циклової комісії))

Дипломна робота

на здобуття ступеня вищої освіти «бакалавр»

(освітньо-кваліфікаційний рівень)

на тему Розробка системи для визначення домінуючих кольорів на зображенні з використанням методів машинного навчання

Development of the image dominant colors determining system using the machine learning method

Виконала: студентка денної форми навчання
спеціальності 123 – Комп'ютерна інженерія.

(шифр і назва напряму підготовки, спеціальності)

Соломко Юлія Олександрівна

(прізвище, ім'я, по-батькові)

Керівник к.ф.-м.н., доц. Шпінарева І. М.

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

Рецензент к.ф.-м.н., доц. Крапівний Ю.М.

(науковий ступінь, вчене звання, прізвище та ініціали)

Рекомендовано до захисту:

Протокол засідання кафедри

№ 11 від «10» червня 2022 р.

Завідувач кафедри

_____ Євгеній МАЛАХОВ

(підпис) (ім'я, прізвище)

Захищено на засіданні ЕК № ___

протокол № __ від «__» ____ 2022 р.

Оцінка _____/_____/_____

(за національною шкалою, шкалою ECTS, бали)

Голова ЕК

_____ Надія КАЗАКОВА

(підпис) (ім'я, прізвище)

Одеса – 2022

АННОТАЦІЯ

У дипломній роботі розроблено систему для визначення домінуючих кольорів на зображенні з використанням методів машинного навчання.

Системи машинного зору дозволяють на основі кольору та колірної моделі об'єктів визначати їх місцезнаходження та наявність, проводити вимірювання та підрахунок у процесі виробництва безпосередньо на конвеєрі чи виробничій лінії, виконувати класифікацію лікарських препаратів за кольором вмісту, виявляти браковані лікарські препарати за відтінком їхнього вмісту. Тому визначення основної палітри кольорів є актуальною проблемою, яку вирішують методами машинного навчання.

Метою роботи є створення системи ідентифікації кольорів з зображення з використанням методів машинного навчання.

В результаті проведених в роботі досліджень за методами виявлення кольору на зображенні спроектована та реалізована система розпізнавання кольору, що складається з модуля згорткової мережі та модуля розкладання на колірні відтінки методом KMeans.

Модуль згорткової нейронної мережі за допомогою моделі та функції збереження найкращих ваг класифікує зображення за приналежністю до одного з семи класів кольорів.

Модуль розкладання на колірні відтінки видає обрану кількість домінуючих відтінків у вигляді діаграми з маркуванням кожного відтінка у форматі HEX.

Система колірної ідентифікації навчена та протестована з використанням двох наборів даних: класичний набор даних для визначення кольору з зображеннями автомобілів та створений спеціально для даної роботи набор даних, що складається з зображень квітів.

ABSTRACT

In the thesis has been developed a system for determining the dominant colors in the image using machine learning methods.

Machine vision systems allow to determine their location and presence on the basis of color and color model of objects, to measure and count in the production process directly on the conveyor or production line, to classify drugs by color content, to identify defective drugs by shade of their content. Therefore, the definition of the main color palette is an urgent problem, which is solved by machine learning methods.

The aim of the work is to create a system of color identification from the image using machine learning methods.

As a result of research on the methods of color detection in the image, a color recognition system was designed and implemented, consisting of a module of convolutional network and a module of decomposition into color shades by KMeans.

The convolutional neural network module uses the best weight conservation model and function to classify images according to one of seven color classes.

The color decomposition module outputs the selected number of dominant shades in the form of a chart with the marking of each shade in HEX format.

The color identification system has been trained and tested using two data sets: the classic color data set with car images and a specially created data set consisting of color images.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ ТА ТЕРМІНІВ

CNN – Convolutional neural network, згорткова нейронна мережа.

Color detection technique – техніка визначення кольору.

HEX – Hexadecimal Colors, шістнадцяткове значення кольору.

KMeans – k-means clustering, алгоритм кластерізації.

KNN – K-nearest neighbors algorithm, алгоритм К-ближчих сусідей.

RGB – формат передачі кольору: червоний, зелений та блакитний.

SVM – Support vector machines, метод опорних векторів.

ВСТУП

Системи машинного зору дозволяють на основі кольору та колірної моделі об'єктів визначати їх місцезнаходження та наявність, проводити вимірювання та підрахунок у процесі виробництва безпосередньо на конвеєрі чи виробничій лінії, виконувати класифікацію лікарських препаратів за кольором вмісту, виявляти браковані лікарські препарати за відтінком їхнього вмісту. Колір широко використовується як один із ознак розпізнавання об'єктів, і для ідентифікації кольору на зображеннях використовуються різні методи машинного навчання.

Популярними методами є колірні моменти та колірна гістограма. Колірні моменти вимірюють колірну подібність між зображеннями за допомогою середнього значення, стандартного відхилення та асиметрії, тоді як колірна гістограма заснована на обчисленні частоти, з якою кольори зустрічаються у зображенні. Ідентифікація кольору на зображенні здійснюється наступними методами машинного навчання: KNN, KMeans, SVM. Останніми роками класифікація кольорів за допомогою глибокого навчання швидко розвивається.

Класифікація кольорів за допомогою машинного навчання має багато застосувань і важко знайти галузь життя сучасної людини. Наприклад, застосування у галузі косметології – визначення тону шкіри, чи у сільському господарстві - розрізняти культуру та ґрунт при використанні робототехніки для визначення та обрізання культур, визначення стиглі плодів, виявлення дефіциту поживних речовин на основі кольору листя. Якщо мова йде про безпеку на дорозі та про автономні транспортні засоби – варто згадати системи виявлення та розпізнавання сигналів дорожнього руху чи розпізнавання кольорів транспортних засобів у різних умовах освітлення. Столярні та побутові послуги, ландшафтна архітектура, графічний дизайн та звичайний пошук інформації в Інтернеті – всі це об’єднує необхідність використання

методів колірної класифікації для модернізації та поліпшення процесів, що протікають у наведених галузях

Метою роботи є створення системи для визначення домінуючих кольорів на зображенні з використанням методів машинного навчання.

Основними задачами даної дипломної роботи є:

- дослідження методів машинного навчання для класифікації кольорів і їх характеристики;
- формування вимог до створюваної системи;
- вибір архітектури, технологій і засобів реалізації системи;
- реалізація алгоритмів класифікації кольорів;
- реалізація системи виявлення атак;
- тестування та порівняння алгоритмів на тестовій виборці.

1 ОГЛЯД ІСНУЮЧИХ СИСТЕМ І МЕТОДІВ ДЛЯ ВИЗНАЧЕННЯ КОЛЬОРІВ

1.1 Огляд існуючих систем класифікації кольорів на зображенні

Color detection technique – це програма, яка ідентифікує колір того пікселя, який запитує користувач. У цій системі порівнюються коди кольорів, які передбачені у програмі із зображенням, кольори якого ми хочемо знати. Система Color detection technique була розроблена з урахуванням людей, які страждають від дальтонізму [1].

Система Color detection technique фіксує, аналізує та порівнює значення R,G,B і надає відповідний колір.Система має величезну колекцію кольорів, які, безумовно, дають найбільш близький вихід данних, навіть якщо колір утворився через суміш двох або багатьох кольорів. Отже, програма забезпечена великою кількістю кольорів, що забезпечує величезний набір порівнянь і дуже низьку ймовірність отримати неправильний колір або відсутність кольору. Програма ColorDetection technique розроблена як програма з дуже високою чутливістю.

Модулі системи Color detection technique.

Модуль I: Зйомка та збереження зображення. У цьому модулі відбувається зйомка зображення для визначення кольору. Система зберігає зображення та змінює його розмір до 800*600 пікселів зображення, яке користувач надає для введення.

Модуль II: Обробка зображень. Система відкриває вікно розміром 800*600 пікселів, зображення обробляються і програма завантажує зображення у вікно. Якщо всі зображення ідеально оброблені програма визначає колір.

Модуль III: Визначення кольору. Після того як система завершила обробку зображень, можна клацнути в будь-якому місці зображення і отримати назву кольору разом зі значеннями R, G, B.

На цьому етап виконання системи визначення кольору завершено. На цій фазі колір виявляється за допомогою зображення, і програма запускається після натискання клавіші esc.

Переваги і недоліки.

Система визначення кольору може працювати в будь-якому комп'ютері з мінімальними характеристиками. Процес виявлення кольору займає менше хвилини і це дуже вигідно для користування та інтеграції. За допомогою Color Detection technique легко визначити колір і дати його назву тим, хто його використовує.

Система має відкритий код та знаходиться у загальному доступі. Недоліком системи є відсутність зручного інтерфейсу користувача, користувач, що не має досвіду роботи з програмним кодом та його запуском. Тобто,система не призначена для масового користування.

HandbagBrandandColorDetection (Визначення бренду сумочки та кольору) – ще одна система для розпізнавання кольору [2]. Система дозволяє автоматично виявляти теми, об’єкти (наприклад, людей, організації, продукти) та ключові слова в статті, опублікованій будь-яким з брендів розробників.

Створено прототип класифікатора брендів сумок, тому що це предмети моди, де Condé Nast (компанія розробників) вже має значну присутність. Крім того, з точки зору комп’ютерного зору, сумки є досить складними об’єктами. Багато брендів мають особливості, які виділяють їх візуально.

Класифікація кольорів сумок виконується нейронною мережею Inception-v3. По двісті зображень з кожного класу було витягнуто з навчальних даних і використано для тестування.

Результати найкращої моделі представлені нижче на рисунку 1.1. Числа в матриці плутанини є результатами тестового набору, де рядки є справжніми мітками, а стовпці — прогнозованими мітками.

Підхід системи включає два кроки: локалізація об’єкта – визначте область зображення, де розташована сумочка та вилучення кольорів – витягує основні кольори з цієї області.

	Chanel	Coach	Gucci	Marc Jacobs	Kate Spade	No Handbag	Prada	Vuitton
Chanel	0.83	0.00	0.01	0.02	0.00	0.00	0.00	0.01
Coach	0.01	0.85	0.00	0.05	0.05	0.01	0.04	0.03
Gucci	0.01	0.00	0.85	0.02	0.00	0.01	0.01	0.02
Marc Jacobs	0.00	0.03	0.01	0.78	0.00	0.01	0.03	0.00
Kate Spade	0.00	0.01	0.01	0.01	0.87	0.00	0.00	0.00
No Handbag	0.09	0.06	0.08	0.09	0.04	0.97	0.04	0.09
Prada	0.03	0.03	0.02	0.03	0.01	0.00	0.85	0.01
Vuitton	0.01	0.00	0.00	0.02	0.00	0.01	0.01	0.81

Рисунок 1.1 – Результати найкращої моделі

У підсумку можна сказати, що ця система використовує існуючі архітектури CNN з відкритим кодом для створення специфічних для предметної області моделей комп’ютерного бачення, які працюють близько до рівня експертів.

Класифікація кольорів допомагає покращити зовнішній вигляд дерев’яних виробів, зібраних з кількох панелей, завдяки відмінності кольорів поверхні масивних дерев’яних панелей [3].

Набір векторів ознак був розділений на різні кластери за допомогою алгоритму K-середніх для досягнення класифікації кольорів, і, таким чином, панелі з масиву з подібним кольором поверхні були класифіковані в одну категорію. Крім того, під час подвійної кластеризації на основі колірного моменту другого порядку було реалізовано розпізнавання текстури на основі класифікації кольорів.

Наведена система перевірила ефективність сортування кольорів поверхні масиву деревини на основі машинного зору та алгоритму навчання без нагляду. Завдяки створенню нових наборів даних та аналізу впливу різних комбінацій кольорових ознак було запропоновано новий метод класифікації багаторівневого сортування за кольором плит з масиву деревини та реалізовано розпізнавання текстур.

1.2 Огляд методів машинного навчання для класифікації кольорів на зображенні

Популярними методами є колірні моменти та колірна гістограма. Колірні моменти вимірюють колірну подібність між зображеннями за допомогою середнього значення, стандартного відхилення та асиметрії, тоді як колірна гістограма заснована на обчисленні частоти, з якою кольори зустрічаються у зображенні.

Для обчислення колірної гістограми може бути обрано кілька варіантів колірного простору, наприклад, колірний простір RGB, колірний простір HSV, колірний простір CIE L*a*b* тощо [4].

Крім кольорових моментів і колірної гістограми, ще один популярний метод заснований на кольорових мітках . Існують два популярних методу для призначення лінгвістичних кольорових міток пікселю зображення, використовуючи або назви кольорів на основі мікросхем, або назви кольорів на основі реального зображення. Назви кольорів на основі мікросхем отримують шляхом зіставлення значень RGB з іменами кольорів позначеного набору кольорових мікросхем. Він добре працює, коли зображення зроблено в умовах ідеального освітлення.

1.2.1 Методи кластеризації

Одним із традиційних методів розпізнавання кольорів є алгоритм машинного навчання K-найближчих сусідів. K-Nearest Neighbor – це один із найпростіших алгоритмів машинного навчання, заснований на техніці навчання під керівництвом. Ідентифікація кольорів KNN здійснюється на основі виділених ознак, таких як колірна гістограма, колірна корелограма, колірні моменти.

Алгоритм KNN передбачає подібність між новим випадком/даними та наявними випадками та поміщає новий випадок у категорію, яка найбільш схожа на наявні категорії. Алгоритм KNN зберігає всі доступні дані та

класифікує нову точку даних на основі подібності. Це означає, що коли з’являються нові дані, їх можна легко віднести до категорії свердловин за допомогою алгоритму KNN. Алгоритм KNN можна використовувати як для регресії, так і для класифікації, але в основному він використовується для задач класифікації. KNN є непараметричним алгоритмом, що означає, що він не робить жодних припущень щодо базових даних.

Алгоритм KNN на етапі навчання просто зберігає набір даних, і коли він отримує нові дані, він класифікує ці дані в категорію, яка дуже схожа на нові дані.

Іншим традиційним алгоритмом машинного навчання є алгоритм KMeans, який використовують для ідентифікації кольорів у зображенні на основі значень колірного простору RGB, CYMK, HSV [5].

1.2.2 Штучні нейронні мережі

Один з основних підходів, що найбільш широко використовувався в галузі розпізнавання об’єктів на зображенні, є нейронні мережі. Для навчання таких моделей використовуються маркована вибірка даних, що складається з масиву зображень та відповідного їм масиву позначок, що визначають категорію, до якої відноситься зображення.

Штучні нейронні мережі дозволили створити обчислювальні моделі, що перевершують за продуктивністю штучний інтелект із традиційними алгоритмами.

Нейронні мережі мають деякі переваги та недоліки.

– Багатошаровий перцептрон є модель з безліччю прихованих параметрів, які залежать від числа нейронів мережі. Параметризована модель потенційно здатна до інкапсуляції більш складних, високорівневих функцій, але при цьому вимагає більше часу та обчислювальних ресурсів для навчання та налаштування параметрів.

– Пошук оптимального значення нейронної мережі вразливий до наявності локального мінімуму, здатного зупинити процес градієнтного спуску.

– Навчена нейронна мережа вимагає мінімальних обчислювальних ресурсів до роботи у режимі розпізнавання (пророцтва категорій).

– Нейронна мережа демонструє розширені здібності до онлайн навчання, коли розмір вибірки не фіксований і поповнюється за рахунок надходження нових даних.

В останні роки класифікація за допомогою глибокого навчання швидко розвивається. Hiary та ін. запропонували двоетапний метод глибокого навчання для класифікації видів квітів [6]. Перший крок складається з сегментації квіткової області за допомогою повністю згорткової мережі, що складається з 5 блоків архітектури VGG16 і ще три деконволюційні шари. Другий крок стосується класифікації типу квітки за допомогою згорткової нейронної мережі, яка також використовує архітектуру VGG16, за якою слідують 3 згорткові шари з 512 картами ознак. Вони обрали VGG16, оскільки він краще відповідає завданням класифікації квітів порівняно з іншими методами глибокого навчання.

Оцінка ефективності їхнього методу була проведена на трьох різних наборах даних, двох із Оксфорда – наборі даних Oxford 17 та Oxford 102 – і наборі даних Zou-Nagy. Результати показують, що запропонований ними метод може досягти принаймні 97% точності на цих наборах даних. Гурнані та ін. порівняли продуктивність архітектури GoogleNet та AlexNet у класифікації різних квітів із набору даних Oxford 102 (датасет з 102 категорій рослин) [7]. Архітектура GoogleNet використовує Inception як основу, тоді як AlexNet використовує вісім шарів, причому перші 5 шарів є згортковими, а останні 3 шари є повністю пов’язаними. Вони зберігали однакові гіперпараметри під час навчання обох архітектур, дійшовши висновку, що GoogleNet дала кращу продуктивність, ніж AlexNet.

1.2.3 Метод опорних векторів

Метод опорних векторів розглядає кожен екземпляр даних (зображення) як точку в N-мірному просторі, де N відповідає розмірності даних або загального числа пікселів зображення. Кожна з точок належить до певного класу (категорії). При цьому завдання розпізнавання подається у вигляді задачі по знаходженню такої гіперплощини в n-мірному просторі, яка б відокремлювала всі точки, що відповідають зображенням даного класу, від інших, що не належать йому. Припускаючи, що таких гіперплощин може існувати багато, метод опорних векторів ставить метою відшукання площини, відстань до якої від найближчої точки максимально в межах множини можливих варіантів – оптимальну розділяючу гіперплощину та відповідний їй оптимальний класифікатор. [8]

Метод опорних векторів має деякі переваги та недоліки.

1) Метод опорних векторів використовує вектори, відібрані з навчальної вибірки, при цьому кількість параметрів обмежена зверху розміром вибірки, але в практиці то, можливо проріджена з допомогою використання інженерії ознак.

2) Метод опорних векторів при коректному виборі метапараметрів гарантує перебування глобального рішення.

3) Метод опорних векторів у деяких випадках, коли число векторів велике в порівнянні з розміром вибірки, будує прогнози значно повільніше.

4) Метод опорних векторів демонструє розширені здібності до онлайн навчання, коли розмір вибірки не фіксований і поповнюється за рахунок надходження нових даних.

1.3 Висновки та уточнення задач

Методи глибокого навчання у задач класифікації кольорів мають потенційну перевагу у досягненні хорошої продуктивності. Порівняння продуктивності деяких архітектур методів машинного навчання дозволяє

припустити, що система, що розробляється в дипломній роботі повинна задовольняти наступним вимогам:

– система повинна виконувати класифікацію домінуючого кольору на зображені;

– система повинна розрахувати кольорні компоненти;

– мати низьку ймовірність помилково-позитивних та помилково-негативних результатів.

Для реалізації даної системи потрібно розглянути наступні задачі:

– аналіз методів машинного навчання для класифікації кольору на зображенні;

– формулювання вимог до системи;

– реалізація модуля класифікації домінуючого кольору на зображенні за допомогою CNN;

– реалізація модуля ідентифікації кольору на основі розрахунку кольорних компонентів методом KMeans;

– навчання нейронної мережі за допомогою двох наборів даних;

– тестування нейронних мереж на тестовій вибірці;

– аналіз результатів системи класифікації домінуючого кольору на зображені та розрахунку кольорних компонентів.

2 ПРОЕКТУВАННЯ СИСТЕМИ ДЛЯ ВИЗНАЧЕННЯ ДОМІНУЮЧИХ КОЛЬОРІВ НА ЗОБРАЖЕННІ

2.1 Архітектура системи

Архітектура системи складається з двох модулів: модуль класифікації домінуючого кольору та модуль розкладання на колірні відтінки (рис.2.1).

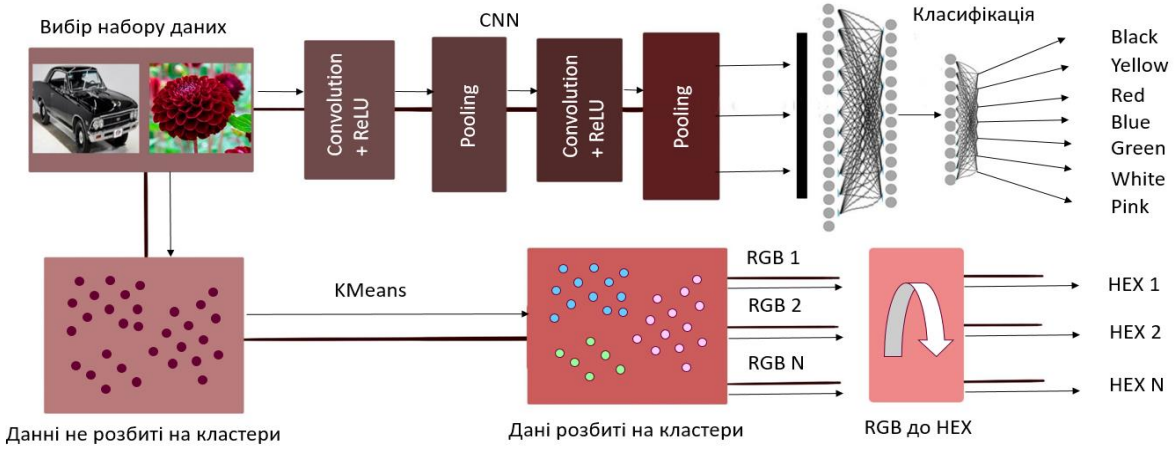

Рисунок 2.1 – Архітектура системи класифікації домінуючого кольору

2.2 Модуль класифікації домінуючого кольору

Класифікація домінуючого кольору на зображенні буде виконуватися за допомогою згорткової нейронної мережі (рис.2.2).

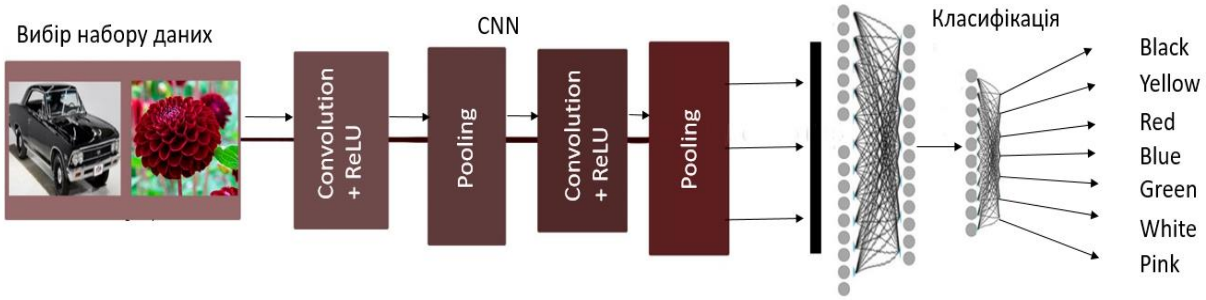

Рисунок 2.2 – Модуль класифікації домінуючого кольору

Модель CNN працює в два етапи: виділення ознак і класифікація. Вилучення ознак – це етап, на якому до зображень застосовуються різні фільтри та шари для вилучення інформації та функцій з них, а після завершення вона передається до наступної фази, тобто класифікації, де вони класифікуються на основі цільової змінної проблеми.

Модель CNN представлена на рисунку 2.3.

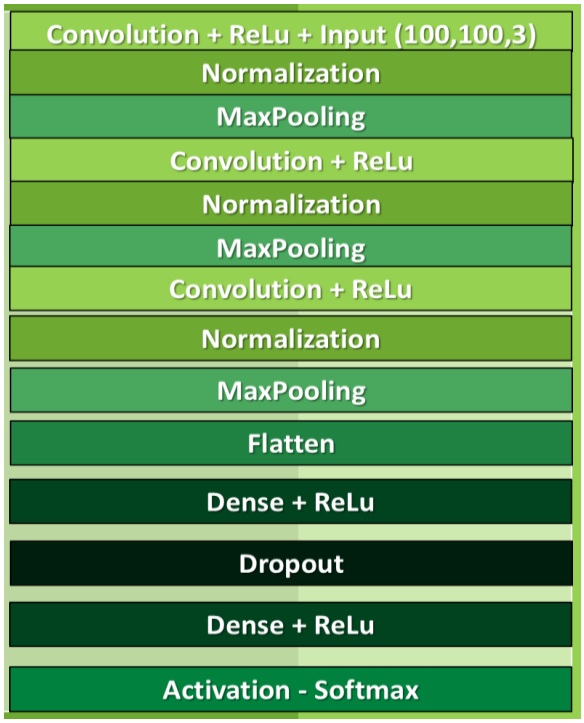

Рисунок 2.3 – Модель CNN

Модель включає в себе вхідний шар, шар згортки та функцію активації, об’єднуючий шар, повністю підключений шар.

Вхідний шар – це вхідне зображення і може бути у RGB. Кожне зображення складається з пікселів у діапазоні від 0 до 255. Потрібно нормалізувати їх, тобто перетворити діапазон від 0 до 1, перш ніж передати його в модель.

Шар згортки – це шар, де фільтр застосовується до вхідного зображення для вилучення або виявлення його ознак. Фільтр застосовується до зображення кілька разів і створює карту об’єктів, яка допомагає класифікувати вхідне

зображення. Результатом застосування фільтра до зображення є те, що користувач отримує карту функцій 4*4, яка містить деяку інформацію про вхідне зображення. На першому кроці фільтр застосовується до виділеної зеленим кольором частини зображення (рис.2.4).

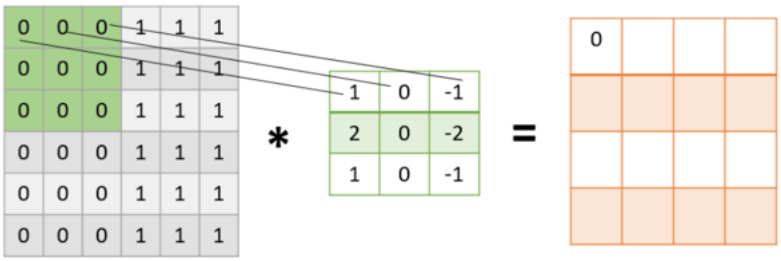

Рисунок 2.4 – Шар згортки

На наступному кроці фільтр зміщується на один стовпець. Аналогічно, фільтр проходить по всьому зображенню (рис.2.5), і користувач отримує остаточну карту функцій.

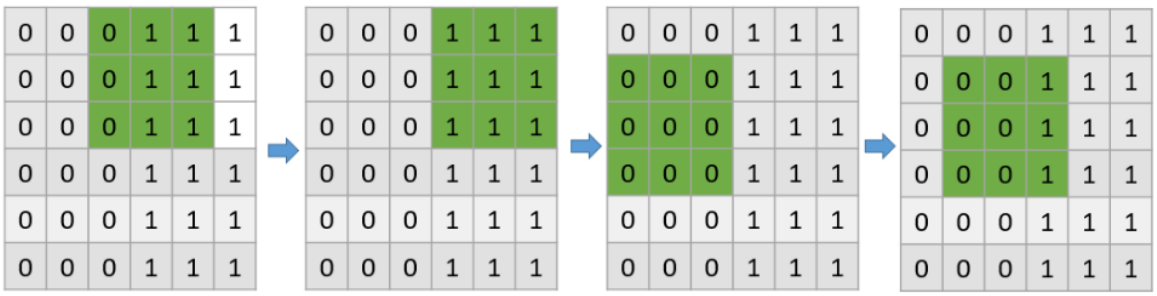

Рисунок 2.5 – Фільтр проходить через все зображення з кроком 1

Як тільки отримано карту ознак, до неї застосовується функція активації для введення не лінійності. Зазначимо, що карта об’єктів, яка отримана, менша за розмір зображення. У міру збільшення значення кроку розмір карти характеристик зменшується.

Об’єднаний шар. Рівень об’єднання застосовується після згорткового шару і використовується для зменшення розмірів карти об’єктів, що допомагає зберегти

важливу інформацію або особливості вхідного зображення та скорочує час обчислення. За допомогою об’єднання в пул створюється версія введення з нижчою роздільною здатністю, яка все ще містить великі або важливі елементи вхідного зображення. Найпоширенішими типами об’єднання є максимальний пул і середній пул. На рисунку 2.6 показано, як працює Max Pooling.

Рисунок 2.6 – Тип об’єднання Max Pooling

З кожної виділеної області береться максимальне значення, і виходить нова версія вхідного зображення розміром 2*2, тому після застосування об’єднання розмірність карти об’єктів зменшується. До цього моменту було виконано кроки вилучення ознак, тепер настала черга класифікації. Повністю підключений шар використовується для класифікації вхідного зображення в мітку. Цей рівень з’єднує інформацію, витягнуту з попередніх кроків (тобто шари згортки та шари об’єднання), з вихідним шаром і в кінцевому підсумку класифікує вхідні дані у потрібну мітку.

2.3 Модуль розкладання на колірні відтінки

Модуль розкладання на колірні відтінки виконується методом KMeans (рис.2.7). Алгоритм KMeans є одним із найбільш використовуваних алгоритмів кластеризації, відомий як кластеризація k-середніх.

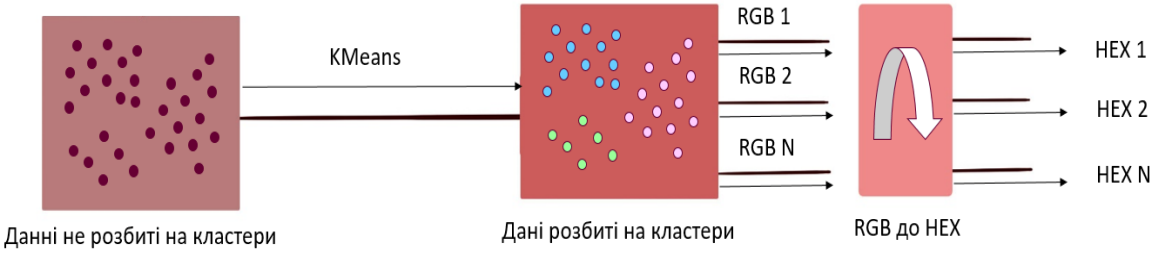

Рисунок 2.7 – Модуль розкладання на колірні відтінки

Алгоритм KMeans – це ітераційний алгоритм, який намагається розділити набір даних на K попередньо визначених окремих підгруп (кластерів), які не перекриваються, де кожна точка даних належить лише одній групі.

У разі використання методу для класифікації об'єкт присвоюється тому класу, який є найпоширенішим серед сусідів конкретного елемента, класи яких вже відомі.

Порядок дії алгоритму наступний.

1) завантаження даних;
2) ініціалізація k шляхом вибору оптимальної кількості сусідів;
3) для кожного зразка в даних:
    а) обчислення відстані між прикладом запиту та поточним прикладом даних;
    б) додавання індексу зразка до впорядкованої колекції, як і його відстань;
    в) сортування впорядкованої колекції відстаней та індексів від найменшої до найбільшої, у порядку зростання;
    г) вибір перших k записів з відсортованої колекції;
    д) вибір позначок вибраних k записів;
    е) повернення вибраних раніше позначок k, що найчастіше зустрічаються.

Алгоритм використовує Евклідову метрику, тобто це найменша можлива відстань між точками A і B. Недоліком евклідової відстані є те, що

вона ігнорує схожість між атрибутами. Кожен їх розглядається як цілком відмінний від інших.

Формула обчислення Евклідова відстані (2.1).

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (qi - pi)^2} \tag{2.1}$$

Іншою важливою складовою методу є нормалізація. Різні атрибути зазвичай мають різний діапазон представлених значень у вибірці. У такому разі значення дистанції можуть сильно залежати від атрибутів з більшими діапазонами. Тому дані здебільшого проходять через нормалізацію.

Нормалізація здійснюється по формулі (2.2).

$$x' = (x - M[X]) / \sigma[X] \tag{2.2}$$

де σ – середньоквадратичне відхилення.

KMeans надає більшої ваги великим кластерам. KMeans приймає сферичні форми кластерів (з радіусом, що дорівнює відстані між центроїдом і найдальшою точкою даних) і не дуже добре працює, коли кластери мають різні форми, такі як еліптичні кластери.

Алгоритм KMeans має недоліки. Алгоритм KMeans добре фіксує структуру даних, якщо кластери мають сферичну форму. Кластери мають складні геометричні форми, KMeans погано справляється з кластеризацією даних та не справляється із завданням, коли об'єкт належить до різних кластерів рівною мірою або не належить жодному.

Алгоритм дуже чутливий до вибору початкових центрів кластерів. Класичний варіант має на увазі випадковий вибір кластерів, що дуже часто було джерелом похибки. Як варіант рішення: необхідно проводити дослідження об'єкта для більш точного визначення центрів початкових кластерів.

2.4 Набір даних

В ході розробки нейронної мережі для її навчання та тестування було використано два набора даних.

Перший набір даних VCoR (Vehicle Color Recognition) Dataset [9] – класичний набір даних з автомобілями, що сортовані за кольором, був перероблений для порівняння з другим тестовим набором. Цей набор частіше за все використовується для реалізації алгоритмів классифікації кольорів тому, що автомобіль має чітку та просту форму і насичений однотонний колір.

Набор даних включає зображення для розпізнавання кольорів транспортних засобів (VCoR) з 7 класами кольорів: червоний, чорний, білий, зелений, рожевий, синій та жовтий. Кількість зображень у наборі - 1725.

Приклад зображення з цього набору наведено на рисунку 2.8. Зображення автомобілю є одним із найкращих об’єктів для дослідження у галузі ідентифікації кольору.

Рисунок 2.8 – Приклад зображень з набору даних з автомобілями

Другий набір даних створений спеціально для цієї роботи, він включає до себе близько трьох тисяч зображень квітів різних кольорів, сортованих за колірною ознакою по різним каталогам. Об’єктом було обрано саме квіти завдяки різноманітним формам та великій кулькості відтінків на зображенні.

Кількість зображень у наборі – 2444, як і попередній набір даних класифікує 7 класів: червоний, чорний, білий, зелений, рожевий, синій та жовтий. Приклад зображення набору з квітами наведено на рисунку 2.9.

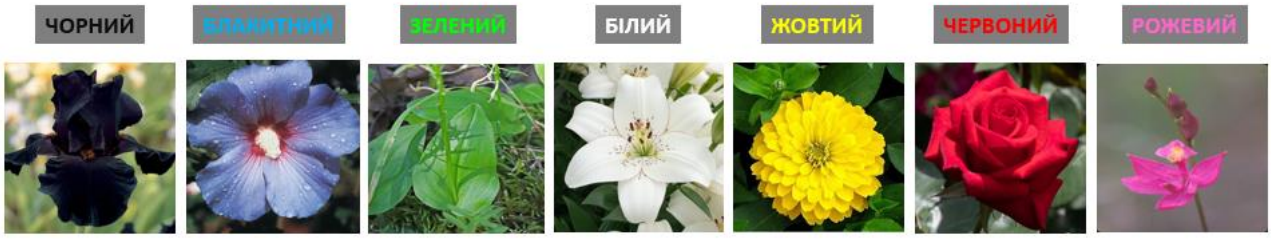

Рисунок 2.9 – Приклад зображень з набору даних з квітами

2.5 Стек технології

Для розробки системи використовуємо мову програмування Python та середовище GoogleColab.

Мова програмування Python вважається найкращою мовою програмування для машинного навчання та найкращою мовою програмування для штучного інтелекту [10]. Щоб реалізувати згорткову нейронну мережу слід використовувати мову програмування Python, яка є стабільною, гнучкою та має доступні інструменти. Переваги, завдяки яким Python найкраще підходить для машинного навчання та проектів на основі штучного інтелекту, включають простоту та послідовність, доступ до великої кількості бібліотек і фреймворків для нейронних мереж та машинного навчання, гнучкість, незалежність від платформи та широку спільноту. Це додає загальній популярності мови.

Python з його багатим стеком технологій має великий набір бібліотек для штучного інтелекту та машинного навчання. Ось деякі з них, які будуть використовуватися в роботі:

а) Keras, TensorFlow і Scikit-learn для машинного навчання;

б) NumPy для високопродуктивних обчислень та аналізу даних;

в) SciPy для просунутих обчислень;

г) Pandas для аналізу даних загального призначення;

д) Scikit-learn має різноманітні алгоритми класифікації, регресії.

Keras – це бібліотека програмного забезпечення з відкритим вихідним кодом, яка надає інтерфейс Python для штучних нейронних мереж. Keras виступає в якості інтерфейсу для бібліотеки TensorFlow .

TensorFlow – це безкоштовна бібліотека програмного забезпечення з відкритим вихідним кодом для машинного навчання та штучного інтелекту . ЇЇ можна використовувати для виконання ряду різних завдань, але вона зосереджена на глибокому навчанні нейронних мереж .

NumPy – це бібліотека для мови програмування Python , яка додає підтримку для великих багатовимірних масивів та матриць , а також містить велику колекцію математичних функцій високого рівня для роботи з цими масивами.

Pandas – це програмна бібліотека, написана для мови програмування Python для маніпуляції та аналізу даних . Зокрема, вона пропонує структури даних та операції для маніпулювання числовими таблицями та часовими рядами.

Scikit-learn містить алгоритми класифікації, регресії та кластеризації, включаючи машини опорних векторів, випадкові ліси, підвищення градієнта, k-середніх та DBSCAN, і призначений для роботи з числовими та науковими бібліотеками Python NumPy та SciPy.

GoogleColab – це онлайн-середовище JupyterNotebooks від Google. Середовище GoogleColab виконує все, що робив би локальний ноутбук (і більше), але він знаходиться в хмарі, тому не потрібно встановлювати програмне забезпечення, і воно доступне з будь-якого комп’ютера, підключеного до Інтернету. Наразі він працює лише на Python 3 і він поставляється з багатьма з найпопулярніших бібліотек Python, які вже встановлені. При необхідності встановлення додаткових пакетів, це можливо зробити це за допомогою pip у клітинці блокнота.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Розробка модуля класифікації домінуючого кольору на основі згорткової нейронної мережі

В ході розробки нейронної мережі першим кроком було обрано декілька бібліотек, що необхідні у роботі з нейронними мережами. Спочатку підключимо необхідні бібліотеки, завантажимо навчальну та тестову вибірки та стандартизуємо вхідні дані. Нижче наведено приклад деяких з бібліотек:

```
from sklearn.cluster import KMeans
import tensorflow as tf
from tensorflow import keras
import cv2
```

Один з найважливіших кроків при створенні згорткової мережі – це якісний набір данних, та, що не менш важливо – корректне завантаження цього набору. З оглядом на те, що розробка відбувається у хмарі та використовується самостійно підібраний набір зображень, завантаження відбувається за допомогою завантаження кожної одиниці набору у масив даних через модуль glob, який находить всі імена шляхів, що відповідають заданому шаблону. Масив data – зберігає зображення, масив labels – зберігає назву класу, до якого це зображення належить. Тут задається також кольорова модель зображення та розмір зображення.

```
black                                                          =
glob.glob('/content/drive/MyDrive/flower_dataset/black/*.*')
for i in black:
    image=tf.keras.preprocessing.image.load_img(i,
color_mode='rgb',
    target_size=(32,32))
    image=np.array(image)
    data.append(image)
    labels.append(0)
```

Наступний крок – це розділення набору даних на тренувальний та тестовий та означення назв класів, що будуть використані для класифікації

зображень. Для згорткової НМ множини x_tarin і x_test потрібно додатково підготувати. Справа в тому, що на вході такої мережі очікується чотиривимірний тензор у форматі:

– (batch, channels, rows, cols) – якщо data_format = 'channels_first';

– (batch, rows, cols, channels) – якщо data_format = 'channels_last'.

Тут channels – це канали на входах згорткових шарів, а параметр data_format за замовчуванням дорівнює 'channels_last', що нас цілком влаштовує. Тобто наші вхідні дані повинні мати розмірність:

(batch, rows = 28, cols = 28, channels = 1)

Але, зараз вони представлені у вигляді тривимірного тензора:

(batch, rows = 28, cols = 28)

Потрібно до них додати ще один вимір (одну вісь) для колірної компоненти (одноканального зображення).

```
y_train_full = y_train_full.reshape(y_train_full.shape[0],)
y_test = y_test.reshape(y_test.shape[0],)
x_train, x_val, y_train, y_val = train_test_split( x_train_full,
y_train_full, test_size=0.2, random_state=42)
classnames = ['black', 'blue', 'green', 'pink', 'red', 'white',
'yellow']
```

Побудова моделі. Параметри згорткового шару: перший параметр – filters. Значення фільтрів показує кількість фільтрів, на яких буде вчитися згортковий шар. З кожного такого фільтра згортковий шар дізнається щось про зображення, наприклад, відтінок, межу, форму / межу. Значення параметрів має бути ступенем 2.

Другий параметр – kernel-size. Ядро означає фільтр, який переміщатиметься за зображенням та витягуватиме елементи деталі за допомогою скалярного твору. Розмір ядра означає розмірність цього фільтра (висота x ширина). Значення розміру ядра зазвичай є непарне число, наприклад 3,5,7.. і т.д. Тут використовували розмір 3 ядра, що означає розмірність фільтра 3x3.

Наступний параметр – padding. Є два типи заповнення: SAME та VALID. У VALID padding на межі зображення немає заповнення нулями. Отже, коли відбувається згортка, відбувається втрата даних, оскільки деякі риси

неможливо знайти захоплені. SAME padding має шар нулів, заповнений по всій межі зображення, тому немає втрати даних. Більше того, розмір виведення зображення після згортки такий самий, як і на вході зображення. Оскільки на початкових шарах можливо втратити дані, то використовували SAME padding.

Шар Pooling використовується для зменшення розміру зображення разом із збереженням важливих параметрів у ролі. Таким чином, це допомагає скоротити обсяг обчислень у моделі. При виконанні згортки згортковий шар зберігає інформацію про точне положення об'єкта. Отже, і не дуже важливі деталі теж розташовані ідеально. В результаті виникає проблема, що навіть невелика зміна пікселя або функції може призвести до значної зміни виведення моделі. Використовуючи Max Pooling, ми звужуємо обсяг, і з усіх функцій беруться до уваги лише найважливіші функції. Таким чином, вищезазначена проблема вирішена. Pooling здійснюється двома способами: Average Pooling або Max Pooling. Зазвичай використовується Max Pooling.

Код функції моделі наведено на лістингу 3.1.

```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Conv2D(32, (3,3), padding='same',
activation='relu', input_shape=(100,100,3)))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.MaxPooling2D(pool_size=(3,3)))
model.add(tf.keras.layers.Conv2D(64, (3,3), padding='same',
activation='relu'))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.MaxPooling2D(pool_size=(3,3)))
model.add(tf.keras.layers.Conv2D(128, (3,3), padding='same',
activation='relu'))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.MaxPooling2D(pool_size=(3,3)))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(128, activation='relu'))
model.add(tf.keras.layers.Dropout(0.4))
model.add(tf.keras.layers.Dense(10, activation='relu'))
model.add(tf.keras.layers.Activation('softmax'))
```

Лістинг 3.1 – Розроблена модель CNN

Коли модель була прописана та створена, потрібно її скомпілювати та почати навчання. При компіляції використовується оптимізатор Adam.

Оптимізація Адама – це метод стохастичного градієнтного спуску, який базується на адаптивній оцінці моментів першого та другого порядку.

```
gpu_model.compile(optimizer=tf.keras.optimizers.Adam(learning_ra
te=1e-3),
loss=tf.keras.losses.sparse_categorical_crossentropy,metrics=[tf
.keras.metrics.sparse_categorical_accuracy])
```

Виведемо структуру цієї мережі та подивимося на кількість вагових коефіцієнтів у кожному шарі. Результат компіляції моделі зображено на рисунку 3.1.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 100, 100, 32)      896

 batch_normalization (BatchN  (None, 100, 100, 32)     128
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 33, 33, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 33, 33, 64)        18496

 batch_normalization_1 (Batc  (None, 33, 33, 64)       256
 hNormalization)

 max_pooling2d_1 (MaxPooling  (None, 11, 11, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 11, 11, 128)       73856

 batch_normalization_2 (Batc  (None, 11, 11, 128)      512
 hNormalization)

 max_pooling2d_2 (MaxPooling  (None, 3, 3, 128)        0
 2D)

 flatten (Flatten)           (None, 1152)              0

 dense (Dense)               (None, 128)               147584

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 10)                1290

 activation (Activation)     (None, 10)                0

=================================================================
Total params: 243,018
Trainable params: 242,570
Non-trainable params: 448
_________________________________________________________________
```

Рисунок 3.1 – Результат компіляції моделі CNN

На рисунку 3.1 перший шар містить 896 параметрів, другий – 18496, наступний шар повнозв'язкової НМ – 73856, та останній – 147584.

Наступним кроком є навчання мережі та їх результати роботи.

Коли модель прописана та скомпільована необхідно розпочати найважливіший єтап – навчання. Для цього використовується функція gpu_model.fit(). Саме тут задається кількість епох для навчання.

```
history = gpu_model.fit(x_train, y_train, epochs=100,
steps_per_epoch=np.ceil(x_train.shape[0]/batch_size),validation_
data = (x_val, y_val))
```

Процес навчання наведено на рисунку 3.2.

```
Epoch 985/1000
2/2 [==============================] - 0s 197ms/step - loss: 1.6795e-04 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.2885 - val_sparse_categorical_accuracy: 0.9783
Epoch 986/1000
2/2 [==============================] - 0s 197ms/step - loss: 5.5572e-05 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.2873 - val_sparse_categorical_accuracy: 0.9783
Epoch 987/1000
2/2 [==============================] - 0s 223ms/step - loss: 1.7270e-05 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.2865 - val_sparse_categorical_accuracy: 0.9783
Epoch 988/1000
2/2 [==============================] - 0s 200ms/step - loss: 1.8874e-05 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.2861 - val_sparse_categorical_accuracy: 0.9783
Epoch 989/1000
2/2 [==============================] - 0s 225ms/step - loss: 4.6336e-05 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.2857 - val_sparse_categorical_accuracy: 0.9783
Epoch 990/1000
2/2 [==============================] - 0s 203ms/step - loss: 5.3055e-05 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.2853 - val_sparse_categorical_accuracy: 0.9783
Epoch 991/1000
2/2 [==============================] - 0s 199ms/step - loss: 3.2397e-05 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.2851 - val_sparse_categorical_accuracy: 0.9783
Epoch 992/1000
2/2 [==============================] - 0s 224ms/step - loss: 7.0990e-05 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.2857 - val_sparse_categorical_accuracy: 0.9783
Epoch 993/1000
2/2 [==============================] - 0s 210ms/step - loss: 3.7942e-05 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.2862 - val_sparse_categorical_accuracy: 0.9783
Epoch 994/1000
2/2 [==============================] - 0s 227ms/step - loss: 4.1206e-05 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.2863 - val_sparse_categorical_accuracy: 0.9783
Epoch 995/1000
2/2 [==============================] - 0s 203ms/step - loss: 1.5255e-04 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.2865 - val_sparse_categorical_accuracy: 0.9783
Epoch 996/1000
2/2 [==============================] - 0s 227ms/step - loss: 1.2201e-04 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.2872 - val_sparse_categorical_accuracy: 0.9783
Epoch 997/1000
2/2 [==============================] - 0s 204ms/step - loss: 1.0723e-05 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.2879 - val_sparse_categorical_accuracy: 0.9783
Epoch 998/1000
2/2 [==============================] - 0s 201ms/step - loss: 3.5532e-05 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.2885 - val_sparse_categorical_accuracy: 0.9783
Epoch 999/1000
2/2 [==============================] - 0s 193ms/step - loss: 4.2769e-05 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.2890 - val_sparse_categorical_accuracy: 0.9783
Epoch 1000/1000
2/2 [==============================] - 0s 203ms/step - loss: 9.5933e-05 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.2904 - val_sparse_categorical_accuracy: 0.9783
```

Рисунок 3.2 – Процес навчання CNN

Точність класифікації для навчальної множини та валідації склала 98%. Після всіх кроків модель навчена та готова до завантаження.

3.2 Розробка модуля розкладання на колірні відтінки методом KMeans

Першим кроком при розробці програми потрібно перетворити колір RGB до значень HEX. Функція RGB2HEX перетворює RGB на HEX, щоб використовувати їх як мітки для кругової діаграми.

```
def RGB2HEX(color):
return "#{:02x}{:02x}{:02x}".format(int(color[0]),
int(color[1]), int(color[2]))
```

Зчитуючи колір, який знаходиться в RGB просторі, повертаємо рядок {:02x}, який відображає шістнадцяткове значення для відповідного кольору.

Метод get_image допоможе отримати зображення в Python в RGB просторі.

```
def get_image(image_path):
    image = cv2.imread(image_path)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
return image
```

Вказуємо шлях до зображення як аргумент. Спочатку читаємо файл за допомогою imread, а потім змінюємо його колірний простір, перш ніж повертати його.

Метод get_colors – викликає для вилучення верхніх кольорів із зображення та відображення їх у вигляді кругової діаграми. Метод get_colors приймає 3 аргументу:

1) image – зображення, кольори якого хочемо витягти;
2) number_of_colors – загальна кількість кольорів, які хочемо отримати;
3) show_chart – логічне значення, яке визначає, показувати кругову діаграму чи ні.

Спочатку змінюємо розмір зображення до розміру 600 x 400. Змінювати його розмір до меншого розміру не потрібно, але це зроблено, щоб зменшити кількість пікселів, що зменшить час, необхідний для вилучення кольорів із зображення. KMeans очікує, що вхідні дані будуть двовимірними, тому

використовується функція зміни форми Numpy для зміни форми даних зображення.

```
modified_image = cv2.resize(image, (600, 400), interpolation =
cv2.INTER_AREA)
modified_image = modified_image.reshape
(modified_image.shape[0]*modified_image.shape[1], 3)
```

Алгоритм KMeans створює кластери на основі наданої кількості кластерів. У цьому випадку це буде утворювати скупчення кольорів, і ці кластери будуть головними кольорами. Потім на тому ж зображенні витягуємо передбачення в змінну labels.

```
clf = KMeans(n_clusters = number_of_colors)
labels = clf.fit_predict(modified_image)
```

Далі використовуємо Counter, щоб отримати кількість міток. Щоб знайти кольори використовуємо clf.cluster_centers_.

Перебираємо ordered_colors ключі, які присутні в count, а потім нормалізуємо кожне значення, виконуючи ділення на 255.

Далі отримуємо кольори hexi . rgb. Оскільки раніше ділили кожен колір на 255, тепер знову множимо його на 255, знаходячи кольори. Show_chartє True, будує кругову діаграму з кожною частиною кольору, визначеною за допомогою count.values(), мітки як hex_colorsi кольори як ordered_colors. Повертаємо той, rgb_colors, який буде використовуватися пізніше (лістинг 3.2).

```
counts = Counter(labels)
center_colors = clf.cluster_centers_
ordered_colors = [center_colors[i] for i in counts.keys()]
hex_colors = [RGB2HEX(ordered_colors[i]) for i in counts.keys()]
rgb_colors = [ordered_colors[i] for i in counts.keys()]
if (show_chart):
    plt.figure(figsize = (8, 6))
    plt.pie(counts.values(), labels = hex_colors, colors =
hex_colors)
return rgb_colors
```

Лістинг 3.2 – Перетворення коліру HEX до RGB

4 ТЕСТУВАННЯ СИСТЕМИ

4.1 Оцінка ефективності нейронної мережи

Коли реалізація кожного з методів колірної ідентифікації було завершено – стоїть задача протестувати реалізовані методи та порівняти результати їх роботи.

Оцінка моделі є невід'ємною частиною процесу розробки моделі. Це допомагає знайти найкращу модель, яка представляє дані, і наскільки добре вибрана модель працюватиме в майбутньому.

Продуктивність системи залежить від побудови матриці неточностей, яка побудована для будь-якого завдання класифікації, причому її розмір залежить від кількості класів, включених в конкретний набір даних. Її мета полягає в тому, щоб порівняти фактичні та прогнозовані мітки, і якщо задача виявлення кольору містить шість класів кольорів квітів, то вона визначається матрицею неточностей 6 на 6 (рис. 4.1).

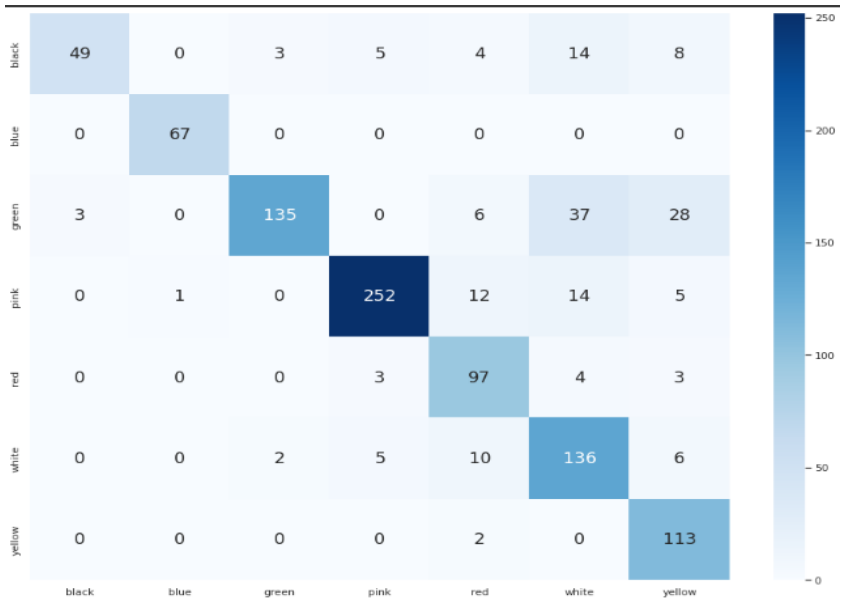

Рисунок 4.1 – Матриця неточностей нейронної мережі для набору даних з зображеннями квітів

Рядки представляють фактичні мітки, а стовпці представляють прогнозовані мітки для матриці неточностей для класифікації кольору. Наприклад: у нашому випадку перше значення 70 вказує на те, що насправді існує 70 зображень червоних кольору, які модель передбачає як червоний колір. Тепер розглянемо перше значення в 3-му рядку (зелений): значення дорівнює 6, що означає, що 6 фактично зелених кольорів передбачені як чорний (стовпець).

Зауважимо, що всі діагональні елементи мають однакову фактичну та передбачувану мітку, що означає, що ці кольори правильно класифікуються моделлю, оскільки передбачена мітка дорівнює фактичній мітці. Таким чином, усі елементи, що не мають діагоналі, вказують на неправильні кольори за класифікатором.

Матриця неточностей на основі набору зображень з автомобілями зображена на рисунку 4.2.

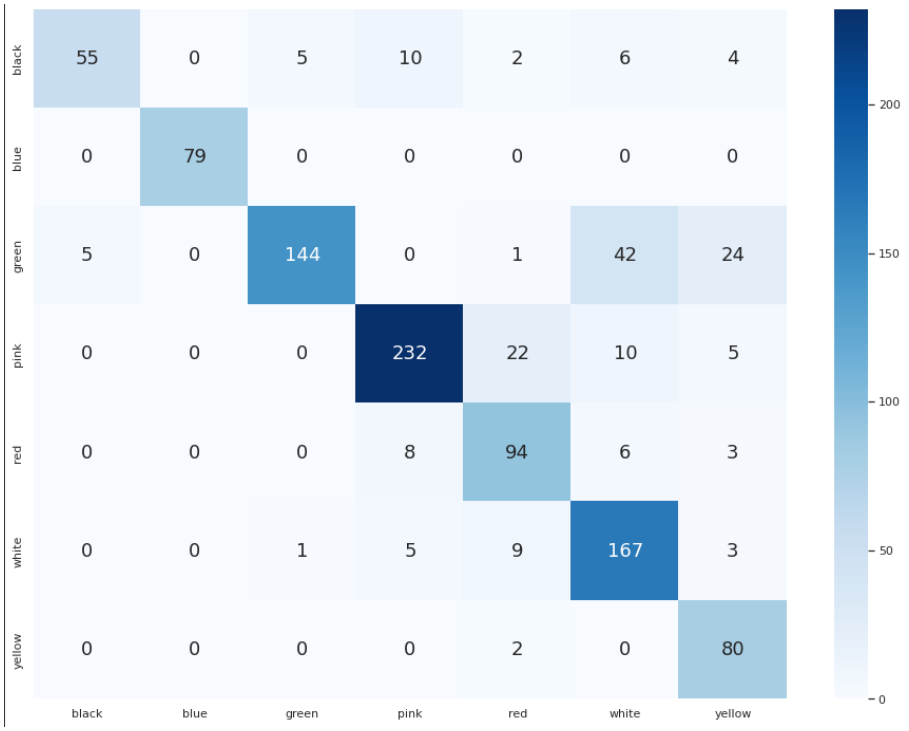

Рисунок 4.2 – Матриця неточностей нейронної мережі для набору даних з зображенням автомобілів

Точність – це показник, який оцінює загальний відсоток виявлення і помилкових тривог, вироблених моделлю IDS, який відображає загальну ймовірність успіху будь-якої IDS і розраховується як:

$$Accuracy = (TN + TP)/(TP + FP + TN + FN) \quad (4.1)$$

Як правило, з нейронними мережами стоїть ціль мінімізувати помилку. Таким чином, цільову функцію часто називають функцією вартості або функцією втрат, а значення, обчислене функцією втрат, називають просто «втратою ». Втрата помилок у квадраті для кожного навчального прикладу, також відомого як L2Loss, є квадратом різниці між фактичним і прогнозованим значеннями.

$$L = (y - f(x))^2 \quad (4.2)$$

Значення показників точності та функції втрат нейронної мережі на основі набору даних з квітами та нейронної мережі на основі набору даних з автомобілями під час тренування зображені на рисунках 4.3, 4.4 та 4.5, 4.6 відповідно. Точність нейронної мережі після 36 епохи трималася на рівні 96-100 відсотків.

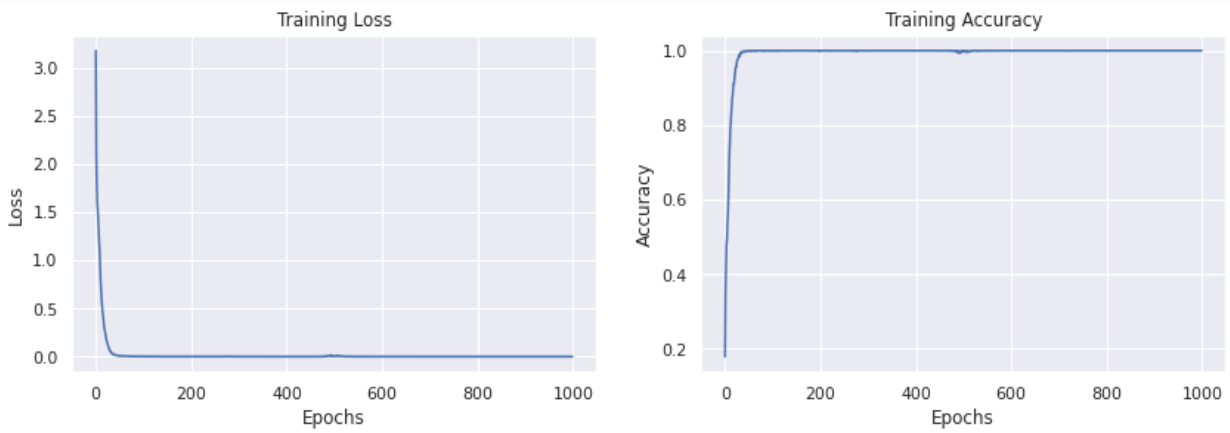

Рисунок 4.3 – Значення показників точності та функції втрат під час тренування модуля згорткової мережі на наборі даних з квітами

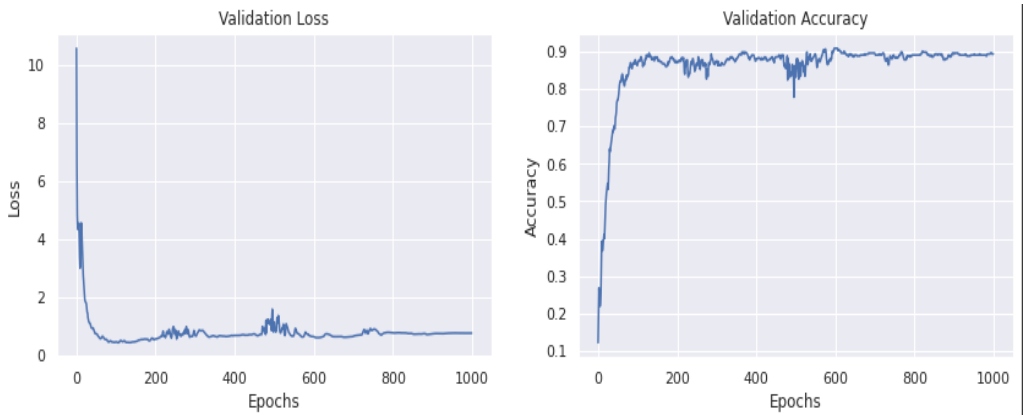

Рисунок 4.4 – Значення показників точності (валідації) та функції втрат під час тренування нейронної мережі на наборі даних з квітами

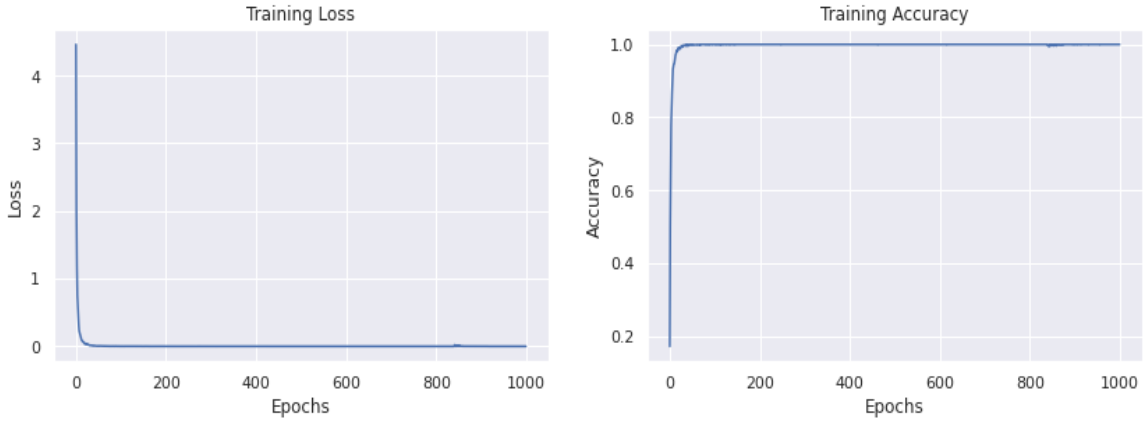

Рисунок 4.5 – Значення показників точності та функції втрат під час тренування нейронної мережі на наборі даних з автомобілями

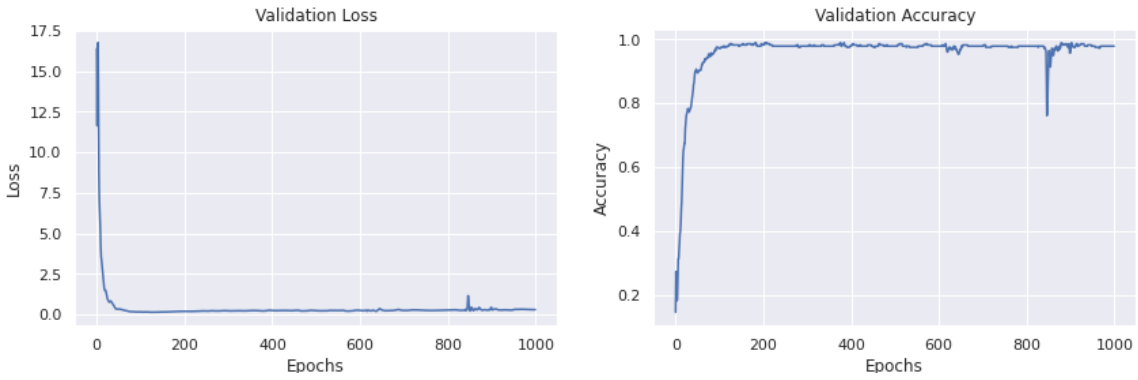

Рисунок 4.6 – Значення показників точності (валідації) та функції втрат під час тренування нейронної мережі на наборі даних з автомобілями

Система протестована на двох наборах даних з типами об’єктів різної складності та при різних умовах освітлення та якості зображення, як показано на рисунку 4.7 та 4.8. Нейронна мережа в процесі тестування показала точність 90% для датасету з квітками та 97% для датасету з автомобілями, що є майже гарною точністю при класифікації.

Рисунок 4.7 – Тестування системи для класифікації зображень квітів

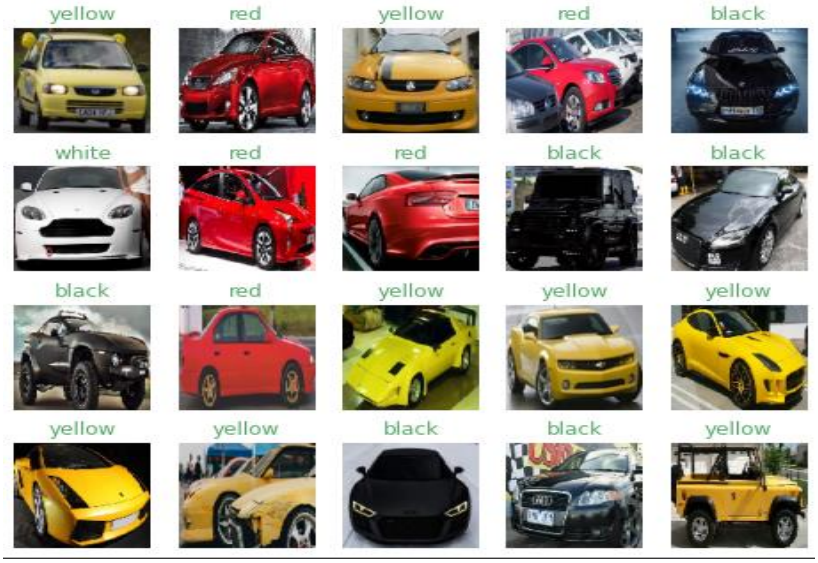

Рисунок 4.8 – Тестування системи для класифікації зображень автомобілів

Якщо порівняти результати розпізнавання з двох наборів даних та точність і кількість помилок, як і було очікувано – система краще розпізнає об’єкт автомобіль, що має більш насичений та однотонний колір та менш складну форму.

4.2 Тестування системи розпізнавання кольору методом KMeans

Наступним кроком проведемо тестування колірної ідентифікації за допомогою алгоритму KMeans. На відміну від нейронної мережі, алгоритм однаково добре розпізнає колір на кожному з наборів даних.

Перевага алгоритму є у більш наглядному виявленню декількох домінуючих відтінків, але обробити та класифікувати за один запуск велику кількість зображень він не здатній.

За результатами тестування в ході порівняння двох методів слід зауважити, що кожний має перевагу відносно до задачі, яка вирішується.

Якщо задачею є обробити велику кількість зображень та відсортувати по колірній ознаці – найбільш ефективним способом буде згорткова нейронна мережа.

Якщо потрібно зробити детальний аналіз одного конкретного зображення та точно вилучити відтінок у форматі RGB чи HEX – перевагу слід надати алгоритму KMeans.

У результаті тестування отримано діаграму з N- кількістю значень, де N – це задана кількість домінуючих відтінків, що буде вилучена з зображення.

Кожне значення супроводжується маркировкою у форматі HEX – та колірним відображенням вилученого відтінку, що може бути використано як палітра кольорів у галузі графічного дизайну, цифрового мистецтва та веб-розробки.

Результат тестування на наборі даних з квітами та автомобілями наведено на рисунку 4.9 та 4.10 відповідно.

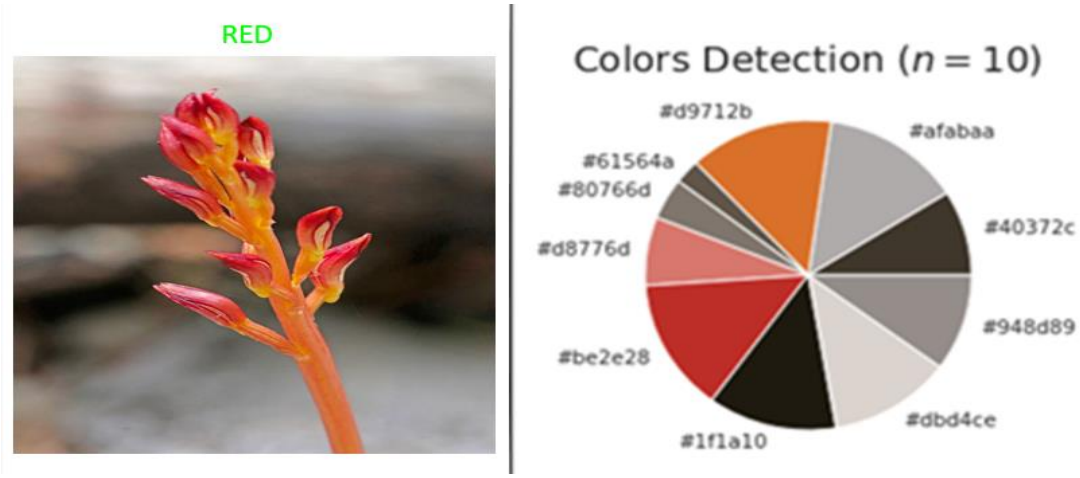

Рисунок 4.9 – Результат роботи алгоритму KMeans

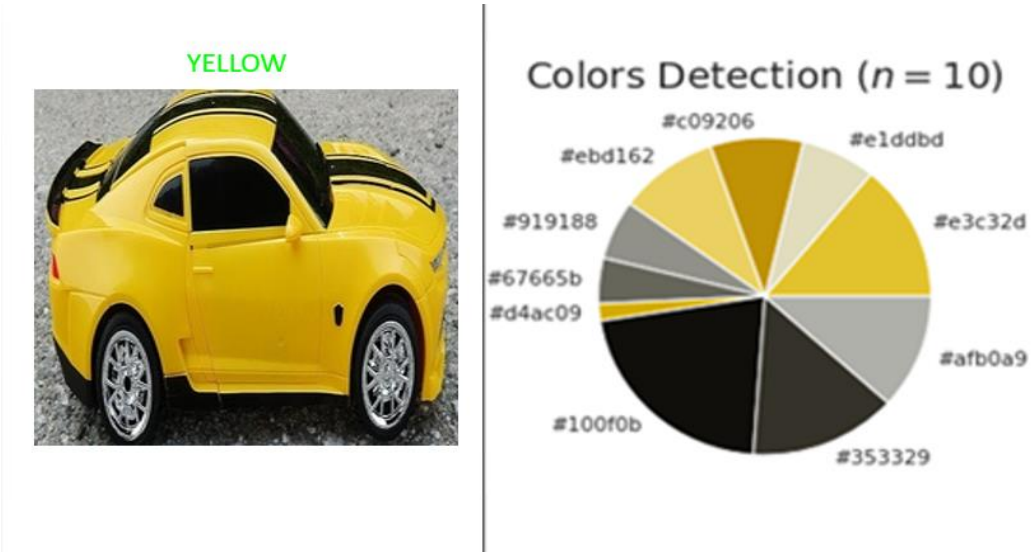

Рисунок 4.10 – Результат роботи алгоритму KMeans

ВИСНОВКИ

В ході виконання дипломної роботи спроектована, реалізована та протестована система, що виконує визначення домінуючих кольорів з зображення. Для цього проведено огляд існуючих систем класифікації кольорів і аналіз методів визначення кольору, та зроблено висновок про доцільність виконання даного проекту з метою аналізу ефективності роботи представлених методів, випробування на різних наборах даних та порівняння роботи методів KMeans та згорткової мережі.

Система складається з модуля згорткової мережі та модуля KMeans. Для визначення кольорів використовується сім класів кольорів та їх числове значення в форматі RGB/HEX.

Модуль згорткової мережі складається з моделі мережі та її навчання з виводом результатів. Модуль KMeans складається з функції перетворення кольорових значень, самого алгоритму та впорядкування результатів з відображенням у діаграмі.

Для навчання використовувалося два набори даних – перший, складається з зображень автомобілів різної якості та зроблених при різних умовах освітлення. Другий – з зображеннями квітів різної колірної насиченості та форми. Другий набір даних створений спеціально для даного проекту з метою перевірити роботу системи на більш складних умовах ідентифікації. Разом набори містять більш ніж 3 тисячі зображень для навчання.

Система протестована з кількістю епох 1000 та за результатами точності на першому наборі (набор квітів) – 90%, та 97% на другому наборі (набор автомобілів).

Алгоритм KMeans та згорткова мережа протестовані на однакових заображеннях. Алгоритм KMeans виявляє задану кулькість домінуючих кольорів однаково ефективно як на зображеннях квітів, так і на зображеннях автомобілів, коли згорткова мережа має більшу точність при классифікації кольору на зображеннях автомобілів.

В ході порівняння роботи методів зроблено висновок, що для классифікації великої кількості данних чи пошуку обьектів за кольором – краще використовувати згорткову мережу, але у випадку, де зображення одне і його потрібно детально дослідити з точки зору кольірних компонентів – слід обирати алгоритм кластерізації KMeans.

Одним з головних призначень системи є використання в якості інструмента складання колірної палітри для користувачів з галузі графічного дизайну, цифрового местецтва та веб розробки.

Подальша робота в цьому напрямку має вестися в декількох напрямках. По-перше, це використання розробленої системи класифікації на змішаному наборі даних без прив’язки до конкретного типу об’єктів. По-друге, якісне доопрацювання – наприклад, створення графічного інтерфейсу для стороннього користувача з можливістю завантажити власне зображення та обрати форму виводу результату самостійно.

Матеріали роботи доповідались на дев’ятнадцятої всеукраїнської конференції студентів і молодих науковців «Інформатика, інформаційні системи та технології».

ПЕРЕЛІК ВИКОРИСТАННИХ ДЖЕРЕЛ

1. Behic Guven , Building a Color Recognizer in Python [Електронний ресурс] – Режим доступу: https://towardsdatascience.com/building-a-color-recognizer-in-python-4783dfc72456.
2. Johan Edvinsson , Machine Learning at Condé Nast, Part 2: Handbag Brand and Color Detection[Електронний ресурс] – Режим доступу: https://technology.condenast-.com/story/handbag-brand-and-color-detection.
3. Zhengguang Wang, Color Classification and Texture Recognition System of Solid Wood Panels[Електронний ресурс] – Режим доступу: https://www.mdpi.com/1999-4907/12/9/1154/htm
4. Zhe Ming Chng, Google Colab for Machine Learning Projects [Електронний ресурс] – Режим доступу: https://machinelearningmastery.com/google-colab-for-machine-learning-projects/
5. KMeans clustering algorithm[Електронний ресурс] – Режим доступу: https://sites.google.com/site/dataclusteringalgorithms/k-means-clustering-algorithm
6. Hazem Hiary, Heba Saadeh, Flower Classification using Deep CNN [Електронний ресурс] – Режим доступу: https://www.researchgate.net/publication-/324478963_Flower_Classification_using_Deep_Convolutional_Neural_Networks
7. Соломко Ю. О., Шпінарева І. М., Ідентифікація кольорів за допомогою машинного навчання. Інформатика, інформаційні системи та технології: тези доповідей дев’ятнадцятої всеукраїнської конференції студентів і молодих науковців. – 2022. – pp. 79.
8. Метод опорных векторов – Supported Vector Machine (SVM) [Електронний ресурс] – Режим доступу: http://statistica.ru/branches-maths/metod-opornykh-vektorov-supported-vector-machine-svm/
9. VehicleColorRecognition Dataset[Електронний ресурс] – Режим доступу: https://-www.kaggle.com/datasets/landrykezebou/vcor-vehicle-color-recognition-dataset
10. Sakshi Gupta, What Is the Best Language for Machine Learning? [Електронний ресурс] – Режим доступу: https://www.springboard.com/blog/data-science/ best-language-for-machine-learning/