**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Західноукраїнський національний університет**

**Факультет комп’ютерних інформаційних технологій**

Кафедра комп’ютерної інженерії

**КОЦУР Тарас Сергійович**

**Інформаційна технологія опрацювання комунальних платежів на основі нейронних мереж / Information technology of processing utility payments on the basis of neural network**

спеціальність: 123 – комп’ютерної інженерії

освітньо-професійна програма – Комп’ютерна інженерія

Кваліфікаційна робота

Виконав студент групи Кім-21

Т.С. Коцур

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Науковий керівник:

О.Й. Піцун

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кваліфікаційну роботу

допущено до захисту

«\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ р.

Завідувач кафедри

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Л.О. Дубчак

ТЕРНОПІЛЬ - 2022

ЗМІСТ

[Вступ 7](#_Toc119847660)

[1 Аналіз підходів та засобів опрацювання комунальних платежів 10](#_Toc119847661)

[1.1 Сучасні системи опрацювання комунальних платежів 10](#_Toc119847662)

[1.2 Засоби машинного навчання для класифікації даних 18](#_Toc119847663)

[1.3 Згорткові нейронні мережі для класифікації зображень 25](#_Toc119847664)

[1.4 Аналіз завдання магістерської роботи та постановка задач дослідження 29](#_Toc119847665)

[1.5 Висновки до розділу 29](#_Toc119847666)

[2 Методи і алгоритми опрацювання даних з використанням нейронних мереж 31](#_Toc119847667)

[2.1 Узагальнений алгоритм роботи системи обліку комунальних послуг 31](#_Toc119847668)

[2.2 Архітектура згорткової нейронної мережі для розпізнавання числових знаків 35](#_Toc119847669)

[2.3 Даталогічна модель бази даних 43](#_Toc119847670)

[2.4 Висновки до розділу 51](#_Toc119847671)

[3 Програмна реалізація системи обліку комунальних послуг 52](#_Toc119847672)

[3.1 Структура програмного модулю 52](#_Toc119847673)

[3.2 Модуль розпізнавання цифрових символів із лічильника 62](#_Toc119847674)

[3.3 Порівняльний аналіз експериментів 68](#_Toc119847675)

[3.4 Висновки до розділу 72](#_Toc119847676)

[Висновки 73](#_Toc119847677)

[Список використаних джерел 74](#_Toc119847678)

[Додаток А Лістинг файлу «ImageProcessor.cpp» 77](#_Toc119847679)

[Додаток В Світлокопія виданої публікації 78](#_Toc119847680)

# Вступ

Значна частина жителів України проживає у великих містах та містечках, що зумовлює необхідність у використанні комунальних послуг, фіксуванні показників лічильників та оплаті комунальних послуг. Розвиток комп’ютерних інформаційних технологій дозволяє спростити рутинні складові життя людини. Можливість застосування сучасних технологій для збору та опрацювання комунальних платежів підвищити ефективність роботи комунальних підприємств за рахунок точнішого обліку наданих ресурсів та зробити процес взаємодії кінцевого користувача із комунальним підприємством більш простішим та комфортнішим.

Особливу увагу варто звернути на процес збору показників лічильників із домогосподарств. Даний процес є складним і надлишковим, тому потребує розробки механізму для спрощення та підвищення ефективності процесу. Більшість інформації, що знаходиться у лічильнику, наприклад електроенергії є у числовому вигляді. Наразі існує велика кількість механізмів, що дозволяє розпізнавати символи на зображенні або відео і точність та ефективність таких систем є достатньо великою та знаходиться у межах від 95 до 99.9%. Виходячи з цього доцільно використовувати існуючи підходи для підвищення точності збору інформації та уникнення людського фактору.

Більшість процесів у нашому житті переходять у цифровий вимір, що означає збільшення обсягу даних, які потрібно обробляти. Показники комунальних послуг не є винятком, особливо враховуючи велику кількість абонентів і наростаючий характер збору даних користувачів. Для оцінки продуктивності роботи того чи іншого підприємства потрібно аналізувати велику кількість даних. Наразі існує велика кількість підходів до класифікації та аналізу даних з використання механізмів штучних нейронних мереж. Використання штучних нейронних мереж для аналізу вхідних показників від абонентів комунальних послуг дозволить проаналізувати інформацію та підвищити як продуктивність підприємства так і підвищити якість надання комунальних послуг.

Розвиток апаратного забезпечення дозволив реалізувати велику кількість алгоритмів комп’ютерного зору та штучного інтелекту. До того ж, доступність високопродуктивноих апаратних складових дозволило розширити сферу використання нейронних мереж. Серед засобів комп’ютерного зору найчастіше використовуванішою є бібліотека OpenCV, які володіє великою кількістю алгоритмів та інтерфейсів для найпоширеніших мов програмування, таких як C++, Java, Python.

Для класифікації даних зараз активно використовують нейронні мережі в промислових масштабах, тобто використання нейронних мереж є не лише можливістю використання в науково-дослідних цілях, але й для промислових. Найпопулярнішими видами нейронних мереж для класифікації даних є згорткові нейронні мережі та їх модифікації.

Метою роботи є розроблення інформаційної технології опрацювання комунальних платежів на етапі формування даних та на етапі їх опрацювання з використанням штучних нейронних мереж.

Наукова новизна отриманих результатів полягає у розробці архітектури згорткової нейронної мережі для розпізнавання даних показників лічильника.

Методи досліджень. Для розв’язання поставлених задач у кваліфікаційній роботі використано: методи: комп’ютерного зору для попереднього оброблення зображень показників лічильників; об’єктно-орієнтованого програмування (для розробки програмної системи).

Орієнтовні напрямки розвитку досліджень: розроблення спеціалізованих програмних засобів збору показників лічильників; розширення функціоналу інформаційно-аналітичних систем.

Кваліфікаційна робота складається із трьох розділів, висновків, списку використаної літератури та додатків. У першому розділі було проаналізовано сучасні системи збору та опрацювання комунальних платежів. Виділено їх переваги та недоліки, що дозволили скласти список задач необхідних для реалізації у даній роботі.

У другому розділі було розроблено архітектуру згорткової нейронної мережі для розпізнавання показників лічильників електроенергії. Також у даному розділі наведено даталогічну модель бази даних розробленої програмної системи опрацювання комунальних платежів.

У третьому розділі програмно реалізовано запропоновану архітектуру згорткової нейронної мережі для розпізнавання показників електролічильників, що дозволило визначити ефективність даної мережі. Додатково програмно реалізовано систему обліку комунальних платежів з використанням веб-інтерфейсу.

Практична цінність одержаних результатів полягає в тому, що:

* розроблено та програмно реалізовано архітектуру згорткової нейронної мережі для розпізнавання показників лічильників електроенергії;
* розроблено даталогічну модель бази даних розробленої програмної системи опрацювання комунальних платежів.

Публікації та апробація до випускної кваліфікаційної роботи. За результатами наукових досліджень, проведених у випускній кваліфікаційні роботі, підготовлено тези доповіді.

# 1 Аналіз ПІДХОДІВ ТА ЗАСОБІВ ОПРАЦЮВАННЯ КОМУНАЛЬНИХ ПЛАТЕЖІВ

## 1.1 Сучасні системи опрацювання комунальних платежів

Підвищення ефективності та якості роботи комунальних підприємств завжди було важливим питанням, яким задавались у різні часи. На перших етапах впровадження подібних систем формували та застосовували бази даних споживачів комунальних послуг, оскільки це був найлегший крок в плані реалізації, хоча і не простим, оскільки необхідно було обробити велику кількість даних. Більш складнішим етапом є збір та обробка показників комунальних платежів. У більшості випадків існуючі системи розробляють на базі спеціальних інформаційно – обчислювальних сервісах організованих при органах місцевої влади. Першими в Україні подібні системи почали впроваджувати у Львові, Кременчузі, Києві. Метою впровадження подібних систем були такі задачі:

* пришвидшити оплату комунальних платежів;
* забезпечити коректне надходження оплати за комунальні послуги на відповідні рахунки;
* спростити та пришвидшити документообіг;
* знизити вартість комунальних послуг за рахунок автоматизації процесів збору платежів;
* зменшити рівень заборгованості громадянами, що дозволить збільшити надходження та покращити якість наданих послуг;
* зменшити кількість черг при оплаті;
* покращити рівень актуального стану справ з оплатою послуг, наявністю спеціальних пільг в окремих громадян тощо;
* забезпечити електронний документообіг для забезпечення громадян субсидіями [1].

Впровадження вищеперелічених пунктів в життя дозволило справді пришвидшити процес оплати комунальних платежів. Автоматизація процесу оплати комунальних платежів дозволяє зменшити кількість помилок зроблених абонентами, зокрема щодо коректності банківських рахунків. Електронний документообіг також сприяє пришвидшенню процесу надання комунальних послуг та зменшує час затрачений абонентом на рутинну паперову роботу. Автоматизація процесу збору даних про показники лічильників та автоматизація обліку абонентів дозволяє зменшити кількість персоналу. Автоматизація процесу обліку комунальних платежів зменшити довжину черг при оплаті послуг та володіти більш актуальнішим станом справ пов’заних із оплатою комунальних послуг населенням міст[2].

Умовно можна виділити дві складові при розробці системи опарцювання комунальних платежів:

- інформаційну;

- фінансову.

Фінансова складова являє собою все те, що необхідно для оплати комунальних послуг громадянином. На рисунку 1.1 наведено завдання, необхідні для реалізації автоматизованої системи збору платежів.

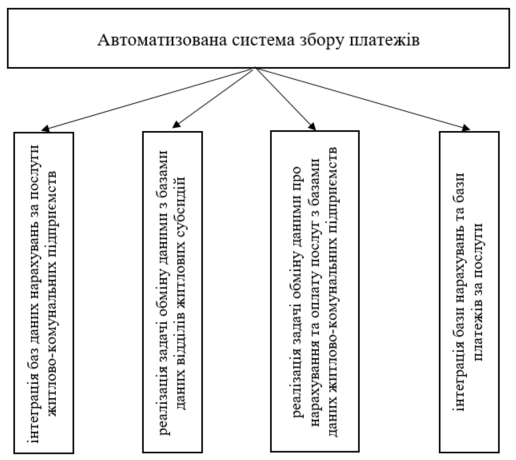


Рисунок 1.1 - Завдання, необхідні для реалізації автоматизованої системи збору платежів

Інтеграція бази нарахувань для комунальних послуг є одним із ключовим завданням для розробки системи опрацювання комунальних платежів [3].

Субсидії є важливим пунктом при оплаті комунальних послуг, тому правильна організація обміну між різними базами даних щодо абонентів, зокрема тих, хто отримують субсидію є важливим кроком для організації коректної роботи будь – якої системи оплати комунальних послуг [4,5].

Коректна організація взаємодії між різними базами даних житлово-комунальних підприємств є ще одним ключовим пунктом для досягнення зручної і ефективної роботи автоматизованої системи.

Виділяють такі особливості реалізації програмних модулів для збору платежів (рисунок 1.2).

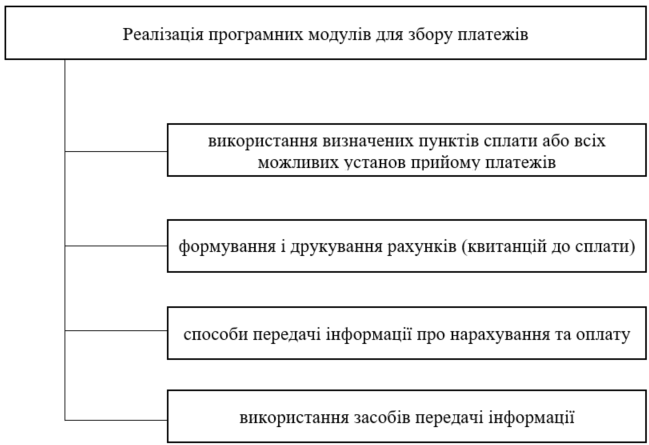


Рисунок 1.2 - Особливості реалізації програмних модулів для збору платежів

Одним із ключових аспектів є поєднання та синхронізація даних між абонентами послуг та комунальними підприємствами. До того ж, організація процесу оплати в багато чому залежить від особливостей роботи конкретного населеного пункту[6].

При проектуванні автоматичної системи оплати комунальних платежів також значну увагу звертаю формування бланку квитанції та способи її видачі, наприклад в паперовому варіанті або в електронному надсилаючи на адресу електронної пошти чи в застосунок.

При організації оплати платежів від місцевої влади потребується консолідація та синхронізація процесів для забезпечення роботи окремих систем із банками. Важливим пункт при проектуванні системи є організація логування дій користувачів, формування звітів, визначення боржників та система їх інформування.

Прикладом централізованого вирішення питання автоматичної оплати комунальних послуг є система ГІОЦ, що розроблена та впроваджена у Києві. Дана система включає в себе як модуль оплати комунальних послуг, так і модуль управління житлово-комунальним комплексом. Характеристики системи ГІОЦ наведено у таблиці 1.1.

Таблиця 1.1 - Характеристики системи комунальних платежів ГІОЦ

|  |  |
| --- | --- |
| Назва характеристики | Значення |
| Кількість рахунків платників | Більше 800 тис. |
| Кількість обслуговуючих підприємств | Більше 420 |
| Кількість банків-партнерів | 92 |

Підхід, впроваджений у даній системі складається із таких базових елементів як збір платежів від користувачів через установи банків, щоденне розщеплення платежів і перерахування їх постачальникам послуг.

Перевагою даної системи також є наявність функціоналу для призначення коштів, пільг, субсидій та інших специфічних платежів. До переваг системи відносять:

* своєчасність отримання платежів;
* оперативна інформація про стан нарахування;
* можливість призначення субсидій;
* можливість призначення додаткових пільг;
* спрощення процесу розрахунків із населенням;
* пришвидшення прийняття рішень [7].

На рисунку 1.3 наведено вигляд графічного інтерфейсу головної веб – сторінки сайту системи ГІОЦ.

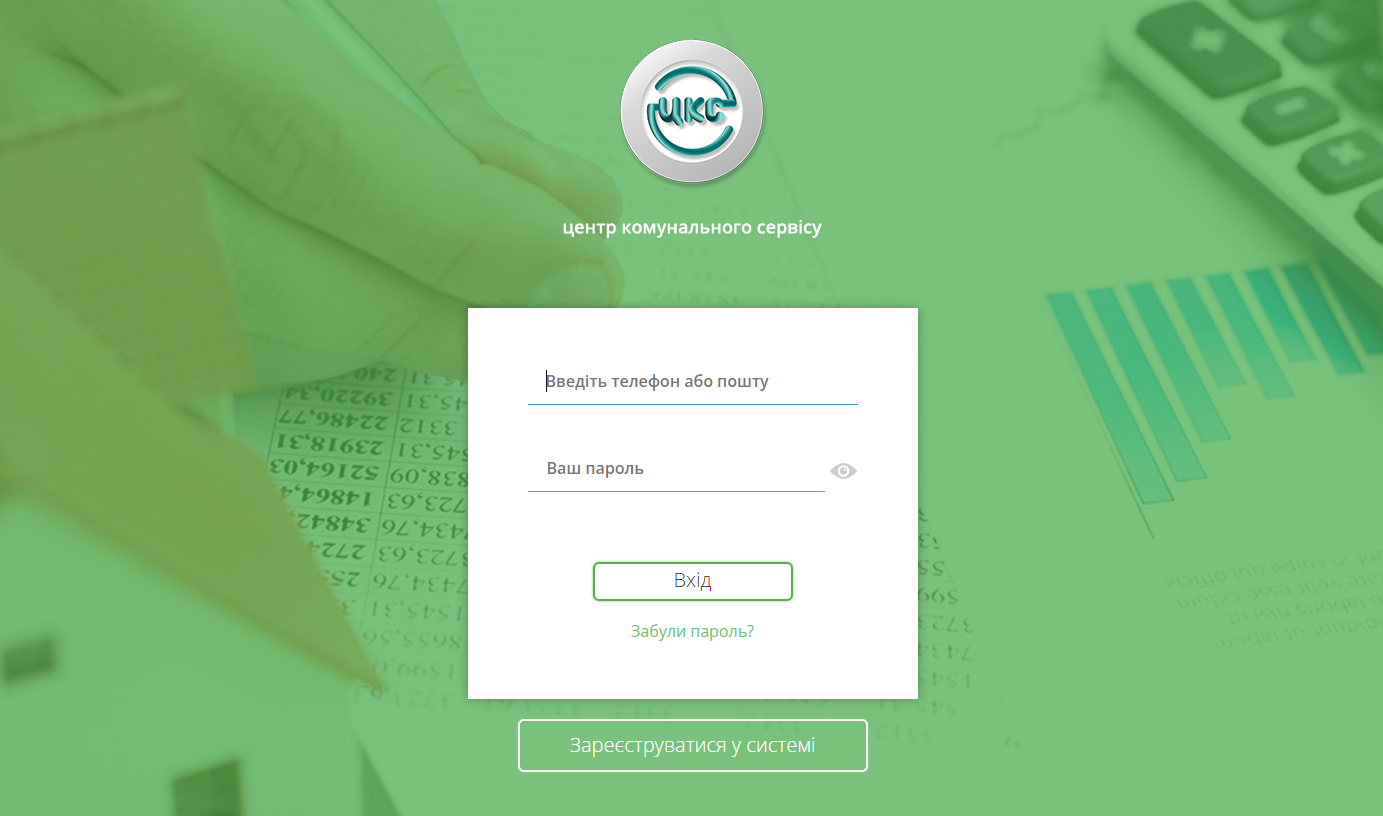


Рисунок 1.3 - Вигляд графічного інтерфейсу головної веб – сторінки сайту системи ГІОЦ

У Хмельницькому розробили власну систему для спрощення процедури оформлення субсидій та оплати комунальних платежів. Розробники вирішили створити інформаційний центр, який координує роботу пов’язану із передачею даних між базами даних комунальних підприємств для забезпечення ефективної та швидкої роботи системи. Щомісячно формується база даних платників, базуючись на інформації з баз даних інших комунальних підприємств. Інформаційний центр займається формування спільної бази даних платників. Також здійснює розподіл між надавачами послуг та отримувачами. Оплата комунальних послуг здійснюється дистанційно і незалежно від місця перебування чи проживання абонента. Квитанція про оплату отримується користувачем відразу ж після успішної оплати [8]. Користувач може використовувати такі поля для пошуку інформації:

* ПІБ платника;
* ідентифікаційний код платника;
* адреса проживання платника.

Обмін інформацією між хостами системи відбувається щоденно, що забезпечує високу оперативність обміну даними.

Розроблений функціонал дозволяє швидко та зручно формувати звіти та отримувати іншу довідкову інформацію, наприклад про стан заборгованості тощо. Вигляд головної сторінки сайту наведено на рисунку 1.4.



Рисунок 1.4 - Вигляд головної сторінки сайту «Інформаційна мережа для автоматизації процедури оплати комунальних послуг і нарахування субсидій у м. Хмельницькому»

Централізована система збору комунальних платежів у місті Луцьку призначена для зберігання, обліку інформації про сплачені платежі і місті Луцьку. Як і попередні системи, дана система працює шляхом збору інформації від різних баз даних для формування повної картини про сплачені послуги, послуги які ще потрібно оплатити та про заборгованість абонентів комунальних послуг. Запровадження даної системи дозволило:

* зменшити необхідність у відвідуванні комунальних закладів для оплати послуг;
* оновлювати актуальну інформацію про склад сім’ї;
* зменшити час на опрацювання квитанцій, зокрема корінців квитанцій при оплаті;
* пришвидшити процес надходження коштів на рахунки банків;
* спростити процес нарахування субсидій;
* зменшити рівень заборгованості користувачів за сплату комунальних послуг[9].

На базі даної системи вперше в Україні було проведено персоніфікований облік пільгових категорій населення.

Результатом роботи даної системи у місті Луцьк стало зменшення заборгованості абонентів послуг та зручніший режим взаємодії з інформацією про поточний стан справ та рівень заборгованості в місті.

Також розроблена система дозволило швидко формувати інформативний звіт про стан розрахунків. Розробники сайту оперативно оновлюють інформацію про поточний стан справ та організовують роботу через соціальні мережі та месенджери для підвищення зручності взаємодії користувачів із системою та швидкості інформування.

На рисунку 1.5 наведено графічний інтерфейс веб –сайту місцевого обчислювального центру міста Луцька.

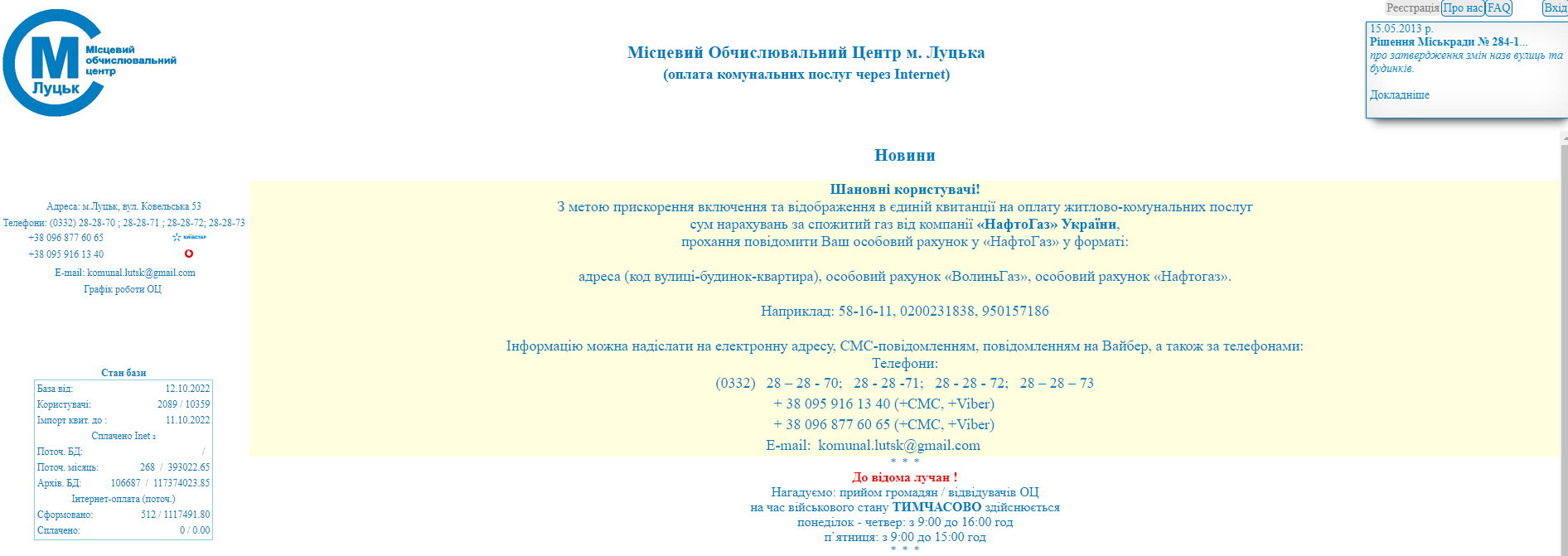


Рисунок 1.5 - Графічний інтерфейс веб –сайту місцевого обчислювального центру міста Луцька

У Кременчузі розробили автоматизовану систему «ВАС», яка працює в режимі реального часу, отримуючи дані із різних баз даних комунальних підприємств. Система «ВАС» обслуговує понад 120 тисяч абонентів та понад 50 підприємств. Актуальна інформація про платежі, заборгованості, пільги, субсидії подаються на центри прийому платежів. Координатором даної системи виступає окремий розрахунковий центр. Відділ субсидій володіє всією необхідною інформацією про конкретного абонента, тобто вичікувати у довгих чергаг немає потреби.

У Тернополі також наявний сервіс для оплати комунальних послуг. Для здійснення оплати необхідно вказати:

* населений пункт;
* номер особового рахунку;

Також можна оплатити рахунок за адресою, для цього необхідно вказати:

* назву населеного пункту;
* вулицю;
* номер будинку;
* номер корпусу;
* номер квартири;
* номер блоку.

## 1.2 Засоби машинного навчання для класифікації даних

Машинне навчання базується на класичних статистичних алгоритмах. Основною метою використання машинного навчання є прийняття рішення на основі вхідних даних. Із розвитком апаратних засобів різко зросла кількість програмних засобів що використовують алгоритми машинного навчання.

Одним із головних елементів в машинному навчанні є безпосередньо процес навчання. Добре навчена система дозволяє класифікувати дані на рівні людини або навіть краще.

Навчання з учителем є найпоширенішим видом навчання і характеризується тим, що інженер вручну розділяє дані на класи, розмічає дані для того щоби система могла зрозуміти які дані до якого класу відносяться. З допомогою даного методу навчання можна вирішувати задачі класифікації, регресії. Узагальнений процес навчання з учителем наведено на рисунку 1.6.

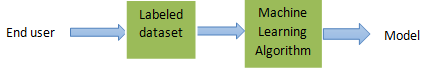


Рисунок 1.6 - Узагальнений процес навчання з учителем

Іншим відомим приклад навчання є навчання без учителя. Прикладом такого виду навчання є задачі:

* кластеризації;
* узагальнення;
* пошук правил.

Узагальнений процес навчання без учителя наведено на рисунку 1.7.

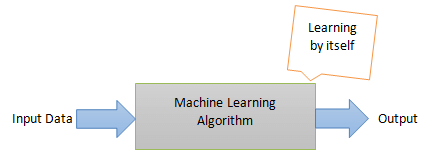


Рисунок 1.7 - Узагальнений процес навчання без учителя

Класифікацію методів навчання візуально наведено на рисунку 1.8.

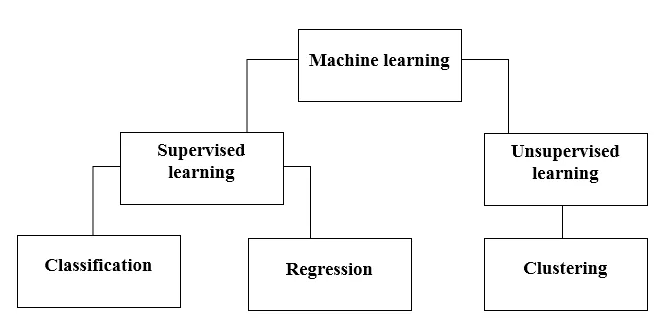


Рисунок 1.8 - Класифікація методів навчання

Класифікація дозволяє поділити елементи з вхідного набору даних за наперед відомою ознакою або ознаками. Даний підхід передбачає наявність промаркованих даних для подальшого навчання з учителем [10]. Прикладами алгоритмів є такі:

* Наївний Баєс;
* Дерева Рішень;
* Метод Опорних Векторів;
* Метод K-найближчих сусідів;

Кластеризація є також потужним підходом до аналізу даних. Перевагою кластеризації є те, що вона не потребує учителя для навчання. Підхід кластеризації базується на виділенні схожих об’єктів на основі їх певних властивостей. Області на поділено набір даних називають кластером. Кількість кластерів задається вручну або автоматично в залежності від вказаної в налаштуваннях чутливості. Дані, що перебувають у межах одного і того ж кластера повинні володіти схожими характеристиками[11].

Приклад кластеризації даних наведено на рисунку 1.9.

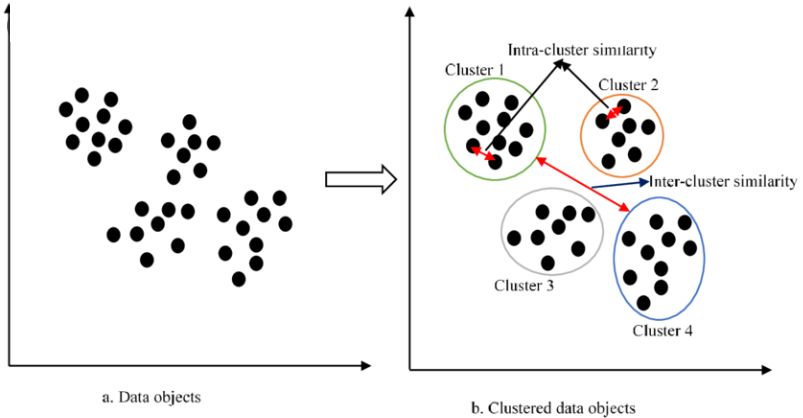


Рисунок 1.9 - Приклад кластеризації даних

Кластеризація — це неконтрольована класифікація шаблонів (спостережень, елементів даних або векторів ознак) у групи (кластери). Проблема кластеризації розглядалася в багатьох контекстах і дослідниками в багатьох дисциплінах; це відображає його широку привабливість і корисність як один із кроків дослідницького аналізу даних[12]. Однак кластеризація є складною комбінаторною проблемою, а відмінності в припущеннях і контекстах у різних спільнотах призвели до того, що передача корисних загальних концепцій і методологій відбувається повільно. У роботі [13] представлено огляд методів кластеризації шаблонів з точки зору статистичного розпізнавання образів з метою надання корисних порад і посилань на фундаментальні концепції, доступні широкій спільноті практиків кластеризації. Автори представили таксономію методів кластеризації та визначаємо наскрізні теми та останні досягнення. Ми також описуємо деякі важливі застосування алгоритмів кластеризації, такі як сегментація зображень, розпізнавання об’єктів та пошук інформації.

Принцип використання лінійної та логістичної регресії наведено на рисунку 1.10.

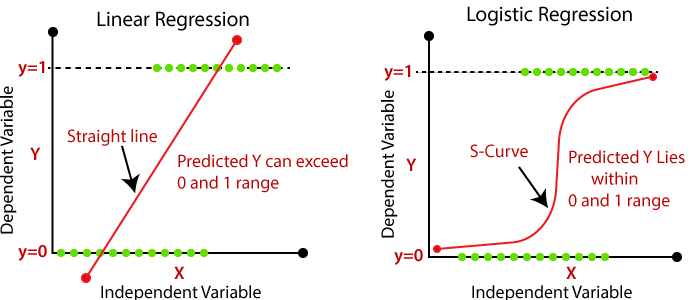


Рисунок 1.10 - Принцип використання лінійної та логістичної регресії

Модель логістичної регресії передбачає залежну змінну даних шляхом аналізу зв’язку між однією чи кількома існуючими незалежними змінними. Наприклад, логістичну регресію можна використати, щоб передбачити, чи виграє чи програє політичний кандидат на виборах або чи буде прийнятий старшокласник до певного коледжу. Ці бінарні результати дозволяють приймати прямі рішення між двома альтернативами.

Модель логістичної регресії може враховувати кілька вхідних критеріїв. У разі прийому до коледжу функція логістики може враховувати такі фактори, як середній бал студента, оцінка SAT і кількість позакласних заходів. Ґрунтуючись на історичних даних про попередні результати з тими самими вхідними критеріями, він оцінює нові випадки за ймовірністю потрапляння до однієї з двох категорій результатів.

Логістична регресія стала важливим інструментом у дисципліні машинного навчання. Це дозволяє алгоритмам, які використовуються в програмах машинного навчання, класифікувати вхідні дані на основі історичних даних. У міру надходження додаткових релевантних даних алгоритми стають кращими у прогнозуванні класифікацій у наборах даних [14].

Лінійна регресія є простою та потужною моделлю для прогнозування числової відповіді на основі набору однієї чи кількох незалежних змінних. Ця стаття здебільшого зосереджуватиметься на тому, як цей метод використовується в машинному навчанні, тому ми не розглядатимемо типові випадки використання, як-от причинно-наслідковий висновок чи експериментальний дизайн. І хоча може здатися, що лінійна регресія ігнорується в сучасному машинному навчанні, де постійно зростає складна архітектура нейронних мереж, алгоритм все ще широко використовується у великій кількості областей, оскільки він ефективний, його легко інтерпретувати та легко розширювати. Ключові ідеї лінійної регресії повторюються всюди, тому розуміння алгоритму є обов’язковим для міцної основи машинного навчання [15].

Лінійна регресія — це алгоритм, який забезпечує лінійний зв’язок між незалежною змінною та залежною змінною для прогнозування результатів майбутніх подій. Це статистичний метод, який використовується в науці про дані та машинному навчанні для прогнозного аналізу [16].

Незалежна змінна також є прогностичною або пояснювальною змінною, яка залишається незмінною через зміну інших змінних. Однак залежна змінна змінюється разом із коливаннями незалежної змінної. Регресійна модель передбачає значення залежної змінної, яка є змінною відповіді або результату, що аналізується або вивчається [17].

Метод дерева рішень є одним із найпопулярніших алгоритмів машинного навчання. Він використовує деревоподібну структуру та їх можливі комбінації для вирішення певної проблеми. Він належить до класу контрольованих алгоритмів навчання, де його можна використовувати як для класифікації, так і для регресії. Принцип роботи методу дерева рішень наведено на рисунку 1.11.

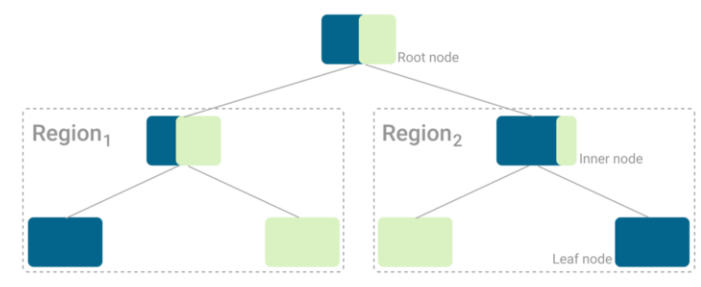


Рисунок 1.11 - Принцип роботи методу дерева рішень

Метод дерева рішень є одним із найбільш часто використовуваних алгоритмів керованого машинного навчання, який можна використовувати як для завдань класифікації, так і для регресії. Інтуїцію, що лежить в основі алгоритму Дерева рішень, дуже просто зрозуміти. Послідовність алгоритму Дерева рішень виглядає наступним чином:

Для кожного атрибута в наборі даних алгоритм дерева рішень формує вузол. Найважливіший атрибут розміщується в кореневому вузлі.

Типи дерев рішень базуються на типі цільової змінної, яку ми маємо. Він може бути двох видів:

* дерево рішень категоріальної змінної: Дерево рішень, яке має категоріальну цільову змінну, а потім називається деревом рішень категоріальної змінної.
* дерево рішень безперервної змінної: Дерево рішень має безперервну цільову змінну, тоді воно називається деревом рішень безперервної змінної.

Приклад: - у нас є проблема передбачити, чи сплатить клієнт страховій компанії премію за поновлення (так/ні). Тут ми знаємо, що дохід клієнтів є важливою змінною, але страхова компанія не має відомостей про доходи для всіх клієнтів. Оскільки ми знаємо, що це важлива змінна, ми можемо побудувати дерево рішень, щоб передбачити дохід клієнта на основі професії, продукту та інших змінних. У цьому випадку ми прогнозуємо значення безперервних змінних.

Важлива термінологія, пов’язана з деревами рішень

Кореневий вузол: він представляє всю генеральну сукупність або вибірку, яка далі поділяється на два або більше однорідних наборів.

Розщеплення: це процес поділу вузла на два або більше підвузлів.

Вузол прийняття рішення: коли підвузол розбивається на додаткові підвузли, він називається вузлом прийняття рішення.

Листовий/кінцевий вузол: Вузли, які не розбиваються, називають листовим або кінцевим вузлом.

Гілка/піддерево: Підрозділ цілого дерева називається гілкою або піддеревом.

Батьківський і дочірній вузол: вузол, який розділений на підвузли, називається батьківським вузлом підвузлів, тоді як підвузли є дочірніми вузлами батьківського вузла [20].

Класифікатори дерева рішень - це дерева рішень, які використовуються для класифікації. Як і будь-який інший класифікатор, класифікатори дерева рішень використовують значення атрибутів/особливостей даних для створення (дискретного) передбачення мітки класу. Структурно класифікатори дерева рішень організовані як дерево рішень, у якому прості умови для (зазвичай окремих) атрибутів позначають межу між проміжним вузлом і його дочірніми вузлами. Листя позначені передбаченнями міток класу. Для класифікаторів дерева рішень запропоновано велику кількість методів навчання. Більшість методів включають фазу вирощування дерева та фазу обрізки. Вирощування дерева є рекурсивним і полягає у виборі атрибута для поділу та фактичних умов поділу, які потім повторюються для дочірніх елементів, доки дані, що відповідають цьому шляху, не стануть чистими або замалими за розміром. Фаза обрізання усуває частину нижньої частини дерева, яка вивчила шум із даних, щоб покращити узагальнення [18].

Для оцінки поточного завдання ми починаємо з кореневого вузла і рухаємося вниз по дереву, дотримуючись відповідного вузла, який відповідає нашій умові чи рішенню.

Цей процес триває, доки не буде досягнуто листкового вузла. Він містить прогноз або результат дерева рішень [19].

## 1.3 Згорткові нейронні мережі для класифікації зображень

Розпізнавання кількох символів у довільних фотографіях на мобільній платформі складне як з точки зору точності, так і з точки зору продуктивності в реальному часі. У роботі [21] автори роблять акцент на області розпізнавання рукописних багатозначних цифр. Згорткова нейронна мережа (CNN) — це сучасне рішення для розпізнавання об’єктів, яке створює робоче навантаження, яке потребує як обчислень, так і даних. Щоб зменшити робоче навантаження, допустиме тренування поверхневий CNN в автономному режимі, досягаючи 99,07% точності. Невід’ємною частиною обробки зображень є використання попередньої обробки та сегментації, щоб зменшити розмір вхідного зображення, що надходить у CNN. Для реалізації CNN на мобільній платформі у більшості випадків приймається та модифікується DeepBeliefSDK для підтримки групування повністю підключених рівнів. Приклади застосування згорткової нейронної мережі наведено на рисунку 1.12.

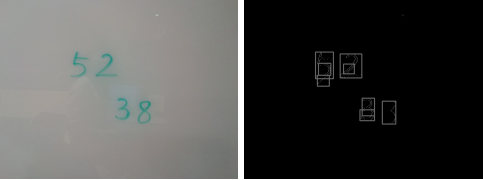




Рисунок 1.12 - Приклади застосування згорткової нейронної мережі

Після сегментації цифр вихідне зображення зсувається та масштабується на фрагменти зображення 28 × 28 окремих цифр. Патчі також надсилаються в CNN для розпізнавання. Для кожної предметної області використовують спеціальну архітектуру CNN, що складається з двох згорткових рівнів (C1 і C3), двох максимальних рівнів об’єднання (S2 і S4) і двох повністю з’єднаних рівнів.

Більш глибокі нейронні мережі складніше навчати. Представлено структуру остаточного навчання, щоб полегшити навчання мереж, які є значно глибшими, ніж використовувані раніше. Шари як вивчення залишкових функцій із посиланням на вхідні дані шару замість вивчення функцій без посилання. Автори [22] надають вичерпні емпіричні докази того, що ці залишкові мережі легше оптимізувати, і вони можуть отримати точність значно збільшена глибина. На наборі даних ImageNet ми оцінити залишкові сітки з глибиною до 152 шарів—8×глибше, ніж мережі VGG, але все ще мають меншу складність. Ансамбль цих залишкових сіток досягає 3,57% помилки на тестовому наборі ImageNet. За цим результатом посів 1 місце на Класифікаційне завдання ILSVRC 2015. Пропонуємо також аналіз на CIFAR-10 з 100 і 1000 шарами. Виключно завдяки нашим надзвичайно глибоким уявленням ми отримуємо 28% відносного покращення набору даних виявлення об’єктів COCO.

Попередня обробка повного набору даних ImageNet. Набір даних ImageNet складається з трьох частин: даних навчання, даних перевірки та міток зображень. Навчальні дані містять 1000 категорій і 1,2 мільйона зображень, упакованих для легкого завантаження. Дані перевірки та тестування не містяться в навчальних даних ImageNet (дублікати видалено).

Дані перевірки та тестування складаються із 150 000 фотографій, зібраних із Flickr та інших пошукових систем, вручну позначених наявністю чи відсутністю 1000 категорій об’єктів. 1000 категорій об’єктів містять як внутрішні вузли, так і кінцеві вузли ImageNet, але не накладаються один на одного. Випадкова підмножина з 50 000 зображень із мітками була оприлюднена як дані перевірки разом зі списком із 1000 категорій. Решта зображень використовуються для оцінки та опубліковані без міток.

Етапи попередньої обробки повного набору даних ImageNet

Є п’ять кроків, щоб підготувати повний набір даних ImageNet для використання в моделі машинного навчання:

Переконайтеся, що у вас є місце на цільовому завантаженні.

Налаштуйте цільові каталоги.

Зареєструйтеся на сайті ImageNet і запитайте дозвіл на завантаження.

Завантажте набір даних на локальний диск або віртуальну машину Compute Engine [23].

Часто набори даних з відкритим кодом можуть не відображати певні місця, користувачів або об’єкти, що призводить до властивих упереджень. Завдяки нашим службам локалізації ми можемо переконатися, що ваша модель не має зміщень, де б вона не була реалізована.

Якщо у вас уже є набір даних, але він занадто вузький, ми можемо допомогти, розширивши його зразками, отриманими з усього світу

За допомогою нашого власного мобільного додатку ми можемо легко координувати збір зображень від реальних користувачів із суворим контролем якості.

Для більш конкретних випадків використання ми можемо організувати пряме отримання зображень людьми в циклі. Це може охоплювати широкий спектр джерел, наприклад:

* мобільні зображення;
* зображення веб-камери;
* зображення камери;
* знімки камер відеоспостереження.

Проект ImageNet був натхненний двома важливими потребами в дослідженні комп'ютерного зору. Першою була необхідність встановити чітку проблему Полярної зірки в комп’ютерному зорі. У той час як у цій галузі було багато важливих завдань, над якими потрібно було працювати, від стереобачення до пошуку зображень, від 3D-реконструкції до сегментації зображень, категоризація об’єктів була визнана однією з найбільш фундаментальних можливостей як людського, так і машинного зору. Тому зростав попит на високоякісний еталон категоризації об’єктів із чітко встановленими показниками оцінки. По-друге, існувала критична потреба в більшій кількості даних для використання більш узагальнених методів машинного навчання. З моменту народження цифрової ери та доступності обміну даними в масштабі Інтернету дослідники в цих галузях наполегливо працювали над розробкою все більш і більш складних алгоритмів для індексування, пошуку, упорядкування та анотування мультимедійних даних. Але хороше дослідження вимагає хороших ресурсів. Щоб розв’язати цю проблему в масштабі (подумайте про вашу зростаючу особисту колекцію цифрових зображень, відео чи базу даних комерційної пошукової системи), було критично важливо надати дослідникам великомасштабну базу даних зображень як для навчання, так і для тестування. Конвергенція цих двох інтелектуальних причин спонукала розробників створити ImageNet [24].

## 1.4 Аналіз завдання магістерської роботи та постановка задач дослідження

Метою роботи є розроблення інформаційної технології опрацювання комунальних платежів на етапі формування даних та на етапі їх опрацювання з використанням штучних нейронних мереж. Для розв’язання поставлених задач у кваліфікаційній роботі використано: методи: комп’ютерного зору для сегментації зображень; об’єктно-орієнтованого програмування (для проектування програмних засобів аналізу зображень), алгоритми машинного навчання для класифікації даних.

Для реалізації поставленої мети потрібно виконати наступні завдання:

* проаналізувати існуючі системи обліку споживачів комунальних послуг;
* проаналізувати функціонал сучасних систем для сплати комунальних послуг;
* розробити архітектуру згорткових нейронних мереж для розпізнавання цифрових символів із показників лічильників;
* розробити даталогічну модель бази даних для обліку абонентів комунальних послуг;
* здійснити програмну реалізацію розробленої архітектури;
* провести тестування розробленої архітектури.

## 1.5 Висновки до розділу

На основі аналітичного підходу проведено порівняльний аналіз систем обліку абонентів комунальних послуг, що дозволило виділити список обов'язкових пунктів, необхідних для функціонування системи. Додатково виділено переваги та недоліки існуючих систем.

Проведено аналіз сучасних алгоритмів машинного навчання дня класифікації, кластеризації даних, що дозволяє виділити їх переваги та недоліки, що дозволить вибрати найбільш необхідний алгоритм для вирішення поствленої задачі.

Проаналізовано сучасні підходи до розробки розпізнавання образів, зокрема цифрових символів на зображенні, що дозволить розробити архітектуру згорткової нейронної мережі для розпізнавання цифрових символів.

# 2 МЕТОДИ І АЛГОРИТМИ ОПРАЦЮВАННЯ ДАНИХ З ВИКОРИСТАННЯМ НЕЙРОННИХ МЕРЕЖ

## 2.1 Узагальнений алгоритм роботи системи обліку комунальних послуг

Розроблена система обліку комунальних послуг є складною і складається з декількох блоків, які поєднані між собою. Передача даних між блоками відбувається з допомогою REST API запитів від клієнта до сервера.

REST — це набір архітектурних обмежень, а не протокол чи стандарт. Коли клієнтський запит робиться через RESTful API, він передає представлення стану ресурсу запитувачу або кінцевій точці. Ця інформація або представлення надається в одному з кількох форматів через HTTP: JSON (нотація об’єктів Javascript), HTML, XLT, Python, PHP або звичайний текст. JSON є найпопулярнішим форматом файлів для використання, оскільки, незважаючи на свою назву, він не залежить від мови, а також читається як людьми, так і машинами [25].

Технології REST зазвичай надають перевагу перед більш надійною технологією простого протоколу доступу до об’єктів (SOAP), оскільки REST використовує меншу пропускну здатність, проста та гнучка, що робить її більш придатною для використання в Інтернеті. Він використовується для отримання або надання певної інформації з веб-служби. Уся комунікація через REST API використовує лише запит HTTP. У HTTP існує п’ять методів, які зазвичай використовуються в архітектурі на основі REST, тобто POST, GET, PUT, PATCH і DELETE. Вони відповідають операціям створення, читання, оновлення та видалення (або CRUD) відповідно. Існують інші методи, які використовуються рідше, наприклад OPTIONS і HEAD. POST найчастіше використовується для створення нових ресурсів. Зокрема, він використовується для створення підпорядкованих ресурсів. Тобто підпорядкований якомусь іншому (наприклад, материнському) ресурсу. Після успішного створення поверніть статус HTTP 201, повертаючи заголовок Location із посиланням на щойно створений ресурс із статусом HTTP 201. Принцип роботи RESTfull API наведено на рисунку 2.1.

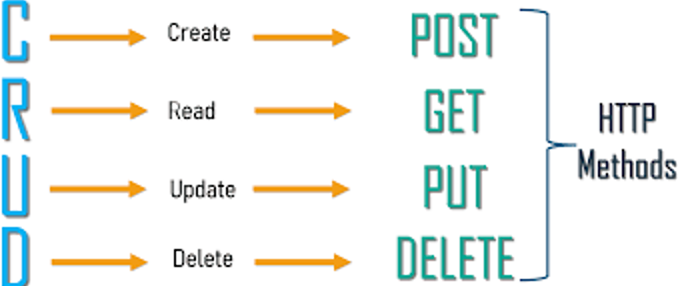


Рисунок 2.1 – Основні методи RESTfull API

Узагальнену структуру модулю обліку комунальних платежів наведено на рисунку 2.2.

Модуль опрацювання даних з лічильника

Блок опрацювання зображення показників лічильника

Блок розпізнавання цифрових символів із зображення

Блок обміну інформацією з сервером

Модуль обліку комунальних послуг

Блок прийому даних

Блок верифікації даних

Блок зберігання даних

Рисунок 2.2 - Узагальнену структуру модулю обліку комунальних платежів

Блок опрацювання зображень – призначений для зчитування зображення з камери, виділення області зображення, де наведено послідовність символів для подальшого формування кінцевого показника. Метою роботи цього блоку є створення копії зображення, але лише з інформативною областю для подальшого дослідження. Приклад роботу функціоналу для виділення необхідного блоку на зображенні наведено на рисунку 2.3.

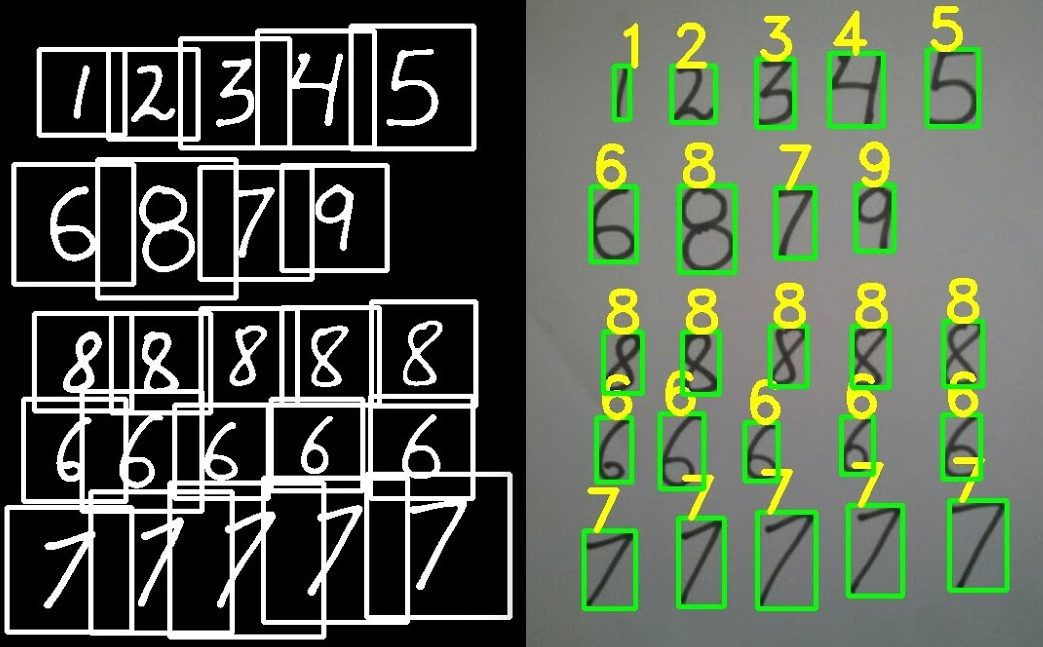


Рисунок 2.3 – Виділення чисел на зображенні

Блок розпізнавання цифрових символів – призначений для виділення цифрових символів, використовуючи алгоритми комп’ютерного зору та згорткові нейронні мереж. Засоби комп’ютерного зору використовуються для здійснення попередньої обробки, фільтрації, корекції яскравості/контрастності зображення. Згорткова нейронна мережа використовується для безпосередньо розпізнавання символів та формування масиву кінцевого результату.

Блок обміну інформацією із сервером – призначений для відправлення даних від користувача на централізований сервер для подальшого зберігання. Обмін даними відбувається з допомогою RESTful API, тому у даному модулі реалізовано функціонал для відправки даних з допомогою POST – запитів на сервер.

Блок прийому даних – це серверна реалізація RESTful API в якій передбачено набір вхідних аргументів для прийому показників від лічильника. алідація є найважливішим аспектом під час розробки програми. Він перевіряє вхідні дані. За замовчуванням базовий клас контролера використовує властивість ValidatesRequests, яка забезпечує зручний метод перевірки вхідних запитів HTTP за допомогою різноманітних потужних правил перевірки. Laravel автоматично перенаправить користувача назад до його попереднього розташування. Крім того, усі помилки перевірки та введені запити автоматично відображатимуться в сеансі.

Блок верифікації даних – призначений для перевірки введених даних, зокрема формату, кількості чисел, додаткових даних від клієнта який надсилає дані.

Блок зберігання даних – реалізовує функціонал для зберігання отриманих даних у базу даних на стороні сервера. Зокрема, реалізовано функції для роботи з базою даних:

* INSERT
* SELECT
* UPDATE

Для прикладу навдено синтаксис команди створення нової таблиці:

CREATE TABLE table\_name (column\_name column\_type constraints);

column\_name – назва конкретного стовпця з будь-яким пробілом.

column\_type – тип даних стовпця. Тип даних залежить від даних опорного стовпця. Тип даних може бути – char(), varchar(), int(), float() тощо.

constraints – щоб надати обмеження певним стовпцям, використовуються обмеження. Обмеженнями можуть бути – не null, первинний ключ, зовнішній ключ тощо. Це ключові слова, які надають набір обмежень для певного стовпця.

## 2.2 Архітектура згорткової нейронної мережі для розпізнавання числових знаків

MNIST («Modified National Institute of Standards and Technology ») — це де-факто «Hello World» набір даних комп’ютерного зору. З моменту випуску в 1999 році цей класичний набір даних рукописних зображень служив основою для порівняльного аналізу алгоритмів класифікації.

Хоча набір даних MNIST ефективно розв’язаний, він може бути корисною відправною точкою для розробки та практики методології вирішення завдань класифікації зображень за допомогою згорткових нейронних мереж.

У наборі даних уже є чітко визначений набір даних навчання та тестування, який ми можемо використовувати.

Щоб оцінити продуктивність моделі для даного тренувального прогону, ми можемо додатково розділити навчальний набір на тренування та набір даних перевірки. Потім можна побудувати графік продуктивності під час навчання та перевірки даних для кожного циклу, щоб отримати криві навчання та зрозуміти, наскільки добре модель вивчає проблему.

Модель має два основні аспекти: інтерфейс вилучення функцій, що складається зі згорткових і об’єднаних рівнів, і сервер класифікатора, який робить прогноз.

Для згорткового інтерфейсу ми можемо почати з одного згорткового шару з невеликим розміром фільтра (3,3) і скромною кількістю фільтрів (32), а потім шар максимального об’єднання. Потім карти фільтрів можна звести, щоб надати функції класифікатору.

Враховуючи, що проблема є завданням багатокласової класифікації, ми знаємо, що нам знадобиться вихідний рівень із 10 вузлами, щоб передбачити розподіл ймовірностей зображення, що належить до кожного з 10 класів. Для цього також знадобиться використання функції активації softmax. Між екстрактором ознак і вихідним шаром ми можемо додати щільний шар для інтерпретації функцій, у цьому випадку зі 100 вузлами []. Приклад структури згорткової нейронної мережі наведено на рисунку 2.4.

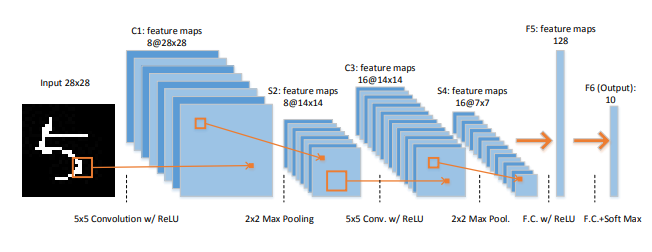


Рисунок 2.4 – Приклад структури згорткової нейронної мережі

Згортковий рівень — це перший рівень згорткової нейронної мережі. «Рівень згортки (CONV) використовує фільтри, які виконують операції згортки». Це означає, що рівень згортки використовує фільтри для отримання корисної інформації для всієї мережі.

Max Pooling вибирає найбільш спеціальні дані, як-от найбільш характерну точку на зображенні, із узагальненої інформації та виводить карту, яка має найбільш характерний параметр із кожної матриці. Max Pooling оптимізує дані та вибирає параметри високої функції для навчання, а вихід моделі шляхом навчання використання в машинах.

Об’єднання середніх значень — це процес обчислення кожного середнього параметра матриці та виведення передачі нової матриці на наступний рівень. Справа в тому, що Max Pooling просто вибирає найвищу функцію, але «Average Pooling об’єднує їх», який використовує всі параметри, які є матрицями 2 × 2, щоб отримати середнє значення.

Приклад використання операції Max Pooling наведено на рисунку 2.5.

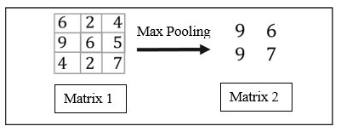


Рисунок 2.5 – Приклад використання операції Max Pooling

Структура згорткової нейронної мережі, яка використовується в цій роботі, зображена на рисунку 2.4. Згорткова нейронна мережа складається з трьох згорткових шарів разом із трьома шарами об’єднання. Для перетворення представленого набору зображень у навчальний набір 3D-фільтрів використовується згортковий шар. Щоб зменшити розмірність карти функцій і усунути зайві деталі, використовується рівень об’єднання для виконання зменшення вибірки.

Шари мінімізують просторову складність використовуваних параметрів, а також усувають проблему переобладнання. Ці рівні використовуються таким чином, щоб вони могли адаптивно вивчати більш розбірливі та оптимальні функції. Група фільтрів, які використовуються в згорткових шарах, обробляє локальні частини вхідних даних. Ці фільтри призводять до сильної реакції при роботі з деякими частинами обличчя, одночасно пригнічуючи інші частини, що призводить до вирішальних локальних структур. Після цих кількох фільтрацій у згортковому шарі виконується зменшення вибірки відфільтрованих результатів у шарі максимального об’єднання, щоб зробити його стійким до дисперсії позиції.

Основним завданням глибокої нейронної мережі, особливо у випадку розпізнавання зображень, обробки відео тощо, є систематичне вилучення функцій шляхом ідентифікації країв і градієнтів, формування поверх них текстур. Загалом згорткові шари в глибоких нейронних мережах утворюють частини об’єктів і, нарешті, об’єкти, які можуть узагальнювати характеристики вхідного зображення.

Average Pooling — це операція об’єднання, яка обчислює середнє значення для ділянок карти об’єктів і використовує його для створення карти об’єктів зі зниженою дискретизацією (об’єднаної). Зазвичай використовується після згорткового шару. Це додає невелику інваріантність перекладу, тобто переклад зображення на невелику величину не впливає суттєво на значення більшості об’єднаних результатів. Він виділяє елементи більш плавно, ніж Max Pooling, тоді як максимальне об’єднання виділяє більш виражені функції, як-от краї. Щоб краще зрозуміти це, давайте розділимо зображення на кілька частин. Якщо ми подивимося на два зображення нижче, які є лише підмножиною зображень, одне зображення містить голову кота разом із фоновим простором. Інше зображення містить лише голову кота. На першому зображенні нам достатньо однієї голови кота, щоб ідентифікувати його як кота, і нам не потрібен фон. На додаток до цього, нам потрібно виділити переважаючі ознаки, такі як око кота, яке діє як диференціатор для ідентифікації зображення [].

Лістинг коду реалізації шару пулінгу на основі Pytorch наведено на рисунку 2.6.

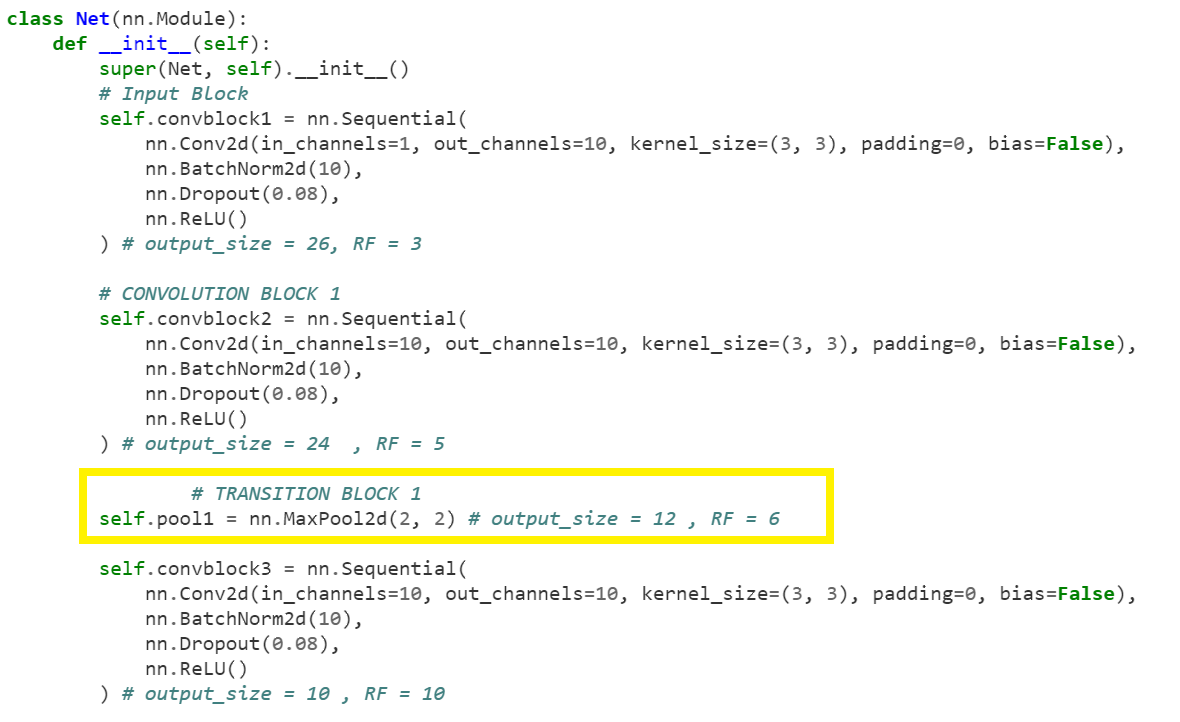


Рисунок 2.6 - Лістинг коду реалізації шару пулінгу на основі Pytorch

Max Pooling — це процес згортання, коли ядро витягує максимальне значення площі, яку воно згортає. Макс Пулінг просто каже згортковій нейронній мережі, що ми передамо лише цю інформацію, якщо це найбільша доступна інформація щодо амплітуди. Максимальне об’єднання на каналі 4\*4 з використанням ядра 2\*2 і кроку 2: Оскільки ми працюємо з ядром 2\*2. Якщо ми спостерігаємо перший набір 2\*2, на якому фокусується ядро, канал має чотири значення 8,3,4,7. Max-Pooling вибирає максимальне значення з цього набору, яке дорівнює «8».

У цьому процесі збереження однакового розміру зображення в усій нейронній мережі призведе до укладання кількох шарів. Це не є стійким через величезні обчислювальні ресурси, яких вимагає. У той же час, нам потрібно достатньо згорток, щоб отримати значущі функції.

Структура мережі VGG-16 була розроблена Симоняном і Зіссерманом і брала участь у конкурсі ILSVRC-2014, де вона досягла точності 92,7% на наборі даних ImageNet [18]. Хоча він подібний до попередніх моделей, які досягли високої точності, таких як AlexNet і LeNet, глибина мережі VGG-16 була збільшена. Ця мережа краще працює з більшими наборами даних і завданнями, пов’язаними з розпізнаванням зображень. Ця реалізація була використана для вираження більшої кількості характеристик набору даних виразів обличчя [19]. Це дозволяє вивчати та правильно класифікувати вирази облич із різним профілем, нахилом та освітленням. Це робить його ідеальним для виявлення виразу обличчя студентів, які сидять перед камерами під час онлайн-занять. Учні можуть не завжди сидіти прямо або дивитися в камеру, наприклад, коли вирішують задачу на папері. Навчальний набір даних містить обличчя в різних позах і виразах обличчя для цього сценарію.

Приклад використання операції Average Pooling наведено на рисунку 2.7.

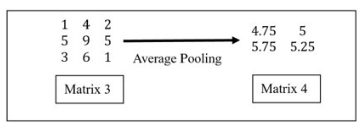


Рисунок 2.7 – Приклад використання операції Average Pooling

«Повністю зв’язаний рівень множить вхідні параметри на матрицю з вагою, а потім додає вектор зміщення», який є додаванням вагових коефіцієнтів нейронів у нейронній мережі, яким не потрібен вхідний параметр [].

Згорткові нейронні мережі (ConvNets), які є основою багатьох нових додатків, що з’являються, досягають чудової продуктивності в задачах аудіо- та візуального розпізнавання. На жаль, досягнення точності часто передбачає значні обчислювальні витрати, що обмежує можливість розгортання. У сучасних ConvNet типово, що рівні згортки споживають переважну більшість обчислювальних ресурсів під час логічного висновку. Це зробило прискорення цих шарів важливою сферою досліджень в наукових колах і промисловості. У цій статті ми розглядаємо наслідки спільної оптимізації внутрішніх структур згорткових шарів і базової реалізації фундаментальної операції згортки. Ми демонструємо, що поєднання цих методів може мати великий вплив на загальне прискорення ConvNet, досягаючи десятикратного збільшення порівняно з базовим рівнем. Ми також представляємо новий клас швидких одновимірних (1D) згорток для ConvNets за допомогою алгоритму Тума–Кука. Ми показуємо, що запропонована нами схема є математично обґрунтованою, надійною та не потребує трудомісткого перенавчання, водночас досягаючи прискорень виключно за рахунок згорткових шарів без втрати базової точності [].

Загалом, для всіх НМ, розглянутих у цій статті, ми використовуємо середньоквадратична помилка (MSE) оцінка втрат, оскільки цей вибір відповідає традиційній функції втрат, яка часто використовується для завдань регресії [26]. Інші види функцій втрат такі як середня абсолютна похибка, втрати Губера та втрати LogCost, також враховувалися для наших НМ, але вони не показують помітні переваги порівняно з MSE. Крім того, важливо підкреслити, що ми вирішили представити просто завдання регресії в цьому документі, оскільки (для нашого тестового випадку сценарій) результати, досягнуті шляхом регресії та класифікації алгоритми були близькі, але потрібно було трохи менше епох випадок регресії для досягнення найнижчого BER [].

Типова глибока нейронна мережа (або глибока згорточна нейронна мережа) має величезний простір параметрів, і за допомогою стохастичного градієнтного спуску (SGD) можна експоненціально прийти до багатьох оптимальних рішень, що складаються з різної кількості шарів, розмірів шарів і кількості параметрів. . Хоча оптимальний розмір мережі (наприклад, кількість шарів, нейронів на шар тощо) для даного набору даних невідомий на початку, ми можемо знайти кілька альтернативних приблизних мереж, які дають таку ж бажану точність. Tishby та ін. нещодавно показали, що організацію глибоких нейронних мереж можна проаналізувати за допомогою теорії інформації. У своєму дослідженні автори продемонстрували, що інформаційно-теоретичний підхід може допомогти нам краще зрозуміти як процес навчання, так і внутрішнє представлення глибинних мереж. Як правило, у глибокій нейронній мережі вхідні дані, позначені X, є багатовимірною змінною, яка є низькорівневим представленням даних, таких як пікселі зображення, тоді як бажаний вихід, Y, має значно нижчу розмірність передбачуваного категорії. Між вхідним і вихідним шарами структура глибокої мережі утворює марковський ланцюг проміжних представлень, що складається з багатьох прихованих шарів — h1,h2,…,hm.

У таблиці 1.2 наведено архітектури згорткових нейронних мереж, які використовувались у даному дослідженні.

Таблиця 1.2 - Архітектури згорткових нейронних мереж

|  |  |
| --- | --- |
| # | Тип архітекутри |
| 1 | Fully connected  Pooling (3x3)  Conv +  ReLU  Pooling (3x3)  Conv +  ReLU  Conv +  ReLU  Pooling (3x3)  Conv +  ReLU  Pooling (3x3)  Conv (3x3) +  ReLU |
| 2 | Fully connected  Pooling (3x3)  Conv +  ReLU  Pooling (3x3)  Conv +  ReLU  Pooling (3x3)  Conv (3x3) +  ReLU  Conv +  ReLU  Pooling (3x3) |
| 3 | Fully connected  Conv +ReLU  Conv +ReLU  Conv + ReLU  Pooling (3x3)  Conv +ReLU  Conv +ReLU  Pooling (3x3)  Conv +ReLU  Pooling (3x3)  Conv +ReLU  Pooling (3x3) |

Проекти машинного навчання зазвичай мають три стадії: переробка даних, побудова моделі та тонке налаштування гіперпараметрів. Багато часу витрачається на першому та останньому етапах проекту, де швидкість прогресу/повернень нижча. Інструменти, які підходять для цих етапів, і досвід значно допоможуть пришвидшити весь проект.

## 2.3 Даталогічна модель бази даних

База даних є важливою складовою будь якої програмної системи, особливо коли мова йде про системи, що призначені для обліку даних сотень тисяч користувачів. Даталогічну модель бази даних розробленої системи наведено на рисунку 2.8.

|  |
| --- |
| **Abonent** |
| Id |
| Surname |
| Name |
| Fathername |
| Address |
| Id\_code |
| Personal\_number |

|  |
| --- |
| **Payment** |
| Id |
| Period |
| Created\_at |

|  |
| --- |
| **Counter Info** |
| Id |
| title |
| Description |

|  |
| --- |
| **Counter data** |
| Id |
| Period |
| Created\_at |
| Data |
| Daily\_volume |
| Night\_volume |
| Counter\_info\_id |

Рисунок 2.8 – Структура бази даних

Розглянемо детальніше таблиці бази даних. У таблиці 2.3 наведено опис таблиці бази даних «Abonent».

Таблиця «Abonent» передбачена для зберігання інформації про абонента комунальних послуг. Присутні поля, що ідентифікують особу за прізвищем. Також присутнє поле, яке зберігає адресу проживання користувача. Для ідентифікації користувача передбачено поле для зберігання інформації про ідентифікаційний код. Поле «Personal\_number» призначене для зберігання інформації про персональний номер абонента комунальних послуг.

Таблиця 2.3 - Опис таблиці «Abonent»

|  |  |
| --- | --- |
| Назва поля | Опис |
| Id | Ідентифікатор |
| Surname | Прізвище абонента |
| Name | Ім’я абонента |
| Fathername | По батькові абонента |
| Address | Адреса проживання |
| Id\_code | Ідентифікаційний код абонента |
| Personal\_number | Особистий номер абонента комунальних послуг |

Опис таблиці бази даних «Payment» наведено у таблиці 2.4. Дана таблиця передбачена для зберігання інформації про оплату комунальних послуг. До складу полів входить інформація про ідентифікатор запису, період, за який відбувається оплата послуг та поле для відображення дати та часу надходження платежу.

Таблиця 2.4 – Опис таблиці «Payment»

|  |  |
| --- | --- |
| Назва поля | Опис |
| Id | Ідентифікатор |
| Period | Період, за який надходить оплата послуг |
| Created\_at | Дата надходження платежу |

Опис таблиці бази даних «Counter data» наведено у таблиці 2.5. Дана таблиця призначена для зберігання інформації про показники лічильника. До складу таблиці входять поля для відображення періоду за який подається показник, безпосередньо сам показник з лічильника електроенергії, кількість споживаної енергії вдень та вночі, ідентифікатор лічильника.

Таблиця 2.5 – Опис таблиці «Payment»

|  |  |
| --- | --- |
| Назва поля | Опис |
| Id | Ідентифікатор |
| Period | Період, за який надходить оплата послуг |
| Created\_at | Дата надходження платежу |

Опис таблиці бази даних «Counter info» наведено у таблиці 2.6.

Таблиця 2.6 – Опис таблиці «Counter info»

|  |  |
| --- | --- |
| Назва поля | Опис |
| Id | Ідентифікатор |
| title | Номер лічильника |
| Description | Додаткова інформація про лічильник. |

Таблиця «Counter info» відповідає за зберігання інформації про лічильник, який знаходиться у абонента.

Розробка різних типів баз даних лежить в основі функціональності, яку вони надають користувачам. Оскільки дані є динамічною сутністю, спосіб їх зберігання дуже різниться. Це також є причиною того, що компанії розробляють власні типи баз даних, які відповідають їхнім потребам. Існують такі види баз даних:

* ієрархічні бази даних;
* мережеві бази даних;
* об'єктно-орієнтовані бази даних;
* реляційні бази даних;
* бази даних NoSQL.

Як і в будь-якій ієрархії, база даних слідує послідовності даних, класифікованих за рангами або рівнями, де дані класифікуються на основі спільної точки зв’язку. У результаті дві сутності даних матимуть нижчий ранг, а спільність матиме вищий ранг.

Мережева база даних є ієрархічною базою даних, але з серйозними змінами. Дочірнім записам надається свобода зв’язування з кількома батьківськими записами. У результаті спостерігається мережа або мережа файлів бази даних, пов’язаних кількома потоками. Зверніть увагу, що кожен з елементів «Студент», «Викладач» і «Ресурси» має записи з двома батьками, які є відділами та клубами.

Ті, хто знайомий з парадигмою об'єктно-орієнтованого програмування, зможуть легко розібратися з цією моделлю баз даних. Інформацію, що зберігається в базі даних, можна представити як об’єкт, який відповідає як екземпляр моделі бази даних.

Вважаючись найбільш зрілими з усіх баз даних, ці бази даних ведуть у виробництві разом із своїми системами керування. У цій базі даних кожна частина інформації пов’язана з будь-якою іншою інформацією. Це завдяки тому, що кожне значення даних у базі даних має унікальну ідентифікацію у формі запису.

База даних NoSQL включає простоту дизайну, простіше горизонтальне масштабування до кластерів машин і точніший контроль над доступністю. Структури даних, які використовуються базами даних NoSQL, відрізняються від тих, що використовуються за замовчуванням у реляційних базах даних, що робить деякі операції швидшими в NoSQL. Придатність даної бази даних NoSQL залежить від проблеми, яку вона повинна вирішити. Структури даних, які використовуються базами даних NoSQL, іноді також вважаються більш гнучкими, ніж таблиці реляційних баз даних [https://www.geeksforgeeks.org/types-of-databases/].

Бази даних NoSQL створені для роботи з певними моделями даних. Вони мають гнучкі схеми, створені для роботи з сучасними програмами. Цей тип бази даних широко використовується, оскільки легко розробляти програми за допомогою NoSQL і забезпечує масштабованість. Їх можна використовувати з великою різноманітністю моделей даних, які можна використовувати для доступу та керування даними. Такі моделі включають документи, графік, ключ-значення, в пам’яті та пошук.

Особливості баз даних NoSQL

Вони забезпечують гнучкі схеми, щоб розробка була швидкою та повторюваною. Така гнучкість моделей NoSQL робить їх ідеальними для використання з напівструктурованими та неструктурованими даними. Бази даних NoSQL призначені для масштабування за допомогою розподілених кластерів апаратного забезпечення замість додавання надійних і дорогих серверів. Певні постачальники хмарних технологій пропонують можливість масштабування баз даних NoSQL як повністю керовану службу. Він оптимізований для обробки конкретних моделей даних і шаблонів доступу, щоб забезпечити вищу продуктивність, замість того, щоб намагатися виконати аналогічні функції з реляційними базами даних. Вони надають високофункціональні API і типи даних, створені спеціально для роботи з відповідними моделями даних. База даних NoSQL робить компроміси, змінюючи властивості ACID, які є важливими та строгими в реляційних базах даних, забезпечуючи тим самим набагато більш гнучку модель даних, яка має можливість горизонтального масштабування. Це причина використання NoSQL, де велика пропускна здатність і низька затримка є основними проблемами. Продуктивність залежить від розміру базового апаратного кластера та затримки мережі. Він змінюється залежно від цих факторів. Бази даних NoSQL можна легко розділити, оскільки шаблон ключ-значення може бути зручним вибором для масштабування програми за допомогою розподіленої архітектури. Така архітектура допомагає збільшити пропускну здатність програми, тим самим покращуючи послідовність продуктивності [https://www.knowledgehut.com/tutorials/aws/aws-non-relational-database].

NoSQL можна використовувати для надання об’єктних API, що дозволяє розробникам програм легко зберігати та отримувати дані зі структур даних у пам’яті. Ключі розділу дозволяють програмам шукати дані на основі пар ключ-значення, наборів стовпців або напівструктурованих документів, які зберігають серіалізовані об’єкти та атрибути програми.

Різні типи баз даних NoSQL розроблені для роботи з OLTP (онлайн-обробкою транзакцій) для кількох шаблонів доступу до даних. Це включає програми з низькою затримкою.

Бази даних NoSQL оптимізовано спеціально для роботи з програмами, які потребують великих обсягів даних, низької затримки та гнучких моделей даних. Цих можливостей можна досягти шляхом зміни деяких обмежень узгодженості даних, які зазвичай дуже суворо застосовуються до інших баз даних.

Приклад запиту до бази даних для отримання інформації про абонентів з обмеженням 100 записів на сторінку:

*SELECT Surname, Name, Fathername*

*FROM Abonent*

*WHERE address=’Ternopil’*

*LIMIT 100;*

Приклад SQL запиту для реалізації пошуку за певним критерієм:

*SELECT \**

*FROM Abonent*

*WHERE Surname*

*LIKE 'Петренко%';*

Приклад SQL запиту для реалізації сортування вибірки за певним критерієм:

*SELECT \**

*FROM Abonent*

*ORDER BY Surname*

*DESC;*

Приклад SQL запиту для додавання запису в таблицю бази даних:

*INSERT INTO*

*Abonent (*

*Surname,*

*Name,*

*FatherName,*

*Address,*

*Id\_code,*

*Personal\_number*

*)*

*VALUES (*

*'Петренко',*

*'Петро',*

*'Петрович',*

*'м. Тернопіль, вул. …',*

*'11111111',*

*'1234567'*

*);*

Для оновлення запису в базі даних використано такий SQL – запит.

*UPDATE Abonent*

*SET*

*Fathername = ‘Іваненко',*

*Address= ‘м. Тернопіль, вул. …Нова’*

*WHERE*

*ID = 111;*

SQL – запит для видалення запису з таблиці бази даних:

*DELETE*

*FROM Customers*

*WHERE*

*Surname='Петренко';*

Оператор UNION використовується для об’єднання даних із результатів двох або більше запитів команди SELECT в один окремий набір результатів. Цей оператор видаляє будь-які дублікати в об’єднаних результатах.

Кількість стовпців, що витягуються кожною командою SELECT у межах UNION, має бути однаковою. Стовпці в одній позиції в кожному операторі SELECT повинні мати подібні типи даних. Наприклад, «char» і «varchar» є ідентичними типами даних. Стовпці мають бути в правильному порядку в операторах SELECT.

Псевдоніми SQL – це тимчасові імена, надані таблицям або стовпцям. Ці псевдоніми існують лише протягом запиту, у якому вони використовуються. Для створення псевдонімів ми використовуємо оператор «AS».

Оператор UNION не допускає дублікатів. Ми можемо використовувати оператор UNION ALL, якщо хочемо, щоб дублікати були присутні в комбінації двох або більше операторів SELECT.

## 2.4 Висновки до розділу

У даному розділі наведено опис узагальненого алгоритму роботи системи обліку комунальних послуг, що включає в себе елементи машинного навчання для розпізнавання цифрових символів із лічильника електроенергії та елементи серверної обробки даних та зберігання інформації у базу даних.

На основі експериментального підходу запропоновано структуру згорткової нейронної мережі для виділення інформативних областей із зображення та виділення цифрових символів.

На основі теорії проектування баз даних розроблено даталогічну модель бази даних для системи обліку комунальних послуг, що включає в себе таблиці для зберігання інформації про абонентів та показників лічильника.

# 3 Програмна реалізація СИСТЕМИ ОБЛІКУ КОМУНАЛЬНИХ ПОСЛУГ

## 3.1 Структура програмного модулю

Розробка системи обліку комунальних послуг реалізовано з використанням мови програмування PHP з використанням фреймворку Laravel. Laravel — це потужний фреймворк MVC PHP, спроектований для розробників, яким потрібен простий і елегантний інструментарій для створення повнофункціональних веб-програм.

Laravel пропонує наступні переваги, коли розробляється веб-додаток на його основі:

- завдяки фреймворку Laravel веб-додаток стає більш масштабованим;

- значна економія часу при розробці веб-програми, оскільки Laravel повторно використовує компоненти з інших фреймворків у розробці веб-програми.

- фреймворк включає простори імен та інтерфейси, таким чином, допомагає організовувати ресурси та керувати ними.

Особливості Laravel

Laravel пропонує наступні ключові функції, що робить його ідеальним вибором для розробки веб-додатків:

* модульність - Laravel надає 20 вбудованих бібліотек і модулів, які допомагають покращити додаток. Кожен модуль інтегровано з менеджером залежностей Composer, що спрощує оновлення.
* перевірка - Laravel містить функції та помічники, які допомагають у тестуванні за допомогою різноманітних тестів. Ця функція допомагає підтримувати код відповідно до вимог.
* маршрутизація - Laravel забезпечує гнучкий підхід до користувача для визначення маршрутів у веб-додатку. Маршрутизація допомагає краще масштабувати програму та підвищує її продуктивність.
* управління конфігурацією - Веб-програма, розроблена в Laravel, працюватиме в різних середовищах, а це означає, що її конфігурація постійно змінюватиметься. Laravel забезпечує послідовний підхід до ефективної обробки конфігурації.

UML – діаграму класів розробленої системи наведено на рисунку 3.1.

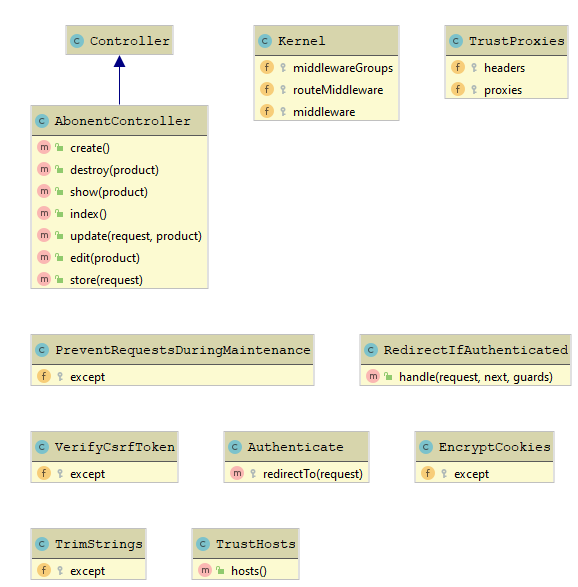


Рисунок 3.1 – UML – діаграму класів розробленої системи

Встановлення значень конфігурації у конфігураційних файлах може бути дуже корисним під час спроби застосувати параметри в програмі, наприклад, коли ви хочете встановити часовий пояс у вашому основному конфігураційному файлі.

Метод config дозволяє програмі отримувати або встановлювати значення в усіх файлах, які знаходяться в каталозі config.

Щоб встановити значення з файлів у каталозі config, ми будемо використовувати метод config().

У методі config() ми використовуємо крапкову нотацію, щоб отримати потрібне значення. Перший параметр, який ми використовуємо, — це ім’я файлу без .php, а потім ми переходимо вниз, ніби намагаємось отримати доступ до значень із масиву, доки не отримаємо конкретне значення, яке шукаємо. Другим параметром, який ми передаємо в метод config(), є нове значення.

Метод config() отримує файл і масив/ключ, до якого потрібно отримати доступ, у форматі крапкової нотації, як показано в прикладі як перший параметр, а нове значення як другий параметр.

Маршрутизація в Laravel дозволяє користувачам направляти всі вимоги додатків до відповідного контролера. Більшість первинних маршрутів у Laravel визнають і приймають уніфікований ідентифікатор ресурсу разом із закриттям, що забезпечує простий і виразний спосіб маршрутизації.

Маршрут — це спосіб створення URL-адреси запиту для вашої програми. Ці URL-адреси не обов’язково зіставляються з певними файлами на веб-сайті, і вони одночасно читаються людиною та зручні для пошукових систем.

У Laravel маршрути створюються в папці routes. Вони створюються у файлі web.php для веб-сайтів. А для API вони створюються всередині api.php.

Ці маршрути призначаються групі мережевого проміжного програмного забезпечення, висвітлюючи стан сеансу та безпеку CSRF. Маршрути в routes/api.php не мають стану та їм призначено групу проміжного програмного забезпечення API.

Стандартна інсталяція Laravel включає два маршрути: один для Інтернету, а інший для API.

Усі маршрути Laravel визначені у файлах маршрутів, які знаходяться в каталозі routes. App\Providers\RouteServiceProvider програми автоматично збирає ці файли. А файл routes/web.php визначає маршрути для вашого веб-інтерфейсу.

Існує можливість визначити маршрут до цієї дії контролера, як:

*Route::get(‘user/{id}’, ‘UserController@show’);*

Route::resource: метод Route::resource може бути контролером Restful, який створює всі основні маршрути, необхідні для програми, і обробляється через клас контролера.

Коли запит відповідає вказаному URI маршруту, викликається метод show в уроці App\Http\ControllersUserController, передаючи параметри маршруту методу.

Що стосується ресурсів, вам потрібно зробити дві речі в додатку Laravel. По-перше, ви повинні створити маршрут ресурсу на Laravel, який забезпечує вставку, оновлення, перегляд і видалення маршрутів. По-друге, створіть контролер ресурсів, який надає метод для вставки, оновлення, перегляду та видалення.

Клас «AbonentController» має методи, що відповідають за відображення списку абонентів. У класі «AbonentController» передбачено методи для відображення CRUD.

Усі маршрути Laravel визначені у ваших файлах маршрутів, які знаходяться в каталозі routes. AppProvidersRouteServiceProvider вашої програми, відповідно, накопичує ці записи. Запис routes/web.php визначає маршрути для вашого веб-інтерфейсу.

Структура маршруту дуже проста. Відкрийте відповідний файл (або `web.php`, або `api.php`) і почніть код з `Route::`. Після цього йде запит, який ви хочете призначити цьому конкретному маршруту, а потім з’являється функція, яка буде виконується в результаті запиту.

Laravel пропонує наступні методи маршрутизації:

- отримати;

- пост;

- поставити;

- видалити;

- патч;

- параметри.

Маршрути визначаються в Laravel у класі Route за допомогою дієслова HTTP, маршруту для відповіді та закриття або стратегії контролера.

У Laravel можна визначити метод Controller для маршруту. Метод контролера виконує всі визначені дії кожного разу, коли користувач приходить на маршрут.

Наступний фрагмент коду, демонструє як призначити метод контролера для маршруту:

*Route::get('/home', 'YourController@functionname');*

Код починається з `Route::`, а потім визначає метод запиту для маршруту. Потім визначте маршрут і контролер разом із методом, додавши символ @ перед назвою методу.

У Laravel ви можете визначити назву свого маршруту. Ця назва часто дуже корисна в різних сценаріях. Наприклад, якщо ви хочете перенаправити користувача з одного маршруту на інший, вам не потрібно визначати повну URL-адресу перенаправлення. Замість цього ви можете просто вказати його назву, і код працюватиме! Ви можете визначити назву маршруту за допомогою методу `name` в екземплярі маршруту.

Використовуючи атрибути PHP, ви можете маніпулювати виявленими маршрутами: ви можете встановити назву маршруту, додати проміжне програмне забезпечення.

Виявлення маршрутів під час кожного запиту програми може мати невеликий вплив на продуктивність. Для підвищення продуктивності необхідно кешувати ваші маршрути як частину процесу розгортання.

Структуру директорій проекту наведено на рисунку 3.2.

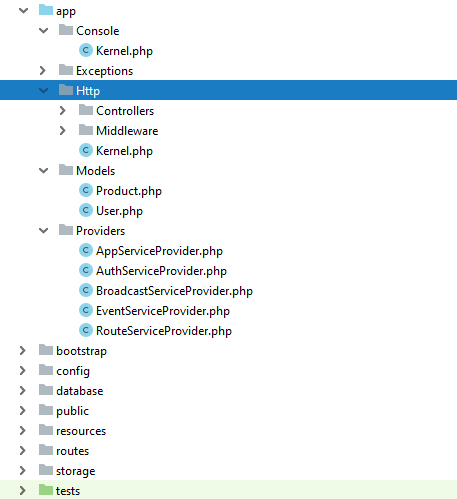


Рисунок 3.2 - Структура директорій проекту

Замість того, щоб визначати всю логіку обробки запитів як замикання у файлах маршрутів, ви можете організувати цю поведінку за допомогою класів «контролера». Контролери можуть групувати пов’язану логіку обробки запитів в один клас. Наприклад, клас UserController може обробляти всі вхідні запити, пов’язані з користувачами, включаючи показ, створення, оновлення та видалення користувачів. За замовчуванням контролери зберігаються в каталозі app/Http/Controllers.

Якщо дія контролера особливо складна, ви можете знайти зручним присвятити цілий клас контролера цій одній дії.

Метод для відображення списку абонентів наведено нижче:

*/\*\**

*\* Display a listing of the abonents.*

*\**

*\* @return \Illuminate\Http\Response*

*\*/*

*public function index()*

*{*

*$abonent = Abonent::latest()->paginate(10);*

*return view('abonent.index',compact('abonent'))*

*->with('i', (request()->input('page', 1) - 1) \* 10);*

*}*

Проміжне програмне забезпечення забезпечує зручний механізм для перевірки та фільтрації HTTP-запитів, що надходять у програму. Наприклад, Laravel містить проміжне програмне забезпечення, яке перевіряє автентифікацію користувача програми. Якщо користувач не автентифікований, проміжне програмне забезпечення перенаправить користувача на екран входу до програми. Однак, якщо користувач автентифікований, проміжне програмне забезпечення дозволить запиту перейти далі в програму.

Структуру директорії «routes» наведено на рисунку 3.3.

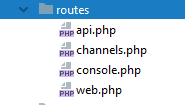


Рисунок 3.3 - Структуру директорії «routes»

Для реалізації доступу до сервера та бази даних реалізовано RESTful API інтерфейс. Приклад частини коду для реалізації доступу до сервера та додавання до бази даних.

Структуру директорії на хостингу наведено на рисунку 3.4. Розміщення на хостингу необхідне для тестування доступу на сервері із публічною IP – адресою. Такий підхід дозволить протестувати роботи сервісу в реальних умовах та визначити можливі проблеми. Помилки чи виключні ситуації.

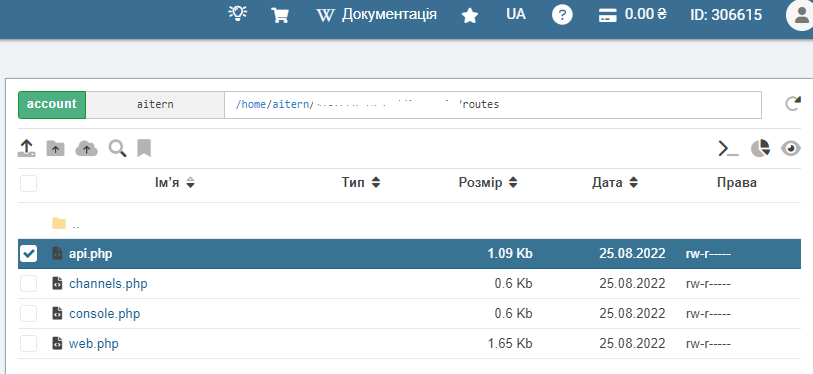


Рисунок 3.4 - Структуру директорії на хостингу

Лістинг файлу api.php, та функції для виділення прав доступу до ресурсу:

*Route::group([*

*'middleware' => 'api',*

*'prefix' => 'auth'*

*], function ($router) {*

*Route::post('login', 'App\Http\Controllers\AuthController@login');*

*Route::post('logout', 'App\Http\Controllers\AuthController@logout');*

*Route::post('refresh', 'App\Http\Controllers\AuthController@refresh');*

*Route::post('me', 'App\Http\Controllers\AuthController@me');*

*});*

У вищенаведеному прикладі зображено перелік шляхів ,якими можна програмно надсилати дані на сервер від клієнта. Для забезпечення конфіденційності переданих даних використовується токен.

Лістинг файлу «VerifyCsrfToken.php» наведено нижче:

*class VerifyCsrfToken extends Middleware*

*{*

*/\*\**

*\* The URIs that should be excluded from CSRF verification.*

*\**

*\* @var array<int, string>*

*\*/*

*protected $except = [*

*//*

*'posts/store',*

*'posts/show',*

*'posts/\*'*

*];*

*}*

Приклад запиту до сервера та бази даних шляхом використання технології REST API та інстурменту Postman наведено на рисунку 3.5.

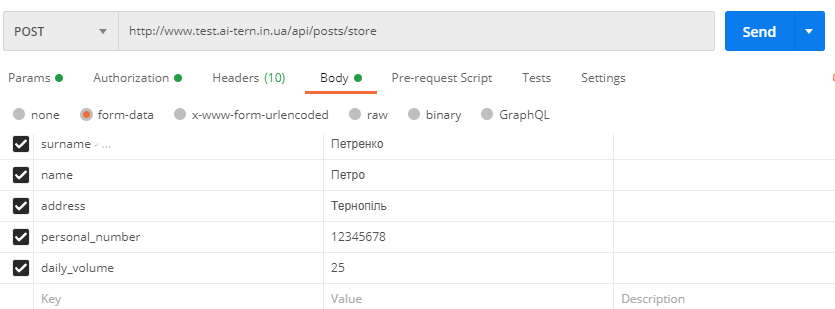


Рисунок 3.5 - Приклад запиту до сервера та бази даних шляхом використання технології REST API

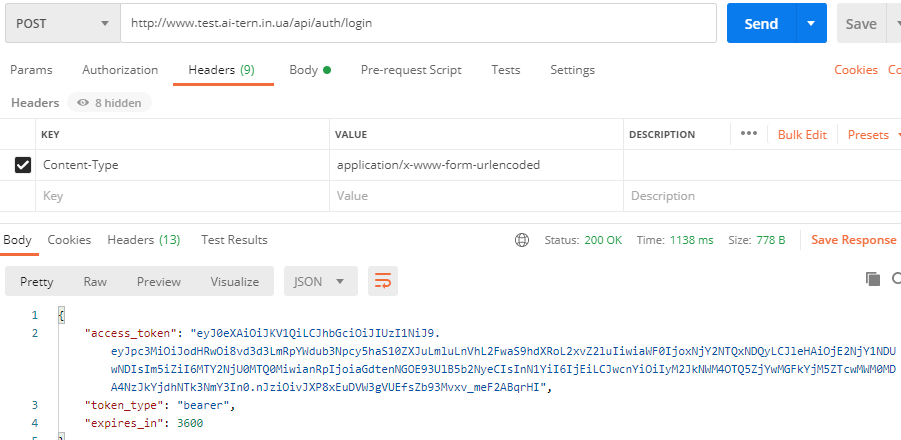
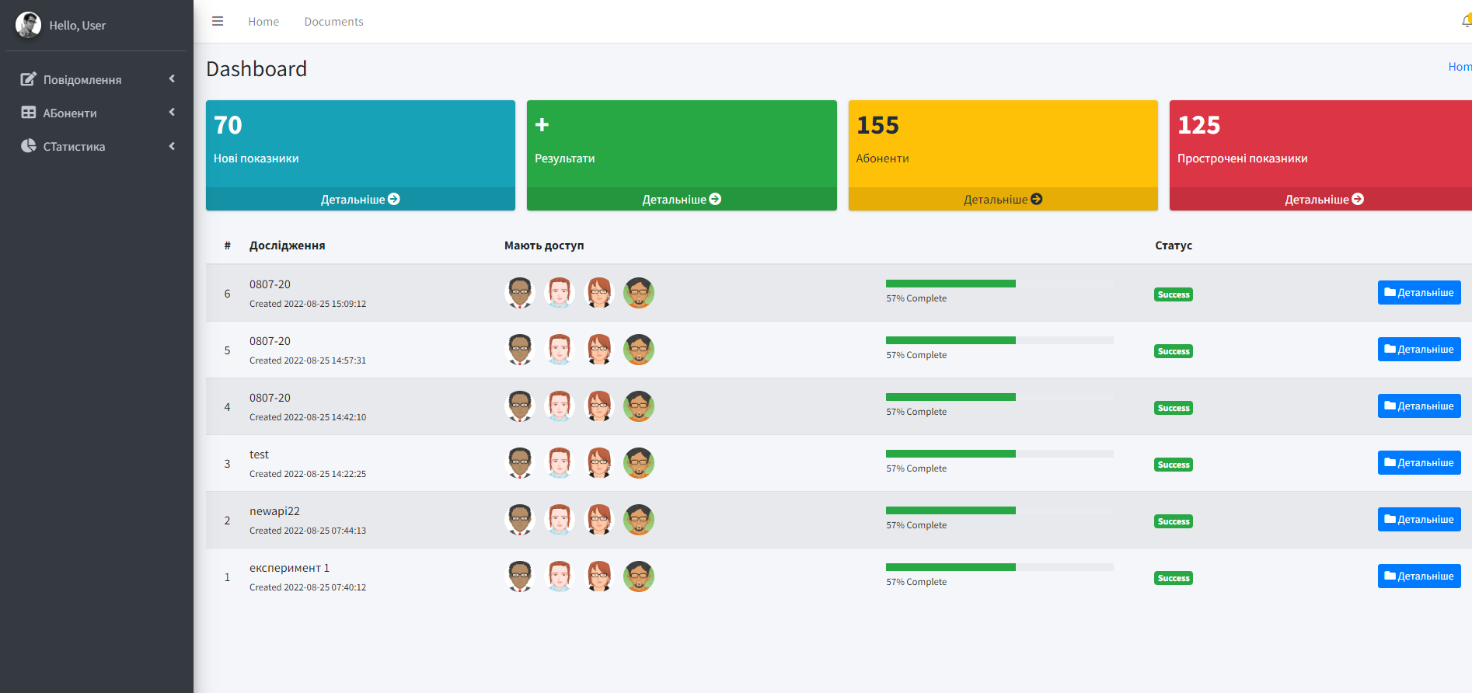
Приклад генерування токена з допомогою API наведено на рисунку 3.6.

Рисунок 3.6 - Приклад генерування токена

В результаті виконання запиту до бази даних отримується access\_token, який в подальшому можна використовувати для відправки даних.

Графічний інтерфейс розробленої системи наведено на рисунку 3.7.

Рисунок 3.7 - Графічний інтерфейс розробленої системи

Інтерфейс складається з бокового меню, де розміщено корисні посилання на необхідні сторінки. У верхній частині головного блоку сторінки знаходяться міні-блоки для зручного відображення інформації про:

- нові показники;

- кількість абонентів, що не подали показників;

- загальна статистика.

У головній частині вікна знаходиться інформація про список абонентів, що подали показники. Список формується у такому порядку, що найновіші показник відображається у самому верху списку. Для зручності інтерфейсу передбачено можливість перегляду інформації про осіб, що можуть переглядати подані користувачем показники, це можуть бути члени родини.

## 3.2 Модуль розпізнавання цифрових символів із лічильника

Розпізнавання символів на лічильниках відбувається за допомогою мобільного додатка, в якому реалізовано модуль перетворення цільового зображення лічильника у дані для передачі у систему. Даний модуль реалізований на основі бібліотеки OpenCV.

OpenCV — це потужне і зручне середовище для реалізації безлічі проектів в області обробки зображень. У цьому посібнику представлені деякі аспекти OpenCV, засновані на практичному застосуванні - зчитуванні показань лічильника електроенергії.

У багатьох будинках все ще існують Лічильники електроенергії з механічним лічильником, які не забезпечують прямого стандартизованого інтерфейсу для зчитування споживання електроенергії за допомогою комп'ютера. Один із способів отримати доступ до цих даних в будь - якому випадку полягає в оптичному виявленні лічильника за допомогою відеокамери і подальшому розпізнаванні символів (OCR — Optical Character Recognition).

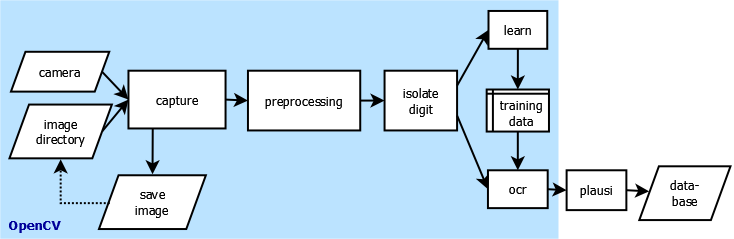


Рисунок 3.8 — Схема розпізнавання даних на лічильнопу пристрої

На рис. 1 показано основe схему по розпізнаванню. Спочатку зображення лічильника електроенергії записується відеокамерою через певні проміжки часу. Після створення зображення програма може альтернативно зберегти зображення у файловій системі, щоб пізніше прочитати його звідти. Це дуже корисно для розробки та тестування, оскільки не завжди потрібно підключати камеру.

Потім зображення проходить попередню обробку. Цей крок повинен змінити і оптимізувати зображення таким чином, щоб на наступному кроці стало можливим виявлення і виділення окремих цифр лічильника.

Потім витягнуті зображення цифр надходять до системи розпізнавання символів (OCR). Однак спочатку він повинен бути навчений набору всіх можливих символів (цифр від 0 до 9) в інтерактивному режимі. Це створює набір навчальних даних, які зберігаються у файлі. Під час нормальної роботи розпізнавач завантажує ці навчальні дані і, таким чином, може класифікувати невідомий символ.

Ця класифікація виконана з певною помилкою. Тому перед подальшою обробкою має сенс перевірити показання на достовірність. Якщо воно передано, то виявлене значення лічильника разом з поточним часом зберігається в базі даних. За допомогою даних, що зберігаються тут, можна проводити більш пізні оцінки, такі як генерація графіки з урахуванням годинного, щоденного та щотижневого енергоспоживання. Перевірка правдоподібності та база даних не реалізовані в OpenCV, але також згадуються в цьому посібнику для повноти.

Стосовно процесу розвізнавання. Камера захоплює зображення лічильника електроенергії (рисунок 3.9). У OpenCV для цього потрібні лише два рядки вихідного коду. Таким же простим є читання зображення з файлу. Оскільки програма призначена для роботи з обома методами введення, тут корисний об'єктно-орієнтований підхід. Давайте визначимо базовий клас Image Input, який зберігає зображення та позначку часу отримання зображення.



Рисунок 3.9 — Зображення, отримане камерою

Перш ніж ми зможемо розпочати розпізнавання символів лічильника, алгоритм повинен ідентифікувати та витягти окремі цифри лічильника. Давайте подивимося на зображення лічильника електроенергії, зняте камерою (рис. 3.9): те, з чим з легкістю справляється наш тренований все життя мозок, є величезною проблемою для "некваліфікованого" комп'ютера. Який критерій відрізняє символ " 0 "механічного лічильника від" 0 " у специфікації частоти 50 Гц?

Алгоритм вилучення відповідної інформації із зображення, який ми хочемо розробити, наведено нижче.

Клас ImageProcessor інкапсулює всі методи, необхідні для цього. Лістинг наведено нижче:

*class ImageProcessor {*

*public:*

*void setInput(cv::Mat & img);*

*void process();*

*const std::vector<cv::Mat> & getOutput();*

*private:*

*cv::Mat \_img;*

*cv::Mat \_imgGray;*

*std::vector<cv::Mat> \_digits;*

*};*

За допомогою setInput() ви передаєте зображення для обробки. Функція process() виконує повну обробку. При успішному виконанні getOutput() видає результат. Він складається з вектора зображень. Кожне зображення містить одну цифру лічильника. Якщо алгоритм пройшов успішно, то довжина вектора повинна бути рівно сім.

Функція process() делегує окремі етапи обробки іншим приватним функціям.

Наше око не засноване на зчитуванні рахунку (наприклад, рис. 3.9) за абсолютними значеннями яскравості або кольору. Гангліозні клітини сітківки влаштовані таким чином, що реагують на високий контраст. Таким чином, мозок здатний дуже швидко розпізнавати краї і лінії і, нарешті, визначати зовнішню форму лічильника і окремі цифри в широкому діапазоні яскравості.

Отож. велика частина OpenCV присвячена ідентифікації ребер і ліній. Корисною процедурою для багатьох ситуацій є алгоритм Canny. Функція Canny () приймає зображення у відтінках сірого як вхідні дані та видає зображення з виявленими краями як вихідні дані, подібні до рисунку 3.10:

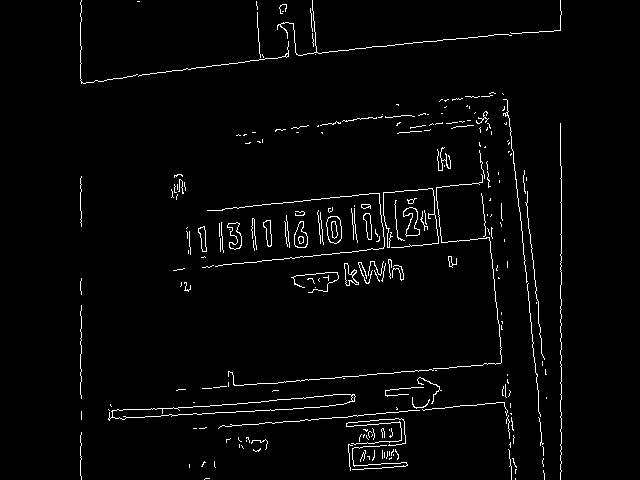


Рисунок 3.10 — Алгоритм Канні виявляє контури

Лістинг наведенго нижче:

*cv::Mat ImageProcessor::cannyEdges() {*

*cv::Mat edges;*

*// detect edges*

*cv::Canny(\_imgGray, edges, \_config.getCannyThreshold1(), \_config.getCannyThreshold2());*

*return edges;*

*}*

Зображення тепер містить ще деякі перешкоди на додаток до використовуваних цифр. Для ідентифікації останнього ми оцінюємо на другому кроці положення та висоти обчислених обмежувальних рамок. Алгоритм намагається знайти найбільшу кількість контурів однакового розміру на горизонтальній лінії з усіх можливих комбінацій обмежувальних рамок. Результуючі поля, вирівняні за вектором, містять, швидше за все, обмежувальні рамки цифр, тому що жодна інша група контурів не вирівняна так значно.

Результат повинен містити цифри в їх розташуванні зліва направо, тому обмежувальні рамки сортуються відповідно до їх положення x.

За допомогою цієї інформації тепер ви можете вирізати окремі цифри із зображення. Оператор () CV:: Mat відповідає за це, він отримує "Регіон інтересу" (ROI) як параметр. Необов'язково cv:: rectangle () малює ROI у вигляді зелених прямокутників на вихідному зображенні (рисунок 3.11). Це дуже корисно при налаштуванні камери та джерела світла.



Рисунок 3.11 — Виявлені обмежувальні рамки цифр наносяться на вихідне зображення

Кожен компонент vector\_digits містить зображення однієї з цифр (рисунок 3.12).

A gallery of exemplary images of digits

Рисунок 3.12 — Контури розпізнаваних цифр даного лічильного пристрою

Комп'ютер витягує інформацію про те, який символ представлений зображенням, за допомогою оптичного розпізнавання символів (OCR). Одним із методів, який часто використовується для цього, є Машинне навчання. На першому етапі системи навчаються з використанням різних тестових даних.

Результатом є модель, яка описує перетворення даних (зображень) в інформацію (кодування символів). Використовуючи цю модель, навчена система може перетворити невідомі дані в потрібну інформацію на другому кроці.

Існує багато алгоритмів машинного навчання, з яких OpenCV значною мірою реалізує. Вибір правильного алгоритму для конкретного завдання вимагає великого досвіду і знань. Надалі використовується один з найпростіших алгоритмів: K-найближчі сусіди. Відомо, що він дуже точний, але, з іншого боку, він споживає багато процесорного часу та пам'яті. Ці недоліки не такі критичні для нашого застосування - часу достатньо, тому що лічильник обертається не дуже швидко.

## 3.3 Порівняльний аналіз експериментів

Швидкість роботи сайту є важливим фактором, який впливає на популярність та довіру користувачів до ресурсу. Тому, розробники сайтів приділяють велику увагу до показників швидкодії з допомогою інструмента Lighthouse.

Швидкодія сайту активно впливає як на відношення користувачів до даного ресурсу, так і на показники SEO оптимізації. Тобто, чим швидше сайт працює, тим вище у пошуковику сайт буде знаходитись.

Наступні значки використовуються для позначення різних типів вмісту.

* HTML HTML документ;
* Файл JavaScript JavaScript;
* Файл CSS CSS;
* Зображення Файл зображення;
* Text/Plain Звичайний текстовий документ;
* Інше Будь-який інший невідомий тип вмісту.

Щоб було легше розрізняти коди відповідей HTTP на діаграмі, ми додали кольорові крапки біля кожної URL-адреси.

* 2xx Сервер відповів успішним кодом;
* 3xx Запит було перенаправлено до іншої цілі;
* 4xx Сталася помилка клієнта, наприклад, сторінка 404 не знайдена;
* 5xx Сталася помилка сервера, наприклад 500 внутрішня помилка сервера.

Показники ефективності сайту з використанням інструменту Lighthouse наведено на рисунку 3.13.

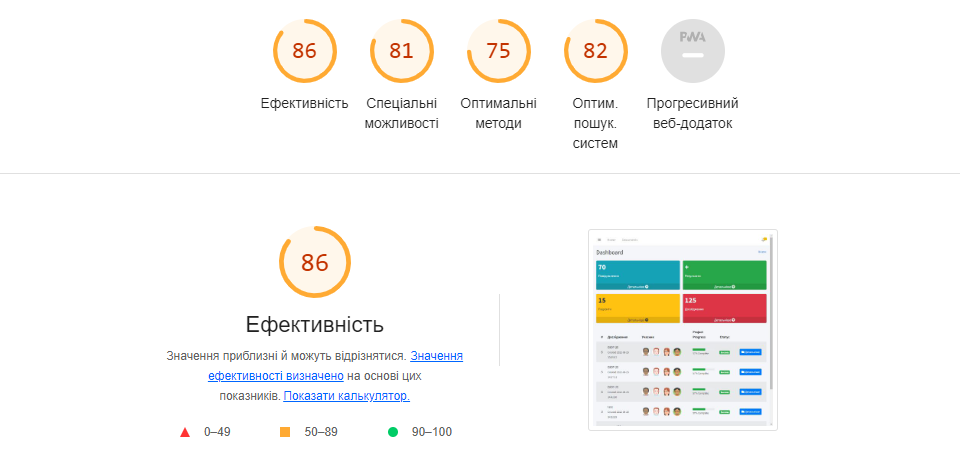


Рисунок 3.13 – Показники ефективності сайту з використанням інструменту Lighthouse

Показники роботи розробленої системи на основі «Observer Metrics» наведено на рисунку 3.14.

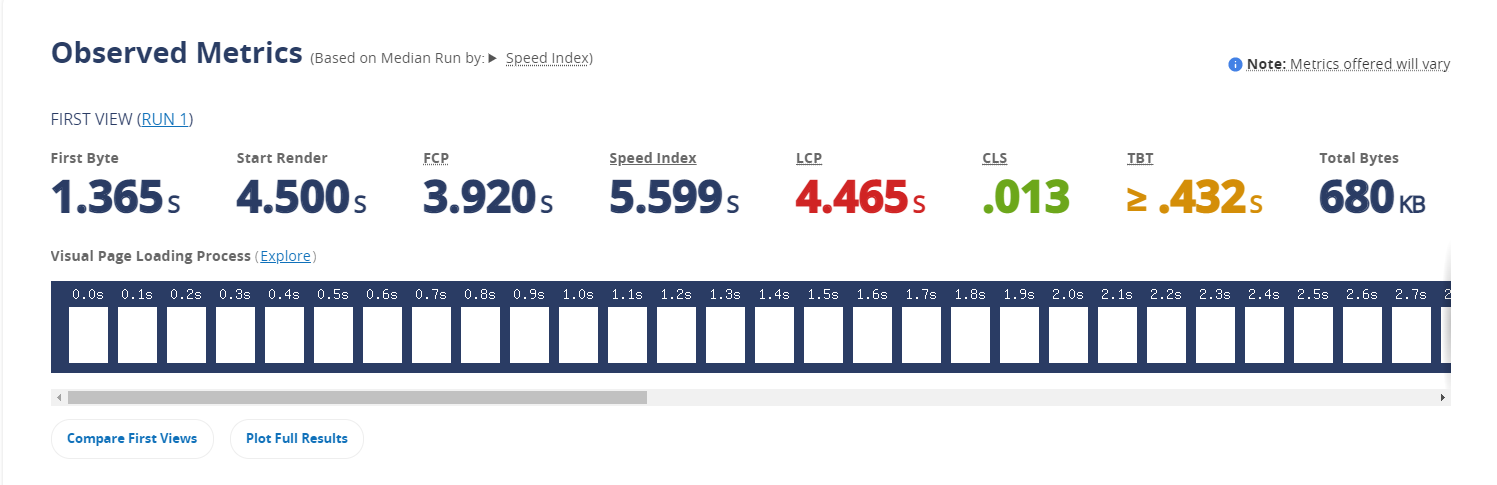


Рисунок 3.14 – Показники роботи розробленої системи на основі «Observer Metrics»

PageSpeed Insights (PSI) звітує про взаємодію користувача зі сторінкою як на мобільних, так і на настільних пристроях, а також надає пропозиції щодо того, як цю сторінку можна покращити.

PSI надає як лабораторні, так і польові дані про сторінку. Лабораторні дані корисні для вирішення проблем, оскільки вони збираються в контрольованому середовищі. Однак він може не охопити реальні вузькі місця. Польові дані корисні для фіксації реального досвіду користувача, але мають більш обмежений набір показників.

Дані реального досвіду користувачів у PSI базуються на наборі даних звіту про взаємодію з користувачем Chrome (CrUX). PSI звітує про досвід реальних користувачів щодо першого фарбування вмісту (FCP), затримки першого введення (FID), найбільшого фарбування вмісту (LCP) і сукупного зміщення макета (CLS) за попередній 28-денний період збору. PSI також повідомляє про досвід експериментальних показників Interaction to Next Paint (INP) і Time to First Byte (TTFB).

Основні веб-показники – це загальний набір сигналів продуктивності, критичних для роботи в Інтернеті. Основними показниками Web Vitals є FID, LCP і CLS, і їх можна агрегувати на рівні сторінки або джерела. Для агрегатів із достатньою кількістю даних за всіма трьома показниками, агрегат проходить оцінку основних веб-показників, якщо 75-й процентиль усіх трьох показників є добрим. В іншому випадку агрегація не проходить оцінку. Якщо агрегація містить недостатньо даних для FID, тоді вона пройде оцінку, якщо 75-й процентиль LCP і CLS є добрими. Якщо в LCP або CLS недостатньо даних, неможливо оцінити агрегацію на рівні сторінки або джерела.

Показники роботи розробленої системи на основі «Waterfall View» наведено на рисунку 3.15.

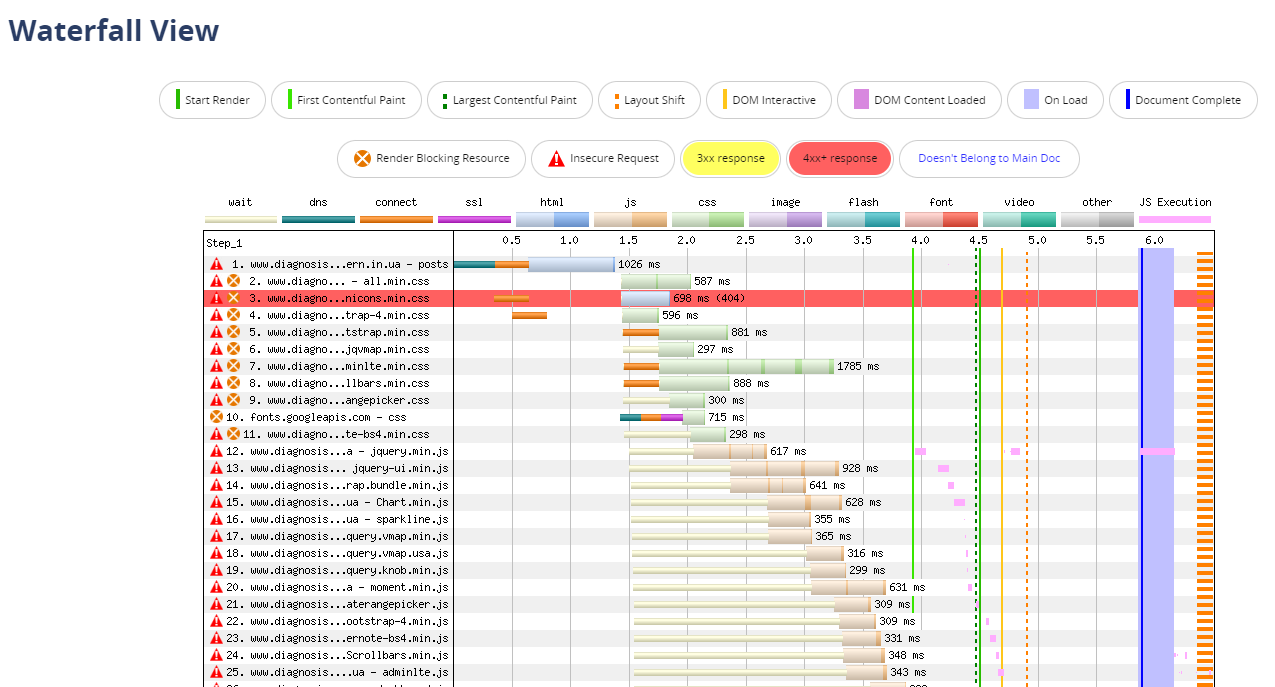


Рисунок 3.15 – Показники роботи розробленої системи на основі «Waterfall View»

Аналізуючи вищенаведені показники, можна виділити наступні показники:

* First Byte = 1.3 секунди;
* Start render = 4.5 секунди;
* FCP = 3.9 секунди;
* Total bytes = 680 Kb;
* ефективність = 86 %;
* оптимізація пошуковими системами = 82%.

PSI представляє розподіл цих показників, щоб розробники могли зрозуміти діапазон досвіду для цієї сторінки чи джерела. Цей розподіл поділено на три категорії: добре, потребує покращення та погано, які представлені зеленими, бурштиновими та червоними смугами. Наприклад, 11% у жовтій смужці LCP означає, що 11% усіх спостережуваних значень LCP знаходяться між 2500 мс і 4000 мс.

## 3.4 Висновки до розділу

Представлено структуру програмного модулю системи контролю комунальних послуг. Визначено основні класи в системі та представлено зв’язки між ними у вигляді UML діаграми. Додатково вказано методи доступу у систему.

Визначено основні етапи функціонування алгоритму. Вказано реалізацію кожного з етапів розпізнавання зображення з застосуванням відкритого середовища OpenCV. В результаті виявлено великі навантаження на певних етапах алгоритму, які нівелюються, завдяки тому, що лічильний пристрій функціонує значно повільніше, ніж алгоритм розпізнавання.

Результати досліджень показали, що показник First Byte = 1.3 секунди, Start render = 4.5 секунди, FCP = 3.9 секунди, Total bytes = 680 Kb, ефективність = 86 %. Дані показники у порівнянні з аналогами є вищими, що впливає на показник ефективності – 86%.

# Висновки

На основі аналітичного підходу проведено порівняльний аналіз систем обліку абонентів комунальних послуг, що дозволило виділити список обов'язкових пунктів, необхідних для функціонування системи. Додатково виділено переваги та недоліки існуючих систем.

Проведено аналіз сучасних алгоритмів машинного навчання дня класифікації, кластеризації даних, що дозволяє виділити їх переваги та недоліки, що дозволить вибрати найбільш необхідний алгоритм для вирішення поствленої задачі.

Проаналізовано сучасні підходи до розробки розпізнавання образів, зокрема цифрових символів на зображенні, що дозволить розробити архітектуру згорткової нейронної мережі для розпізнавання цифрових символів.

Наведено опис узагальненого алгоритму роботи системи обліку комунальних послуг, що включає в себе елементи машинного навчання для розпізнавання цифрових символів із лічильника електроенергії та елементи серверної обробки даних та зберігання інформації у базу даних.

На основі експериментального підходу запропоновано структуру згорткової нейронної мережі для виділення інформативних областей із зображення та виділення цифрових символів.

На основі теорії проектування баз даних розроблено даталогічну модель бази даних для системи обліку комунальних послуг, що включає в себе таблиці для зберігання інформації про абонентів та показників лічильника.

Представлено структуру програмного модулю системи контролю комунальних послуг. Визначено основні класи в системі та представлено зв’язки між ними у вигляді UML діаграми. Додатково вказано методи доступу у систему.

Результати досліджень показали, що показник First Byte = 1.3 секунди, Start render = 4.5 секунди, FCP = 3.9 секунди, Total bytes = 680 Kb, ефективність = 86 %. Дані показники у порівнянні з аналогами є вищими, що впливає на показник ефективності – 86%.

# Список використаних джерел

1. Арбузов С. Г. Розрахунок // Банківська енциклопедія / С. Г. Арбузов, Ю. В. Колобов, В. І. Міщенко, С. В. Науменкова. — Київ : Центр наукових досліджень Національного банку України : Знання, 2011. — 504 с. — (Інституційні засади розвитку банківської системи України).
2. Шемшученко Ю. С. Розрахунки // Юридична енциклопедія : [у 6 т.] / ред. кол.: Ю. С. Шемшученко (відп. ред.) [та ін.]. — К. : Українська енциклопедія ім. М. П. Бажана, 2003. — Т. 5 : П — С. — 736 с.
3. Шемшученко Ю. С. Житлово-комунальні послуги [Архівовано 9 серпня 2016 у Wayback Machine.] // Юридична енциклопедія : [у 6 т.] / ред. кол.: Ю. С. Шемшученко (відп. ред.) [та ін.]. — К. : Українська енциклопедія ім. М. П. Бажана, 1998. — Т. 2 : Д — Й. — 744 с. — ISBN 966-7492-00-8.
4. Воронова Л. К. Субсидія //Словник фінансово-правових термінів / за заг. ред. д. ю.н., проф. Л. К. Воронової. – 2-е вид., переробл. і доповн. – К.: Алерта, 2011– 558 с.
5. Шемшученко Ю. С. Житлова субсидія // Юридична енциклопедія : [у 6 т.] / ред. кол.: Ю. С. Шемшученко (відп. ред.) [та ін.]. — К. : Українська енциклопедія ім. М. П. Бажана, 1998. — Т. 2 : Д — Й. — 744 с.
6. Бережнюк І. Г. Умови платежів // Митна енциклопедія : у 2 т. / І. Г. Бережнюк (відп. ред.) та ін.. — Хм. : ПП Мельник А. А., 2013. — Т. 2 : М — Я. — С. 425. — 536 с.
7. Центр комунальних сервісів [Електронний ресурс] – Режим доступу: <https://cks.com.ua/cabinet/> (19.03.2022)
8. Оплата комунальних послуг [Електронний ресурс] – Режим доступу: <https://mereja.km.ua/> (10.09.2022)
9. Тарифи Луцької міської територіальної громади станом на 01.11.2022 [Електронний ресурс] – Режим доступу: https://www.lutskrada.gov.ua/pages/lutsk-tariffs(01.11.2022)
10. Березький О.М. Методи, алгоритми та програмні засоби опрацювання біомедичних зображень: монографія / [ О.М. Березький, Ю.М. Батько, К.М, Березька, С.О. Вербовий, Т.В. Дацко, Л.О. Дубчак, І.В. Ігнатєв, Г.М. Мельник, В.Д. Николюк, О.Й. Піцун ]; під наук. ред. Березький О.М., Тернопіль . ТНЕУ «Економічна думка», 2017. - 330 с.
11. Berezsky O.M. Evaluation methods of image segmentation quality / O.M Berezsky, O.Y. Pitsun // Радіоелектроніка, інформатика, управління.- 2018.- №1. – С. 41-61
12. Berezsky O. Automated Processing of Cytological and Histological Images / O. Berezsky, O. Pitsun // Proceedings of XII International Conference Perspective Technologies and methods in mems design (MEMSTECH 2016) 20-24 April, 2016, Lviv-Polyana, Ukraine, pp. 51-53
13. Jain A. K. Data clustering: a review / A. K. Jain , M. N. Murty , P. J. Flynn // ACM Computing Surveys, Volume 31 Issue 3, Sept. 1999 pp 264–323https://doi.org/10.1145/331499.331504
14. DEFINITION logistic regression [Електронний ресурс] – Режим доступу: <https://www.techtarget.com/searchbusinessanalytics/definition/logistic-regression> (19.03.2022)
15. LINEAR REGRESSION A Visual Introduction To (Almost) Everything You Should Know [Електронний ресурс] – Режим доступу: <https://www.techtarget.com/searchbusinessanalytics/definition/logistic-regression> (28.10.2022)
16. Березький О. М. ЗАСТОСУВАННЯ МЕТОДУ ЛІНІЙНОЇ РЕГРЕСІЇ ДЛЯ АНАЛІЗУ КІЛЬКІСНИХ ХАРАКТЕРИСТИКИ ЦИТОЛОГІЧНИХ ЗОБРАЖЕНЬ / О. М. Березький, О. Й. Піцун, Г. М. Мельник, Т. В. Дацко // Український журнал інформаційних технологій Volume 3 Issue 1 Pages 73-77
17. Kumari K. Linear regression analysis study / Khushbu Kumari, Suniti Yadav // 2018, Volume: 4 , Issue Number: 1 , Page: 33-36
18. Dobra, A. (2009). Decision Tree Classification. In: LIU, L., ÖZSU, M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA. <https://doi.org/10.1007/978-0-387-39940-9_554>
19. Decision-Tree Classifier Tutorial [Електронний ресурс] – Режим доступу: <https://www.kaggle.com/code/prashant111/decision-tree-classifier-tutorial/notebook> (28.10.2022)
20. Decision Tree Algorithm [Електронний ресурс] – Режим доступу: https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html (28.10.2022)
21. Yang X. MDig: Multi-digit Recognition using Convolutional Nerual Network on Mobile / Xuan Yang // In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pages 3642–3649. IEEE, 2012
22. Kaiming He. Deep Residual Learning for Image Recognition/ Kaiming He , Xiangyu Zhang // ImageNet Large Scale Visual Recognition Challenge 2015 (ILSVRC2015)
23. Cloud TPU Documentation Guides Was this helpful? Send feedback Download, pre-process, and upload the ImageNet dataset <https://cloud.google.com/tpu/docs/imagenet-setup>
24. imagenet-1k [Електронний ресурс] – Режим доступу: <https://huggingface.co/datasets/imagenet-1k> (28.10.2022)
25. What is a REST API? [Електронний ресурс] – Режим доступу: redhat.com/en/topics/api/what-is-a-rest-api(28.10.2022)