НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра прикладної математики

«На правах рукопису»

УДК 519.688:004.855.5

«До захисту допущено»

Завідувач кафедри

_____ Олег ЧЕРТОВ

«___» _____ 2023 р.

Магістерська дисертація
на здобуття ступеня магістра
за освітньо-науковою програмою «Наука про дані та математичне моделювання»
зі спеціальності 113 «Прикладна математика»
на тему: «Математичне та програмне забезпечення системи класифікації сканованих документів для ділового документообігу»

Виконала: студентка ІІ курсу, групи КМ-11мн
Пащенко Катерина Михайлівна _____

Керівник:
Доцент, д-р фіз.-мат. наук
Норкін Богдан Володимирович _____

Консультант з нормоконтролю:
старший викладач,
Мальчиков Володимир Вікторович _____

Рецензент:
Старший викладач кафедри програмного
забезпечення комп'ютерних систем, канд. техн. наук
Гречко Анастасія Валеріївна _____

Засвідчую, що в цій магістерській дисертації
немає запозичень із праць інших
авторів без відповідних посилань.

Студент _____

Київ — 2023

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра прикладної математики

Рівень вищої освіти — другий (магістерський)
Спеціальність – 113 «Прикладна математика»
Освітньо-наукова програма «Наука про дані та математичне моделювання»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олег ЧЕРТОВ
«___» _____ 2023 р.

ЗАВДАННЯ

на магістерську дисертацію студентці

Пащенко Катерині Михайлівні

1. Тема дисертації: «Математичне та програмне забезпечення системи класифікації сканованих документів для ділового документообігу», науковий керівник дисертації доцент, д-р фіз.-мат. наук Норкін Богдан Володимировив, затверджені наказом по університету від «30» ___03____ 2023 р. № _1359__-С.

2. Термін подання студенткою дисертації: «15» травня 2023 р.

3. Об’єкт дослідження: методи обробки та розпізнавання цифрових зображень документів, типи цифрових зображень документів: відсканові документи, фото документу, електронний документ; техніки представлення зображення у вигляді

числових характеристик; методи розпізнавання на основі нейронних мереж (методи в TensorFlow, Keras), модель згорткової нейронної мережі для класифікації зображень документів, згорткові нейронні мережі, методи тестування системи класифікації, метрики ефективності роботи алгоритму класифікації, моделі верифікації (перевірки адекватності) алгоритму, дослідження залежності точності моделі від кількості даних для її навчання. Бібліотеки для глибокого навчання (Python); бібліотека Keras Tuner для автоматичної оптимізації гіперпарметрів нейронної мережі; набір даних для тренування моделей: Tobacco-3482, RVL-CDIP датасети. Існуючі системи та застосунки класифікації цифрових зображень документів: Neuroph, Simbrain, Encog, згорткові нейронні мережі типу LeNet, VGGNet, Google Net.

4. Предмет дослідження: модель системи класифікації зображень сканованих документів; техніки обробки зображень різних форматів та методи обробки візуальних об'єктів. Модель згорткової нейронної мережі для класифікації зображень. Метрики ефективності роботи алгоритму класифікації. Моделі верифікації (перевірки адекватності) алгоритму. Математичне, програмне, методичне та інші види забезпечення системи класифікації зображень текстових документів.

5. Завдання для досягнення мети: виконати аналіз існуючих систем класифікації документів; виконати аналіз існуючих методів аналізу, передбачення та класифікації категоріальних даних; визначити підсистеми майбутньої розробленої системи класифікації; обрати алгоритм машинного навчання для кожної з підсистеми; розробити процедуру підготовки вхідних даних; спроектувати автоматизовані підсистем; здійснити програмну реалізацію спроектованих підсистем; провести тестування розробленої ситеми та верифікацію результатів.

6. Орієнтовний перелік ілюстративного матеріалу: огляд математичних методів та розроблених рішень повз’янаних з класифікацією документів,

представлення компонентної моделі системи класифікації, розробка програмного забезпечення, огляд результату роботи розробленої системи коасифікації, презентація.

7. Орієнтовний перелік публікацій: тези за темою: «Математичне та програмне забезпечення системи класифікації цифрових зображень текстових документів».

8. Дата видачі завдання: 1 лютого 2023 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Скласти та затвердити план написання дисертації	15.09.2022	
2	Пошук та опрацювання літератури, аналіз істуючих рішень	01.10.2022	
3	Робота над першим розділом «Огляд існуючих рішень»	15.11.2022	
4	Розробка компонентної моделі системи. Робота над розділом «Опис розробленої системи». Підготовка статті за результатами наукового дослідження.	20.12.2022	
5	Робота над розділом «Огляд та обгрунтування математичних методів для реалізації розробленої моделі»	01.03.2023	
6	Робота над розділом «Програмна реалізація». Розробка програмного забезпечення системи.	01.04.2023	
7	Робота над розділом «Верифікація та валідація системи»	15.04.2023	
8	Оформлення текстової і графічної частин магістерської дисертації	01.05.2023	

Студентка _____ Катерина ПАЩЕНКО

Науковий керівник дисертації _____ Богдан НОРКІН

Реферат

Дисертацію виконано на 96 аркушах, вона містить 2 додатки та перелік посилань на використані джерела з 28 найменувань. У роботі наведено 51 рисунки.

Актуальність теми. На сьогодні все більше документів, таких як текстові документи, таблиці, схеми, платежів, заявки на роботу, різноманітні форми зберігаються та обробляються в форматі цифрового зображення. Також постає необхідність систематизовувати попередньо створені документи в паперовому вигляді та вилучати з них корисну інформацію. Тому актуальною є тематика пов'язана з класифікацією документів, адже саме це відіграє важливу роль у завданнях систематизації, сортуванні, класифікації сканованих чи сфотографованих документів в процесі ділового документообігу.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційна робота виконувалась згідно з планом науково-дослідних робіт кафедри прикладної математики Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

Мета і задачі дослідження. Метою дисертаційної роботи є підвищення ефективності роботи в завданнях класифікації та систематизації сканованих документів в процесі ділового документообігу.

Для досягнення мети було визначено наступні завдання:

- виконати аналіз існуючих систем класифікації документів;
- виконати аналіз існуючих методів аналізу, передбачення та класифікації категоріальних даних;
- визначити підсистеми майбутньої розробленої системи класифікації;
- обрати алгоритм машинного навчання для кожної з підсистеми;
- розробити процедуру підготовки вхідних даних;

- спроектувати автоматизовані підсистем;
- здійснити програмну реалізацію спроектованих підсистем;
- провести тестування розробленої ситеми та верифікацію результатів.

Методи дослідження. Для досягнення поставленої мети використовувалися такі методи: алгоритми машинного навчання, методи для обробки цифорових зображень, методи оптимізації гіперпараметрів та параметрів згорткової нейронної мережі, методи проектування систем Data Science, методи теорії алгоритмів та програмування, методи аналізу даних та математичної статистики.

Об'єктом дослідження є методи класифікації документів на основі структури документу.

Предметом дослідження є реалізація моделі класифікації сканов документів для підвищення ефективності в системах електронних офісів.

Наукова новизна. Удосконалено архітектуру згорткової нейронної мережі, яка за показниками точності та повноти не поступається відомим аналогам, але потребує менше часу на навчання, швидше класифікує цифрові зображення сканованих документів та потребує менше ресурсів для розгортання і використання.

Практична цінність одержаних результатів. На основі запропонованої системи для класифікації реалізовано програмний модуль для класифікації сканованих документів, що дозволяє провести систематизацію та сортування одиниць в сфері документообігу з можливістю подальшої обробки документів.

Апробація результатів дисертації. Основні положення та результати роботи представлено та опубліковано на конференції ПМК 2022 (Прикладна Математика та Комп'ютинг).

Публікації. Норкін Б. М., Пащенко К.М., Математичне та програмне забезпечення системи класифікації цифрових зображень текстових документів. Прикладна математика та комп'ютинг. ПМК-2022: п'ятнадцята науково-практична

конференція магістрантів та аспірантів, Київ, 16-17 лист. 2022 р.: зб. Тез доп./ [редкол.: Дичка І. А. та ін.]. — К. : Просвіта, 2022. — С. 86-93.

Ключові слова: згорткова нейронна мережа, сканований документ, цифрове зображення, структура документу, класифікація, набір даних, нейронні мережі.

ABSTRACT

The thesis is completed on 96 sheets, it contains 2 appendices and a list of references to the used sources from 28 names. The work contains 51 drawings.

Actuality of theme. Today, more and more documents such as text documents, tables, charts, payments, job applications, various forms are stored and processed in digital image format. There is also a need to systematize previously created documents in paper form and extract useful information from them. Therefore, the topic related to the classification of documents is relevant, because it plays an important role in the tasks of systematization, sorting, and classification of scanned or photographed documents in the process of business document circulation.

Thesis connection to scientific programs, plans, and topics. The dissertation work was carried out in accordance with the plan of research works of the Department of Applied Mathematics of the National Technical University of Ukraine "Ihor Sikorskyi Kyiv Polytechnic Institute".

The purpose and objectives of the research. The aim of the dissertation work is to increase the efficiency of work in the tasks of classification and systematization of scanned documents in the process of business document circulation.

To achieve the goal, the following tasks were defined:

- perform an analysis of existing document classification systems;
- perform an analysis of existing methods of analysis, prediction and classification of categorical data;
- to determine the subsystems of the future developed classification system;
- choose a machine learning algorithm for each of the subsystems;
- develop a procedure for preparing input data;
- to design automated subsystems;

- implement the software implementation of the designed subsystems;

- test the developed system and verify the results.

Methods of research.To achieve the goal, the following methods were used: machine learning algorithms, methods for processing digital images, methods for optimizing hyperparameters and parameters of a convolutional neural network, methods for designing Data Science systems, methods for the theory of algorithms and programming, methods for data analysis and mathematical statistics.

The object of the study is the methods of document classification based on the document structure.

The subject of the study is the implementation of a model for the classification of scanned documents to improve efficiency in electronic office systems.

Scientific contribution. The architecture of a convolutional neural network has been improved, which in terms of accuracy and completeness is not inferior to known analogues, but requires less time for training, classifies digital images of scanı documents faster and requires fewer resources for deployment and use.

Practical value of the obtained results. On the basis of the proposed system for classification, a software module for the classification of scanned documents has been implemented, which allows systematization and sorting of units in the field of document circulation with the possibility of further processing of documents.

Approbation of the results of the dissertation. The main provisions and results of the work were presented and published at the PMK 2022 (Applied Mathematics and Computing) conference.

Publications. Norkin B.M., Pashchenko K.M., Mathematical and software support of the classification system of digital images of text documents. Applied mathematics and computing. PMK-2022: the fifteenth scientific and practical conference of master's and postgraduate students, Kyiv, November 16-17. 2022: Coll. Theses add./ [edited by: I. A. Dychka and others]. — K.: Prosvita, 2022. — P. 86-93.

Keywords: convolutional neural network, scanned document, digital image, document structure, classification, dataset, neural networks

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CNN (Convolutional Neural Network) - згорткова нейронна мережа

DAS (Document Analysis Systems) - система аналізу документів

NN (Neural Network) - штучна нейронна мережа

ReLU (Rectified Linear Units) – шар зрізаниз вузлів

RGB (Red, Green, Blue) – колірна модель: червоний, зелений, синій

ВСТУП

Все більше й більше документів сьогодні, таких як текстові документи, таблиці, схеми, платежів, заявки на роботу, різноманітні форми зберігаються та обробляються в форматі цифрового зображення. Також слід відмітити, що постає необхідність проведення систематизації попередньо створених документів в паперовому вигляді та вилученні з них корисної інформації. Для таких завдань розглядаються методи класифікації цифрових зображень текстових документів, адже саме вони віріграють важливу роль у завданнях систематизації, сортуванні, класифікації сканованих чи сфотографованих документів.

Прикладами застосування системи класифікації цифрових зображень документів може бути сферу управління, де постає необхідність автоматично класифікувати та систематизувати заяки, форми, звіти чи інші ділові папери. В сфері фінансів додатково до перелічених функції системи класифікації працюють над виявленням некоректних платежів чи чеків, які у подальшому можуть відправитись на повторний перегляд. Така функція може не лише автоматизувати процедуру обробки даних, а й додатково попередити незаконні фінансові махінації. Варто звернути увагу на медичну сферу, адже швидка ефективність в роботі з медичними документами є важливо необхідним фактором для збереження життя та здоров’я населення. Більш того, в даному напряму документи відрізняються від звичних паперових форм чи листів, тому сортування та класифікація документів вимагає додаткових перевірок. Як результат, це ще раз вказує, що системи класифікації сканованих документів стають все більш популярними, вони допомагають автоматизувати, спростити та пришвидшири роботу з документами в усіх сферах, де необхідно сортувати чи класифікувати велику кількість паперів.

Існує декілька способів класифікації зображень текстових документів, в основному їх можна розділити на два класи: методи, що базуються на аналізі структури самого зображення документу та методи, що базуються на основі аналізу вмісту документу [1]. В деяких завданнях, наприклад виділення підписів чи пошук об’єкта на зображенні для більш ефективної класифікації зображень документів використовується поєднання вказаних способів. Проте слід враховуватити тип документів та бажаний кінцевий результат для вибору методу класифікації. В загальному, для документів, що мають чітку структуру та представлені у вигляді схем, форм, таблиць, чітко сформованих списків, доцільно використовувати перший методи класифікації зображення. Відповідно для документів з текстом, наприклад листи, звіти, статті, варто звернути увагу саме на методи з акцентом на оптичне розпізнаванням тексту. Однак обробляючи велику кількість цифрових зображень документів, важко наперед визначити який саме спосіб необхідно обрати. В більшості випадків роботи з документами метод класифікації зображення за макетом документа має більший вплив та перевагу над методом класифікації за змістом, адже спочатку продовиться робота з цифровою картинкою, класифікування до відповідної групи документів, а потім проводиться робота з самим текстом в залежності від задачі. Дана робота присвячена реалізації системи, що дозволяє будувати нейромережеві розпізнавачі для класифікації сканованих документів.

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Існуючі реалізації нейронних мереж

В відкритому доступі існує декілька прикладів нейронних мереж, які можуть використовуватися для задачі класифікації цифрових зображень документі. Проте вони мають сформовану архітектуру та визначені параметри мережі. Такий варіант обмежує розробників у можливості змінювати структуру шарів, додавати додаткові параметри в залежності від типу вхідних даних, а одже ускладнює задачу оптимізації системи.

Штучні нейронні мережі є одним з типів алгоритмів машинного навчання та активно використовуються взалежності від скадності будови та реалізації у задачах класифікації об’єктів, зокрема і розпізнавання картинок. Нейронні мережі мають спільну загальну структуру та складаються з декількох шарів: вхідний, вихідний шари та певна кількість прихованих шарів, до складу яких можуть входити шари оптимізації чи додаткових фільтрів.

1.1.1 Згорткова нейронанна мережа

Згорткова нейронна мережа[2] – різновид згорткових мереж, яка здебільного використовується в задачах класифікації зображень. Також існує декілька покращених програмних реалізацій даного типу нейронної мережі або додаткові модулі чи бібліотеки, які містять додаткові вбудовані функції чи шари для роботи з окремим типов вхідних даних та збільшують ефективність задачі класифікації.

Згорткова нейронна мережа здобула широкого застосування в задачах класифікації зображень, бо має ряд переваг порівняно з іншими алгоритмами. Як і класична штучна нейронна мережа, ЗНМ має стандартну аріхітектру – вхідний, вихідний шари та набір прихованих шарів – рис 1.1. Більш детальний розбір архітектури, шарів та додаткових параметрів ЗНМ буде розглянуто в наступних розділах.

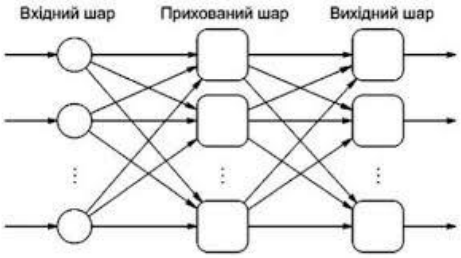


Рисунок 1.1 – Спрощена архітектура згорткової нейронної мережі[2]

Мережа ефективно працює з вхідними даними такими як, зображення, обробка зображень в процесі навчання базується на просторовій локальності самої картинки, обробка даних відбувається поступово та невеликими частинами, що забезпечує ефективність класифікації для зображень різного розміру. Ефективно підібрані параметри та вагові коефіцієнти мережі забезпечують універсальність мережі для вхідних даних різної розмірності та типу сфери застосування. Будова архітектури ЗНМ є інтуїтивно зрозумілою для розробки. Рівні обробки зображення розташоваі в порядку від простого до складного, на нижчих рівнях використовуються простіші функції обробки, в той час як на вищих рівнях функції мають вищу складність. ЗНМ може працювати з зображеннями без попередньої додаткової обробки, типу зміни кольорової гами, чи підготовки шаблонів, що дозволяє застосовувати ЗНМ в задачах класифікації зображень різної складності та деталізації.

Кількість та різновид задач класифікації зображень разом з різноманіттям самих зображень щороку збільшується, це допомагає ефективно навчити та підготувати ЗНМ. Ефективність роботи ЗМН може наближатися до ефективності роботи людини, або, навіть, в деяких випадках перевищити людську ефективність роботи. В задачах класифікації об’єктів на зображеннях до підтипів певної групи, наприклад класифікація порід тварин, рослин, дрібних деталей нейронна мережа є більш ефективним та універсальним інстументом, ніж людські ресурси.

Ефективність ЗНМ залежить від вхідного набору даних та глибини самої мережі. В деяких випадках глибина мережі впливає сильніше на ефективність роботи, ніж будь-яка складова архітектури, проте існує великий ризик перенавчання мережі, коли ЗНМ працює ефективно з навчальними даними та повертає високу точність класифікації, проте з тестовим набором даних ефективність мережі падає. Для уникнення даного конфлікту у згортковій мережі передбачено ряд параметрів та метрик для оцінки залежності ефективності роботи мережі від її розмірності.

Для досягнення максимальної ефективності ЗНМ в задачах класифікації існує декілька різновидів архітектури мережі, додаткових параметрів та фільтрів. Для оптимального навчання нейронних мереж застосовують оптимізатори - алгоритми, що слідкують за процесом навчання мережі та регуляризують значення ваг для уникнення перенавчання чи недонавчання.

1.1.2 Згорткова нейронна мережа типу LeNet

Мережа LeNet[3] – згорткова нейронна мережа, яка була представлена для задач класифікації зображень. Раніше застосувалась в задачах розпізнавання цифр для обробки чеків ти банкових форм. За ефективністю дану згорткову нейронну

мережу можна порівняти з методом машинного навчання – метод опорних векторів. Архітектура LeNet мережі не є складною, порівняно з іншими реалізаціями, та має зрозумілу структуру – рис 1.2.

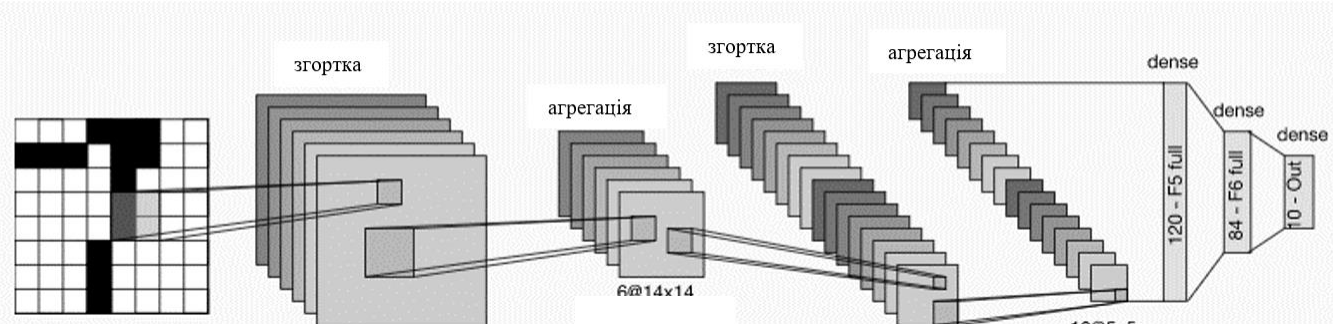


Рисунок 1.2 – Приклад архітектури згорткової нейронної мережі LeNet[3]

Приховані шари згорткової нейронної мережі місять пару згорткових шарів, шарів об’єднання, включають функції активації та стандартні шари входу та виходу. З моменту розробки даної моделі, LeNet застосовувалась в завданнях класифікації та розпізначання зображень з числами та класифікації медичних зображень. LeNet була одною з перших ЗНМ з нескладною архітектурою для задач класифікації зображень.

1.1.3 Згорткова нейронна мережа типу VGGNet

Одна з поширених реалізацій ЗНМ, що розроблена для збільшення глибини шарів мережі та покращення ефективності роботи моделі класифікації зображень є VGGNet [4] мережа, яка в залежності від кількості шарів в архітектурі поділяється на наступні типи: VGG-11, VGG-13, VGG-16 та VGG-19, відповідно числовий індекс в назві моделі показує глибину розглянутої згорткової мережі. На практиці найбільш популярними є моделі VGG-16 та VGG-19.

VGGNet є більш ефективною за класичну згорткову мережу, так як навчання даної моделі базується на більш ефективних ознаках у порівнянні з класичною мережею. Результатом роботи мережі є список значень імовірностей, до якого з можливих класів може належати поточний приклад. Архітектура даної мережі має свої особливості – рис 1.3.

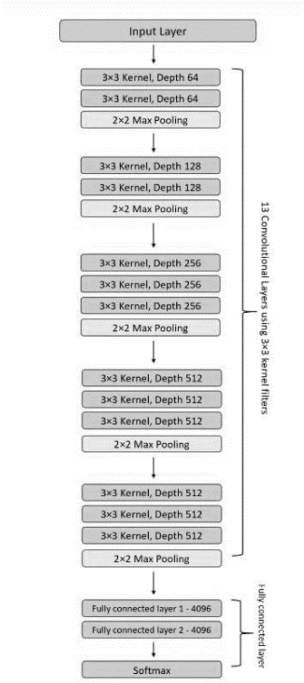


Рисунок 1.3– Приклад архітектури мережі VGGNet[4]

На вхід моделі подаються зображення розмірністю 224×224. Згорткові шари мережі мають наперед визначену сприятливу розмірність, яка становить 3×3 на кожному шарі. Додатково до попереднього шару включено згорткові шари розмірністю 1×1. В більшості випадків функція активації даної мережі є функція ReLU, яка використовується в усіх прихованих шарах мережі. VGGNet з глибиною шарів 16 має три повністю звязані шари. Хоча VGGNet є однією з найефективніших та швидких алгоритмів, що використовуються в задачах класифікації зображень,

питання про удосконалення даної архітектури є актуальним та розглядається випадки збільшення кількості шарів даної мережі до 20.

Мережа VGGNet є популярною в задачах розпізнавання людських облич та емоцій, задачах класифікації об’єктів одного роду до різних підгруп – класифікація порід тварин, рослин та є широко популярною в медичній сфері.

1.1.4 Згорткова нейронна мережа Google Net

Мережа Google Net[5] була розроблена компанією Google в 2014 році, але досі не втратила своєї популярності та ефективності в задачах класифікації зображень, а саме зображень ділових паперів.

Особливість архітектури мережі Google Net схожа на попередньо розглянуту VGG (див. 1.1.3) тим, що в архітектуру включена техніка шару згортки розмірністю 1×1 між шарами мережі, що допомагає зменшити кількість параметрів та ваг в мережі та уникнути перенавчання мережі. Повністю зв’язані шари використовуються в кінці архітектури, це зменшує кількість параметрів та підвищує ефективності до 60% - рис 1.4.

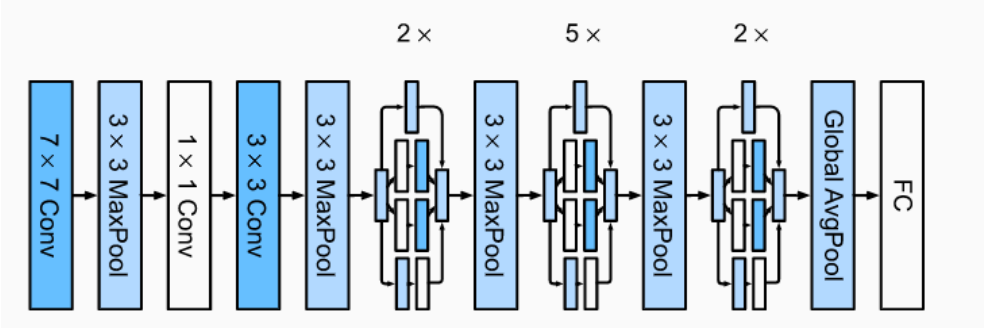


Рисунок 1.4 – Приклад архітектури Google Net[5]

Модулі згортки в даній архітектурі обробляють дані паралельно та об’єднуються разом у згенерований кінцевий результат. Такі фільтри згорткової мережі дозволяють працювати та обробляти об’єкти в різних масштабах.

Крім задач класифікації зображень, GoogLeNet використовується в задачах виділення предмету з зображення чи відео, задачах сегментації зображень та задачах відслідковування об’єктів в режимі реального часу.

1.2 Огляд розроблених рішень

Крім класичних та удосконалених реалізацій згорткових нейронних мереж, які використовуються для задач класифікації зображень вцілому, існує велика кількість готових програм розпізнавання та систем для побудови нейромережевих (і не тільки) класифікаторів.

1.2.1 Сервіс Neuroph

Сервіс Neuroph[6] - фреймворк, розроблений на мові програмування Java для проектування архітектури нейронних систем. Сервіс інтегрований з Java API, що підтримують соновні класи та утиліти для створення окремих типів нейронних мереж - рис 1.5.

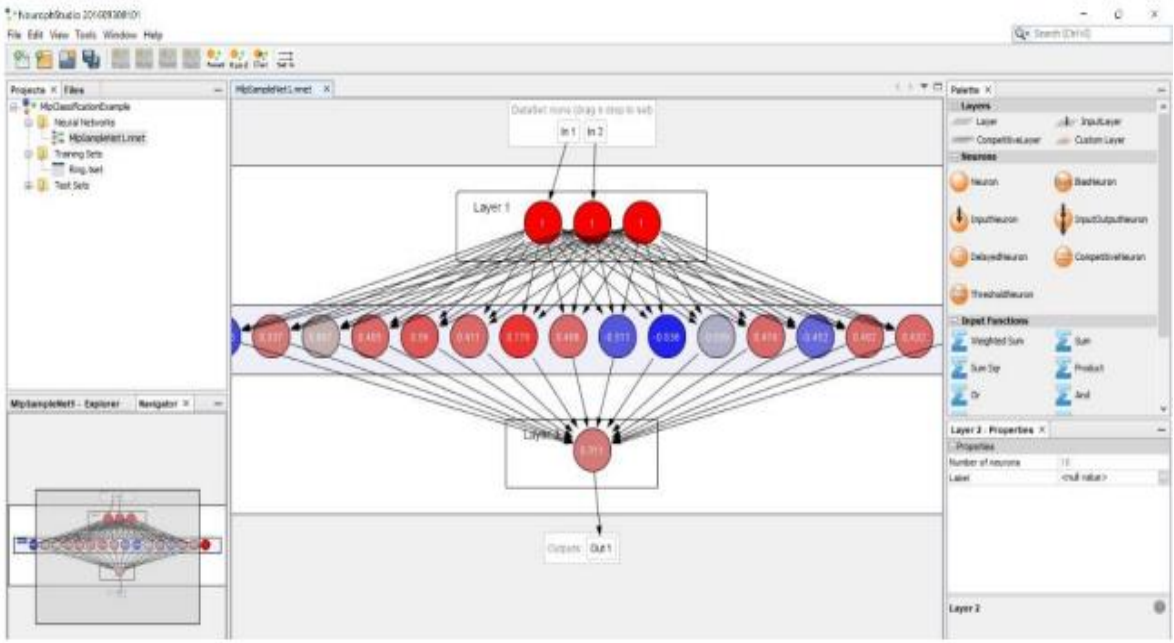


Рисунок 1.5 – Графічний інтерфейс програми Neuroph[6]

Програма має графічний інтерфейс з додатковими інструкціями для користування. У середовищі реалізовані наступні алгоритми:

- багатошаровий персептрон з алгоритмом зворотного поширення помилки;
- адалайн;
- персептрон;
- мережа Кохонена;
- мережа Хебба;
- мережа Хопфілда.

Крім побудови нейронних мереж в Neuroph Neuroph інтегровані додаткові фреймворки - Spring та Hibernate.

1.2.2 Сервіс Simbrai

Сервіс Simbrain[7] – програмне забезпечення з відкритим кодом, що орієнтована на розробку та візуалізацію архітектури штучних нейронних мереж – рис 1.6.

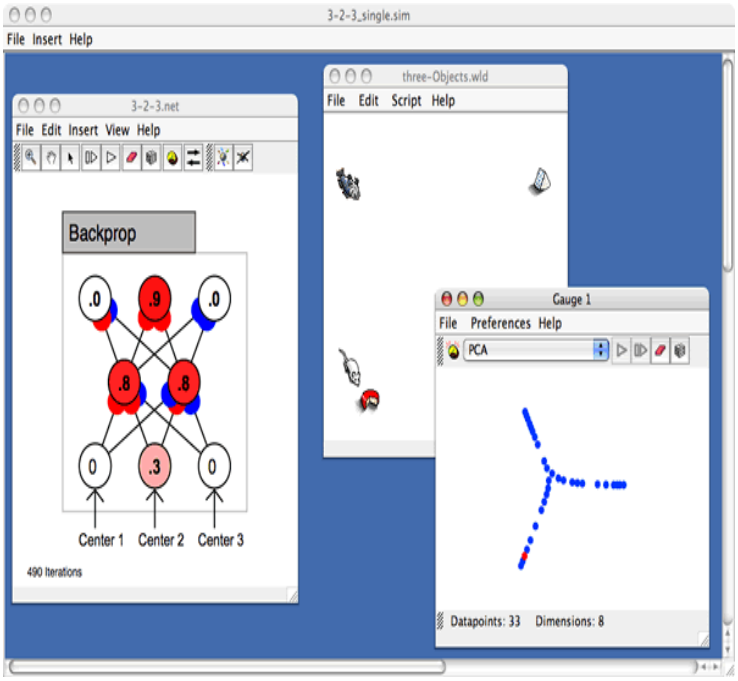


Рисунок 1.6 - Інтерфейс програми Simbrain[7]

В системі розроблено велику кількість різних варіантів нейронів і типів мереж, присутнє навчання без учителя та генетичні алгоритми. Підтримується візуалізація створеної архітектури нейронних мереж, тому розробник має змогу бачити візуалізацію процесу побудови шарів мережі, тасування ваг під час навчання та алгоритм зворотнього поширення помилки.

В Simbrain немає спеціальних налаштувань для реалізації зготкових нейронних мереж класифікації зображень. Вхідні дані необхідно завантажувати вручну, що не є зручним у випадку використання масштабних наборів даних.

1.2.3 Сервіс Encog

Сервіс Encog[8] є найбільш удосконаленим програмним забезпеченням з відкритим кодом, що розроблене на основі Java, C#, and C++ мовах програмування. В системі підтримується реалізація багатошарового персептрона, згорткової нейронної мережі, рекурентної нейронної мережі. Також підтримується розробка алгоритмів зворотнього поширення та Левенберг-Марквардт алгоритму, алгоритми машинного навчання – метод опорних векторів, дерева рішень та генетичні алгоритми.

Encog може бути встановленим на різних операційних системах, таких як Windows, Mac OS, and Linux. Має графічний інтерфейс, з інтерактивними підказками для ефективної побудови штучних нейронних мереж – рис 1.7.

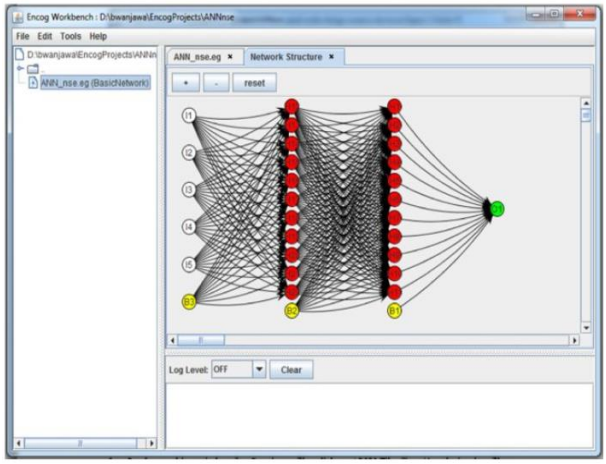


Рисунок 1.7 - Інтерфейс програми Encog[8]

В додадок до розглянутих функцій система підтримує паралельне функціонування та багатопроцесорну розробку. Проте в програмі так само не реалізована можливість розробки згорткових нейронних мереж, щоб дозволила працювати з задачами класифікації зображень.

1.3 Висновки до розділу

В розділі було виконано аналіз існуючих програмних продуктів на ринку для розпізнавання зображень, а саме цифрового зображення документів. Було визначено сильні та слабкі сторони досліджуваних продуктів. Виявлено, що запропоновані на ринку програмні продукти є багатофункціональні та високоефективні. Для аналізу та класифікації зображень в системах використовуються одночасно декілька алгоритмів та прийомів класифікації: нейронні мережі такі як: мережа Хопфілда, двостороння асоціативна пам'ять, мережа Кохонена, мережа Хебба та інші, наявність функції попередньої і пост обробки для поліпшення процесу навчання і оцінки продуктивності мережі та інші.

Водночас такі системи складаються з декількох блоків та вимагають значних комп’ютерних ресурсів також системи не є безкоштовними. Враховуючи вище наведені проблеми постає необхідність розробити простішу автоматизовану систему класифікації зображень документів, яка ефективно працюватиме в системах електронних офісів та стане допоміжним інструментом для малого бізнесу.

2 ОПИС РОЗРОБЛЕНОЇ СИСТЕМИ КЛАСИФІКАЦІЇ СКАНОВАНИХ ДОКУМЕНТІВ

2.1 Структура системи класифікації сканованих документів

Головна ідея роботи полягає в побудові та тренуванні нелінійного класифікатора, функцією якого є визначати різні класи документів на основі їх структури та віднести зображення документу до попередньо визначеного класу.

Модель системи класифікації цифрових зображень сканованих документів складається з таких компонентів:

- блок збору та збереження даних;
- блок обробки даних;
- блок навчання класифікатора;
- блок оцінки ефективності роботи класифікатора;
- класифікатор;
- блок програмного модуля.

Компонента модель наведена на рис 2.1.

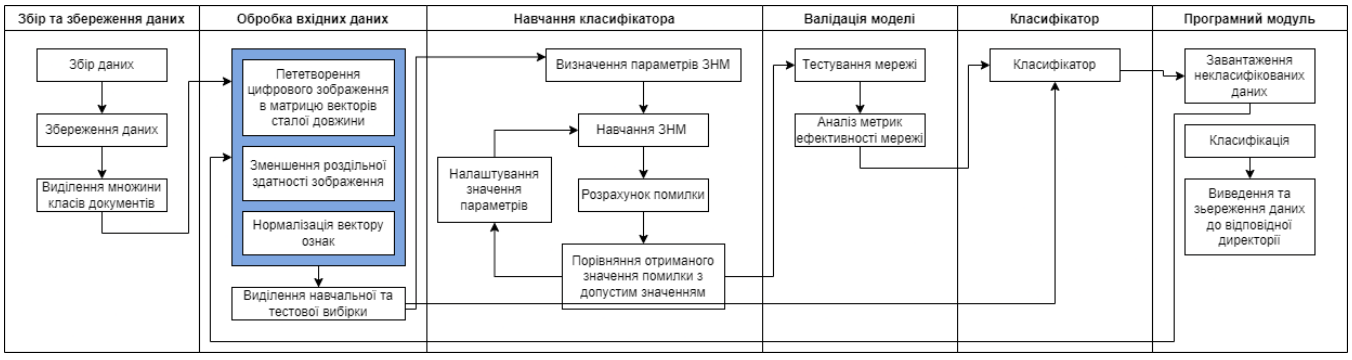


Рисунок 2.1 — Модель системи класифікації сканованих документів

2.2 Опис компонентів системи класифікації

Призначення системи полягає обробці структури різних типів текстових документів та класифікувати зображення документу до попередньо визначеного класу. Така система має зменшити час навчання, пришвидшити процес класифікації цифрових зображеннь, потребуючи менше ресурсів для розгортання і використання.

Кожен компонент системи призначений для виконання окремої задачі:

- компонент «Збереження даних» призначений для збору зображень сканованих документів. Багато текстових документів, таблиць, схем, платежів створюються, обробляються та зберігаються у формі цифрового зображення. Такі дані для даної системи можуть збиратися з різних типів електронних офісів, де зображення відразу класифікуються за певною категорією. Також отримати необхідні дані для навчання можливо за допомогою фото;
- компонент «Обробка даних» призначений для аналізу, чистки від шумів чи некоректних зображень та перетворення вхідних зображень у зображення нижчої роздільної здатності, які відповідають вхідному формату нейронної мережі;
- компонент «Навчання класифікатор» призначений для навчання моделі на основі даних зібраних та оброблених в попередніх компонентах коректно класифікувати зображення;
- компонент «Оцінка ефективності» призначений для аналізу роботи натренованого класифікатора та порівняння отриманих результатів з очікуваними. У разі не відповідності та отримання нижчної ефективності моделі від очікуваного, є можливість повернутися до компоненту «Навчання класифікатора»;

– компонент «Класифікатор» призначений для роботи з вхідними зображеннями документів, що необхідно прокласифікувати до визначеного класу документів;

– компонент «Програмний модуль» призначений для інтеграції класифікатора з іншими програмами для зручного користування та введення, виведення отриманого результату в найбільш зруний для користувача спосіб.

2.3 Вимоги до розробленої системи класифікації сканованих документів

Додатково було сформовано ряд бізнес-правил та вимог, яким має відповідати розроблена система класифікації[9]. Набір сформованих вимог буде використовуватися при аналізі отриманих результатів та процесу верифікації системи класифікації. Отже, розроблена система класифікації сканованних документів має відповідати наступним вимогам:

- в системі має бути реалізована функціональність обробки сирих даних та трансформування вхідного зображення в вектор ознак сталої довжини;
- наявність алгоритму машинного навчання, що вирішує задачу класифікації цифрових зображень текстових документів;
- системи повинна класифікувати зображення за наступними класами: «лист», «меморандум», «реклама», «email», «бланк», «нотанки», «реклама», «резюме», «науковий звіт» і «звіт»;
- розроблений програмний продукт в кінцевому вигляді повинен бути предсталений у виді вікна з можливістю завантаження зображень документів;
- система повинна проводити процедуру класифікації зображень документів з точністю не нижче 70-90%;

- в системі має бути передбачений функціонал завантаження сканованих документів з директорії.
- результат роботи системи класифікації має зберігатися в окремих папках, що відповідають кожному класу документів.

2.4 Висновки до розділу

У даному розділі була представлена компонентна модель системи класифікації цифрових зображень сканованих документів. Проаналізовано призначення кожного компонентту системи та описано як вони повзязані один з одним. Наведено список основних вимог, яким має відповідати розроблена система класифікації сканованих документів.

3. ОГЛЯД ТА ОБГРУНТУВАННЯ МАТЕМАТИЧНИХ МЕТОДІВ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ КЛАСИФІКАЦІЇ СКАНОВАНИХ ДОКУМЕНТІВ

3.1 Згорткова нейронна мережа

Різновидом штучних нейронних мереж є згорткова нейронна мережа(ЗНМ)[10], яка, як і класична мережа, за своєю топологією містить вхідний, вихідний та приховані шари – рис 3.1.

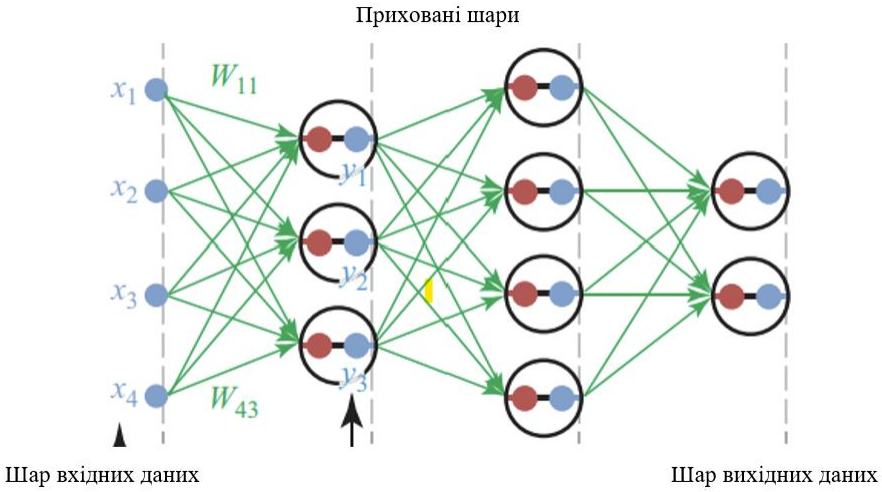


Рисунок 3.1 – Схематичне представлення архітектури ЗНМ[10]

Кожен вузол ЗНМ має ваги та відхилення, які навчаються та оптимізуються в процесі тренування мережі. Додаткво для більшої ефективності до прихованих шарів додані функції втрат, повнозвязні шари, шари нормалізації, фільтри та оптимізатори, детальніше про них буде описано в наступних пунктах. Головна відмінність ЗНМ полягає в тому, що така мережа застосовується в основному в задачах класифікації зображень, а отже розробник має повне представлення про вхідні дані, що допомагає вносити додаткові шари чи параметри в архітектуру мережі. Зображення подається на вхід мережі у вигляді матриці ознак, що в ході

тренування, застосовуючи математичні перетворення, згортається до матриці менших розмірів – цей принцип і називається згортоковістю мережі.

3.2 Опис шарів ЗНМ

3.2.1 Шари згортки ЗНМ

Основним шаром ЗНМ, де відбувається процес згортки є згортковий шар (англ. convolutional layer)[11]. Кожен згортковий шар представляє собою набір нейронів у вигляді прямокутної сітки – матриці значень розмірністю M×N, та вимагає, щоб вхідними даними шару була також прямокутна сітка значень. Розмір прямокутної частини, тобто сприятливого шару, є однаковим для кожного нейрону шару - m×n, разом з вагами - w та значенням зсуву. Останні значення зберігаються в окремому масиві, що називається ядром згортки.

Вхідними даними згорткового нейронного шару є зображення двовимірного розміру – I(n×m) та двовимірне ядро-фільт - K(i, j). Значення вихідних даних згортки також буде прямокутна матриця значень – X, але з меншою розмірністю (n-i+1×m-j+1) (3.1).

$$X(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n), \tag{3.1}$$

де I – вхідна двовимірна матриця значень,

K – ядро згортки.

Як видно з (3.1) результат зготки є комутативний, тому порядок множення не впливає на результат. Проте слід враховувати, в такому випадку спостерагіється явище крос-кориляції(3.2).

$$X(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n) \quad (3.2)$$

Для простоти реалізації на практиці формулу згортки реалізовують за допомогою бібліотеки, для оптимізації підбору усіх допутимих значень сприятливого поля (n×m). У контексті машинного навчання алгоритм згортки визначає відповідні значення ядра К у відповідному місці розташування зв'язного сприятливого поля, проте у випадку крос-корелації (3.2) ядро обретається навколо відповідної прямокутної матриці значень.

На рисунку 3.2 предсталено схематичну модель шару згортки використовуючи (3.1) без оберту ядра. Як видно вхід обмежено лише до тих позицій, де ядро лежить у межах зображення. Компонент вихідного шару формується шляхом застосування ядра до відповідної області вхідного шару.

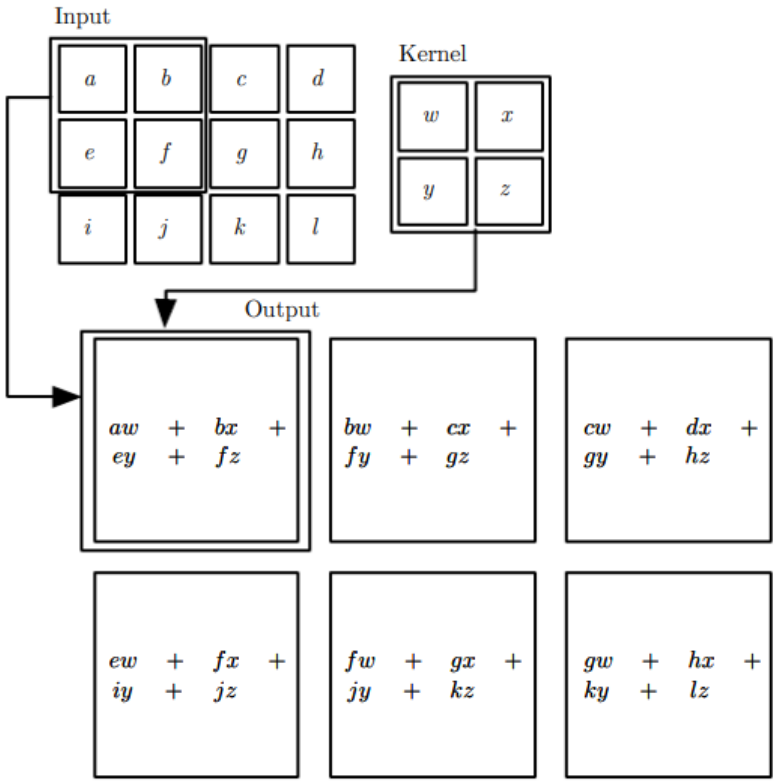


Рисунок 3.2 – Приклад двовимірної згортки [11]

3.2.2 Агрегувальні шари

Агрегувальний шар (анг. pooling layer) [10] ЗНМ використовується для об’єднання виходу кластерів на локальних чи глобальних ділянках прямокутної матриці входу одного шару до іншого. Агрегувальний шар додається після кожного шару згортки (див. 3.2.1), результатом роботи якого є єдиний блок значень отриманих в результаті об’єднання вихідної матриці шару зготки. Шар об’єднання є простіший за реалізацією, ніж шар зготки, оскільки не вимагає додаткових параметрів чи процесу навчання, а лиш застосовує математичне переворення для вхідних даних. Отримані значення є вхідними даними наступного рівня. Агрегація є

операцією нелінійного зничення розмірності вхідних даних, тобто призначення даного шару є зменшити розмірність, не зважаючи на значення ознак.

Існує декілька типів агрегування, які підбираются взалежності від задачі. Одним з популярних методів є максимізаційне агрегування (англ. max pooling)[10]. Ідея полягає в тому, щоб обрати максимальне значення з набору вхідних даних в конкретному патчі даних з подальшм об'єднанням в матрицю ознак (3.3). Максимізаційне агрешування повертає найважливіші ознаки з вхідних даних.

$$MaxPolling(X)_{i,j,k} = max_{m,n} X_{i*s_x+m,j*s_y+n,k}, i = 0 \dots m, j = 0 \dots m, \quad (3.3)$$

де X – матриця вхідних значень шару,

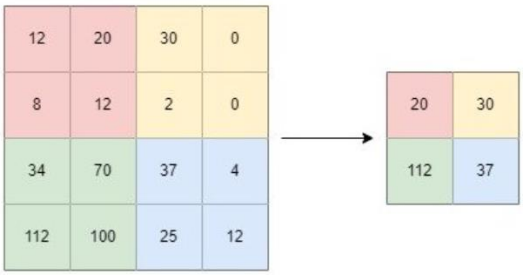
k – індекс патчу даних,

(m, n) – індекси патчів даних в матриці входу X,

(i, j) – відповідні індекси агрегованого значення даних (m, n) входу X в матриці результатів,

s_x, s_y - локальні індекси в патчі, де проходить максимізаційне агрегування.

Схематичне представлення роботи максимізаціного способу агрегування наведено на рисунку 3.3.



Риснунок 3.3 – Приклад максимізаційного агрегування

Усереднювальне агрегування (англ. average pooling) [10] повертає середнє значення конкретного пачу вхідної матриці значень з подальшм об’єднанням в матрицю ознак (3.4).

$$AvgPolling(X)_{i,j,k} = \frac{1}{f_x * f_y} \sum_{m,n} X_{i*s_x+m,j*s_y+n,k}, i = 0 \dots m, j = 0 \dots m, \quad (3.4)$$

де X – матриця вхідних значень шару,

k – індекс патчу даних,

(m, n) – індекси патчів даних в матриці входу X,

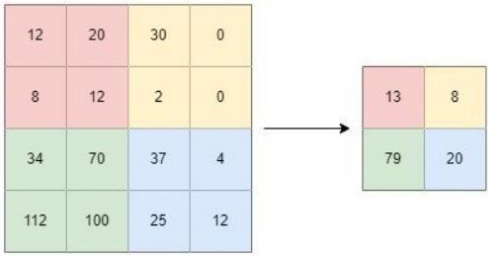
(i, j) – відповідні індекси агрегованого значення даних (m, n) входу X в матриці результатів,

s_x, s_y - локальні індекси в патчі, де проходить максимізаційне агрегування,

f_x, f_y – розміри конкретного патчу вхідної матриці значення.

Даний метод є ефективним у випадку, коли потрібне зглажування даних входу. В такому випадку метод допомагає визначити наявність викидів.

Схематичне представлення роботи максимізаціного способу агрешування наведено на рисунку 3.4.



Риснунок 3.4 – Приклад усереднювального агрегування

Стохастичне агрегування(англ. stochastic pooling)[11] – операція агрегування значень, яка додає до максимізованого об’єднання значення імовірності(3.5).

$$StochasticPooling(X)_{i,j,k} = \begin{cases} X_{i,j,k}, & \text{з імовірністю } p_{i,j} \\ 0 \ , & \text{з імовірністю } 1 - p_{i,j} \end{cases} \quad (3.5)$$

де X – матриця вхідних значень шару,

k – індекс патчу даних,

(i, j) – відповідні індекси агрегованого значення даних (m, n) входу X в матриці результатів,

$p_{i,j}$ - ймовірність збереження значення в позиції (i,j) у вхідній матриці ознак.

Імовірності визначається випадковим чином для кожного патчу вхідної матриці даних та зазвичай пропорційні значенням у конкретному патчі. Метод ефективний для випаків, коли постає необхідність підвищити стійкість моделі до невеликих варіацій у вхідних даних.

Основна мета рівня агрегування це поступове зменшення розмірів вхідних даних мережі, а отже зменшення кількості параметрів, які використовуються в навчанні ЗНМ. В архітектурі ЗНМ звичайною практикою є включення рівня об’єднання після кожного згорткового рівня. Операція об’єднання вводить новий тип інваріантності, який робить мережу нечутливою до невеликих змін у вхідних даних. Рівень об’єднання працює незалежно на кожному зрізі глибини вхідних даних і зменшує його просторові розміри.

3.2.3 Шар зрізаних лінійних вузлів

Шар зрізаних лінійних вузлів (англ. Rectified Linear Units layer, ReLU)[12], це тип функції активації, що використовується в нейронних мережах, зокрема і в ЗНМ. Це нелінійна функція, яка замінює всі від'ємні вхідні значення нулем, всі значення вище нуля залишає незмінними (3.6).

$$f(X) = \max(0, \quad x), \tag{3.6}$$

де x – масив вхідних даних шару.

ReLU застосовує ненасичену фукнцію активації, тому для посилення активації та підвищення ефективності ЗНМ існують функції такі як насичений гіперболічний тангенс чи сигмоїдна функція[12]. Це позитивно спливає на властивості мережі приймати рішення, без додаткового налаштування шарів згортки.

Проте така нелінійна функція має недолік пов'язаний з проблемою вибухового градієнту. Під час навчання ЗНМ можуть накопичуватись помилки, ваги та параметри навчання піддаються значним змінам, в результаті модель не є ефективною у процесі навчання даних. Результат шару зрізаних лінійних вузлів можу наближатися до 0. Дана проблема вирішується за допомогою зменшення швидкості навчання мережі та використання удосконаленої функції - Leaky ReLU[12](3.7).

$$f(X) = \max(0.01 * x, \quad x), \tag{3.7}$$

де x – масив вхідних даних шару.

Удосконалена функція представляє собою надзвичайно виправлену функцію та повертає певне значення вхідного масиву замість 0, як у попередній формулі. Отже для підвищення ефективності мережі та усунення помилок при навчанні в роботі використано саме останній варіант функції Leaky ReLU.

3.2.4 Шар активації софтмакс

На вихідному рівні ЗНМ в задачах багатокласової класифікації зображень застосовується шар з функцією активації софтмакс (англ. Softmax)[10]. Саме ця функція при отриманні вхідних даних повертає значення імовірностей за різними класами, які мережа навчена розпізнавати.

На вхід функція софтмакс отримує вхідний вектор дійсних значень, отриманих при тренуванні мережі на попередніх шарах. Попереднім шаром в ЗНМ може виступати шар згортки, або шар зрізаних вузлів. Далі відбувається перетворення вхідних даних на набір імовірностей того, до якого класу найбільше підходить обраний приклад(3.8).

$$Softmax(X)_i = \frac{s^{X_i}}{\sum_j s^{X_i}}, j = 0 \dots n, \tag{3.8}$$

де n – розмір вхідного вектору значень,

s^{X_i} – елемент вхідного вектору X_i.

Важливо, сума імовірностей отриманого вектора повинна дорівнювати 1. Після отримання вектору імовірностей, легко визначити до якого саме класу належить досліджуваний об’єкт, беручи максимальне значення імовіності.

3.3 Методи оптимізації ЗНМ

3.3.1 Техніка виключення

При розробці архітектури нейронної менежі часто застосовують повнозв’язні шари, проте це не є ефективним для всіх задач. При повному зв’язку між шарами кількість ваг та параметрів навчання збільшується, що може призвести до перенавчання моделі. Для уникнення даної проблеми пропонується метод виключення зв’язків(англ. dropout)[10, 11] в ЗНМ. Даний метод є ефективною технікою регуляризації зв’язків та нейронів в шарах мережі, яку можна застосовувати на різних рівнях. Прицип роботи техніки полягає в тому, що на визначених рівнях деякі вузли мережі зберігаються з імовірністю p, або навпаки – видаляються з імовірністю 1- p (3.7).

$$g(h) = Da(h), \tag{3.7}$$

де $a(h)$ – функція активації (див. розділ 3.2.3),

D = ($X_{0,}$ ..., X_i) – n-вимірний вектор випадкових величин розподілених за законом Бернулі.

Величина X_i предасталена у вигляді (3.8).

$$f(k,p) = \begin{cases} p, & \text{якщо } k = 1 \\ 1-p, & \text{якщо } k = 0 \end{cases}, \tag{3.8}$$

де k – умовне позначення усіх вхідних значень певного шару мережі.

Під час епохи навчання кожному вузлу мережі присвоюється імовірність p, для того щоб певний нейрон був викинутий стохастично. Поняття видалення вузла мережі означає, що датий вузол повертає значення 0(3.9).

$$G_i = X_i * a_i(h) = \begin{cases} a_i(h), \text{ якщо } X_i = 1 \\ 0\text{ , якщо } X_i = 0 \end{cases} \quad (3.9)$$

В підсумку видалені вузли покладаються на інші вузли і навчання проходить у скороченій мережі – рис.3.5.

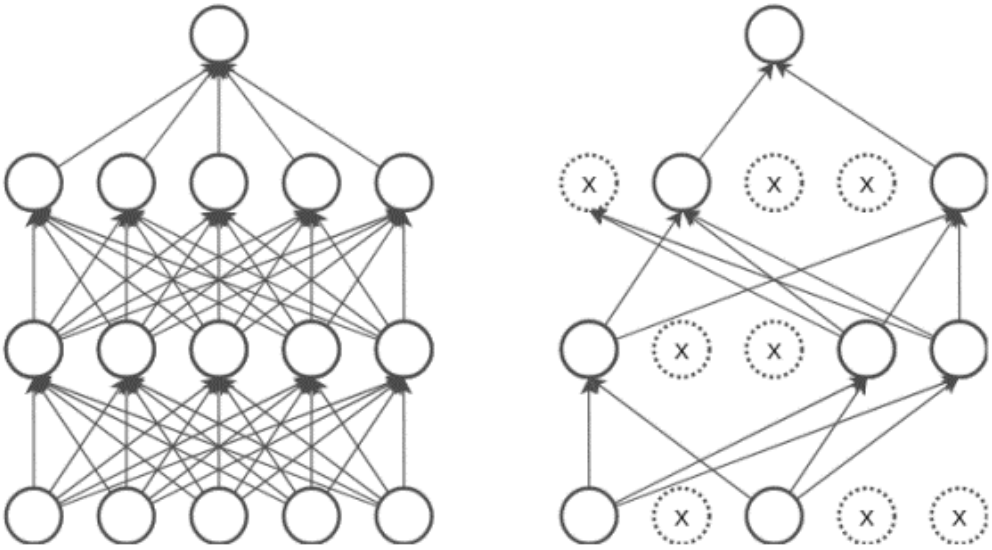


Рисунок 3.5 – Візуальна демонстрація застосування техніки виключення в повнозв’язній мережі[11]

Техніка виключення застосовується після шарів згортки та агрегування. Після проходження серії навчання, де вузли були виключені з навчання, ті ж самі вузли можуть бути повернені назад. В такому випадку значення вагів та інших параметрів навчання не змінюються, а повертаються до попередньо заданого значення. Крім

зменшення імовірності перенавчання мережі, метод виключення також підвищує швидкість тренування мережі.

3.3.2 Гіперпараметри та параметри ЗНМ

Важливою складовою будь-якої нейронної мережі є гіперпараметри, які слідкують за процесом навчання мержі. Ці значення визначаються перед навчанням мережі та є сталими протягом усього процесу тренування. Підбір параметрів можуть впливати на результат навчання як позитивно, так і негативно зменшувати ефективність, сприяти перенавчанню мережі.

Кількість шарів[13]– це тип гіперпараметру, який визначає глибину мережі, а саме кількість шарів фільтру, що будуть застосовуватись до вхідного зображення. З додаванням додакових шарів до мережі, продуктивність може збільшуватись, проте це впливає на час навчання та обсяг обчилення. Кількість шарів визначається екпериментально, попередньо оцінивщи складність обчислень, складність обробки вхідних даних або розробивши елементарну мережу з декількох шарів для тестування.

Розмір ядра згортки[13] – даний гіперпараметр визначає розмір того фільру, що застосовується до вхідної матриці ознак трансформованого зображення. Розмір ядра згортки залежиться від розміру вхідного зображення.

Швидкість навчання[13]– гіперпараметр, що контролює процес зміни ваг під час тренування. Порівняння відбувається відносно градієнта фукнції втрат. Значення даного параметру визначає з якою швидкістю відбувається рух до точки мінімуму різниці між прогнозованим значенням та реальним. Чим меншне значенням тим

навчання мережі є повільнішим, проте на практиці значення гіперпараметру визначається випадковим чином та оптимізується в ході екпериментів.

Швидкість виключення[13] – гіперпараметр, що визначає імовірність, з якою вузли видаляються в процесі навчання мережі відповідно до техніки згортки (див. 3.2.5).

3.3.3 Пакетна нормалізація

Кожеш шар згорткової мережі приймає значення з попереднього шару та передає оброблені дані наступному. В ході такого транспортування даних може з’явитися проблема нестабільних градієнтів, тобто значення параметрів у всіх наступних вхідних шарах мережі погіршуються та спостерігається явище перенавчання або недонавчання. Метод, який застосовується для нормалізації вхідних даних кожного шару мережі є пакетна нормалізація (англ. batch-normalization)[14], саме він бориться з проблемою внутрішнього коваріатного зсуву.

При роботі з великим набором даних відбувається розділення вхідних навчальних даних на набори, метод пакетної нормалізації нормалізує дані кожного патчу за раз. Дані кожного вхідного шару проходять нелінійну функція активації, проте для регулювання параметрів навчання мережі необхідно звести середнє значення та середньоквадратичне відхилення до 0 (3.10), (3.11).

$$\hat{x}_i = \frac{x_i - \mu_b}{\sqrt{\sigma_b^2}} \tag{3.10}$$

або

$$\hat{x}_i = \frac{x_i - \mu_b}{\sqrt{\sigma_b^2 + \mathrm{e}}}, \tag{3.11}$$

μ_b – середнє значення кожного вхідного пакету даних,

σ_b^2 – середньоквадратичне відхилення кожного вхідного пакету даних,

x_i – вхідне значення попереднього шару,

e – можлива похибка фідхилення(параметр не обов'язковий).

На практиці пакетна нормалізація застосовується відразу після зготкового шару, або після шару агрегування, проте можуть бути варіанти застосування пакетної нормалізації і після функції активації.

3.3.4 Оптимізатори

В задачах багатокласової класифікації зображень використовуються додаткові фукнції, такі як оптимізатори. Оптимізатор контролює в сукупності значеннями гіперпараметрів(див. 3.6), фільтрів та межі налаштування ваг в процесі тренування ЗНМ.

Adam (Adaptive Moment Estimation)[15] – алгоритм оптимізації нейронних мереж, що має перевагу над іншими методами оптимізаї, наприклад в порівнянні з послідовним адаптивним алгоритмом LMS, в паралельності обчислень, що дозваляє отримувати значення оптимальних параметрів глобально, протистояти помилці синхронізації та повертати оптимальний результат швидше.

Принцип роботи даного алгоритму оптимізації полягає в пришвидшенні градієнтного спуску з врахуванням екпоненціального середнього(3.12).

$$w_{t+1} = w_t - a * m_t, \quad (3.12)$$

де w_i – сукупність ваг у потосний момент часу t.

Обчислення m_i відбувається за формулою (3.13)

$$m_i = \beta_1 * m_{t-1} + (1 - \beta_1)[\frac{\delta X}{\delta w_i}], \quad (3.13)$$

де m_t - сукупність градієнтів у поточний момент часу t (початкове значення mt = 0),

w_t – сукупність ваг у поточний момент часу t,

$a_t -$ коефіцієнт швидкості навчання в момент часу t,

δX – похідна фукнції втрат,

δw_t – похідна функція ваг в момент часу t,

β_1 – параметр швидкості розпаду середнього градієнта(за замовчуванням рекомендоване значення 0.09).

3.4 Алгоритм зворотньої поширення помилки

Дані в ЗНМ поширюються та обробляються в прямому напряму, тобто від шарів вхідних даних, проходячи через вузли прихованих шарів, піддаючись впливу вагам тренування, прямо до шарів виходу. Так зване пряме поширення даних проводиться доти, доки мережа не досягне бажаної ефективності роботи. В свою чергу алгоритм зворотнього поширення[10, 16] працює над налаштуванням ваг та інших параметрів мережі, щоб мережа змогла навчитися та досягнути до бажаної точки. В алгоритмі зворотнього поширення змінам піддаються ваги та значення дельт, тобто значення зміщення. Алгоритм працює з градієнтом функція втрат. В

процесі знаходиться, якою мірою зміна ваги в одному пікселі впливає на функцію витрат - $\frac{\partial E}{\partial w^l_{m^l,n^l}}$. Розмір вхідного зображення - M×N , розмір ядра ваг - $k_1 \times k_2$, тож градієнтна складова обчислюється послідовним чином(3.14).

$$\frac{\partial E}{\partial w^l_{m^l,n^l}} = \sum_{i=0}^{M-k_1} \sum_{j=0}^{M-k_2} \frac{\partial E}{\partial x^l_{i,j}} \frac{\partial x^l_{i,j}}{\partial w^l_{m^l,n^l}} = \sum_{i=0}^{M-k_1} \sum_{j=0}^{M-k_2} \delta^l_{i,j} \frac{\partial x^l_{i,j}}{\partial w^l_{m^l,n^l}} \tag{3.14}$$

де $w^l_{m^l,n^l}$ – значення ваги матриці значення шару *l*,

$x^l_{i,j}$ – вектор входів до шару l (3.15)

В свою чергу $x^l_{i,j}$ визначається, як (3.15), тоді загальний вигляд $\frac{\partial x^l_{i,j}}{\partial w^l_{m^l,n^l}}$ предаставлений в (3.16)

$$x^l_{i,j} = \sum_m \sum_n w^l_{i+m,j+n} * \mathrm{o}^{l-1}_{i+m,j+n} + b^l, \tag{3.15}$$

де $\mathrm{o}^{l-1}_{i,j}$ – вихідний вектор значень шару *l-1*,

b^l – зміщення шару *l*.

$$\frac{\partial x^l_{i,j}}{\partial w^l_{m',n'}} = \frac{\partial}{\partial w^l_{m',n'}} \left(\sum_m \sum_n w^l_{m,n} * \mathrm{o}^{l-1}_{i+m,j+n} + b^l \right) \tag{3.16}$$

Оскільки в рівнянні (3.17) при знаходженні суми всі значення компонентів рівні нулю, крім тих, де m = m' та n = n' в

$$\frac{\partial x^l_{i,j}}{\partial w^l_{m',n'}} = \frac{\partial}{\partial w^l_{m',n'}} \left(w^l_{m,n} * \mathrm{o}^{l-1}_{i+0,j+0} + \cdots w^l_{m',n'} * \mathrm{o}^{l-1}_{i+m,j+n} + \cdots + b^l \right) =$$

$$= \frac{\partial}{\partial w^l_{m',n'}} \left(w^l_{m',n'} * o^{l-1}_{i+m,j+n} \right) = \left(o^{l-1}_{i+m',j+n'} \right) \quad (3.17)$$

(3.18)

Виходячи з (3.17) отримано наступний результат градієнту функції втрат (3.18)

$$\frac{\partial E}{\partial w^l_{m^l,n^l}} = \sum_{i=0}^{M-k_1} \sum_{j=0}^{M-k_2} \delta^l_{i,j} \; o^{l-1}_{i+m',j+n'} = \Delta W$$

Отже ваги наступного шару будуть приймати значення як в (3.19)

$$W \leftarrow W - a * \Delta W, \quad (3.19)$$

де α∈(0,1].

3.5 Методи попередньої обробки вхідних даних

3.5.1 Нормалізація вхідних даних

Ефективність роботи мережі залежить не лише від топології нейронної мережі та кількості параметрів навчання, а й безпосередньо від вхідних даних. У випадку задач класифікації зображень на вхід подається матиця ознак зображення, значення якої приймають діапазон від 0 до 255. Якщо розглядати RGB зображення, то значення в матриці ознак може варіюватися в більшому діапазоні, оскільки інтенсивність одного каналу може бути набагато більша за інший. В такому випадку більші значення каналів при навчанні мережі будуть домінувати над іншими значеннями.

Щоб вирішити таку проблему вхідні дані потрібно підготувати, тобто нормалізувати до визначеного масштабу[17]. На практиці в задачах класифікації зображень ефективно застосовувати масштабування ознак (3.20) та стандартизовану оцінку(3.21).

$$x = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3.20}$$

$$x = \frac{x - \mu}{\sigma}, \tag{3.21}$$

де μ - середнє вибірки вхідних даних,

σ – середньоквадратичне відхилення вхідного набору даних.

У випадку з RGB зображеннями значення нормалізуються окремо для кожного вхідного каналу. Значення μ та σ також визначаються в межах одного каналу вхідних даних та повинні бути обчисленими лише на базі тренувальних даних, для тестових даних чи даних для валідації використовують ті ж самі значення.

3.5.2 Техніки зменшення роздільшої здатності вхідного зображення

Навчання моделі та власне сама класифікації цифрового зображення документу проводиться на зображенні, проте мережа працює лише з числовими даними, тому постає необхідність трансформувати вхідне зображення формату .jpeg довільної роздільної здатності в набір ознак конкретного розміру.

Роздільна здатніть вхідного зображення є довільною та може досягати більше ніж 1000 пікселів, такий розмір не є оптимальний входом згорткоової нейронної мережі, може призвести до надлишковості параметрів навчання, громіздкості

обчислень та перенавчання мережі. Модель працює з самою структурою документу, без врахування змісту картинки, тому доцільно є зменшення роздільної здатності картинки до тих пір, доки структура документу залишається зрозумілою для класифікації. Одним із методів змершення роздільної здатності зображення є інтерполяція[18]. Ідея полягає в переоцінці значень пікселів в зображенні зміненого розміру на основі відомих пікселів. Існує декілька видів інтерполяції: сусідня, білінійна та бікубічна інтерполяції. Білінійна інтерполяція предсталяє собою обчилення фукнції двох змінних, що накладається на варіант двовимірного зображення. Узагальнена формула білінінйної інтерполяції функції двох змінних – (3.22).

$$F(x,y) = \sum_{i=0}^{1}\sum_{j=0}^{1} a_{ij}\, x^i y^j \tag{3.22}$$

де a_{ij} – коефіцієнт інтепроляції.

У випадку білінійної інтерполяції коефіцієнти знаходяться з системи лінійних рівнянь відносно відомих значень – 3.23 – 3.26.

$$f(x_1y_1) = a_{00} + a_{10}x_1 + a_{01}y_1 + a_{11}x_1y_1 \tag{3.23}$$

$$f(x_1y_2) = a_{00} + a_{10}x_1 + a_{01}y_2 + a_{11}x_1y_2 \tag{3.24}$$

$$f(x_2y_1) = a_{00} + a_{10}x_2 + a_{01}y_1 + a_{11}x_2y_1 \tag{3.25}$$

$$f(x_2y_2) = a_{00} + a_{10}x_2 + a_{01}y_2 + a_{11}x_2y_2 \tag{3.26}$$

Таким чином роздільна здатність зображення може бути зменшена та приведена до єдиного розміру - рис 3.6, 3.7, щоб потім бути поданою на вхід класифікатора. Як видно з рисунка 3.7, струкрута документу залишається незмінною, проте зміст та чіткість тексту змінилась.

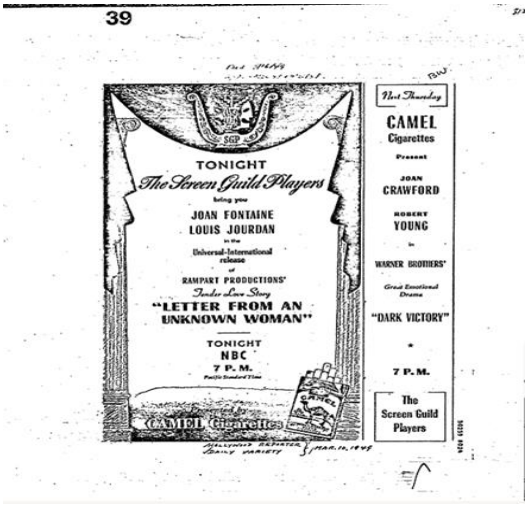


Рисунок 3.6 - Початкове зображення

Рисунок 3.7 - Зображення після зменшення роздільної здатності до 224×224

3.2.1 Техніки виділення з зображення вектору ознак

Нейронна мережа на вихід повертає вектор імовіpностей, до якого з класів може належати поточний приклад. Для отримання даного результату на вхід мережі має подаватися вектор ознак. Оскільки в роботі вхідними даними є зображення, постає питання, як трансформували зображення до вигляду вектору значень. Кожне зображення складається з окремих частин, пікселів, кожен з яких містить значення трьох каналів RGB (англ. Red, Green, Blue). В результаті вхідне зображенння

трансформується в трьохвимірний вектор, координати якого відповідають за висоту, широту на кількість каналів зображення. Значення кожного пікселі знаходиться в діапазоні від 0 до 255. Зі зростанням потреби в задачах компютерного зору або роботи з зображеннями математичне преставлення трансформування зображення в вектор ознак пресвтавлено матиматичне в ряді програмних фукнції, наприклад img_to_array()[15, 18].

3.6 Метрики оцінки ефективності класифікатора

Оцінка ефективності та адекватності моделі багатокласової класифікації зображень вимагає більшого об’єму обчислень та має свої особливості.

Один з найпростіших способів оцінки ефективносіт роботи класифікатора – є матриця помилок[10, 19]. Матриця помилок дозволяє одночасто порівняти ефективінсть моделі для різних класів та складається наступних значень для кожного класу: істинно позитивне, істинно негативне, хибно позитивне, хибно негативне – рис 3.8.

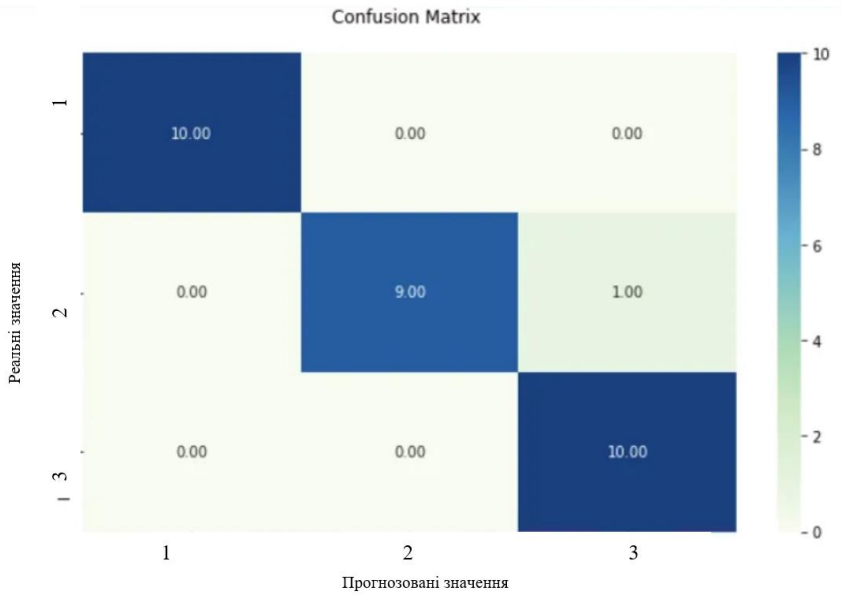


Рисунок 3.8 – Візуалізація матриці помилок для багатокласової моделі класифікації.

По діагоналі матриці вказана кількість одиниць кожного класу, що визначені як істинно позитивні – приклад був прокласифікований до первного класу j, і в дійсності цей приклад належить до цього класу j.

Значення, які вказані справа та зліва відносно діагональної комірки класу j вказують на кількість хибно негативних прогнозів – приклад належав до класу j, проте класифікатор відніс приклад до іншого класу.

Значення, що знаходсть в комірках вище та нижче діагональної комірки класу j вказують на кількість прикладів, що було визначені класифікатором як хибно позитивні – приклад не належав до класу j, проте класифікатор відніс його саме до даного класу j.

Значення у всіх інших комікаx матриці помилок вказують на кількість істинного негативних прогнозів – приклад, що був прокласифікований до класу відмінного від класу j, і в дійсності поточний приклад не належить до класу j.

Коефіцієнт точної відповідності (анг. Exact Match Ratio)[10] визначає частку прикладів, що були прокласифіковано правильно (3.27).

$$EMR = \frac{\sum I(y^i == \widehat{y^i})}{n}, \tag{3.27}$$

n – кількість прикладів тренування, y^i – дійні значення, $\widehat{y^i}$ – прогнозоване значення в тренувальній вибірці.

Дана метрика обчислюється для кожного класу прикладів окремо. Недоліком даної метриє є те, що оцінка є грубою, оцінюється лише правильно класифіковані приклади.

1/0 Loss метрика[19] є протилежною до метрики коефіцієнт точної відповідності, оскільки тут оцінюється загальна кількість прикладів, які було прокласифіковано неправильно (3.28).

$$one\_sero\_loss = \frac{\sum I(y^i \neq \widehat{y^i})}{n}, \tag{3.28}$$

n – кількість прикладів тренування, y^i – дійні значення, $\widehat{y^i}$ – прогнозоване значення в тренувальній вибірці.

Метрика Хеммінга удосконалена версія 1/0 Loss метрика, оскільки визначає пропорцію неправильно прокласифікованих прикладів до загальної кількості для кожного класу даних (3.29).

$$Hem\_Los = \frac{1}{nL}\sum_{i=1}^{n}\sum_{j=1}^{L} I(y_j^i \neq \widehat{y_j^i}) \tag{3.29}$$

де n – кількість прикладів тренування, L – кількість класів, y_j^i– дійсні значення для і-го прикладу класу j, $\widehat{y_j^i}$–прокласифіковані значення для і-го прикладу класу j.

Як видно з формули (3.29) для багатокласової моделі враховується усі значення які прокласифіковано неправильно, а потім усереднюється для всіх класів.

Точність (англ. Accuracy)[19] є метрикою, яка вцілому описує ефективність роботи моделі в усіх класах. Метрика оцінює кількість правильно класифікованих зображень до загальної кількості та використовується у задачах, де усі класи є одноково важливими. Макро точність спочатку визначається для кожного класу прикладів окремо(3.30), а потім усереднюється для усіх класів (3.31).

$$A_{macro}^{j} = \frac{\sum_{i=1}^{n} I(y_j^i \cap \widehat{y_j^i})}{\sum_{i=1}^{n} I(y_j^i \cup \widehat{y_j^i})} \tag{3.30}$$

де n – кількість прикладів тренування, L – кількість класів, y_j^i– дійсні значення для і-го прикладу класу j, $\widehat{y_j^i}$–прокласифіковані значення для і-го прикладу класу j.

$$A_{macro} = \frac{1}{k} \sum_{j=1}^{k} A_{macro}^{j} \tag{3.31}$$

де k – кількість класів.

Мікро точносіть обчислюється глобально для всіх прикладів всіх класів(3.32).

$$A_{micro} = \frac{\sum_{j=1}^{k} \sum_{i=1}^{n} I(y_j^i \cap \widehat{y_j^i})}{\sum_{j=1}^{k} \sum_{i=1}^{n} I(y_j^i \cup \widehat{y_j^i})} \tag{3.32}$$

Збіжність (англ. Precision)[19] – метрика, яка оцінює частку прикладів, що було прокласифіковано правильно. Метрика обчислюється, як частка усіх правильно прокласифікованих об’єктів до класу j до загальної кількості об’єктів, які віднесено класифікатором до класу j. Аналогічно до метрики точності, метрика збіжності в задачах мультикласової класифікації поділяється на два типи: макро збіжність(3.33, 3.34) та мікро збіжність моделі по всіх класах одночасно(3.35).

$$P_{macro}^{j} = \frac{\sum_{i=1}^{n}(y_j^i \cap \widehat{y_j^i})}{\sum_{i=1}^{n} \widehat{y_j^i})} \tag{3.33}$$

$$P_{macro} = \frac{1}{k} \sum_{j=1}^{k} P_{macro}^{j}, \tag{3.34}$$

де к - кількість класів.

$$P_{micro} = \frac{\sum_{j=1}^{k}\sum_{i=1}^{n}(y_j^i \cap \widehat{y_j^i})}{\sum_{j=1}^{k}\sum_{i=1}^{n} \widehat{y_j^i}} \tag{3.35}$$

В попередньо наведених метриках ефективність роботи класифікатора оцінювалась лише з урахуванням частики правильно прокласифікованих прикладів. Такий підхід може призводити до збитковості значення, оскільки при цьому не враховуються неправильно класифіковані приклади даних, кожному класу незалежно від кількості прикладів, які до нього належіть, присвоюється однакова важливість. В такому випадку на оцінку ефективності будуть впливати класи, в яких кількість об’єктів більша, що не є коректним підходом. Для вирішення даної проблеми пропонується використовувати збалансовану ментику оцінки – повнота (англ. Recall))[19]. Для моделей класифікації метрика повноти преставлена двома типами: макро та мікро повнота(3.36, 3.37, 3.38).

$$R_{macro}^{j} = \frac{\sum_{i=1}^{n}(y_j^i \cap \widehat{y_j^i})}{\sum_{i=1}^{n} y_j^i)} \quad (3.36)$$

$$R_{macro} = \frac{1}{k}\sum_{j=1}^{k} R_{macro}^{j}, \quad (3.37)$$

де к - кількість класів.

$$Recall_{micro} = \frac{\sum_{j=1}^{k}\sum_{i=1}^{n}(y_j^i \cap \widehat{y_j^i})}{\sum_{j=1}^{k}\sum_{i=1}^{n} y_j^i} \quad (3.38)$$

3.7 Висновки до розділу

В розділі було прорведено аналіза математичних технік та алгоритмів, які будуть використано для досягнення кінцевого результату – створення системи коасифікації сканованих документів. Для розробки коасифікатора в системі класифікації було обрано згорткову нейронну мережу. В розділі було проаналізовано топологію ЗНМ, описано основні шари в архітектурі нейронної мережі – шари згортки та агреграції; наведено приклад функцій активації, які застосовуються в моделях багатокласової класифікації, проаналізовано вплив повнозв'язних шарів в аріхтектурі ЗНМ та описано техніку - виключення для оптимізації роботи мережі та зменшення імовірності перенавчання мережі.

Проведено огляд рішень, які доцільно використовувати для підготовки вхідних даних – регуляризація та нормалізація, а також пакетна нормалізація, яка застосовується в прихованих шарах мережі для уникнення перенавчання та пошуку оптимальних значень ваг тренування.

В ході досліження було визначено основні фактори та гіперпараметри, що впливають на ефективність роботи мережі та швидкість навчання – це кількість шарів(глибина), параметр швидкості навчання мережі, параметр зміщення ваг та інші. Розглянуто можливі варіанти оптимізаторів, зокрема оптимізатор Adam, що допоможе у регулюванні процесу тренування мережі.

Наведено основні метрики оцінки ефективності мережі саме для оцінки моделі багатокласової класифікації: матриця помилок, макро та мікро метрики точності, збіжності та повноти.

4 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ КЛАСИФІКАЦІЇ СКАНОВАНИХ ДОКУМЕНТІВ

4.1 Огляд інструментів для реалізації програмного модуля системи класифікації сканованих документів

Для реалізації програмного модуля класифікації цифрових зображень документів було обрано мову програмування Python[20]. Python – високорівнева мова програмування, код якої є мінімалістичний у реалізації модулів та зручний для зчитування. Дана мова програмування включає велику кількість додаткових бібліотек, модулів програмування чи фреймворків, що дозволяють досягти бажаного кінцевого результатут у різних сферах: веб-розробка, аналіз та обробка даних, реалізації функції штучного інтелекту. Python може працювати з різними типами та структурами даних, таких як списки, словники, множини, чи удосконалені датафрейми та серії даних, тому це є ще одна з причин вибору саме цієї мови програмування, адже великий акцент даної роботи покладено на роботу з даними, зокрема з зображеннями. Розроблений код зручно представляти як окремі блоки, класи або функції, що додатково підкреслює читабельності коду, забезпечує модульну структурність коду та додаткову функцію стійкості до помилок.

Для розробки та тестування системи класифікації було обрано середовище розробки Jupyter Notebook. Середовище встановлено локально та підтримує хмарні сервіси, такі як Google Colab чи Microsoft Azure.

Модуль os[18]– модуль, що забезпечує реалізацію операційних запитів в системі: доступ до директорії та маніпулюванням шляхами доступу, зчитування, збереження чи редагування файлу за вказаною директоріє, створення тимчасових системних файлів чи перегляд системних каталогів. В роботі модуль використано для доступу до директорії з зображеннями.

Модуль random[20] – модуль для генерації випадкових чисел. В роботі використано для тасування вхідних даних, отримання випадкових прикладів для тестування класифікатора.

Зовнішня бібліотека Matplotlib[20]– стандартна біоліотека Python, що використовується для створення статистики на основі даних, візуалізацію даних чи анімацію змін. Крім раніше вказаних функції додатково містить функці для отримання та виведення файлів різних форматів, зокрема зображень: .png, .jpeg, .tif та інші. Дозволяє будувати двовимірні та тривимірні графіки.

Бібліотека Pillow[20] - бібліотека для роботи з зображеннями: зчитування, обробка, представлення, додаткова анімаці або передача зображення до інших бібліотек чи модульів.

Зовнішня бібліотека Pandas[21] – бібліотека для аналізу та швидкої обробки сирих даних. Пітримує два формати з позначками – серія та датафрейм. Бібліотека підтримує обробку одновимірних та двовимірних масивів даних. Наявна інтеграція до графічних бібліотек, що забезпечує додаткову можливість візуазізації результату.

Зовнішня бібліотека NumPy[22] – бібліотека, що забезпечує додаткову та швидку підтримку та обробку багатовимірних структур даних. Забезпечує додаткову векторизацію даних, індексацію даних. Від бібліотеки Pandas відрізняється великим набором математичних обчислень для різних структур даних.

Зовнішня бібліотека Scikit-learn[23] – бібліотека для машинного навчання, що включає в себе більшість стандартних алгоритмів машинного навчання, техніки оптимізації, техніки оцінки ефективності алгоритмів. Бібліотека також включає додаткові фунції обробки вхідних даних, функції роботи з тестовими та навчальними даними.

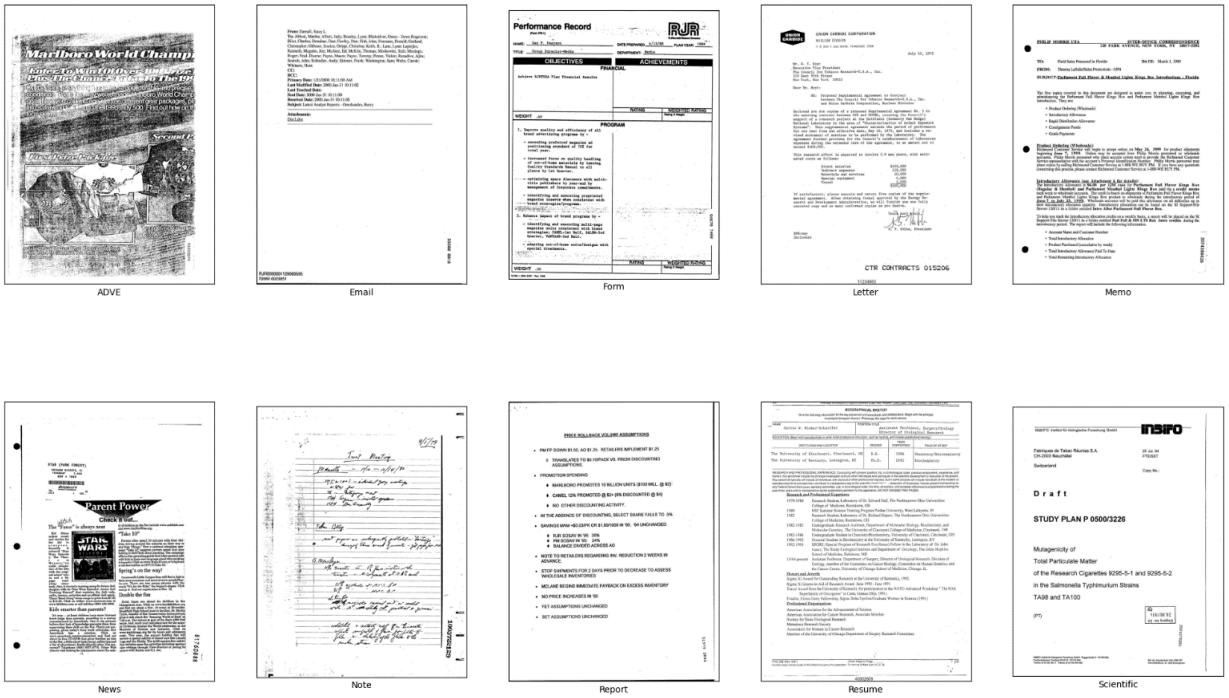
Фреймворк TensorFlow[24] - наскрізна платформа з відкритим доступом до використання для реалізації задач машинного навчання: класифікації, прогнозування, кластеризації, сегментації. Забезпечує ресурси для швидкої

побудови ефективних моделей машинного навчання з можливістю їх розгортання на зовнішніх платформах. Включає додаткові функції обробки даних.

Бібліотека Keras[25] – бібліотека відкритого коду, призначена для реалізації задач глибинного навчання – зокрема нейронними мережами різного типу. Бібліотека працює поверх фреймворку Tensor Flow. Додатково бібліотека підтримує функції підготовки вхідних даних різного роду, оптимізатори, метрик оцінки моделей та функції виведення результатів.

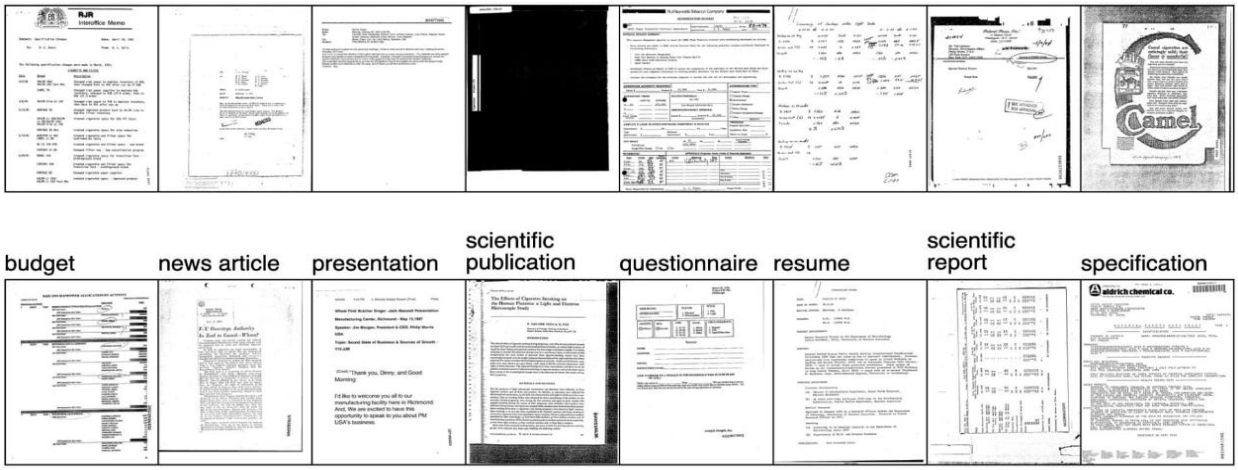
4.2 Опис набору даних

Для якісного навчання класифікатора цифрових зображень документів використано підготовлені набори даних у вигляді сховищ зображень сканованих документів. В ході роботи було використано набір даних Tobacco-3482[26], що складається з 3482 зображень документів різної роздільності зданості у форматі .jpg. Зображення представленні в чорно-білому форматі та розділені між десятьма класами, а саме: «лист», «форма», «електронний лист», «мемо», «новина», «нотанки на папері», «звіт», «резюме», «реклама», «науковий звіт» - рис 4.1.



Рисунку 4.1 –Приклад зображень кожного класу набору даних Tobacco-3482

Для надійності та підвищення ефективності мережі в процесі навчання було розглянуто інший набір зображень - RVL-CDIP (RyersonVisionLabComplexDocumentInformationProcessing) [27]. Даний набір даних складається з 400 000 зображень чорно-білого предстaлення, як і попередній набір, проте кількість класів збільшилась до 16. В наборі даних представлено 16 категорій: «лист», «меморандум», «реклама», «email», «структура папки» «бланк», «рукописний документ», «рахунок-фактура», «реклама», «бюджет», «стаття в журналі», «презентація», «наукова публікація», «анкета», «резюме», «науковий звіт» і «документ-опис» - рис 4.2.



Рисунку 4.2 – Зразок набору даних RVL-CDIP [27]

Оскільки основний набір даних для тренування моделі в даній роботі є Tobacco-3482, то з датасету RVL-CDIP використано тільки зображення класів, що перетинаються з класами Tobacco-3482.

Зображення кожного класу збережено в коремій директорії, відведеній саме для цього класу – рис.4.3

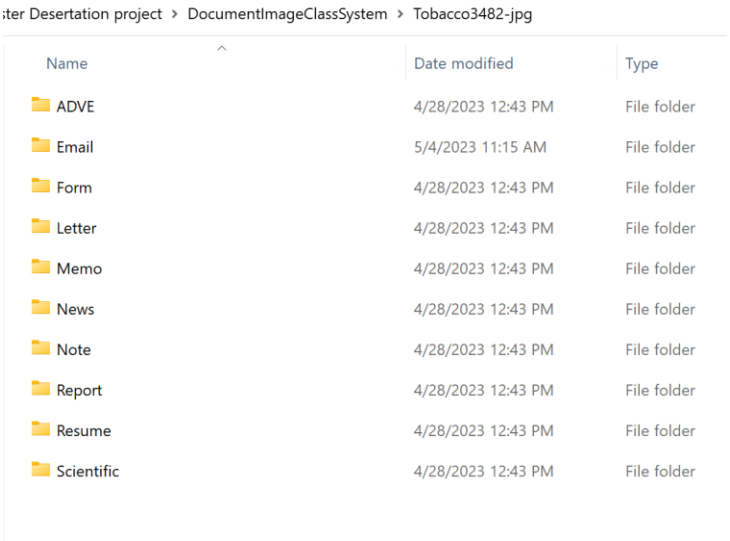


Рисунок 4.3 – Формат збереження зображень

Даний спосіб збереження зображень враховано при реалізації модулів зчитування зображень та класів.

4.3 Опис реалізованих функцій

Функція getAllDataClassesAndPaths() реалізована для отримання доступу до директорій кожного класу зображень. Як зазначалося вище усі вхідні дані зберігаються локально в різних папках, відповідно то дого, до якого класу належить зображення. На вхід функція приймає змінну, що зберігає шлях до загальної папки що є батькіською до папок-класів. Результатом роботи функції є список шляхів до кожної папки зображень окремого класу – рис 4.4 та список усіх можливих класів між якими розділені зображення документів – рис 4.5.

```
paths
```

```
['C:\\Users\\pashc\\OneDrive\\Documents\\Magister Desertation project\\DocumentImageClassSystem\\Tobacco3482-jpg\\ADVE',
 'C:\\Users\\pashc\\OneDrive\\Documents\\Magister Desertation project\\DocumentImageClassSystem\\Tobacco3482-jpg\\Email',
 'C:\\Users\\pashc\\OneDrive\\Documents\\Magister Desertation project\\DocumentImageClassSystem\\Tobacco3482-jpg\\Form',
 'C:\\Users\\pashc\\OneDrive\\Documents\\Magister Desertation project\\DocumentImageClassSystem\\Tobacco3482-jpg\\Letter',
 'C:\\Users\\pashc\\OneDrive\\Documents\\Magister Desertation project\\DocumentImageClassSystem\\Tobacco3482-jpg\\Memo',
 'C:\\Users\\pashc\\OneDrive\\Documents\\Magister Desertation project\\DocumentImageClassSystem\\Tobacco3482-jpg\\News',
 'C:\\Users\\pashc\\OneDrive\\Documents\\Magister Desertation project\\DocumentImageClassSystem\\Tobacco3482-jpg\\Note',
 'C:\\Users\\pashc\\OneDrive\\Documents\\Magister Desertation project\\DocumentImageClassSystem\\Tobacco3482-jpg\\Report',
 'C:\\Users\\pashc\\OneDrive\\Documents\\Magister Desertation project\\DocumentImageClassSystem\\Tobacco3482-jpg\\Resume',
 'C:\\Users\\pashc\\OneDrive\\Documents\\Magister Desertation project\\DocumentImageClassSystem\\Tobacco3482-jpg\\Scientific']
```

Рисунок 4.4 – Приклад результату роботи функції getAllDataClassesAndPaths()

```
: classes
```

```
: ['ADVE',
   'Email',
   'Form',
   'Letter',
   'Memo',
   'News',
   'Note',
   'Report',
   'Resume',
   'Scientific']
```

Рисунок 4.5 – Список усіх отриманих класів вхідних даних

Функція countSampleInClass() обчислює кількість прикладів, що входять до кожного класу. На вхід отримує список шляхів до директорій-класів.

Функція getEntityPathAndLabel() розроблена для об’єднання прикладів з усіх класів. Зображення різних класів зберігаються в різних папках, тому функція призначена для зчитування вміст кожної папки-класу та зберження результату в єдиний список. На вхід функція приймає список шляхів до кожної директиви окремого класу, результатом функції є обєднаний список шляхів усіх зображень. Додатково додано функцію рандомного тасування, тому шляхи предсталені в результуючому списку в випадковому порядку.

Функція getClass() повертає клас кожного зображення з загального списку. На вхід приймає список шляхів, отриманих в функції getEntityPathAndLabel() – рис. 4.5

```
#get examples of entity and related info
print(total_set[0])
print('GetClass : ', getClass(total_set[0]))
print('Label : ', total_labels[0])
```

```
C:\Users\pashc\OneDrive\Documents\Magister Desertation project\DocumentImageClassSystem\Tobacco3482-jpg\ADVE\0000136188.jpg
GetClass :  ADVE
Label :  ADVE
```

Рисунок 4.6 – Приклад результату роботи функцій getEntityPathAndLabel() та getClass() для зображення з індексом 0

Функція process_images() розроблена для попередньої обробки вхідих зображень. В функції реалізовано техніку зменшення розмірності зображення до роздільної здатності 224×224. Отримане зображення визначеної роздільної здатності трансформується в матрицю ознак та зберігається як список – рис 4.7.

```
x_train[0]
```

```
array([[[253, 253, 253],
        [254, 254, 254],
        [254, 254, 254],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       ...,

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[253, 253, 253],
        [254, 254, 254],
        [254, 254, 254],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[  0,   0,   0],
        [  0,   0,   0],
        [  0,   0,   0],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]]], dtype=uint8)
```

Рисунок 4.7 – Результат роботи process_images() функції

Функція train_test_split() з бібліотеки sklearn.model_selection використано для розділення датасету на навчальну та тренувальну вибірку. Вхідними параметрами функції є список усіх зображень в дата сеті, назви їх класів відповідно, параметр відношення тренувальних даних до тестових та значення рандомності. Результатом функції є 4 списки: список навчальних даних з відновідними лейбами та список тестових даних з відповідними до них лейбами – рис. 4.8.

```
#get exapmles of traint and test data with labels
print("Приклад даних в навчальній вибірці")
print(X_train_l[0])
print(y_train_l[0])
print("\nПриклад даних в тестовій вибірці")
print(X_test_l[0])
print(y_test_l[0])
```

```
Приклад даних в навчальній вибірці
C:\Users\pashc\OneDrive\Documents\Magister Desertation project\DocumentImageClassSystem\Tobacco3482-jpg\ADVE\503146051+-6051.jp
g
ADVE

Приклад даних в тестовій вибірці
C:\Users\pashc\OneDrive\Documents\Magister Desertation project\DocumentImageClassSystem\Tobacco3482-jpg\Note\2030105137.jpg
Note
```

Рисунок 4.8 – Приклад результату роботи функції train_test_split()

Вбудована функція .imshow() викоистовується для виведення зображення в двовимірному просторі – рис 4.9. Вхідним параметром функції є шлях до обраного обораження.

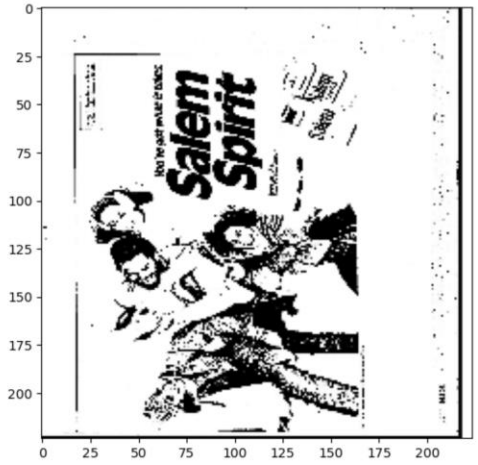


Рисунок 4.9 – Результат роботи функції .imshow()

Клас Sequential()[23] бібліотеки Keras використовується для підбору шарів, що далі будуть застосовані в моделі згорткової мережі. Аргументи в даному класі передаються послідовно. Для реалізації структури нейронної мережі підключаються додаткова функціональність - метод add() – додавання шару та метод pop() - виключення шарів з створеної структури. Результатом роботи класу є сукупність шарів мережі в заданій послідовності.

Вбудований модуль compile()[25] використовується для компіляції створеної моделі ЗНМ. Вхідними параметрами методу є гіперпараметр функції втрат, тип оптимізатора для розглядаємої мережі та типи метрик, які будуть використані при оцінці ефективності мережі протягом тренування. Клас повертає поністю труктруровану усіма шарими та гіпертапарметрами мережу, яка готово до тренування на навчальних даних.

Функція fit()[25] використовується для застовування створеної в попередніх функціях моделі до тренувальних даних. На вхід функції подають навчальні дані, а також гіперпараметр epochs - кількість епох, під час яких буде відбуватися навчання, batch_size – розмір кодного пакету даних, що подається на вхід, параметр validation_data - набір даних для тестування навченої моделі. Результатом функції є оновлена модель згорткової мережі з оновленими вагами протягом навчання. Для збереження натренованої моделі класифікації додатково використовується функція model.save().

Клас predict()[25] використовується для застосування навченої моделі класифікації до валідаційних даних або інших даних, які необхіно прокласифікувати.

Функція manualValidateModel() використовується для практичного аналізу отриманих результатів класифікації. На вхід функції подається список прокласифікованих зображень та список оригінальних зображень. Результатом функції є повідомлення з інформацією про дійсний клас зображення та прогнозований, і повідомлення чи результати співпадають: true – зображення прокласифіковано корентно, false – зображення прокласифіковано некоректно – рис. 4.10

```
Прогнозований клас зображення:  Note    |   Справжній клас зображення :  Note    |   Результат класифікації :  True
Прогнозований клас зображення:  Memo    |   Справжній клас зображення :  Memo    |   Результат класифікації :  True
Прогнозований клас зображення:  Form    |   Справжній клас зображення :  Form    |   Результат класифікації :  True
```

Рисунок 4.10 – Приклад результату робоботи функції manualValidateModel()

Функція CreateOrCheckDir() створена для перевірки наявності папки відповідного класу, до якого буде збережено прокласифіковане зображення. У разі відсутності відповідної директорії, функція створює нову під назвою необхідного класу. Вхідними параметрами фукнції є шлях до батьківської директорії та назва класу, з якою має бути створена або знайдена директорія.

Функція DocsClaffier() сворена для застосування моделі натреновного класифікатода до непрокласифікованого зображення. На вхід моделі подаються шлях до директорії, де знаходяться непрокласифіковані документи, та назва класифікатора. Після виконання даної функції зображення зберігаютсья у відповідну категоріюклас, до якого визначив класифікатор.

4.4 Навчання класифікатора у системі класифікації сканованих документів

4.4.1 Завантаження та підготовка вхідних даних

Вхідними даними системи класифікації є цифрові зображення документів (див. розділ 4.1). Навчання моделі класифікації проводиться за десятьма класами: «лист», «форма», «електронний лист», «мемо», «новина», «нотанки на папері», «звіт», «резюме», «реклама», «наукова робота». Проведено аналіз розподілу даних у вхідному датасеті (рис 4.11).

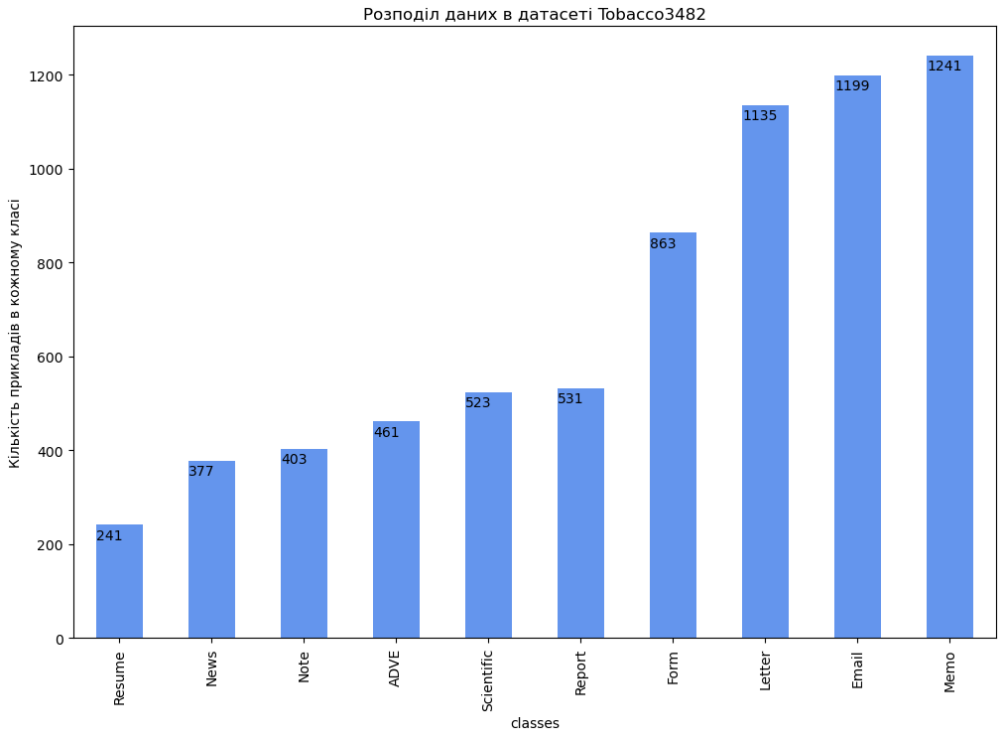


Рисунок 4.11 – Розподіл даних вхідного набору даних Tobacco-3482

З діаграми розподілу 4.11 видно, що найбільше прикладів – 1241 відноситься до класу «мемо», меншими класами є класи «електронний лист», «лист», «форма» з кількістю прикладів 1199, 1135, 863 відповідно. Класами, що місять найменшу кількість прикладів є класи «резюме» та «новини». Зображення зберігают в окремих директоріях призначених для конкретного класу. Попередньо проведено аналіз файлів в директоріях на наявність інших форматів, відмінних від .jpg. Отримано, що в розглянутих директоріях додатково знаходяться файли з розширенням .txt та файли бази даних – рис. 4.12.

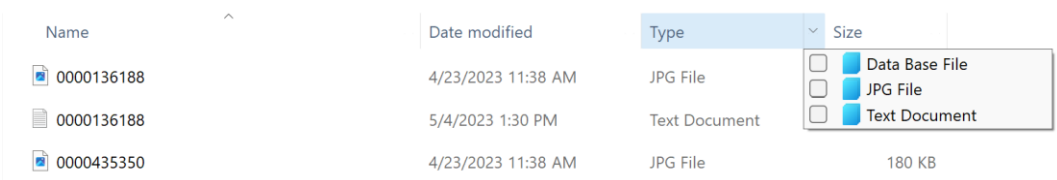


Рисунок 4.12 – Огляд формату файлів в директоріях тренувальних даних

При імпорті вхідних даних імпортуються лише файли формату .jpg. Отриманий датасет містить шляхи до зображень усіх класів та назву класу, до якого входить це зображення – рис 4.13. Окремо кожному існуючому класу зображень було присвоєно індекс – рис 4.14.

	image_path	image_name	in
0	C:\Users\pashc\OneDrive\Documents\Magister Des...	Scientific	
1	C:\Users\pashc\OneDrive\Documents\Magister Des...	Email	
2	C:\Users\pashc\OneDrive\Documents\Magister Des...	Memo	
3	C:\Users\pashc\OneDrive\Documents\Magister Des...	Letter	
4	C:\Users\pashc\OneDrive\Documents\Magister Des...	Memo	
...	...	...	
3477	C:\Users\pashc\OneDrive\Documents\Magister Des...	Email	
3478	C:\Users\pashc\OneDrive\Documents\Magister Des...	Resume	
3479	C:\Users\pashc\OneDrive\Documents\Magister Des...	Memo	
3480	C:\Users\pashc\OneDrive\Documents\Magister Des...	ADVE	
3481	C:\Users\pashc\OneDrive\Documents\Magister Des...	Email	

Рисунок 4.13 – Приклад вхідного набору даних

```
{'ADVE': 0,
 'Email': 1,
 'Form': 2,
 'Letter': 3,
 'Memo': 4,
 'News': 5,
 'Note': 6,
 'Report': 7,
 'Resume': 8,
 'Scientific': 9}
```

Рисунок 4.14 – Номерація існуючих класів

Список усіх шляхів до зображень розташовано в рандомній послідосності, що забезпечить оптимальний розподіл даних між навчальним та тестовим набором даних. Для тренування моделі зобржаження було транспортовано до матриці ознак. Кожному пікселю зображення було присвоєно значення в діапазоні від 0 до 255 – відповідно до кольорової гами.

Наступний етап підготовки вхідних даних - зменшення роздільної здатності зображення, оскільки мережа приймає на вхід дані сталої розмірності. Роздільна здатність вхідних зображень перевищує 1000 пікселів, тому зображення були стиснуті до розміру 224×224 – рис 4.15, 4.16.

Рисунок 4.15 – Початкове зображення. Роздільна здатність 1728×2292

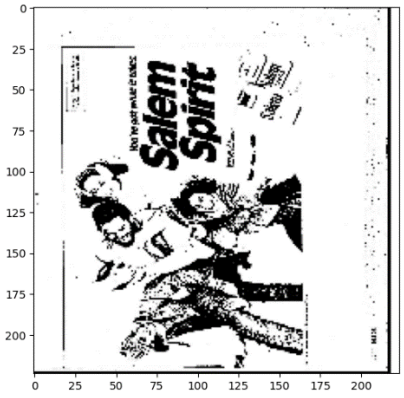


Рисунок 4.16 –Зменшене зображення. Роздільна здатність 224×224

Представлення зображення у вигляді матриці ознак – рис 4.17, 4.18. Розмірність двовимірного списку ознак – 224×224, кількість каналів входу – 3.

```
: print("Формат зображень навчальнох вибірки : ", x_train.shape)
  print("Розмір лейбів навчальної вибірки : ", y_train.shape)

Формат зображень навчальнох вибірки :  (2332, 224, 224, 3)
Розмір лейбів навчальної вибірки :  (2332, 10)
```

Рисунок 4.17 – Формат вхідних даних після транформації зображень

```
x_train[0]

array([[[253, 253, 253],
        [254, 254, 254],
        [254, 254, 254],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       ...,
```

Рисунок 4.18 – Приклад трансформації вхідного зображення до матриці ознак

4.4.2 Підготовка навчального та тестового наборів даних

Початоковий датасет був розділений на навчальну та тестову вибірку у відношенні 67% навчальних даних та 33% - тестових. Розподіли даних для навчального на тестового набору наведено на рис 4.19 та 4.20. Навчальна вибірка складається з 2332 приклади, до тестової – 1150.

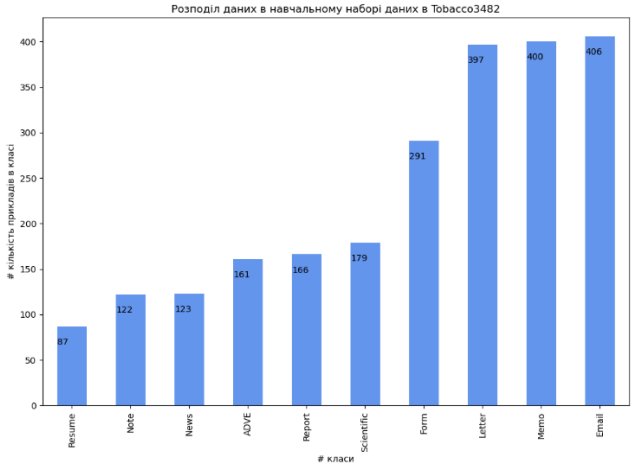


Рисунок 4.19 – Розподіл даних в навчальній вибірці

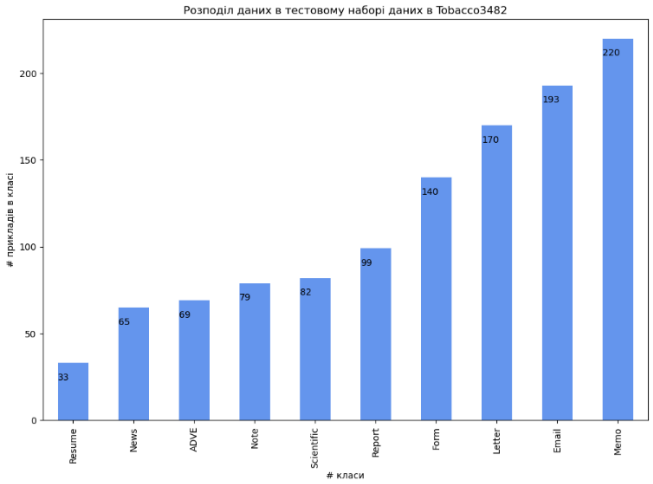


Рисунок 4.20 – Розподіл даних в тестовій вибірці

4.5 Реалізація системи класифікації цифрових зображень документів

Топологія спроектованої нейронної мережі включає 16 основних шарів – шари згортки та шари агрегування та додаткові шари з’єднання пакетів даних, оскільки дані для навчання подавалися на вхід мережі у вигляді окреми наборів даних – пакетів, шару активації з функцією активації Relu – рис 4.21. Додатково

для оптимізації роботи застововується техніка виключення вузлів мережі при повнозв’язних шарах та функцію активації софтмакс.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

Рисунок 4.21 – Топологія спроектованої згорткової мережі

На вхід моделі подаються дані з навчальної вибірки розмірністю (224, 224, 3). На виході модель мережі повертає список імовіртостей для кожного прикладу класифікації окремо по кожному класу. Клас, до якого належить поточний приклад, визначається відповідно до більшого значення імовірності. Для оптимізації навчання ЗНМ застосовано оптимізаційни алгоритм Adam – рис 4.22.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 flatten (Flatten)           (None, 25088)             0

 dense (Dense)               (None, 128)               3211392

 dropout (Dropout)           (None, 128)               0

 predictions (Dense)         (None, 61)                7869

=================================================================
Total params: 17,933,949
Trainable params: 17,933,949
Non-trainable params: 0
_________________________________________________________________
```

Рисунок 4.22 – Огляд додаткових шарів розробленої ЗНМ

4.6 Навчання класифікатора

Для отримання ефективної моделі класифікації було проведено декілька спроб навчання ЗНМ взалежності від кількості епох, розміру вхідних пакетів даних та значення гіперпараметрів.

Навчання моделі під час першої спроби тренування було проведено на 10 епохах. Гіперпараметр швидкості навчання(learning rate) становив 0.001. Кількість прикладів в одному вхідному пакеті даних станов 32. Для детальнішого аналізу побудовано графік залежності функції втрат для навчальних та тестових даних - рис 4.23 та графік залежності метрики точності класифікації для навчальних та тестових даних – рис 4.24.

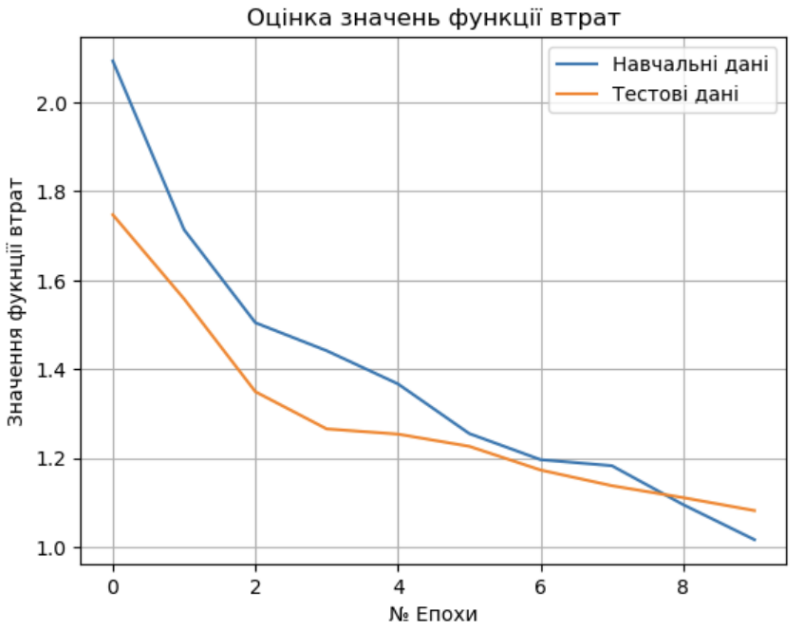


Рисунок 4.23 - Графік залежності значення функції втрат протягом навчання мережі в ході 1-го експерименту

Як видно з графіку 4.11 після 8-ї епохи навчання значення функції витрат для тестових даних перевищує значення функції витрат для навчальних даних, в свою чергу значення точності прогнозування на тестових даних менша – рис 4.12, ніж на навчальних, що свідчить про імовірність перенавчання моделі[28]. Причиною може бути велика швидкість навчання мережі, гіперпарамерт learning rate та складність топології моделі.

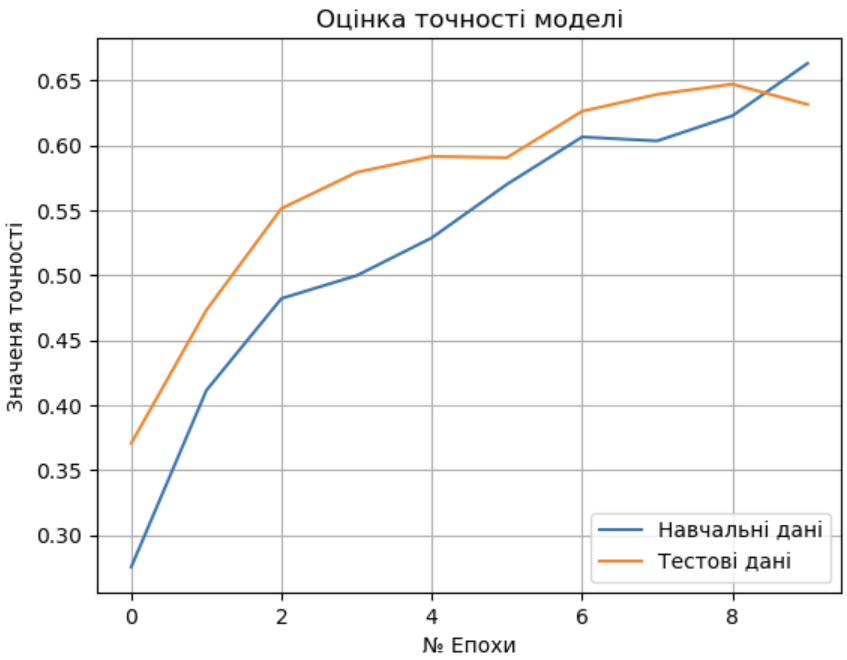


Рисунок 4.24 - Графік залежності значення метрики точності класифікації протягом навчання мережі в ході 1го експерименту

В результаті після першої спроби тренування моделі класифікації отримано модель з ефективністю класифікації 0.66391. Отриманий результат не відповідає очікуванням та показує низький результат на тестових даних.

Під час другої спроби навчання моделі кількість епох було більшено до 15, кількість прикладів в одному пакеті, що подається на вхід мережі було збільшено до 64, значення гіперпараметру навчання мережі було зменшено до 0.00001.

Для детальнішого аналізу побудовано графік залежності функції втрат для навчальних та тестових даних - рис 4.25 та графік залежності метрики точності класифікації для навчальних та тестових даних – рис 4.26.

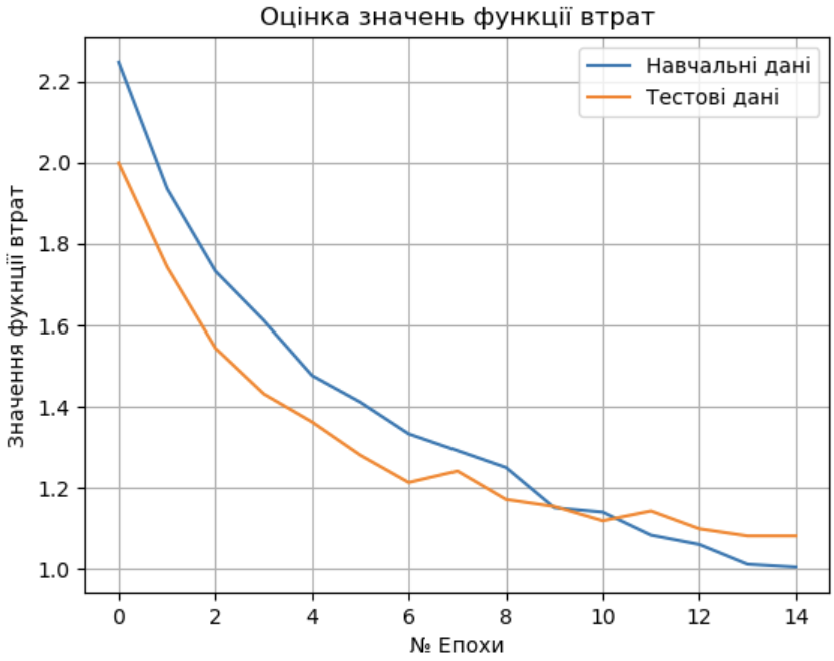


Рисунок 4.25 - Графік залежності значення функції витрат протягом навчання мережі в ході 2-го експерименту

З графіку 4.25 видно, як і в попереньому випадку значення фукнції втрат для тестових даних, більша за значення тієї ж функції на навчальних даних. Проте аналізуючи графік залежності значень точностей , точності класифікації моделі для навчальних та тестових даних знаходяться в близькому діапазоні, тому модель показує однаковий результат при навчанні та тестуванні.

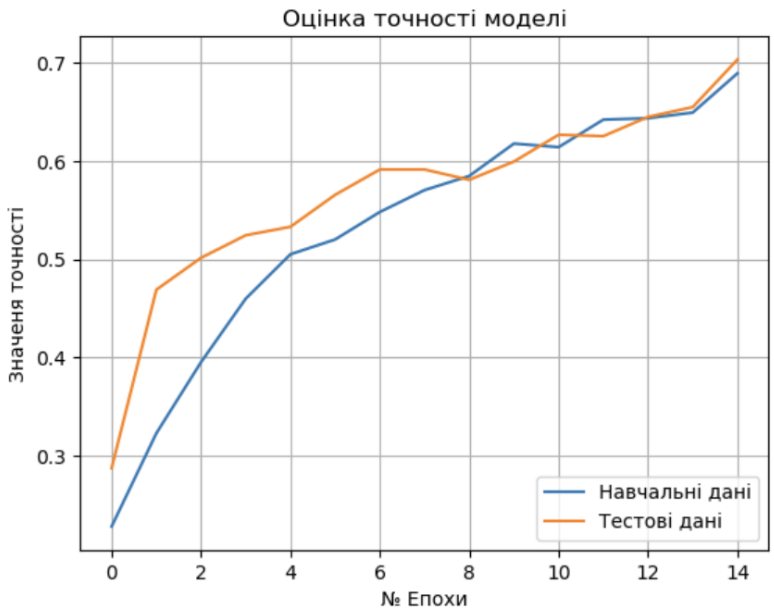


Рисунок 4.26 - Графік залежності значення метрики точності класифікації протягом навчання мережі в ході 2-го експерименту

Ефективність моделі класифікації після навчання в другому випадку становить 0.6994. Даний результат є на порядок вище за попередній результат, модель не піддається до перенавчання, проте отриманий результат не є очікуваним результатом прогнозування.

В попередніх випадках було знайдено підходяще значення кількості прикладів в пакеті даних, що подається на вхід моделі та значення гіперпаратемну learning_rate, проте для досягнення кращогого значення точності мережі проводиться наступна спроба навчання на 20-ти епохах. Кількість прикладів в одному пакеті, що подається на вхід мережі - 64, значення гіперпараметру learning_rate - 0.00001.

Детальний аналіз можна зробити на основі побудованих графіку залежності функції втрат для навчальних та тестових даних - рис 4.27 та графіку залежності метрики точності класифікації для навчальних та тестових даних – рис 4.28.

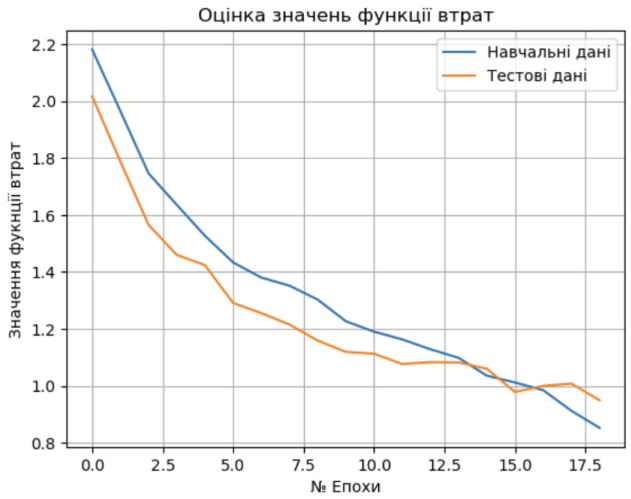


Рисунок 4.27 - Графік залежності значення функції витрат протягом навчання мережі в ході 3-го експерименту

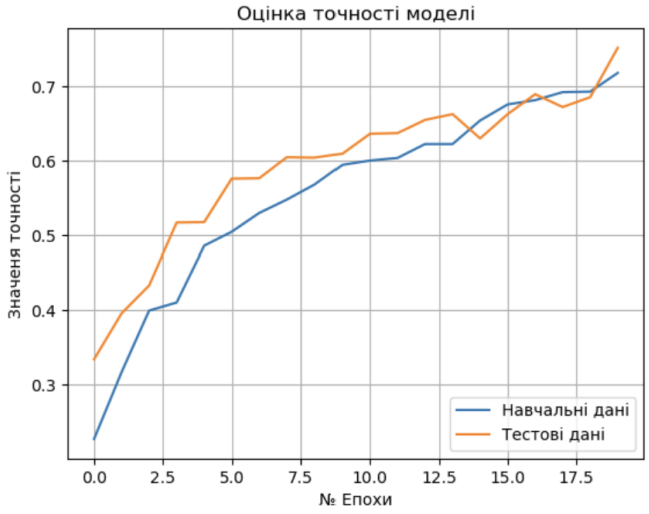


Рисунок 4.28 - Графік залежності значення метрики точності класифікації пртягом навчання мережі в ході 3-го експерименту

Ефективність моделі класифікації після навчання в третьому випадку становить 0.8034. Отримане значення точності класифікації, задовольняє очікуваному результату.

4.7 Метрики оцінки ефективності системи класифікації

Відповідно до результатів роботи системи класифікації сканованих документів розраховано ряд метрик для кращої оцінки точності системи. Розраховано значення метрики збіжності, метрики, що оцінює кількість прикладів, які прокласифіковано до певного класу, до загальної кількості прикладів в розглянутому класі – рис 4.29.

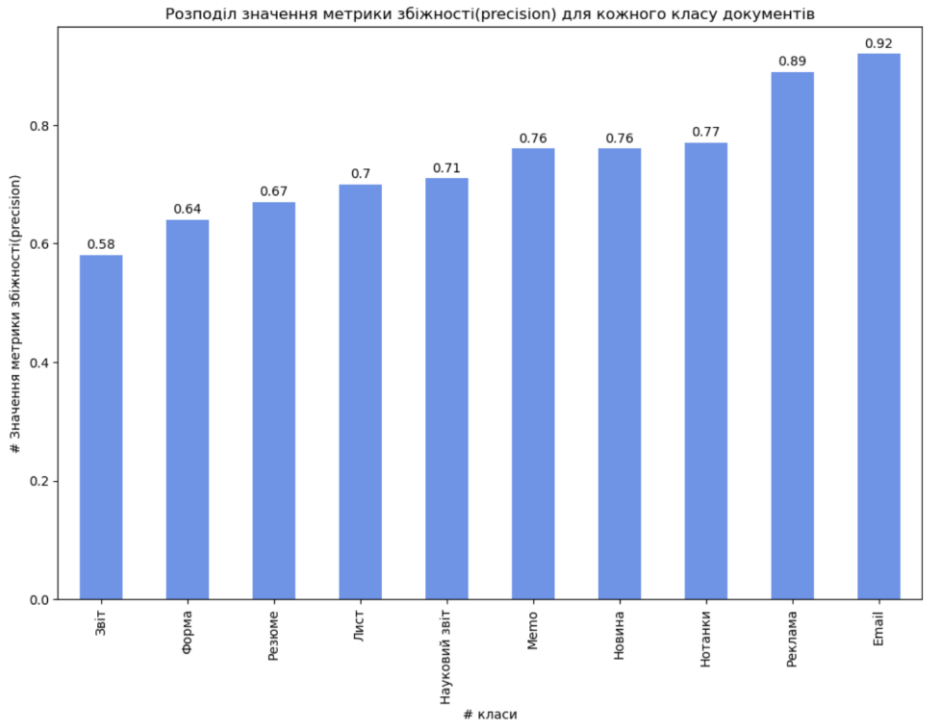


Рисунок 4.29 – Розподіл значень метрики збіжності класифікації

Як видно з діаграми 4.29 найгірше було прокласифіковано приклади з класу «Звіт», значення збіжносіт становить 0.58. Найкраще за метрикою збіжності було прокласифіковано декілька класів: «Реклама» та «Email», де значення збіжності становить 0.89 та 0.92 відповідно.

Для уточнення попередньої метрики було розраховано значення метрики повноти(recall) по кожному класу документів. В даній метриці проводиться оцінка кількості прикладів, які було прокласифіквано правильно в поточному класі до загальної кількості прикладів потосного класу - рис. 4.30

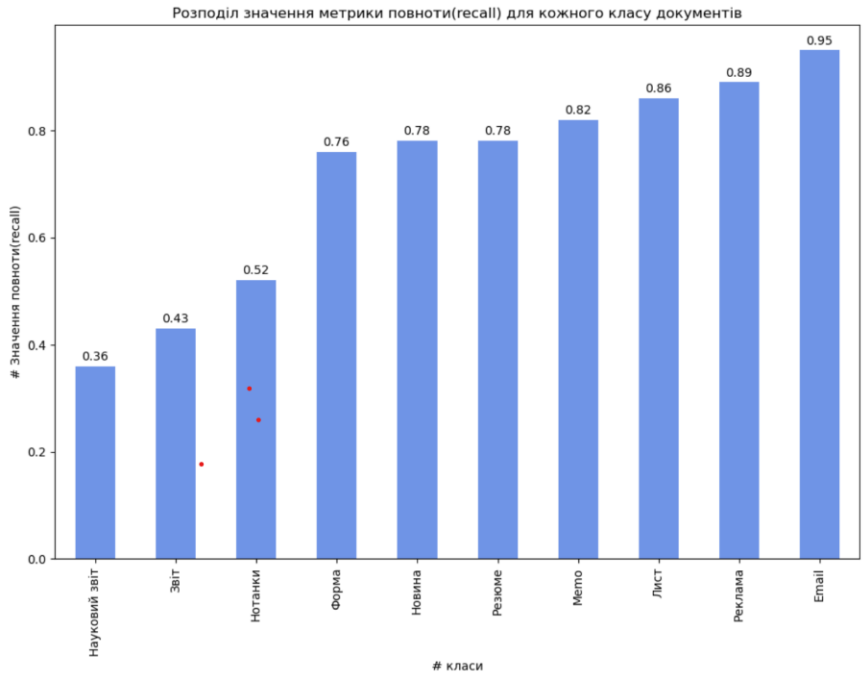


Рисунок 4.30 – Розподіл метрики повноти класифікації

З діаграми 4.30 видно, що найгірше було прокласифіковано класи «Науковий звіт», «Звіт» та «Нотанки», де значення повноти класифікації становить 0.36, 0.43 та 0.52 відповідно. Натомість класи «Реклама» та «Email» мають найвищі показники, як і в попередньому випадку оцінки.

Як гармонійне середнє між збіжністю та точністю класифікації було розраховано значення метрики F1-оцінка – рис 4.31

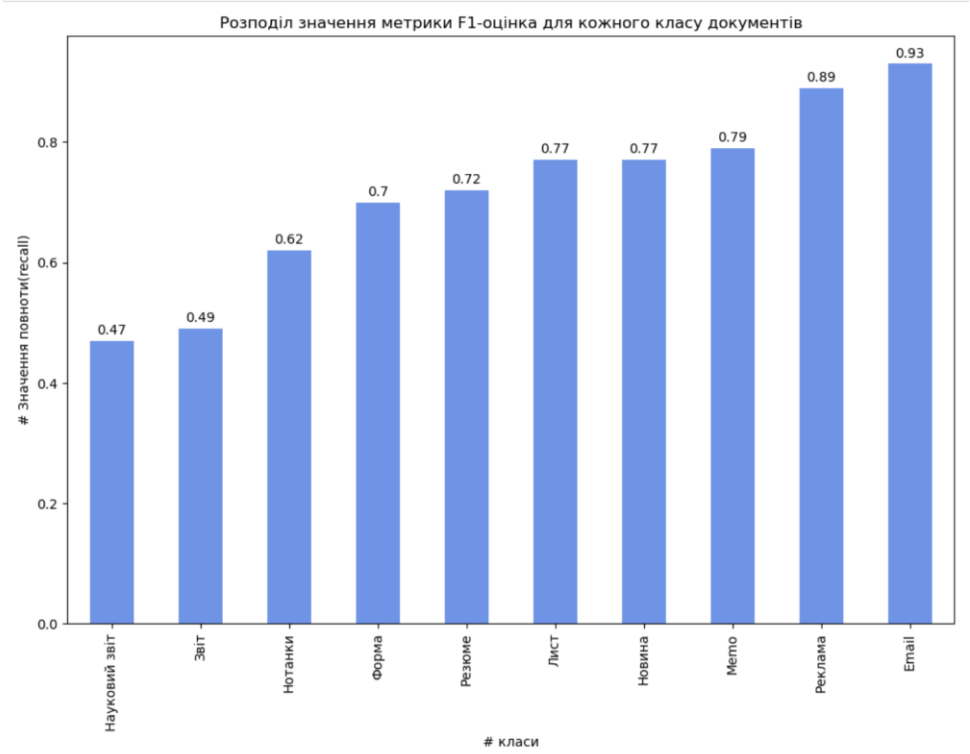


Рисунок 4.31 – Розподіл значення метрики F1-оцінка по кожному класу

Відповідно до діаграми 4.31 аналогічно до попередніх оцінок найгірний результат отримано для класів «Науковий звіт» та «Звіт» - 0.47 та 0.49, найкращі показники класифікації мають класи - «Реклама» та «Email».

Для детального аналізу роботи системи, побудовано матрицю помилок – рис 4.32, де вказана точна кількіть тестових прикладів, що були прокласифіковано до кожного класу. Також під кожним кількісним значенням наведено відсоткове відношення прокласифікованих прикладів.

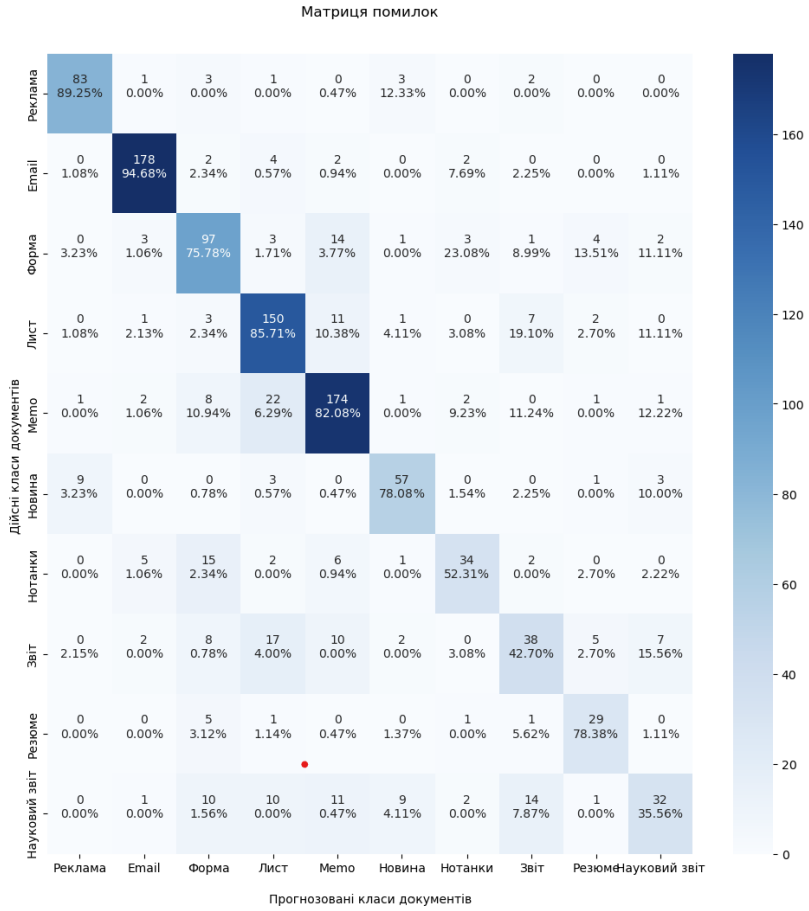


Рисунок 4.32 – Матриця помилок

Для оцінки ефективності роботи системи класифікації вцілому було розраховано точність - accuracy класифікації за всіма розглянутими класами – рис 4.33.

	precision	recall	f1-score	support
Реклама	0.89	0.89	0.89	93
Email	0.92	0.95	0.93	188
Форма	0.64	0.76	0.70	128
Лист	0.70	0.86	0.77	175
Memo	0.76	0.82	0.79	212
Новина	0.76	0.78	0.77	73
Нотанки	0.77	0.52	0.62	65
Звіт	0.58	0.43	0.49	89
Резюме	0.67	0.78	0.72	37
Науковий звіт	0.71	0.36	0.47	90
accuracy			0.76	1150
macro avg	0.74	0.71	0.72	1150
weighted avg	0.76	0.76	0.75	1150

Рисунок 4.33 – Порівняльна шкала розразованих метрик оцінки роботи системи

Згідно з 4.33 найгірші результати за всіма оцінками системи класифікації показала при класифікації документів з класів «Науковий звіт», «Звіт» та «Резюме». Значення точності класифікації даних класів варіюється від 0.49 до 0.72 за середнім гармонійним значенням між повнотою та збіжністю. Причиною отриманих низьких результів класифікаці може виступати те, що структура документів вище вказаних класів схожа між собою і представлена у вигляді набору тексту. Додаткові таблиці, поля чи зображення відсутні в даних класах, що ускладнює процес розрізнення. Як видно з матриці помилок, деякі приклади класу «Науковий звіт» були прокласифіковані як «Звіт», і навпаки. Отже, скановані документи вище вказаних проблемних класів можуть бути подані на подальший розгляд, з метою додавання додаткових позначок розрізнення або застосування для них алгоритмів ком.ютерного зору з метою аналізу змісту документу, а не лише його макету.

Найкраще система прокласифікувала документи з класів «Реклама» та «Email» - 0.89 та 0.93 за гармонійним середнім між повнотою та збіжністю. Оскільки документи даних класів маю чітке представлення макету документу – зображення з коротким описом, як у випадку класу «Реклама», чи розділення тексту на привітання, основну частину та підпис у випадку «Email». Ефективніть роботи системи класифікації за іншими класами документів таносить 0.70 – для касу «Форма», 0.77 – клас «Лист», 0.79 – клас «Мемо», 0.77 – клас «Новина», 0.62 – клас «Нотатки».

Ефективність роботи системи класифікації за всіма класами документів оцінено в 0.76, що задовольняє вимогам системи, а саме – ефективність системи класифікації має бути в межах 70% - 90%.

4.7 Висновки до розділу

В розділі було описано систему класифікації з огляду на розробку програмного забезпечення. Наведено етапи реалізації програмного модулю, основні інстументи розробки, мову програмування та середовище розробки.

Наведено опис розроблених функцій для завантаження та обробки вхідних даних, розробки класифікатора, збереження отриманих результатів. Проведено аналіз вхідного набору даних на загальну кількість класів в наборі, кількість прикладів в кожному з розглянутих класів, особливості предсталення вхідних даних в системі та розділення вхідного набору на тренувальну та тестову частини.

Проведено аналіз та порівняння результатів роботи розробленого програмного забезпечення системи класифікації в залежності від типу метрики оцінки якості ефективності. Проаналізовано отриманий результат на тестових даних та визначено основні складності в реалізації системи класифікації.

5 ВЕРИФІКАЦІЯ ТА ВАЛІДАЦІЯ

5.1 Функціональність та область застосування

Метою даної роботи було підвищення ефективності роботи в завданнях класифікації та систематизації сканованих документів шляхом розробки системи класифікації на основі згорткової нейронної мережі.

Функціональність розробленої системи пвідповідає поставленим вимогам, а саме:

- система дозволяє завантажувати зображення формату .jpeg, .tiff, .png;
- система обробляє вхідні зображення високої роздільної здатності та трансформує їх в вектор сталої довжини розмірністю 224*224;
- система класифікує скановані документи за наступними категоріями: «лист», «форма», «електронний лист», «Мемо», «новина», «нотанки на папері», «звіт», «резюме», «реклама», «наукова робота»;
- ефективінсть системи класифікації становить 76% , що задовольняє початковій вимозі системи класифікації з ефективністю не нижче 70-90%;
- користувач має змогу завантажити дані для класифікації з локальної директорії. Результат роботи системи класифікації зберігається в указаній користувачем локальній директорії у відповідних папках-класах.

Розроблена система класифікації сканованих докуметів складається з наступних компонентів: збору та збереження даних, обробки вхідних даних, навчання та тестування класифікатора, класифікатор.

Досягнення кінцевого результатут як було зазначено у вимогах до ситеми було здійснення за допомогою реалізації алгоритму машинного навчання – згорткої нейронної мережі з застосуванням додаткових технік та оптимізаторів для підвищення еефктивності моделі.

Розроблена система може бути ефективно використана в практичних задачах систематизації, сортуванні чи класифікації сканованих та сфотографованих документів. Також система може бути інтегорованою в інші системи, що працюють з завчаннями вилучення інформації з конкретного типу сканованих документів.

5.2 Приклад роботи системи класифікації сканованих документів

Практичне призначення розробленої системи класифікації сканованих документів є підвищення ефективності роботи в задачах класифікації та систематизації документів. В більшості випадків на практиці одночано обробляється певний набір документів, тому система може ефективно класифікувати відразу декілька сканованих документів, однак ефективіність роботи не знизиться і у випадку роботи з одиничним сканованих документом.

Для початку роботи з системою користувачу потрібно вказати шлях до директорії, де знаходиться набір непрокласифікованих сканованих докмуентів – рис 5.1.

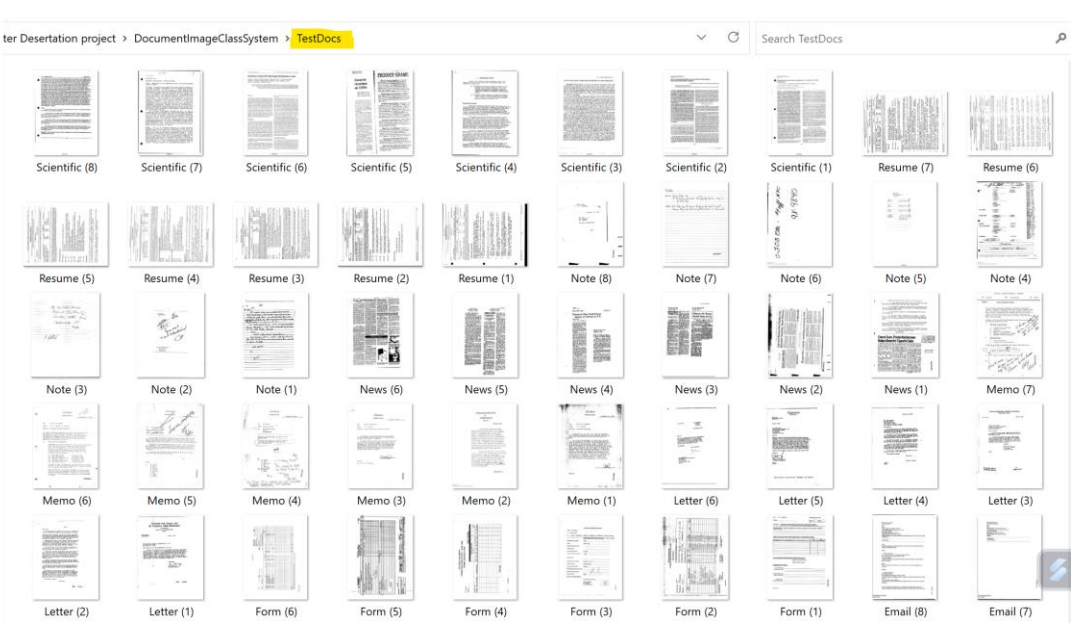


Рисунок 5.1 – Приклад директорії з непрокласифікованими сканованими документами

Додатково потрібно вказати назву директорії, де б користувач хотів мати папки з прокласифікованими документами. В результаті запуску системи в сказаній директорії створюються папки з назвами класів документів, прокласифіковані документи зберігаються у підповідній папці – рис.5.2.

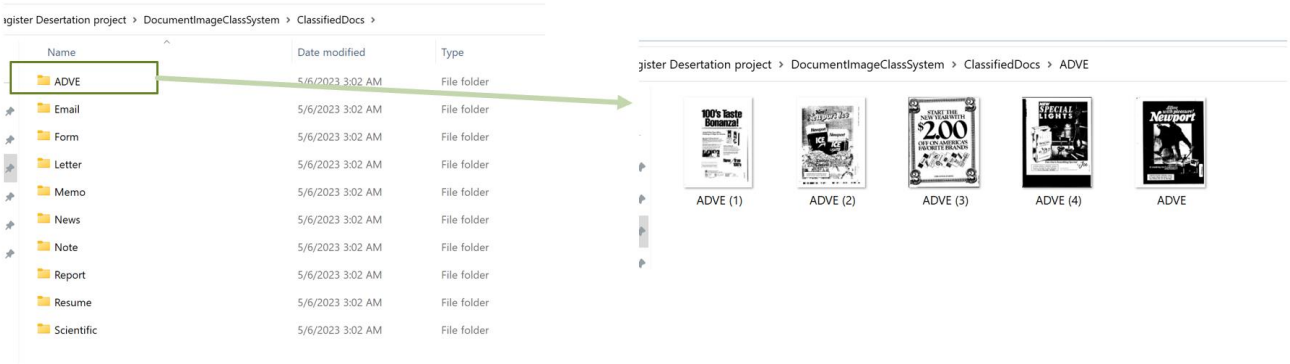


Рисунок 5.2 – Результат роботи розробленої системи класифікації

5.3 Висновки до розділу

В даному розділі було проаналізовано актуальність розробленої системи класифікації сканвоаних документів та вказано напрями, де може застосовуватися розроблена система. Проведено огляд отриманих результатів відповідно до розроблених вимог. Наведено приклад роботи системи непокласифікованих сканвоаних документах.

ВИСНОВКИ

У даній магістерській дисертаційній роботі отримано наступні нові теоретичні та практичні результати:

1. Досліджено існуючі математичні підходи та програмні засоби, що розроблені для завдань багатокласової класифікації зображень, зокрема це згорткові нейронні мережі типу LeNet, VGGNet, Google Net, а також розроблені готові рішення проектування нейронних мереж Neuroph, Simbrain, Encog. В ході аналізу було встановлено, що наведені вище методи та практичні рішення не включають додаткові налаштування для класифікації саме зображень сканованих документів, або якщо і включають, то процес класифікації документу вимагає додаткових ресурсів та є дороговартісними. На основі проведеного дослідження було встановлено, що найкращим варіантом для проектування системи класифікації сканованих документів є модель згорткової нейронної мережі з урахуванням параметрів для класифікації документу за макетом, методі попередної обробки даних: білінійна інтерполяція для зменшення розмірності зображення та методи перевірки ефективності роботи ситеми, такі як матриця помилок.

2. Розроблено компоненту модель системи класифікації сканованих документів, яка складається з шести основних компонентів. Проаналізовано призначення кожного з компонентів, вхідні дані та результат роботи компонентів, взаємозв’язок компонентів системи.

3. Підібрано та розроблено математичне забезпечення системи класифікації сканованих документів. Для досягнення посталеної задачі було підібрано методи підготовки вхідних даних: техніка трансформації вхідного зображення у вектор ознак, техніка зменшення розмірності вхідного зображення, методи нормалізації даних. Описано архітектуру згорткової нейронної мережі, включаючи математичне забезпечення шарів мережі та додаткові техніки

оптимізації навчання мережі. Наведено ряд метрик оцінки точності системи класифікації – точність, повнота, гармонійне середнє в макро та мікро представленнях, матрицю помилок.

4. Розроблено програмне забезпечення системи класифікації сканованих документів, використовуючи мову програмування Python у поєднанні із вбудованими та сторонніми бібліотеками: skilit-learn, pandas, NumPy, matplotlib. Модель згорткової нейронної мережі, додаткові техніки оптимізації навчання мережі та підбір гіперпараметрів мережі було спроектовано за допомогою сторонніх фреймворків мови програмування Python – TensorFlow. Розроблено програмний модуль, що дає можливість використовувати систему для класифікації сканованих документів з локальної директорії користувача з подальшою можливістю збереження результатів в обрану користувачем директорію. Розроблений програмний мобудь забезпечує класифікацію як набору документів, так і одиничного прикладу сканованого документу.

5. Розраховано значення ефективності та точності класифікації системи в залежності від гіперпараметрів згорткової нейронної мережі, а саме: кількість епох навчання, значення гіперпараметру навчання мережі та значення кількості прикладів в пакеті даних, що подається на всіх мережі. Отримані результати точності класифікації предсталено у вигляді стовпчикових діаграм з розподілом значення точності класифікації кожного класу сканованих документів, а також побудовано матрицю помилок з кількісними та відсотковими значеннями прикладів, що прокласифікавона коректно та некоректно. В результаті при наступних значеннях гіперпараметрів мережі: 20 епоха навчання, 64 - кількість прикладів в одному пакеті, що подається на вхід мереж, значення гіперпараметру learning_rate - 0.00001, - було отримано класифікатор з точністю роботи 76% в сукупності по всім класам сканованих документів. При аналізі роботи системи класифікації за кожним класом отримано - найточніше було прокласифіковано класи «Реклама» та «Email» з

точносітю 89% та 93%, найгірше було прокласифіковано документи класів «Науковий звіт» та «Звіт» з точністю 50% в обох випадках.

В ході аналізу подальших досліджень та способів інтеграції розробленої системи класифікації сканованих документів можливі наступні варіанти дослідження:

- застосування додаткових методів класифікації документів на основі аналізу змісту самого документу для класів «Науковий звіт» та «Звіт» для підвищення точності класифікації документів, що є схожими за макетом та не мають виразних об’єктів у структурі;
- додавання нових класів сканованих документів для розширення області застосування системи класифікації сканованих документів;
- збільшення додаткових ресурсів апаратного забезпечення для підвищення швидкості класифікації на великих наборах сканованих документів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Nawei C. A survey of document image classification: problem statement, classifier architecture and performance evaluation / C. Nawei, B. Dorothea. // International Journal on Document Analysis and Recognition (IJDAR). – 2007. – Vol.6 - pp. 1–16.
2. Le Kang. Convolutional Neural Networks for Document Image Classification / Le Kang, K. Jayant, Peng Ye. // International Conference on Pattern Recognition. – 2014. –Vol.3, №22. – С. 1–7.
3. Convolutional Neural Networks - LeNet [Електронний ресурс] – Режим доступу до ресурсу: https://d2l.djl.ai/chapter_convolutional-neural-networks/lenet.html. – (20.04.2023)
4. VGG-16 convolutional neural network [Електронний ресурс] – Режим доступу до ресурсу: https://www.mathworks.com/help/deeplearning/ref/vgg16.html. - (20.04.2023)
5. GoogLeNet convolutional neural network [Електронний ресурс] – Режим доступу до ресурсу: https://www.mathworks.com/help/deeplearning/ref/vgg16.html. - (20.04.2023)
6. Neuroph sourceforge documentation [Електронний ресурс] – Режим доступу до ресурсу: https://neuroph.sourceforge.net/ - (20.04.2023)
7. SIMBRAIN [Електронний ресурс] – Режим доступу до ресурсу: https://simbrain.net/ - (20.04.2023)
8. Encog Machine Learning Framework [Електронний ресурс] – Режим доступу до ресурсу: https://www.heatonresearch.com/encog/ - - (20.04.2023)
9. Норкін Б. М., Пащенко К.М., Математичне та програмне забезпечення системи класифікації цифрових зображень текстових документів. Прикладна

математика та комп'ютинг. ПМК-2022: п'ятнадцята науково-практична конференція магістрантів та аспірантів, Київ, 16-17 лист. 2022 р.: зб. Тез доп./ [редкол.: Дичка І. А. та ін.]. — К. : Просвіта, 2022. — С. 86-93.

10. Efficient Processing of Deep Neural Networks / Vivienne Sze, Yu-Hsin Chen, Joel S. Emer, Tien-Ju Yang. – Massachusetts: Morgan & Claypool Publishers, 2020. – 341 с.
11. Ian G. Deep Learning / G. Ian, B. Yoshua, C. Aaron. – Massachusetts: Massachusetts Institute of Technology, 2016. – 802 с.
12. Yuhan Bai. RELU-Function and Derived Function Review / Yuhan Bai. // 2022 International Conference on Science and Technology Ethics and Human Futur. – 2022. - Vol. 144 – С. 1–7.
13. Aszemi N. Hyperparameter Optimization in Convolutional Neural Network using Genetic Algorithms. / Dominic P. // International Journal of Advanced Computer Science and Applications. - 2019. - Vol.10, no. 6. - С. 1–10.
14. Loffe S. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift / S. Loffe, C. Szegedy. // arXiv:1502.03167. – 2015. – С. 1–11.
15. Kingma D. Adam: A Method for Stochastic Optimization / D. Kingma, B. Jimmy. // International Conference for Learning Representations. – 2017. – №3. – С. 1–15.
16. Schmidt R. M. Recurrent Neural Networks (RNNs): A gentle Introduction and Overview // Computer Science. - 2019. - С. 1–11.
17. Nixon M. Feature Extraction and Image Processing for Computer Vision / M. Nixon, A. Aguado. – London: Replika Press Pvt Ltd, 2013. – 609 с. – (3)
18. A. Semma et al. WRITER IDENTIFICATION: THE EFFECT OF IMAGE RESIZING ON CNN PERFORMANCE / The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. 2021. XLVI-4/W5 - 2021. P. 501–507

19. Grandini M. METRICS FOR MULTI-CLASS CLASSIFICATION: AN OVERVIEW / M. Grandini, B. Enrico, V. Giorgio. // arXiv:2008.05756 – 2020. – С. 1–17.
20. Python [Електронний ресурс] – Режим доступу до ресурсу: https://www.python.org/. - (30.04.2023)
21. Pandas documentation [Електронний ресурс] – Режим доступу до ресурсу: https://pandas.pydata.org/.
22. NumPy documentation [Електронний ресурс] – Режим доступу до ресурсу: https://numpy.org/doc/stable/. - (30.04.2023)
23. Scikit-learn [Електронний ресурс] – Режим доступу до ресурсу: https://scikit-learn.org/stable/. - (30.04.2023)
24. Tensorflow [Електронний ресурс] – Режим доступу до ресурсу: https://www.tensorflow.org/learn. - (30.04.2023)
25. Keras [Електронний ресурс] – Режим доступу до ресурсу: https://keras.io/.
26. Tobacco - 3482 DataSet [Електронний ресурс] – Режим доступу до ресурсу: https://www.kaggle.com/datasets/patrickaudriaz/tobacco3482jpg. - - (30.04.2023)
27. RVL-CDIP-I Dataset [Електронний ресурс] – Режим доступу до ресурсу: https://www.kaggle.com/datasets/nbhativp/first-half-training. - - (30.04.2023)
28. Fabien A. Deep Convolutional Neural Networks for Document Classification / A. Fabien, B. Montrucchio. – Turin: POLITECNICO DI TORINO. – 110 с. – (2018).

Додаток А
Лістинг коду mainClassify.py

```
# ### Import modules and librarius

# In[170]:


import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
import random
#import sys
#import gc
get_ipython().run_line_magic('matplotlib', 'inline')
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from sklearn.model_selection import train_test_split

import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelBinarizer
import PIL
from PIL import Image
import keras
from keras import layers
from keras import metrics
from keras.models import load_model
from keras.layers import Dense, Flatten, Conv2D, Dropout, MaxPooling2D, GlobalAveragePooling2D, Dropout
from keras import optimizers
from keras import models
from keras.models import Sequential
from keras import preprocessing
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
from keras.applications import VGG16
from keras.utils import plot_model
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
d = 'C:\\Users\\pashc\\OneDrive\\Documents\\Magister Desertation project\\DocumentImageClassSystem\\Tobacco3482-
jpg\\'
PATH = '../'
#fucntion to get oath to all possible classes of data
def getAllDataClassesAndPaths(d):
    classes = (os.listdir(d))
    paths = [os.path.join(d, o) for o in os.listdir(d)
                    if os.path.isdir(os.path.join(d,o))]
    return classes, paths
classes, paths = getAllDataClassesAndPaths(d)
countSampleInClass(classes, paths):
    nbEntries = []
```

```
    for i in range(len(classes)):
        nbEntries.append(len(os.listdir(paths[i])))

    return nbEntries
nbEntries = countSampleInClass(classes, paths)
df = pd.DataFrame({'classes':classes, 'Entries':nbEntries})
ax = df.sort_values(by='Entries', ascending=True).plot.bar(x='classes', y='Entries', color='cornflowerblue',legend=False,
figsize=(12,8))
ax.set_title('Розподіл даних в датасеті Tobacco3482')
ax.set_ylabel("Кількість прикладів в кожному класі")
for p in ax.patches:
    ax.annotate(str(p.get_height()), xy=(p.get_x(), p.get_height()-30))
#get access to all samples and collect data
def getEntityPathAndLabel(d):
    total_set = []
    total_labels = []
    for root, dirs, files in os.walk(d):
        for file in files:
            if file.endswith(".jpg"):
                path = os.path.join(root, file)
                total_set.append(path)
                total_labels.append(root.split(os.path.sep)[-1])
    return total_set, total_labels
total_set, total_labels = getEntityPathAndLabel(d)

# Return image class based on list entry (path)
def getClass(img):
    return img.split(os.path.sep)[-2]

#get examples of entity and related info
#visualize input data
random.Random(seed).shuffle(total_set)

for ima in total_set[0:3] :
    print(ima)
    img = mpimg.imread(ima)
    plt.figure(figsize=(7,7))
    imgplot = plt.imshow(img, cmap="gray")
    plt.show()

def get_file_paths_and_labels(data_root):
    """
    Returns a list of paths to image and text files, corresonding
    class labels and mapping of class names to label index.
    """
    image_paths = sorted([str(path) for path in data_root.glob('*/*.jpg')])#return path according to pattern in gaps
    random.shuffle(image_paths)

    text_paths = []
    for path in image_paths:
        txt_path = os.path.splitext(path)[0] + '.txt'
        with open(txt_path, 'w+') as fp:
            text_paths.append(txt_path)
```

```
    label_names = sorted(item.name for item in data_root.glob('*/') if item.is_dir())#checking if item is present in directory
                                                                #return label of images group
    label_to_index = dict((name, index) for index, name in enumerate(label_names)) #return dict of numenate labels

    labels = [label_to_index[pathlib.Path(path).parent.name] for path in image_paths]

    #image_names = [ for img_name, img_ind in labels]
    image_names = []
    for label_id in labels:
        for img_name, img_id in label_to_index.items():
            if img_id == label_id:
                image_names.append(img_name)

    return image_paths, text_paths, labels, image_names, label_to_index

## import data
data_root = pathlib.Path('C:/Users/pashc/OneDrive/Documents/Magister Desertation
project/DocumentImageClassSystem/Tobacco3482-jpg/')

image_paths, text_paths, labels, image_names, label_to_index = get_file_paths_and_labels(data_root)

data = {'image_path': image_paths, 'image_name': image_names, 'index_label': labels}
dfinput = pd.DataFrame(data)

print(label_to_index)

#get test and train set
X_train, X_test, y_train, y_test = train_test_split(dfinput['image_path'], dfinput['image_name'], test_size=0.33,
random_state=123)
X_train_l = X_train.tolist()
y_train_l = y_train.tolist()
X_test_l = X_test.tolist()
y_test_l = y_test.tolist()

#Visualize classes distribution (bar chart)
# TRAIN SET
instances = [0] * len(classes)
for index, val in enumerate(classes) :
    for e in X_train:
        if(val == getClass(e)) :
            instances[index] += 1


df = pd.DataFrame({'classes':classes, 'entries':instances})
ax = df.sort_values(by='entries', ascending=True).plot.bar(x='classes', y='entries', color='cornflowerblue',legend=False,
figsize=(12,8))
ax.set_title('Розподіл даних в навчальному наборі даних в Tobacco3482')
ax.set_ylabel("# кількість прикладів в класі")
ax.set_xlabel("# класи")
for p in ax.patches:
    ax.annotate(str(p.get_height()), xy=(p.get_x(), p.get_height()-20))
# transform image in mantix of features
def process_images(img_set) :
    processed_img = []

    for i in range(len(img_set)) :
```

```
        processed_img.append(cv2.resize(cv2.imread(img_set[i], cv2.IMREAD_COLOR), (img_size, img_size)))

    return processed_img
data_train = process_images(X_train_l)
data_test = process_images(X_test_l)
lb = LabelBinarizer()
lb.fit(list(classes))

x_train = np.array(data_train)
y_train =lb.transform(np.array(y_train_l))

x_test = np.array(data_test)
y_test = lb.transform(np.array(y_test_l))

base_model = VGG16(weights = None, include_top=False, input_shape = (img_size, img_size, channels))
base_model.summary()

model = models.Sequential()

model.add(base_model)
model.add(layers.Flatten())
model.add(layers.Dense(128, activation='relu',  name='dense'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(len(classes), activation='softmax',  name='predictions'))

model.summary()
model.compile(optimizer=optimizers.Adam(learning_rate=learning_rate), loss='categorical_crossentropy',
metrics=['accuracy'])


# In[31]:


train_model = model.fit(x_train, y_train,
                        batch_size=batch_size,
                        epochs=15,
                        verbose=1,
                        validation_data=(x_test, y_test))


model.save('trained_modelVGGversion15.h5')
new_model = models.load_model('trained_modelVGGversion15.h5')

# Check its architecture
new_model.summary()
new_model_20 = models.load_model('trained_modelVGGversion20.h5')

# Check its architecture
new_model_20.summary()
score_20 = new_model_20.evaluate(x_test, y_test, verbose=1)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

score_20 = new_model_20.evaluate(x_test, y_test, verbose=1)
```

```
def manualValidateModel(predictions_list, y_test_l):
    count_true = 0;
    count_false = 0;

    predicted_label_l = []

    for i, prediction in enumerate(predictions_list):
        state = True
        #print(np.argmax(prediction))
        if (predicted_classes[np.argmax(prediction)] != y_test_l[i]) :
            state = False
            count_false += 1
        else :
            count_true += 1
        predicted_label_l.append(predicted_classes[np.argmax(prediction)])
        print("Прогнозований клас зображення: ", predicted_classes[np.argmax(prediction)], "   |   Справжній клас
зображення : ", y_test_l[i],  "   |   Результат класифікації : ", state)

    return count_true, count_false, predicted_label_l

count_true, count_false, predicted_label_l  = manualValidateModel(predictions_list, y_test_l)
confusion = confusion_matrix(y_test_l, predicted_label_l)

cm_display = ConfusionMatrixDisplay(
    confusion_matrix=confusion,
    display_labels = list(predicted_classes))

cm_display.plot()
plt.show()

# ### classificator

path_to_test_file = "C:\\Users\\pashc\\OneDrive\\Documents\\Magister Desertation
project\\DocumentImageClassSystem\\TestDocs"


# In[319]:


def DocsClaffier(path_to_test_file, new_model_20):
    total_set_test_image, total_labels_test = getEntityPathAndLabel(path_to_test_file)

    random.Random(seed).shuffle(total_set_test_image)

    data_format_of_test_docs = process_images(total_set_test_image)
    result_test_image = np.array(data_format_of_test_docs)

    predictions_new_image1 = new_model_20.predict(result_test_image, verbose=1)

    predictions_list_new = predictions_new_image1.tolist()

    return  predictions_list_new


# In[320]:
```

```
path_new_to_classified_docs = 'C:\\Users\\pashc\\OneDrive\\Documents\\Magister Desertation
project\\DocumentImageClassSystem\\'


# In[321]:


def CreateOrCheckDir(parent_dir, directory):
    # Path
    path_to_dir = os.path.join(parent_dir, directory)

    # Create the directory
    if not os.path.exists(path_to_dir):
        os.mkdir(path_to_dir)
        print("Folder %s created!" % directory)
    #else:
        #print("Folder %s already exists" % directory)

    return path_to_dir

path_to_parent_dir = CreateOrCheckDir(path_new_to_classified_docs, 'ClassifiedDocs')

predictions_list_new = DocsClaffier(path_to_test_file, new_model_20)

for i, prediction in enumerate(predictions_list_new):
    ind_prediction = np.argmax(prediction)
    label_predict = predicted_classes[ind_prediction]

    dir_to_docs_class = CreateOrCheckDir(path_to_parent_dir, label_predict)

    path_of_current = total_set_test_image[i]

    shutil.copy(path_of_current, dir_to_docs_class)
```

Додаток Б
Ілюстративний матеріал

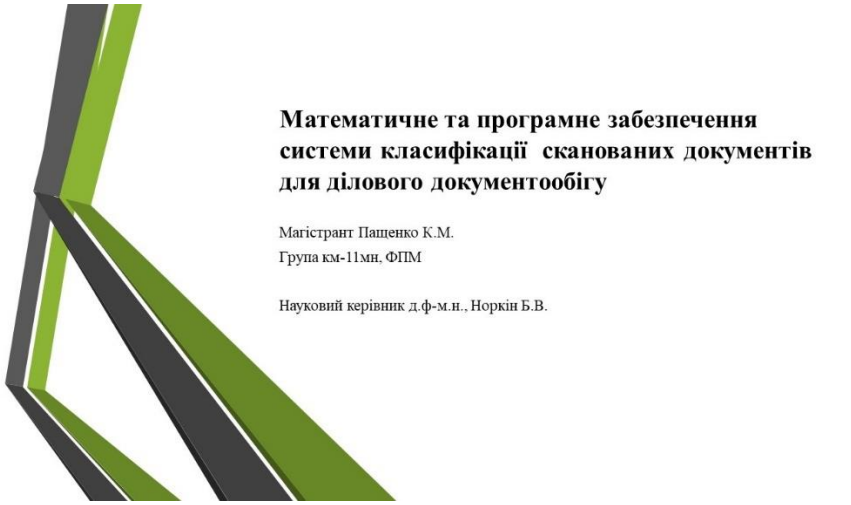


Рисунок Б.1 — Слайд 1

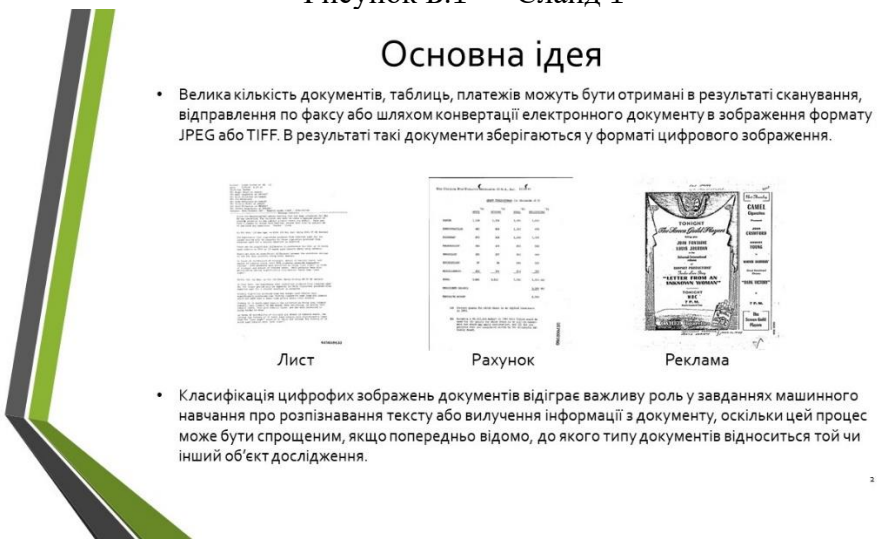


Рисунок Б.2 — Слайд 2

Актуальність

Класифікація сканованих документів є важливою задачею в процесі документообігу:

- дозволяє автоматично розподіляти або архівувати документи;
- покращує ефективність індексуванняу створенні цифрової бібліотеки. (класифікація документів на сторінки змісту або титульну сторінку допомагає звузити кількість сторінок, з яких потім можна витягти певні метадані);
- відіграє важливу роль у пошуку конкретного виду документу (наприклад з бази різнорідної колекції зображень документів користувачі мають багато запитів на пошук, наприклад паперів з одного конкретного журналу або пошуку документа, що містять таблиці або графіки. Класифікація зображень документів на основі візуальної подібності допомагає звузити пошук).

3

Рисунок Б.3 — Слайд 3

Аналіз існуючих рішень

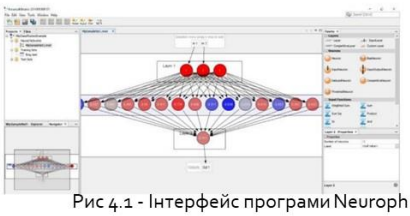


Рис 4.1 - Інтерфейс програми Neuroph

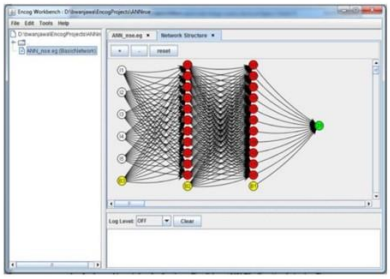


Рис 4.3 -Інтерфейс програми Encog

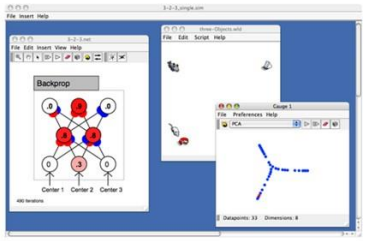


Рис 4.4 -Інтерфейс програми Neuroph

4

Рисунок Б.4 — Слайд 4

Аналіз існуючих рішень

- Розглянувши системи для розпізнавання образів, був зроблений висновок, що всі вони не підходять для реалізації завдання саме класифікації цифрових зображень сканованих документів;
- одним із найголовніших факторів стало те, що вони не включають в себе реалізацію згорткових нейронних мереж а саме такого типу нейронні мережі найбільше підходять для створення модулю класифікації зображень;
- такі системи складаються з декількох блоків та вимагають значних комп'ютерних ресурсів. Системи не є безкоштовними.

5

Рисунок Б.5 — Слайд 5

Постановка задачі

Мета роботи: підвищення ефективності процесу обробки ділових документів в застосунках для аналізу документів шляхом розробки математичного та програмного забезнечення для класифікації цифрових зображень документів.

Предмет дослідження: засоби автоматизації класифікації сканованого документу на основі згорткової нейронної мережі.

6

Рисунок Б.6 — Слайд 6

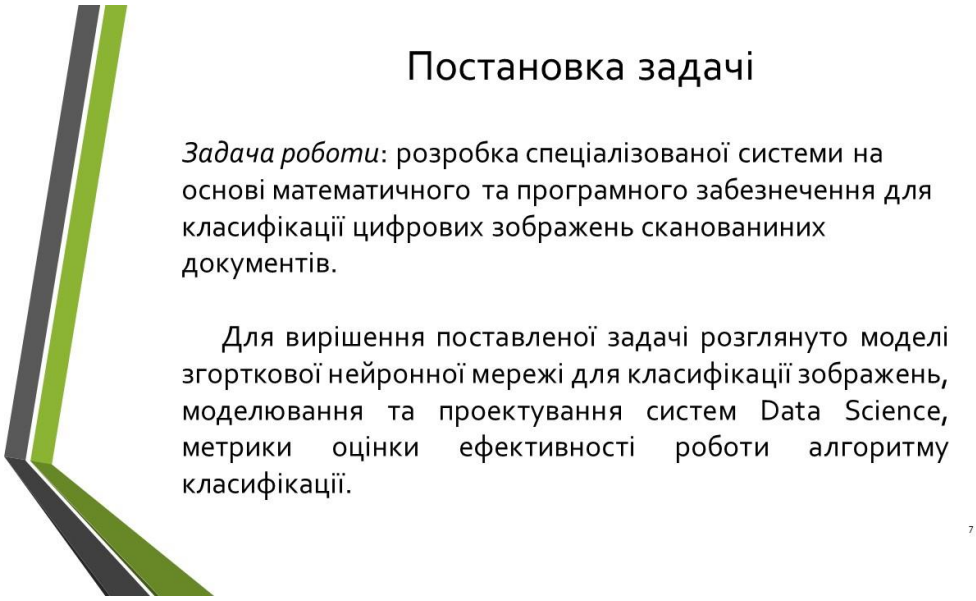


Рисунок Б.7 — Слайд 7

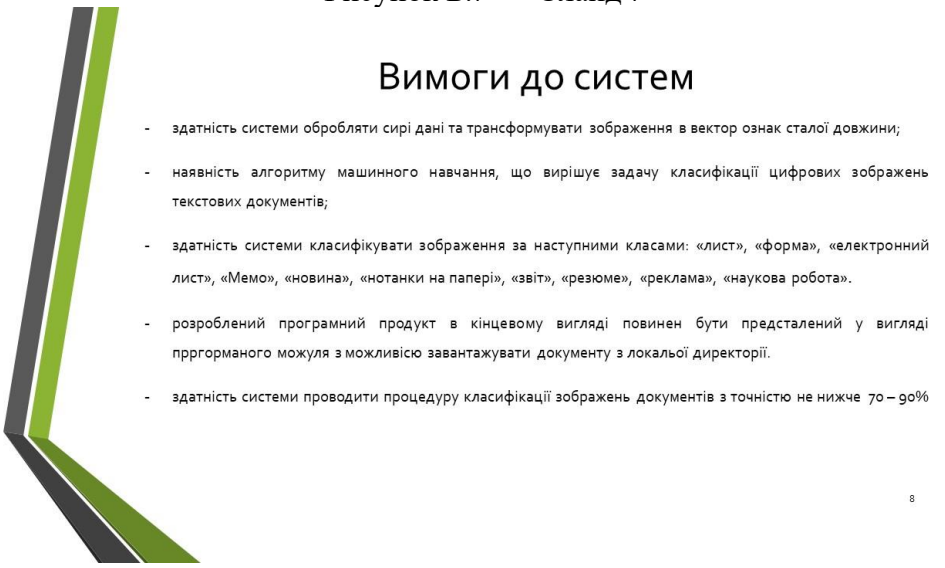
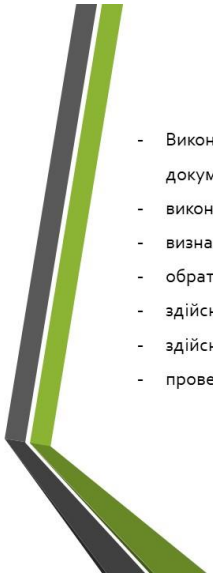


Рисунок Б.8 — Слайд 8

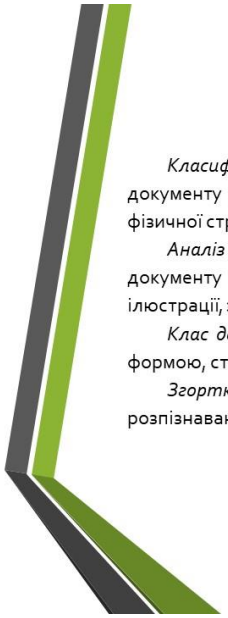


Задачі для реалізації поставленої мети

- Виконати аналіз робіт та досліджень існуючих систем класифікації цифрових зображень документів;
- виконати аналіз існуючих методів аналізу, передбачення та класифікації категоріальних даних;
- визначити підсистеми, з яких має скаладатись система;
- обрати моделі методів машинного навчання для кожної з підсистеми;
- здійснити попередню обробку сирих;
- здійснити програмну реалізацію спроектованих підсистем;
- провести тестування розроблених підсистем.

9

Рисунок Б.9 — Слайд 9



Термінологія

Класифікація зображень цифрових документів – завдання визначення односторінкового документу до одного з попередньо заданих класів документів. Класифікація базується на аналізі фізичної структури документу (англ. document layout analysis).

Аналіз макету документу – процес, що застосовується для визначення фізичної структури документу та його компонентів. Прикладом компонентів документу можуть бути основний текст, ілюстрації, зображення, таблиці.

Клас документу – множина документів, що характеризується подібною фізичною структурою, формою, стилем (наприклад клас «рахунок», «анкета», «реклама» та інші).

Згорткова нейронна мережа – різновид глибинних нейронних мереж, що використовується для розпізнавання двовимірних або тривимірних даних.

10

Рисунок Б.10 — Слайд 10

Компонентна модель системи

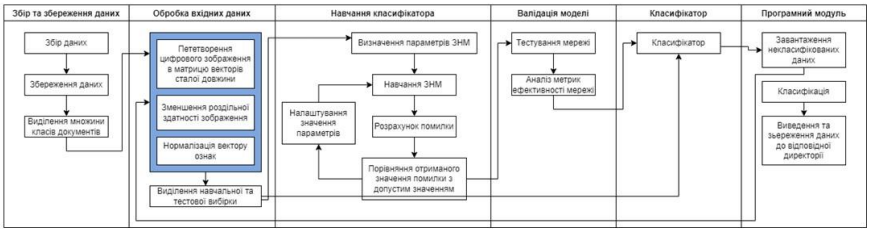


Рис 11.1 Компонентна модель системи класифікації цифрових зображень документів

11

Рисунок Б.11 — Слайд 11

Опис набору даних

- Набір даних Tobacco-3482 складається з 3482 зображень у відтінках сірого;
- Набір розділені між десятьма класами, а саме: «лист», «форма», «електронний лист», «мемо», «новина», «нотанки на папері», «звіт», «резюме», «реклама», «наукова робота»
- Зображення мають роздільну здатність більшу за 1000 пікселів.
- Зображення зберігаються в форматі .jpg

Рис 12.1 – Приклади вхідних даних

12

Рисунок Б.12 — Слайд 12

Опис блоків збереження та обробка даних

- Блок збереження даних системи класифікації зображень документів охоплює оцінку простору документів та перелік можливих класів документів;
- Простір документів – це множина всіх документів, які подаються на вхід класифікатора для обробки - вхідні дані моделі. Звідси визначаються марковані екземпляри для навчання та тренування класифікатора.
- Множина класів зображень документів визначає як розділений простір документів, а назва класу документу є результатом роботи.
- В даній роботі розглядається варіант розділення простору документів зображення, коли документу належить лише одному класу

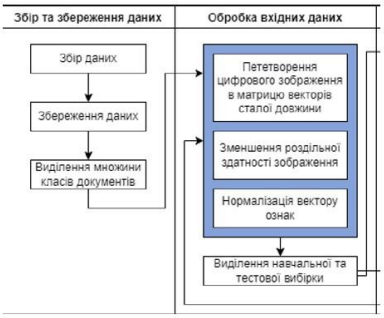


Рис 13.1 – Блок збереження та обробки даних

13

Рисунок Б.13 — Слайд 13

Опис класифікатора

- Модель згорткової нейронної мережі, що є різновидом нейронних мереж, відзначається особливою швидкістю прийняття рішень та забезпечує високу ефективність класифікації зображень;
- навчання моделі проводиться на маркованих даних, які використовуються для ітеративного налаштування ваг;
- мережа вимагає ручного налаштування топології моделі;
- Фреймворк Keras – є високорівневий фреймворк, написаний на Python. Keras надає підтримку для побудови моделей глибинного навчання, включаючи нейронні мережі

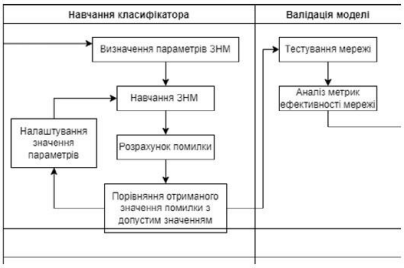


Рис 12.1 – Блок «Навчання класифікатора»

14

Рисунок Б.14 — Слайд 14

Рисунок Б.15 — Слайд 15

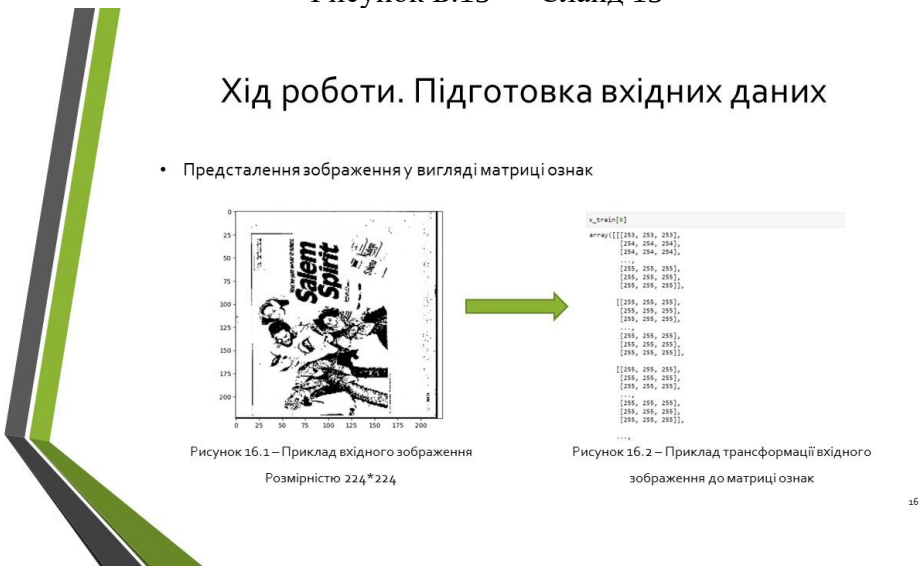


Рисунок Б.16 — Слайд 16

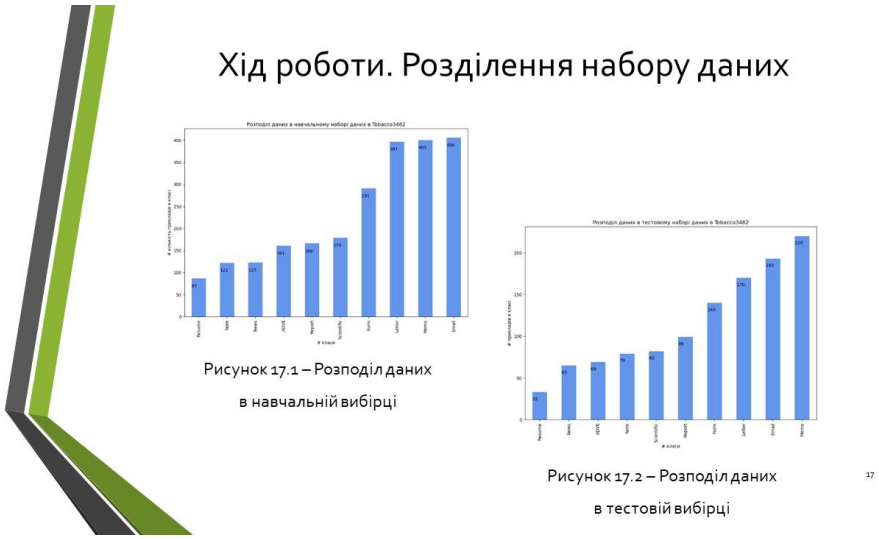


Рисунок Б.17 — Слайд 17

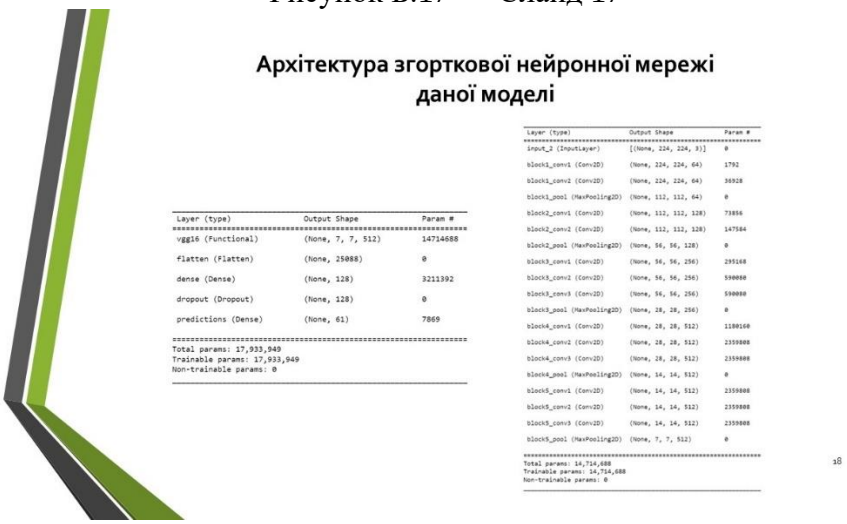


Рисунок Б.18 — Слайд 18

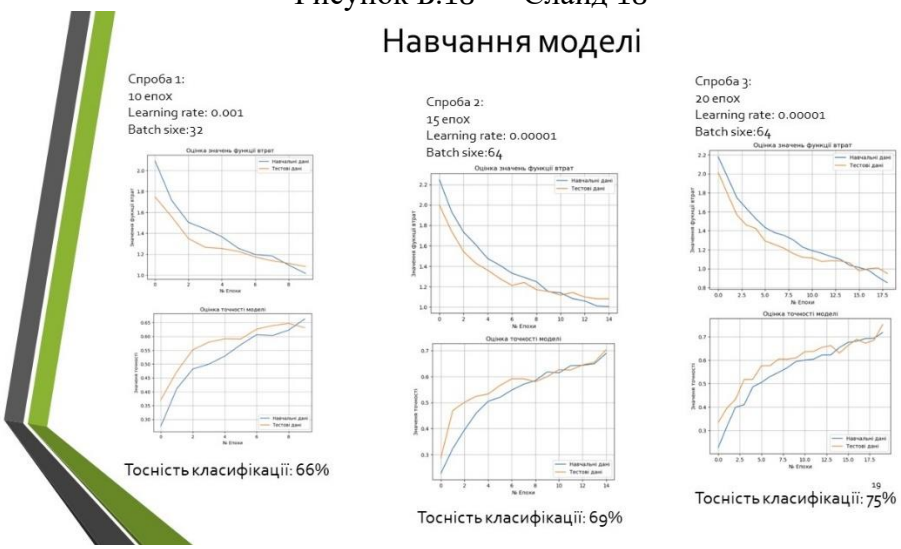


Рисунок Б.19 — Слайд 19

Приклад процесу навчання мережі

```
Epoch 8/20
22/22 [==============================] - 1207s 56s/step - loss: 1.3517 - accuracy: 0.5474 - val_loss: 1.2152 - val_accuracy: 0.
6043
Epoch 9/20
22/22 [==============================] - 3845s 181s/step - loss: 1.3028 - accuracy: 0.5675 - val_loss: 1.2121 - val_accuracy:
0.6038
Epoch 10/20
22/22 [==============================] - 4470s 208s/step - loss: 1.2267 - accuracy: 0.5941 - val_loss: 1.1588 - val_accuracy:
0.6091
Epoch 11/20
22/22 [==============================] - 4051s 188s/step - loss: 1.1901 - accuracy: 0.5999 - val_loss: 1.1191 - val_accuracy:
0.6359
Epoch 12/20
22/22 [==============================] - 3379s 155s/step - loss: 1.1629 - accuracy: 0.6034 - val_loss: 1.1127 - val_accuracy:
0.6368
Epoch 13/20
22/22 [==============================] - 1727s 77s/step - loss: 1.1282 - accuracy: 0.6221 - val_loss: 1.0764 - val_accuracy: 0.
6545
Epoch 14/20
22/22 [==============================] - 1203s 55s/step - loss: 1.0981 - accuracy: 0.6221 - val_loss: 1.0831 - val_accuracy: 0.
6622
Epoch 15/20
22/22 [==============================] - 1188s 55s/step - loss: 1.0355 - accuracy: 0.6537 - val_loss: 1.0819 - val_accuracy: 0.
6297
Epoch 16/20
22/22 [==============================] - 1319s 61s/step - loss: 1.0116 - accuracy: 0.6753 - val_loss: 1.0599 - val_accuracy: 0.
6622
Epoch 17/20
22/22 [==============================] - 1375s 63s/step - loss: 0.9846 - accuracy: 0.6810 - val_loss: 0.9781 - val_accuracy: 0.
6890
Epoch 18/20
22/22 [==============================] - 1205s 55s/step - loss: 0.9458 - accuracy: 0.6918 - val_loss: 0.9997 - val_accuracy: 0.
6718
Epoch 19/20
22/22 [==============================] - 1201s 55s/step - loss: 0.9121 - accuracy: 0.6925 - val_loss: 1.0077 - val_accuracy: 0.
6550
```

На етапі навчання було задіяно 2090 документів 10-ти категорій, вказаних в описі даних 500 прикладів, було взято для валідації результатів.
На вхід подавалися зображення розміром 224*224

20

Рисунок Б.20 — Слайд 20

Огляд методів оцінки ефективності роботи класифікатора

Для визначення ступеня задовільності результату навчання згорткової нейронної мережі використовується ряд метрик:

- Точність (Accuracy) оцінює кількість правильно класифікованих зображень до загальної кількості та використовується у задачах, де усі класи є одноков важливими

  Макро точність : $A^j_{macro} = \frac{\sum_{i=1}^{n} I(y^j_i \cap \hat{y}^j_i)}{\sum_{i=1}^{n} I(y^j_i \cup \hat{y}^j_i)}$ $A^{\square}_{macro} = \frac{1}{k}\sum_{j=1}^{k} A^j_{macro}$

  Мікро точність : $A^{\square}_{micro} = \frac{\sum_{j=1}^{k}\sum_{i=1}^{n} I(y^j_i \cap \hat{y}^j_i)}{\sum_{j=1}^{k}\sum_{i=1}^{n} I(y^j_i \cup \hat{y}^j_i)}$

- Збіжність (Precision) обчислюється, як частка усіх правильно прокласифікованих обєктів до класу j до загальної кількості обєктів, які віднесено класифікатором до класу j.

  Макро збіжність: $P^j_{macro} = \frac{\sum_{i=1}^{n} (y^j_i \cap \hat{y}^j_i)}{\sum_{i=1}^{n} \hat{y}^j_i)}$ $P^{\square}_{macro} = \frac{1}{k}\sum_{j=1}^{k} P^j_{macro},$

  Мікро точність : $P^{\square}_{micro} = \frac{\sum_{j=1}^{k}\sum_{i=1}^{n} (y^j_i \cap \hat{y}^j_i)}{\sum_{j=1}^{k}\sum_{i=1}^{n} \hat{y}^j_i}$

21

Рисунок Б.21 — Слайд 21

Огляд методів оцінки ефективності роботи класифікатора

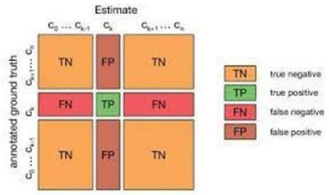
- повнота (англ. Recall)

$$R^j_{macro} = \frac{\sum_{i=1}^{n}(y_i^j \cap \hat{y}_i^j)}{\sum_{i=1}^{n} y_i^j)}$$

$$Recall^{\square}_{micro} = \frac{\sum_{j=1}^{k}\sum_{i=1}^{n}(y_i^j \cap \hat{y}_i^j)}{\sum_{j=1}^{k}\sum_{i=1}^{n} y_i^j}$$

$$R^{\square}_{macro} = \frac{1}{k}\sum_{j=1}^{k} R^j_{macro},$$

- Матриця помилок (багатокласова класифікація)



22

Рисунок Б.22 — Слайд 22

Звіт оцінки ефективності розробленої моделі класифікації

	precision	recall	f1-score	support
ADVE	0.86	0.88	0.87	69
Email	0.92	0.95	0.93	193
Form	0.67	0.82	0.74	140
Letter	0.68	0.81	0.74	170
Memo	0.71	0.78	0.74	220
News	0.79	0.80	0.79	65
Note	0.90	0.58	0.71	79
Report	0.63	0.54	0.58	99
Resume	0.71	0.67	0.69	33
Scientific	0.74	0.28	0.41	82
accuracy			0.75	1150
macro avg	0.76	0.71	0.72	1150
weighted avg	0.76	0.75	0.74	1150

23

Рисунок Б.23 — Слайд 23

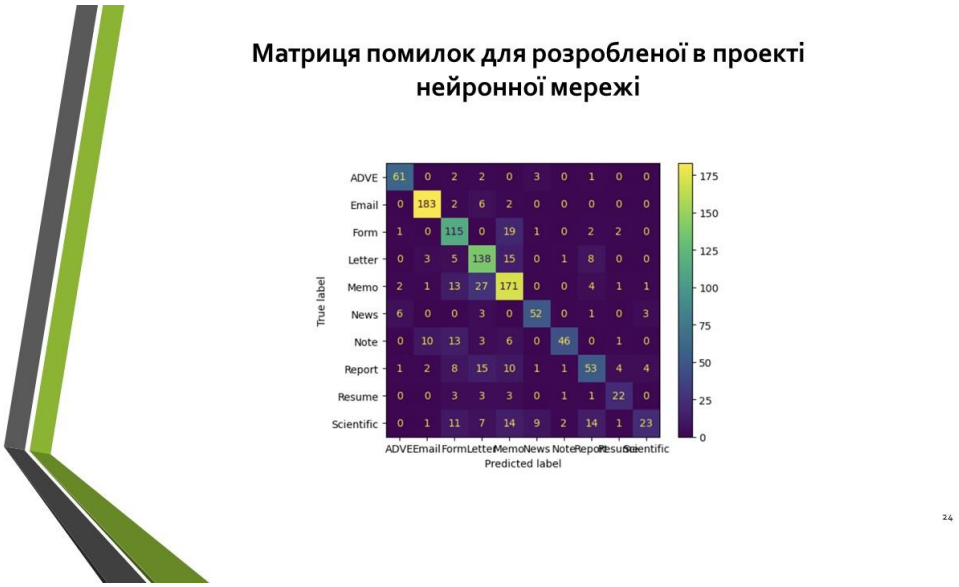


Рисунок Б.24 — Слайд 24

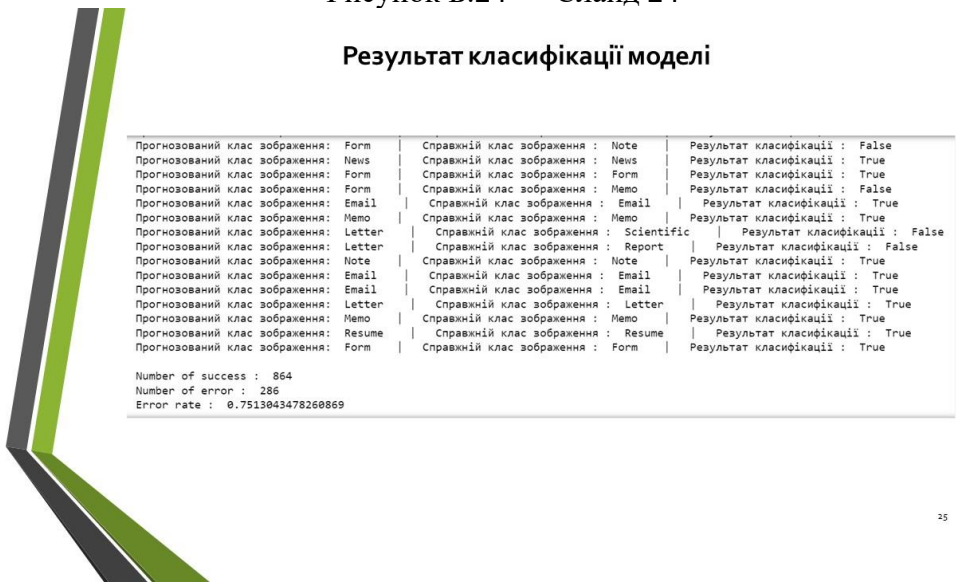


Рисунок Б.25 — Слайд 25

Приклад роботи програмного модуля класифікації

- В директорії TestDocs зібрано скановані документи, які треба прокласифікувати

- Шлях до директорії завантажується до програмного модуля
- Запуст класифікатора

26

Рисунок Б.26 — Слайд 26

Приклад роботи програмного модуля класифікації

- Прокрасифіковані документи завантажують у відровідно директорію класу

27

Рисунок Б.27 — Слайд 27

Висновки

- **Досліджено** актуальні методи класифікації зображень сканованих документів та виявлено, що красифікація документів використовується в задачах обробки документів; систематизовано методи класифікації зображень документів на основі фізичної структури документу.
- **Запропоновано** компонентну модель реалізації системи класифікації цифрових зображень документів, з метою автоматизації класифікації цифрових зображень документів та підвищення ефективності класифікації. Система призначена для застосування в системах електорнних офісів та може стати допоміжним інструментом для малого бізнесу.
- **Здійснено** реалізацію компонент системи, що базується на розробці та налаштуванні згорткової нейронної мережі, з урахуванням попередньої обробки даних,а саме зменшення роздільної здатності зображення. Ефективніть розробленої моделі класифікації становить 75%

28

Рисунок Б.28 — Слайд 28

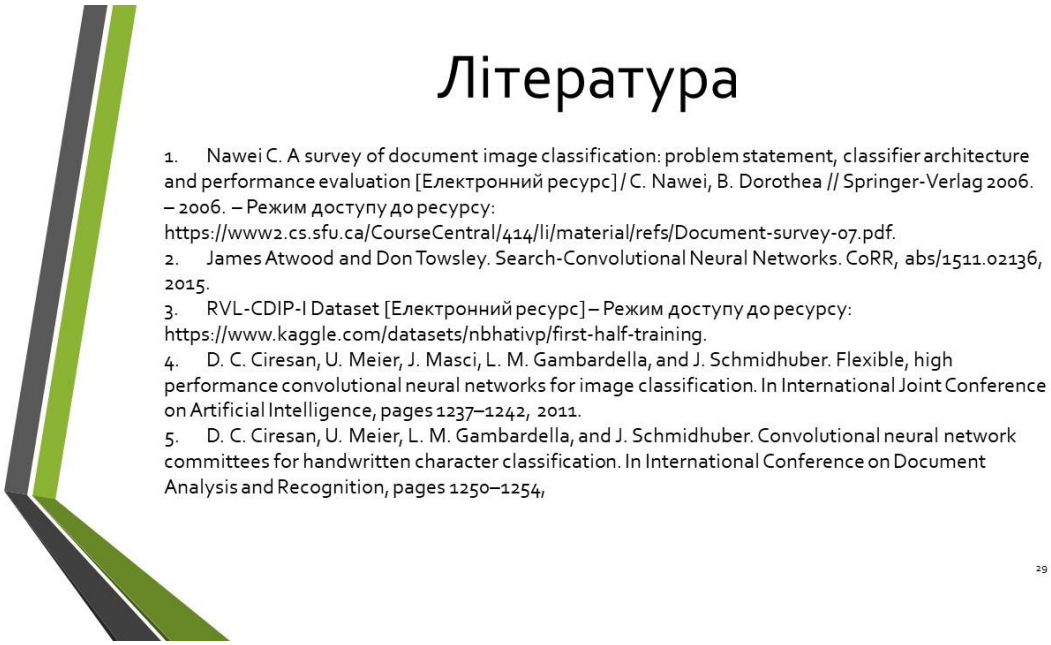


Рисунок Б.29 — Слайд 29

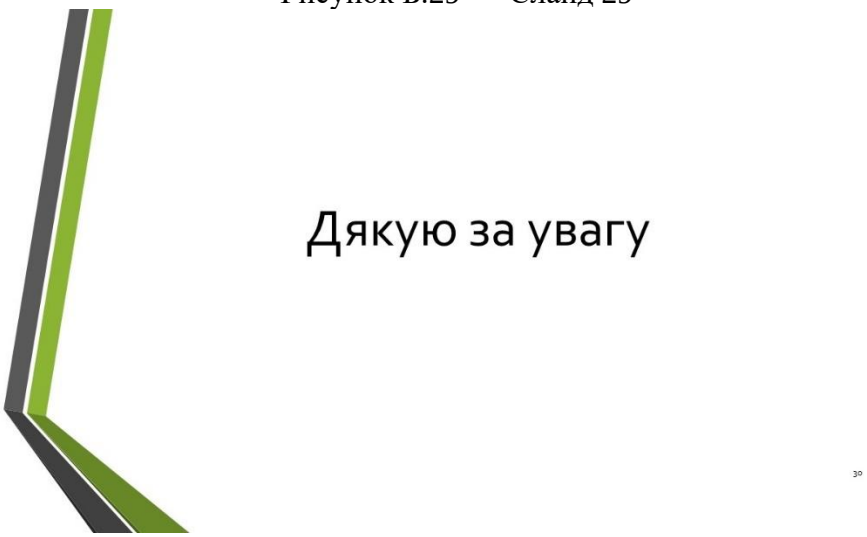


Рисунок Б.30 — Слайд 30