НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
ім. Ігоря Сікорського

Навчально-науковий комплекс «Інститут прикладного системного аналізу»
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
“___”_____2017 р.

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки 6.050101 Комп’ютерні науки
(код і назва)

на тему: «Дослідження штучних нейронних мереж для роспізнавання і класифікації дорожніх знаків»

Виконав (-ла): студент (-ка) __IV__ курсу, групи __ДА-31__
(шифр групи)

_____Євтухов Сергій Костянтинович_____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник_____доц., к.т.н., Харченко К.В. _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант Економічний доц., к.е.н., Рощина Н.В._____ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Нормоконтроль _____старший викладач Бритов О.А._____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студент _____
(підпис)

Київ – 2017 року

Національний технічний університет України
«Київський політехнічний інститут»
ім. Ігоря Сікорського

Інститут (факультет) ННК «Інститут прикладного системного аналізу
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050101 Комп'ютерні науки
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
«___»_____2017 р.

ЗАВДАННЯ
на дипломну роботу студенту
Євтухову Сергію Костянтиновичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження штучних нейронних мереж для розпізнавання та класифікації дорожніх знаків,
керівник роботи Харченко К.В., к.т.н., доцент_____ ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом роботи 12.06.2017

3. Вихідні дані до роботи Операційна система Windows 10, Частота процессору 2.4 ГГц, Середовище розробки – Jupyter Notebook, Мова програмування – Python, Фреймворк машинного навчання – Tensorflow, Бібліотеки, що використовувались: numPy, scikit-learn

4. Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити)

1. Спроектувати архітектуру штучної нейронної мережі для класифікації дорожніх знаків.
2. Проаналізувати математичні методи навчання нейронної мережі. Оптимізувати нейронну мережу.
3. Проаналізувати підходи до передобробки даних. Підготувати датасет.

4. Дослідити вплив особливостей архітектури мережі на результат класифікації.
5. Дослідити вплив гіперпараметрів моделі на результат класифікації.
6. Провести функціонально-вартісний аналіз отриманого програмного
7. Проаналізувати результати роботи, зробити висновки.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) _____
_____
_____

6. Консультанти розділів роботи*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Рощина Н.В., доцент, к.е.н.		

7. Дата видачі завдання 01.02.2017

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	01.02.2017	
2	Ознайомлення з технічною літературою і підготовка теоретичної частини	28.02.2017	
3	Аналіз вимог завдання, вибір методів і засобів розв'язання поставленої задачі	15.03.2017	
4	Розробка і дослідження моделі. Проведення експериментів	25.04.2017	
5	Тестування моделі. Перевірка відповідності завданню	05.05.2017	
6	Оформлення дипломної роботи	31.05.2017	
7	Отримання допуску до захисту та подача роботи в ДЕК	10.06.2017	

Студент _____ С.К. Євтухов
(підпис) (ініціали, прізвище)

Керівник роботи _____ К.В. Харченко
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено керівника дипломної роботи.

АНОТАЦІЯ

бакалаврської дипломної роботи Євтухова Сергія Костянтиновича на тему:
«Дослідження штучних нейронних мереж для класифікації дорожніх знаків»

Дипломна робота присвячена проектуванню моделі на основі штучної нейронної мережі для вирішення задачі класифікації дорожних знаків.

В роботі описано основні принципи проектування нейронних мереж для вирішення задач класифікації зображень. Спираючись на ряд апріорних припущень щодо архітектури нейронної мережі, створено й оптимізовано власну модель, в основі якої знаходиться композиція згорткової нейронної мережі і повнозв'язного класифікатора.

Згорткова штучна нейронна мережа є багаторівневою нейромережевою архітектурою, що самостійно визначає найважливіші характеристики на вхідних даних.

Над отриманою моделлю проведено ряд експериментів, що включають різні підходи до передобробки даних, зміну елементів архітектури нейромережі. Досліджено вплив початкових значень гіперпараметрів на результат класифікації.

Розпізнавання і класифікація дорожніх знаків мають прикладне значення при проектуванні безпечних автопілотованих і самокерованих транспортних засобів. Результати даної роботи можуть бути застосовані для вирішення більш глобальної задачі детектування і розпізнавання дорожних знаків у відеопотоці, що отримано з камер самокерованого транспортного засобу.

Загальний обсяг роботи: 70 сторінок, 25 рисунків, 9 таблиць, 10 посилань

Ключові слова: згорткова нейронна мережа, класифікатор, оптимізація, backpropagation, розпізнавання, самокерований автомобіль

АННОТАЦИЯ

бакалаврской дипломной работы Евтухова Сергея Константиновича по теме: «Исследование искусственных нейронных сетей для классификации дорожных знаков»

Дипломная работа посвящена проектированию модели на основе искусственной нейронной сети для решения задачи классификации дорожных знаков.

В работе описаны основные принципы проектирования нейронных сетей для решения задачи классификации изображений. Опираясь на ряд априорных предположений об архитектуре нейронной сети, создана и оптимизирована собственная модель, в основе которой находится композиция свёрточной нейронной сети и полносвязного классификатора.

Свёрточная искуственная нейронная сеть есть многоуровневой нейросетевой архитектурой, которая самостоятельно определяет наиболее важные характеристики входных данных.

Над полученой моделью проведено ряд экспериментов, среди которых разные подходы к предобработке данных, изменение элементов архитектуры нейросети. Исследовано влияние начальных значений гиперпараметров на результат классификации.

Распознавание и классификация дорожных знаков имеют прикладное значение при проектировании безопасных автопилотируемых и самоуправляемых транспортных средств. Результаты данной работы могут быть применены для решения более глобальной задачи детектирования и распознавания дорожных знаков в видеопотоке, полученном с камер самоуправляемого транспортного средства.

Общий объем работы: 70 страниц, 25 рисунков, 9 таблиц, 10 ссылок

Ключевые слова: свёрточная нейронная сеть, классификатор, оптимизация, backpropagation, распознование, самоуправляемый автомобиль

ABSTRACT

for a bachelor thesis written by Yevtukhov Serhii Kostyantinovich on: «Investigation of Artificial Neural Networks For Traffic Signs Classification Problem»

Thesis is dedicated to development of the model based on the artificial neural network to solve a problem of traffic signs classification.

The paper describes basic principles of designing neural networks for solving image classification problems. Using the number of apparent assumptions about neural network architecture, a new model, that based on a combination of convolutional neural network and fully connected classifier, was created and optimized.

The convolutional artificial neural network is a multi-level neural network architecture that independently determines the most important characteristics of the input data.

The resulting model was examined with several experiments, including different approaches to data preprocessing and changing of network architecture elements. I have also investigated how the classification accuracy depends on hyper parameters initialization.

Recognition and classification of traffic signs have a practical importance in designing of safe autopilot and self-driving vehicles. The results of this work can be applied to solve a more global problem of detecting and recognition of traffic signs in a video stream received from the cameras of a self-driving vehicle.

Total volume of work: 70 pages, 25 figures, 9 tables, 10 references

Key words: convolutional neural network, classifier, optimization, backpropagation, recognition, self-driving car

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

ГС – градієнтний спуск

СГС – стохастичний градієнтний спуск

НМ – нейронна мережа

ЗНМ – згорткова нейронна мережа

Датасет – набір даних, вибірка

ReLU – Rectifier – активаційна функція

Backpropagation – метод зворотного розповсюдження помилки

Tensorflow – назва бібліотеки машинного навчання

Пулінг – операція зменшення розмірності шару згорткової нейронної мережі

Multi-Scale архітектура – нейромережева архітектура, в якій на вхід класифікатора подають результат згортки декількох згорткових шарів

ВСТУП

Сьогодні як ніколи чітко окреслився вектор розвитку машинобудівельної індустрії. Це розвиток в напрямку проектування і розробки досконалих безпілотних (самокерованих) транспортних засобів. Дослідницькі центри найвідоміших автомобільних гігантів як то Tesla Motors, Mercedes-Benz, BMW, Toyota активно працюють на передовій лінії сучасної науки і інженерії, намагаючись створити найдосконалішу модель. Навіть компанії, що не стосуються прямо машинобудування, на кшталт Google, розроблюють власні концепти самокерованих автомобілів.

При створенні безпілотного транспортного засобу виділяють два основні напрями робіт: розробка алгоритмів машинного навчання і комп'ютерного зору для вирішення задач розпізнавання та прогнозування; та розробка фізичної автономної рухомої платформи (композиції датчиків, локалізація, контроль). Ці дві частини самокерованого автомобіля можна асоціювати з його мозком і тілом.

Дана робота присвячена одній з найважливіших функцій «мозку» безпілотного транспортного засобу – розпізнаванню ключових образів у середовищі застосування (таких як автомобілі, пішоходи, дорожні знаки, сигнали світлофору, дорожня розмітка) для подальшого прийняття рішень. А конкретно, вирішується задача класифікації дорожніх знаків.

Мінімальною задачею інженерів-розробників безпілотних автомобілів при класифікації дорожніх знаків є досягти точності, що порівнюється з людською – 98%. Звичайно, основні зусилля спрямовані на те, що подолати цей поріг і досягти точності, близької до 100%. Саме така постановка цілей є важливою, оскільки мова йде про безпеку пасажирів. Неточність, допущена при розробці моделі, може призвести до катастрофічних наслідків.

Розв’язувати задачі розпізнавання і класифікації з високою точністю дозволяють штучні глибокі нейронні мережі. Проте щоб досягти максимального результату необхідно обирати такі підходи при проектуванні нейронної мережі, які враховують особливості задачі. Результат класифікації може залежати як від розробленої архітектури нейромережі, так і від методів її оптимізації; як від початкового налаштування гіперпараметрів, так і, безумовно, від особливостей набору даних, що використовується при навчанні.

Таким чином, необхідно виділити наступні підзадачі при створенні моделі для класифікації дорожніх знаків:

- дослідження і передобробка датасетів;
- проектування архітектури нейронної мережі;
- дослідження впливу компонентів нейронної мережі на точність класифікації;
- дослідження впливу гіперпараметрів на точність класифікації.

Метою цієї роботи є розробка моделі на основі згорткової штучної нейронної мережі, що здатна класифікувати зображення дорожних знаків. Розроблена модель може бути використана для вирішення більш глобальної задачі детектування і розпізнавання дорожних знаків у відеопотоці, що надходить з камер самокерованого автомобіля.

1 ПРОЕКТУВАННЯ ШТУЧНОЇ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ

1.1 Огляд класичних архітектур НМ і задач, що вирішуються за допомогою них

1.1.1 Перцептрон

Перцептрон - одна з перших моделей нейромереж. Незважаючи на свою простоту, перцептрон здатний навчатися і вирішувати досить складні завдання. Основна математична задача, з якою він справляється, - це лінійне розділення будь-яких нелінійних множин, так зване забезпечення лінійної сепарабельності. Типова архітектура багатошарового перцептрону схематично зображена на рис. 1.

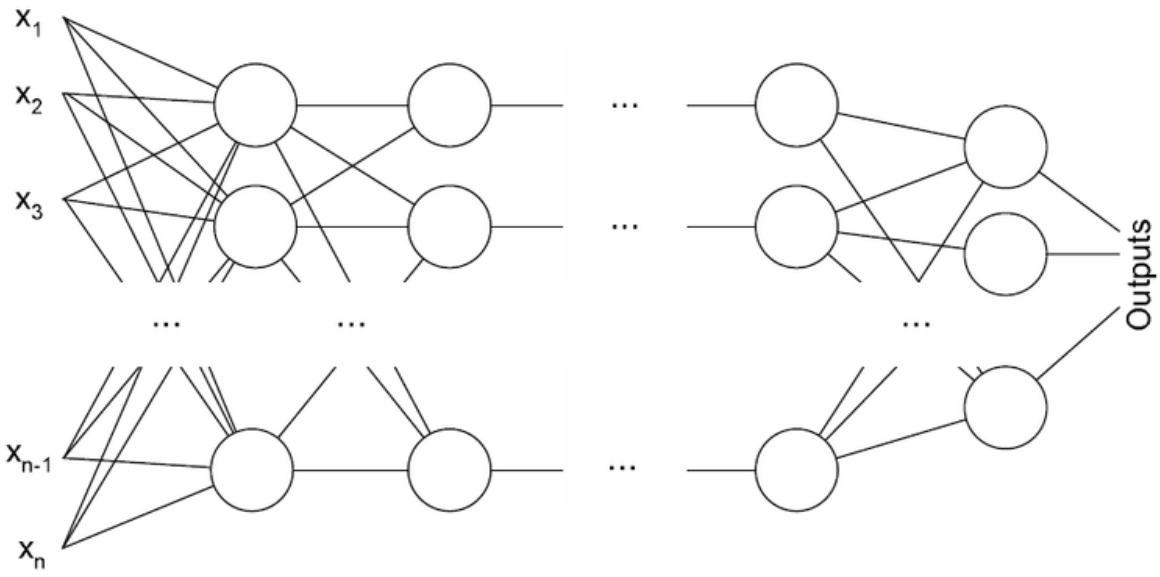


Рисунок 1 – Архітектура багатошарового перцептрону

Перцептрон складається з трьох типів елементів, а саме: сигнали, що надходять від давачів, передаються до асоціативних елементів, а відтак до реагуючих. Таким чином, перцептрони дозволяють створити набір «асоціацій» між

вхідними стимулами та необхідною реакцією на виході. В біологічному плані це відповідає перетворенню, наприклад, зорової інформації у фізіологічну відповідь рухових нейронів.

Перцептрон застосовується для вирішення класичних задач машинного (класифікація, регресія) навчання як окрема модель, так і в складі більш складних моделей.

1.1.2 Згорткова НМ

Згорткові мережі можуть включати шари локальної або глобальної підвибірки, які поєднують виходи кластерів нейронів [1]. Вони також складаються з різних комбінацій згорткових та повноз'єднаних шарів, із застосуванням поточкової нелінійності в кінці кожного шару. Для зниження числа вільних параметрів та покращення узагальнення вводиться операція згортки на малих областях входу. Однією з головних переваг згорткових мереж є використання спільної ваги у згорткових шарах, що означає, що для кожного пікселя шару використовується один і той же фільтр (банк ваги); це як зменшує обсяг необхідної пам'яті, так і поліпшує продуктивність.

Класичними реалізаціями ЗНМ, що стали проривом в індустрії, є LeNet5 та AlexNet.

1.1.2.1 LeNet5

LeNet-5 - новаторська 7-шарова згорткова мережа, розроблена Яном ЛеКуном в 1995, яка класифікує рукописні цифри. Застосовується банками для розпізнавання рукописних цифр на оцифрованих чеках. На вхід приймає зображення розміром 28x28 пікселів у напівтонах сірого. Можливість обробляти зображення з більш високою роздільною здатністю вимагає більшої кількості шарів згортки, так що ця методика обмежена доступністю обчислювальних ресурсів.

1.1.2.2 AlexNet

Велика глибока згорткова нейронна мережа для класифікації зображень з високою роздільною здатністю була тренована на датасеті ImageNet LSVRC-2010, що налічував 1,3 мільйона екземплярів в 1000 різних класів. Архітектура мережі зображена на рис. 2. Нейронна мережа, яка має 60 мільйонів параметрів і 500000 нейронів, складається з п'яти шарів згортки, за деякими з яких знаходяться пулінгові шари (шари підвибірки), і повнозв'язний перцептрон в кінці. Для того, щоб навчання відбулося швидше, були розроблено паралельну реалізацію згортальних мереж на GPU. Для зменшення перенавчання в повнозв'язних шарах було використано новий метод регуляризації, який виявився дуже ефективним.

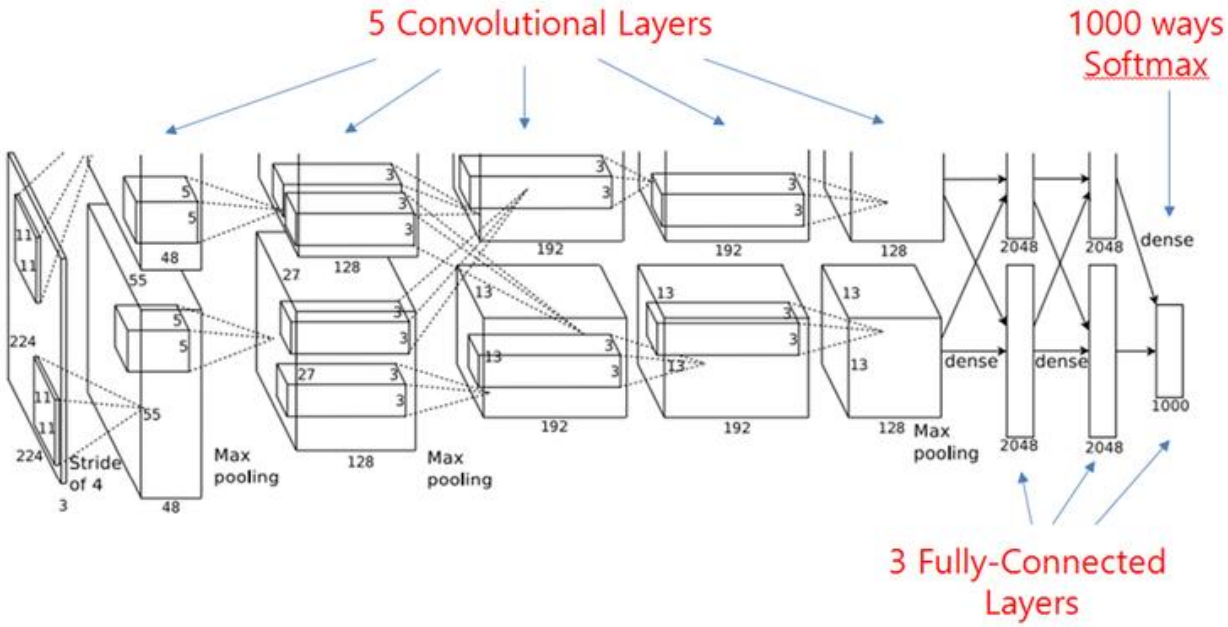


Рисунок 2 – Архітектура AlexNet

1.2 Визначення задачі

1.2.1 Опис вхідних даних

Вхідними даними до задачі класифікації дорожніх знаків є датасет німецьких дорожніх знаків, що налічує близько 50000 екземплярів. Екземпляром є кольорове зображення розміром 32x32 пікселя, що містить мітку одного з 43 класів, до якого

належить дорожній знак. Таким чином датасет містить близько 40000 розмічених екземплярів. Решта є нерозміченими і будуть використані для валідації і тестування моделі.

Усього у датасеті представлені 43 німецьких дорожніх знака (табл. 1), які мають відповідники в Україні.

Таблиця 1 – Класи знаків, представлених у датасеті

ClassId	**SignName**
0	Speed limit (20km/h)
1	Speed limit (30km/h)
2	Speed limit (50km/h)
3	Speed limit (60km/h)
4	Speed limit (70km/h)
5	Speed limit (80km/h)
6	End of speed limit (80km/h)
7	Speed limit (100km/h)
8	Speed limit (120km/h)
9	No passing
10	No passing for vehicles over 3.5 metric tons
11	Right-of-way at the next intersection
12	Priority road
13	Yield
14	Stop
15	No vehicles
16	Vehicles over 3.5 metric tons prohibited
17	No entry
18	General caution
19	Dangerous curve to the left
20	Dangerous curve to the right
21	Double curve
22	Bumpy road
23	Slippery road
24	Road narrows on the right
25	Road work
26	Traffic signals
27	Pedestrians
28	Children crossing

Таблиця 1 (закінчення)

29	Bicycles crossing
30	Beware of ice/snow
31	Wild animals crossing
32	End of all speed and passing limits
33	Turn right ahead
34	Turn left ahead
35	Ahead only
36	Go straight or right
37	Go straight or left
38	Keep right
39	Keep left
40	Roundabout mandatory
41	End of no passing
42	End of no passing by vehicles over 3.5 metric tons

Після аналізу вибірки і побудови гістограми частот екземплярів (рис. 3), можна побачити, що вона є надто нерівномірною. Деякі єкземпляри недопредставлені, а деякі представлені в надлишковому об’ємі. Щоб не допустити перенавчання моделі на екземплярах, що зустрічаються частіше, можливо, знадобиться вирівняти датасет шляхом генерації додаткових екземлярів.

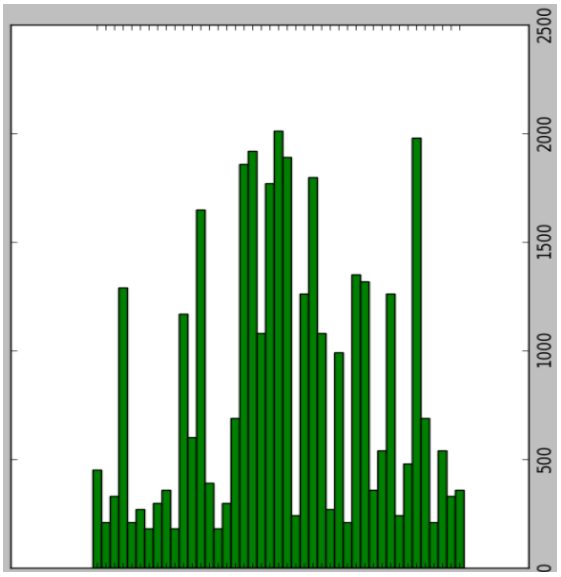


Рисунок 3 – Гістограма частот екземплярів навчальної вибірки

1.2.2 Опис вихідних даних

Вихідними даними задачі є клас дорожнього знаку, зображення якого було «зкормлено» нейронній мережі, і яке вона до того не бачила. Таким чином буде побудована модель, що здатна класифікувати дорожні знаки на 43 класи, що представлені в навчальній вибірці.

1.2.3 Розмірність задачі

Таким чином, маємо датасет дорожніх знаків, розміри якого в першу чергу визначають розмірність задачі. Ключова інформація про дані наведена у таблиці 2.

Таблиця 2 – Основні статистики датасету

Об’єм навчальної вибірки	34799
Об’єм валідаційної вибірки	4410
Об’єм тестувальної вибірки	12630
Розмірність екземпляра (зображення)	32x32x3
Кількість класів	43

1.3 Вибір початкової архітектури НМ

1.3.1 Поєднання згорткової і повнозв’язної архітектур

Зважаючи на викладений вище огляд типових архітектур нейронних мереж і на клас задачі, що вирішується в даній роботі (задача класифікації зображень), архітектура нейронної мережі буде спроектована наступним чином: поєднаймо згорткову нейронну мережу для вичлененння фіч і повнозв'язний перцептрон для класифікації екземплярів [2]. Отримаємо мережу схожу до мереж сімейства LeNet (рис. 4).

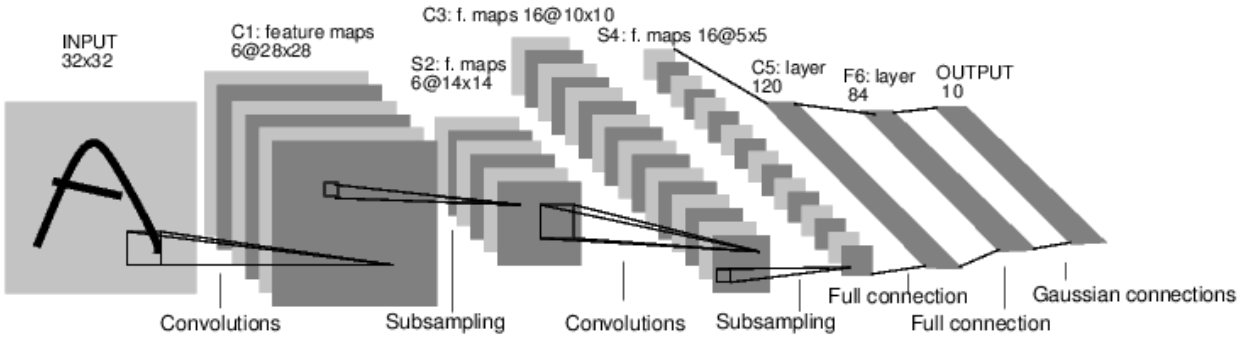


Рисунок 4 – Архітектура LeNet-подібної згорткової нейромережі

1.3.2 Обрання початкових розмірів мережі

Спроектована нейронна мережа приймає на вхід зображення, представлене масивом цілих чисел розмірністю 32x32x3. Такий масив містить 3072 елементів. Конволюційна нейронна мережа дозволяє вичленити з зображення ключові його елементи, по яких можна класифікувати або розпізнати ті чи інші об'єкти.

Емпірично встановлено, що для даного формату вхідних даних конволюційні фільтри розміром 5x5 дозволяють виділити найважливіші характеристики зображення, не пропустивши важливих фіч. Застосуємо декілька таких фільтрів до вхідного зображення.

Наприклад, маємо 6 фільтрів розмірністю 5x5, що проініціалізовані випадковими значеннями ваг. Тоді після першого шару згортки отримаємо 6 карт фіч (feature map) розміром 28x28.

Для того щоб мінімізувати кількість параметрів нейромережі і виділити тільки найважливіші фічі, застосовується пулінг. Візьмемо розмірність сітки пулінгу 2x2 і крок сітки 1. Тоді після шару пулінгу отримаємо 6 карт фіч розміром 14x14.

Застосуємо 16 фільтрів розмірністю 5x5. І після другого шару згортки отримаємо 16 карт фіч (feature map) розміром 10x10. Після другого шару пулінгу маємо 16 карт фіч розміром 5x5.

Таким чином, кількість параметрів нейромережі зменшується майже в 8 разів.

Далі приєднаний повнозв’язний перцептрон з одним прихованим шаром. Задамо кількість нейронів в прихованому шарі 120.

Зведемо отримані значення до однієї таблиці (табл. 3).

Таблиця 3 – Розмірності шарів в початковій архітектурі нейромережі

Вхідне зображення	32*32*3
1 шар згортки	28*28*6
1 шар пулінгу	14*14*6
2 шар згортки	10*10*16
2 шар пулінгу	5*5*16
1 повнозв’язний шар	120
2 повнозв’язний шар	43

1.3.3 Обрання активаційних функцій

У якості активаційних функцій доцільно обрати будь-які неспадні дифиренційовні функції, що діють на множині дійсних чисел. Можливі варіанти можуть бути такі: лінійна функція з насиченням, ReLU, сигмоїд, тангенс гіперболічний, порогова функція. Деякі з них зображені на рис. 5.

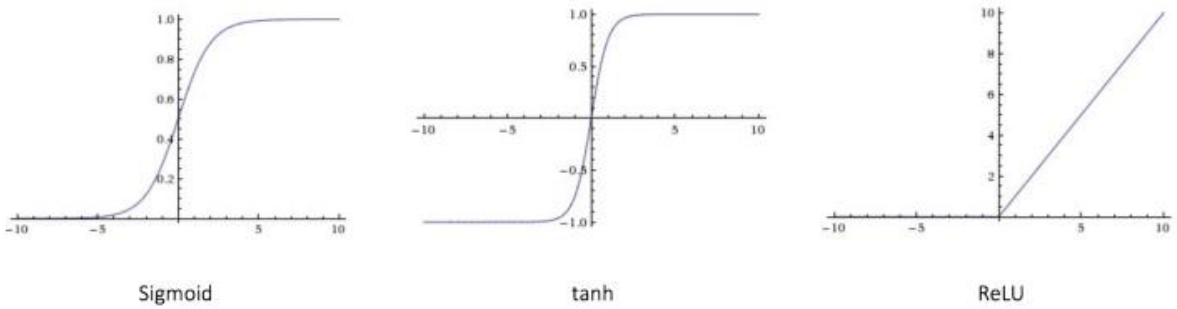


Рисунок 5 – Види активаційних функцій

Використаємо ReLU: $h(a) = \max(0, a)$, де $a = WX + b$. Застосування ReLU забезпечує дві ключові переваги, що дозволяють пришвидшити навчання нейромережі: розріженість та менша ймовірність розмиття градієнту в порівнянні з використанням інших активаційних функцій.

Розрідженість виникає, коли a < 0. Чим більша кількість нейронів з ReLU-активацією в шарі, тим більша розрідженість результуючого представлення. Сигмоїд же, наприклад, завжди повертає деяке ненульове значення, що призводить до «згущення» представлення. З точки зору швидкості навчання розріджене представлення завжди краще за «згущене».

1.3.4 Інтерпретація виходів повнозв'язного перцептрона

Побудована за такою схемою мережа має на своїх виходах деякі числа. Кожен вихід характеризує вхідний екземпляр відповідно конкретному класу. Для підвищення інтерпретованості моделі є сенс виразити отримані величини у ймовірнісній формі, так, щоб модель давала відповідь на питання: до якого классу найімовірніше належить вхідний екземпляр. Для цього до виходів повнозв’язного перцептрона підключимо Softmax-функцію (функція експоненційної нормалізації). Така функція дозволяє перетворити деякий k-мірний вектор z в k-мірний вектор s(z) дійсних чисел на проміжку від 0 до 1, такий що сума його елементів завжди дорівнює 1. Формула перетворення j-го елементу наведена нижче:

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

Тепер мережа готова дати відповідь на поставлене питання. Але для її навчання цього недостатньо.

1.3.5 Перехресна ентропія і функція втрат

Для того, аби нейронна мережа працювала згідно наших очікувань, її необхідно навчити. Для цього, в свою чергу, треба визначити «оцінку якості» роботи нейромережі. Застосуємо підхід, що називається перехресною ентропією. Представимо усі можливі класи у якості ймовірнісних векторів (one-hot encoding), так що для і-го класу і-й елемент вектора дорівнює 1, а всі інші елементи вектора дорівнюють 0. Тепер, під час навчання нейромережа отримує на вхід екземпляр розміченого датасету – наприклад, зображення дорожнього знаку «Стоп». Даний клас має відповідне векторне представлення. На виходах мережі сформуються 43 ймовірності належності знака до кожного з класів. Отже, необхідно навчити мережу таким чином, щоб вихід, що відповідає класу «Стоп», мав найближче до 1 значення серед усіх інших виходів. Інакше кажучи, треба мінімізувати різницю між виходом нейромережі і вектором знаку «Стоп».

Питання полягає в тому, яким чином виразити цю різницю між двома векторами. Є багато різних способів. Одним з ефективних є перехресна ентропія. Величина перехресної ентропії двох векторів визначається за формулою на рис. 6:

$$\hat{\mathbf{y}} \qquad\qquad \mathbf{y}$$

$$\begin{bmatrix} 0.1 \\ 0.5 \\ 0.4 \end{bmatrix} \qquad D(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_j y_j \ln \hat{y}_j \qquad \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Рисунок 6 – Перехресна ентропія двох векторів

Оскільки функція логарифму набуває від'ємних значень на інтервалі (0; 1), то чим ближче вихід нейромережі для правильного класу до 1, тим менше значення величини перехресної ентропії D. D=0 означатиме, що нейромережа класифікувала екземпляр як правильний клас з імовірністю 100%.

Отже, величина перехресної ентропії і є оцінкою якості роботи нейромережі, оцінкою якості класифікації. Така функція, що дозволяє оцінити якість роботи нейромережі залежно від її параметрів, називається функцією втрат. Чим менше значення функції втрат, тим краще відпрацювала нейромережа і тим вища точність класифікації.

1.4 Висновки

У даному розділі були розглянуті класичні архітектури згорткових нейронних мереж, що використовуються для класифікації зображень: LeNet5 і AlexNet. Першу покладено за основу для вирішення задачі класифікації дорожніх знаків.

Було проаналізовано набір даних. З аналізу очевидно, що датасет є незбалансованим, що може призвести до низької якості моделі.

В якості активаційної функції обрано ReLU, оскільки вона зменшує ймовірність розмиття градієнту при навчанні. Функція втрат виведена на основі перехресної ентропії результатів Softmax-функції і one-hot векторів класів.

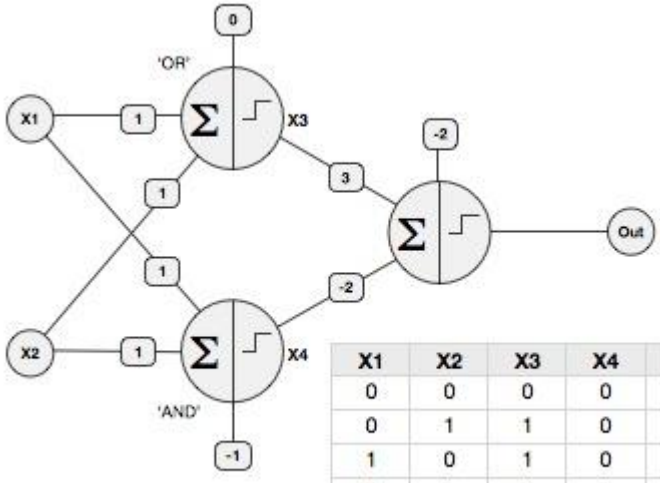
2 МАТЕМАТИЧНІ МЕТОДИ НАВЧАННЯ НЕЙРОННИХ МЕРЕЖ

2.1 Розповсюдження інформації у нейронній мережі

За своєю сутністю, нейронна мережа є обчислювальний граф, який може обчислити функцію будь-якої складності. Інформація у нейронній мережі розповсюджується, починаючи від вхідного шару нейронів, через приховані шари до вихідного шару і на вихідних нейронах отримується результат опрацювання сигналу. В мережах такого виду немає зворотних зв’язків. Прикладом нейронної мережі прямого поширення є перцептрон Розенблатта, від якого і беруть свій початок нейромережі прямого розповсюдження.

У вершинах обчислювального графа знаходяться деякі оператори, що виконують перетворення над вхідними операндами. Так, у нейронах нейронної мережі зазвичай відбувається лінійна комбінація вектору вхідних сигналів і векторів ваг.

На рис. 7 наведена реалізація функції XOR на нейронній мережі з одним прихованим шаром. У якості активації застосовано порогову функцію.



X1	X2	X3	X4	Out
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

Рисунок 7 – Реалізація функції XOR на нейронній мережі

У випадку задачі, що вирішується в даній роботі, на вхід нейромережа отримує 3-канальні пікселі зображення розміром 32x32. На виході маємо число, що характеризує якість класифікації.

2.2 Зведення задачі навчання НМ до задачі оптимізації

Необхідно побудувати модель – навчити нейронну мережу – яка зможе достатньо точно відновити функцію, що діє із множини зображень дорожніх знаків (які представлені в датасеті) на множину класів.

У попередньому розділі була отримана оцінка якості класифікації нейронної мережі. Оцінка є значенням функції втрат (перехресна ентропія), яка залежить від виходів нейронної мережі, а отже і від усіх параметрів (ваг) нейронної мережі.

Модель тим точніше апроксимує шукану залежність, чим менше значення має функція втрат. Таким чином, щоб навчити нейронну мережу, необхідно мінімізувати функцію втрат стосовно ваг нейронної мережі. Саме так задача навчання нейронної мережі зводиться до задачі багатовимірної оптимізації функції.

Отже, мета навчання нейромережі – знайти такі значення параметрів (вагів), при яких помилка класифікації буде мінімальною.

2.3 Методи оптимізації багатовимірних задач

Одним з найпопулярніших методів оптимізації, що застосовуються для оптимізації нейронних мереж, є методи засновані на обчисленні градієнта. Зокрема градієнтний спуск. Існує багато технік і модифікацій для того, щоб пришвидшити збіжність методу.

Градієнтний спуск (ГС) – це спосіб мінімізувати цільову функцію $C(\theta)$, де $\theta \in \mathbb{R}^d$ – параметри моделі, шляхом оновлення параметрів у напрямі, протилежному

градієнту цільової функції $\nabla_\theta C(\theta)$ [3]. Параметр η – означає крок алгоритму, який виконується в напрямі (локального) мінімуму. Інакше кажучи, відбувається рух в напрямі схилу по поверхні цільової функції аж поки не буде досягнуто «долини».

Існує три варіанти градієнтного спуску, які застосовуються в залежності від кількості даних, що використовуються. В залежності від кількості даних обирається «золота середина» між точністю оновлення параметрів і часом, який необхідний для оновлення.

2.3.1 Градієнтний спуск

Відомий також як пакетний градієнтний спуск, розраховує величину оновлення параметрів функції втрат на цілому датасеті.

$$\boldsymbol{\theta = \theta - \eta \nabla_\theta C(\theta)},$$

де $\boldsymbol{\theta}$ – параметри моделі;

$\boldsymbol{\eta}$ – швидкість навчання – крок градієнтного спуска, який виконується в напрямі локального мінімуму.

Оскільки необхідно розрахувати градієнти на цілому датасеті для того, щоб зробити одне оновлення, пакетний градієнтний спуск може дуже повільно сходитися і займати дуже великий об’єм оперативної пам’яті. Пакетний градієнтний спуск також не дозволяє оптимізувати модель «онлайн», наприклад, коли датасет розширюється нальоту.

Алгоритм пакетного градієнтного спуску реалізується [3] таким чином, як показано на рис. 8.

```
for i in range(nb_epochs):
  params_grad = evaluate_gradient(loss_function, data, params)
  params = params - learning_rate * params_grad
```

Рисунок 8 – Реалізація пакетного градієнтного спуску

Для заздалегідь визначеного числа епох, спочатку обчислюється вектор градієнта *params_grad* функції втрат для всього набору. Потім оновлюються параметри в напрямку анти-градієнту з деякою швидкістю навчання, що визначає, наскільки великий крок виконується. Пакетний ГС гарантовано сходиться до глобального мінімуму для опуклих поверхонь помилок і до локального мінімуму для неопуклих.

2.3.2 Стохастичний градієнтний спуск

Навпроти, стохастичний градієнтний спуск (СГС) виконує оновлення параметрів для кожного екземпляра з навчальної вибірки $(x^{(i)}, y^{(i)})$:

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \boldsymbol{\eta}\boldsymbol{\nabla}_{\boldsymbol{\theta}}\boldsymbol{C}\left(\boldsymbol{\theta}; \boldsymbol{x}^{(i)}; \boldsymbol{y}^{(i)}\right)$$

Пакетний ГС виконує надлишкові обчислення для великих масивів даних, оскільки розраховує градієнт для подібних екземплярів датасету, а потім лише раз оновлює параметри. СГС позбавлений такої надлишковості, оскільки виконує одне оновлення для кожного екземпляру. У зв'язку з цим, як правило, СГС сходиться набагато швидше, і може бути використаний для навчання нальоту.

СГС виконує часті оновлення з високою дисперсією, що призводить до сильних флуктуацій цільової функції [3], як показано на рис. 9.

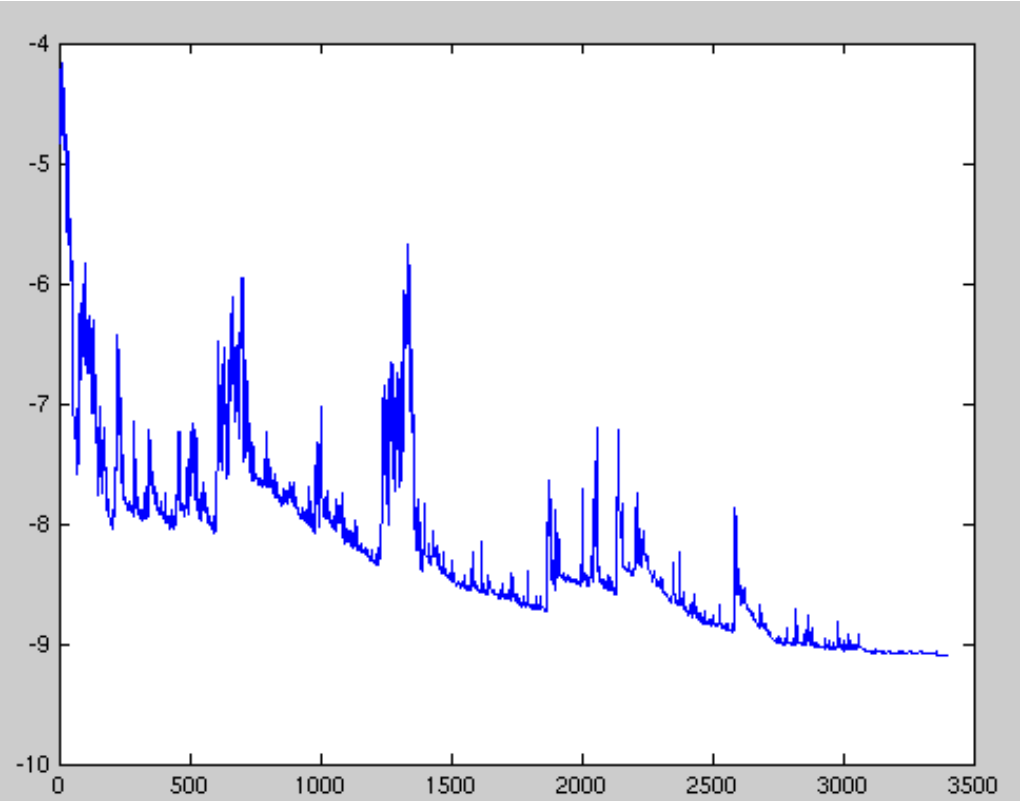


Рисунок 9 – Коливання СГС

У той час як пакетний ГС застрягне в ямі локального мінімуму, флуктуації СГС «вибивають» його з ями і, з одного боку, дозволяють йому перейти на нові і потенційно нижчі локальні мінімуми. З іншого боку, це в кінцевому підсумку ускладнює збіжність до точного мінімуму. Проте доведено, що якщо повільно зменшувати швидкість навчання, СГС поводить себе як і пакетний ГС. Тобто сходиться до локального або глобального мінімуму для неопуклої і опуклої поверхні відповідно.

На рис. 10 наведено типову реалізацію [3] стохастичного градієнтного спуску. Усередині циклу по епохах додається цикл по екземплярах датасету. Для кожного окремого екземпляру розраховується градієнт і відбувається оновлення параметрів.

```
for i in range(nb_epochs):
  np.random.shuffle(data)
  for example in data:
    params_grad = evaluate_gradient(loss_function, example, params)
    params = params - learning_rate * params_grad
```

Рисунок 10 – Реалізація стохастичного градієнтного спуску

2.3.3 Міні-пакетний градієнтний спуск

Міні-пакетний градієнтний спуск є «золотою серединою» між пакетним ГС і стохастичним ГС.

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \boldsymbol{\eta}\boldsymbol{\nabla}_{\boldsymbol{\theta}}\boldsymbol{C}\left(\boldsymbol{\theta}; \boldsymbol{x}^{(\boldsymbol{i}:\boldsymbol{i}+\boldsymbol{n})}; \boldsymbol{y}^{(\boldsymbol{i}:\boldsymbol{i}+\boldsymbol{n})}\right)$$

Таким чином, міні-пакетний ГС зменшує флуктуації під час оновлення параметрів, що може призвести до кращої збіжності. Типовий розмір міні-пакету обирається між 50 і 256 екземплярами. Зазвичай для навчання нейромереж обирається саме міні-пакетний ГС. Часто, коли мова йде про використання стохастичного ГС, мається на увазі сама міні-пакетний ГС.

Однак застосування міні-пакетного ГС не гарантує збіжність. Виникають декілька викликів, які необхідно вирішити для ефективної оптимізації:

- Вибір правильної швидкості навчання може бути складним. Мала швидкість навчання призводить до вкрай повільної збіжності, в той час як швидкість навчання, що є занадто великою, може перешкоджати збіжності, викликаючи коливання функції втрат навколо мінімуму або навіть призводити до розбіжності.
- Графіки зміни швидкості навчання [4] намагаються регулювати швидкість навчання в процесі навчання, наприклад, за допомогою «відпалу» («annealing»), тобто зменшення швидкості навчання відповідно до заздалегідь визначеного розкладу, або коли значення цільової функції між епохами падає нижче деякого

порогового значення. Ці графіки і пороги, однак, повинні бути визначені заздалегідь, і, таким чином, не можуть пристосовуватися до характеристик конкретного набору даних [5].

- Більше того, одна й та сама швидкість навчання застосовується для оновлення всіх параметрів. Якщо дані розріджені і фічі мають дуже різні частоти, не треба оновлювати всі фічі однаково. Можливо, краще сильніше оновлювати фічі, що зустрічаються рідко.
- Ключовою проблемою мінімізації неопуклих функцій є «застрягання» в локальних мінімумах. Дофін і ін. [6] стверджує, що труднощі виникають справді не в локальних мінімумах, а в сідлових точках. Ці сідла зазвичай оточені плато, що унеможливлює правильну роботу стохастичного ГС, так як градієнт близький до нуля по всіх напрямках.

2.3.4 Техніка моменту

Стохастичний ГС має труднощі з пошуком ярів [7] (областей, в яких криві поверхні набагато похиліші в одному вимірі, ніж в іншому), які зазвичай зустрічаються навколо локальних оптимумів. В таких випадках, Стохастичний ГС стрибає по схилах яру, дуже повільно наближаючись до локального оптимуму (рис. 11).

Рисунок 11 - Стохастичний ГС без моменту

Метод (або техніка) моменту – це метод, який допомагає прискорити Стохастичний ГС у відповідному напрямку і згладжує коливання, як можна бачити на рис. 12.

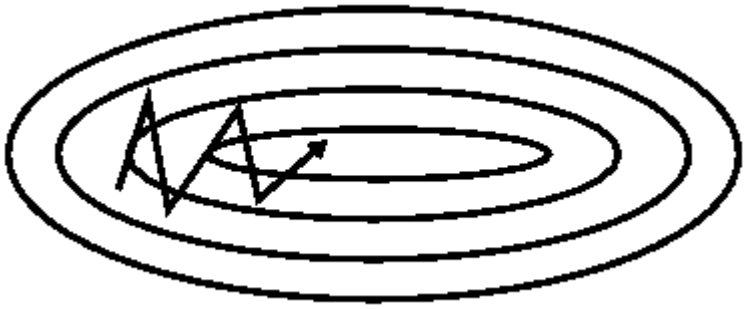


Рисунок 12 – Стохастичний ГС з моментом

Це відбувається завдяки додаванню компоненти γ, що містить інформацію про оновлення вагів на попередньому кроці. Концептуально, це означає використання накопичених з попередніх кроків знань про те, в якому напрямку відбувається оптимізація. Тепер рівняння оновлення параметрів можна переписати наступним чином:

$$\boldsymbol{v_t = \gamma v_{t-1} + \eta \nabla_\theta C(\theta)} \quad (1)$$

$$\boldsymbol{\theta = \theta - v_t}$$

Величина моменту γ у формулі (1) зазвичай встановлюється близько 0.9. Момент буде збільшувати величину оновлення параметрів для вимірів, градієнти яких вказують в тому ж напрямі, куди відбувається рух, і зменшувати величину оновлення для вимірів, градієнти яких змінюють напрям.

2.3.5 Метод Адам

Метод Адам (Adaptive Moment Estimation) [8] дозволяє регулювати швидкість навчання залежно від параметра, виконуючи більші оновлення для

рідких параметрів і маленькі оновлення для частих параметрів. Метод використовує не лише накопичені значення градієнтів з попередніх кроків, а і накопичені значення квадратів градієнтів. Накопичення відбувається шляхом так званого експоненційного розпаду середнього (exponentially decaying average). Завдяки цьому значення з останніх кроків роблять набагато більший внесок в сумарне значення, ніж значення градієнтів з перших кроків.

$$\boldsymbol{m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t} \quad (2)$$

$$\boldsymbol{v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2}, \quad (3)$$

де $\boldsymbol{m_t}$ – оцінка першого моменту (середнє);

$\boldsymbol{v_t}$ – оцінка другого моменту (дисперсія) градієнту.

Оскільки m_t і v_t у формулах (2) і (3) ініціалізуються нулями, було помічено, що вони тяжіють до нулів, особливо під час початкових кроків і, особливо, коли коефіцієнт розпаду малий (β_1 і β_2 близькі до 1). Для вирішення цієї проблеми на значення моментів накладається штраф:

$$\widehat{\boldsymbol{m_t}} = \frac{\boldsymbol{m_t}}{\boldsymbol{1 - \beta_1^t}}$$

$$\widehat{\boldsymbol{v_t}} = \frac{\boldsymbol{v_t}}{\boldsymbol{1 - \beta_2^t}}$$

Далі отримані значення можна використовувати для оновлення параметрів за формулою:

$$\boldsymbol{\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\widehat{v_t}} + \epsilon} \widehat{m_t}}$$

Автори методу пропонують наступні значення для параметрів: $\beta_1 = 0.9$, $\beta_2 = 0.999, \epsilon = 10^{-8}$. Емпірично доведена конкурентність методу Адам у порівнянні з аналогами як то метод Нестерова, Adagrad, Adadelta, RMSprop [8].

2.4 Метод зворотного розповсюдження помилки

Метод зворотного поширення помилки (англ. Backpropagation) - метод навчання нейронної мережі, що базується на градієнтних методах оптимізації. Це ітеративний градієнтний алгоритм, який використовується з метою мінімізації помилки роботи нейронної мережі і отримання бажаного виходу.

Основна ідея цього методу полягає в поширенні сигналів помилки від виходів мережі до її входів, в напрямку, зворотному прямому поширенню сигналів у звичайному режимі роботи [9]. Для можливості застосування методу зворотного поширення помилки передавальна функція нейронів повинна бути диференційована.

Метою застосування методу зворотного поширення помилки є знаходження кроку зміни вагів нейронної мережі для методу градієнтного спуску або його модифікації. Необхідно зрозуміти, як зміна параметрів мережі призводить до зміни значення функції втрат. Інакше кажучи, якщо визначені функція втрат *C*, матриці вагів *w* і вектори зміщень *b*, треба знайти значення похідних $\frac{\partial C}{\partial w_{jk}^{l}}$ і $\frac{\partial C}{\partial b_{j}^{l}}$.

Для того щоб порахувати значення похідних, введемо проміжну величину δ_j^l, яку назвемо *помилкою* в *j*-му нейроні *l*-го шару і визначимо як значення часткової похідної функції втрат по виходу лінійної комбінації вектору вагів і входів нейрона:

$$\boldsymbol{\delta_j^l = \frac{\partial C}{\partial z_j^l},} \tag{4}$$

У формулі (4) закладено глибокий математичний сенс помилки. Якщо значення часткової похідної функції втрат по виходу лінійної комбінації – велике за модулем число, це означає, що даний вектор ваг є неоптимальним, а відповідно помилка є великою. І навпаки: якщо $\frac{\partial C}{\partial z_j^l}$ у формулі (4) є близьким до нуля, це

означає, що майже неможливо покращити значення функції втрат за рахунок зміни зваженого входу нейрону. А відповідно, вектор ваги знаходиться біля оптимуму і помилка в нейроні невелика. Саме тому помилка може бути визначена як $\frac{\partial C}{\partial z_j^l}$.

Метод зворотного поширення помилки ґрунтується на чотирьох фундаментальних рівняннях. Разом ці рівняння дозволяють визначити помилку нейронів, а відповідно і градієнт функції втрат.

2.4.1 Рівняння помилки у вихідному шарі

Дане рівняння дозволяє отримати помилку на кожному нейроні вихідного шару.

$$\boldsymbol{\delta_j^L = \frac{\partial C}{\partial a_j^L} \sigma'(z_j^L)} \tag{5}$$

Використовуючи правило похідної складної функції, отримуємо залежність величини помилки від зваженої суми входів нейрона вихідного шару. Кожна з компонент даного рівняння є простою обчислювальною задачею. У формулі (5) z_j^L обчислюється під час прямого ходу мережі, а $\sigma'(z_j^L)$ залежить від форми передатної функції. Наприклад, для ReLU це буде просто порогова функція. Складність обчислення $\frac{\partial C}{\partial a_j^L}$ залежить від форми функції втрат. Так, обчислення похідної функції перехресної ентропії від активації є, очевидно, простою задачею.

Формула (5) є покомпонентним виразом для δ^L. Однак простіше для сприйняття переписати рівняння у векторній формі:

$$\boldsymbol{\delta^L = \nabla_a C \cdot \sigma'(z^L),} \tag{6}$$

де $\boldsymbol{\nabla_a C}$ – градієнт, вектор, чиї компоненти – часткові похідні $\boldsymbol{\frac{\partial C}{\partial a_j^L}}$.

Таким чином, $\nabla_a C$ виражає величину зміни *C* відносно виходів нейромережі. Таким чином рівняння записане у векторній формі (див. формулу (6)) і може бути просто обчислене за допомогою бібліотеки NumPy.

2.4.2 Рекурентне рівняння помилки у довільному шарі

Одразу запишемо рівняння у векторній формі.

$$\boldsymbol{\delta^l} = ((\boldsymbol{w^{l+1}})^{\boldsymbol{T}}\boldsymbol{\delta^{l+1}}) \cdot \boldsymbol{\sigma'}(\boldsymbol{z^l}), \tag{7}$$

де w^{l+1} – транспонована матриця вагів (*l+1)-го* шару.

Інтуїтивно, вираз $((w^{l+1})^T \delta^{l+1})$ демонструє, як помилка розповсюджується від (*l+1)-го* шару до *l-го.*

Комбінуючи формули (6) і (7), можна розрахувати вектор помилок на будь-якому шарі нейронної мережі. Починаючи з розрахунку вектора помилок на вихідному шарі, поступово розраховуємо помилку на кожному наступному шарі, рухаючись від виходів мережі до її входів.

2.4.3 Рівняння зміни функції втрат відповідно до зміни зміщення

Покомпонентне рівняння виглядає наступним чином:

$$\frac{\boldsymbol{\partial} \mathrm{C}}{\boldsymbol{\partial b_j^l}} = \boldsymbol{\delta_j^l}$$

Таким чином, помилка δ_j^l точно дорівнює величині зміни функції втрат стосовно зміщення b_j^l. Переписавши в векторній формі, маємо:

$$\frac{\boldsymbol{\partial} \mathrm{C}}{\boldsymbol{\partial b}} = \boldsymbol{\delta} \tag{8}$$

2.4.4 Рівняння зміни функції втрат відповідно до зміни ваг

Покомпонентне рівняння виглядає наступним чином:

$$\frac{\partial \mathrm{C}}{\partial w_{jk}^{l}} = a_k^{l-1}\delta_j^l$$

Дане рівняння дозволяє визначити, як зміна вагів впливає на зміну значення функції втрат через відомі значення активацій $(l-1)$-*го* шару і помилки на l-*му* шарі. Рівняння може бути переписане без надлишкових індексів:

$$\frac{\partial C}{\partial w} = a_{in}\delta_{out} \tag{9}$$

З даного рівняння очевидно, що якщо значення активації нейрона близьке до нуля, то і компонента градієнту $\frac{\partial C}{\partial w}$ також буде малою. В такому випадку, кажемо, що вага навчається повільно, маючи на увазі, що вона не змінюється надто під час градієнтного спуску. Інакше кажучи, ваги, що зважують нейрони з малою активацією, навчаються повільно.

2.4.5 Алгоритм зворотного розповсюдження помилки

Чотири вище викладених рівняння забезпечують методику обчислення градієнта функції втрат. Підсумовуючи, знайти усі компоненти градієнту можна наступним чином:

1) Вхід. Розрахувати відповідні активації вхідного шару.
2) Прямий хід: для кожного шару *l = 2, 3, … L* розрахувати $z^l = w^l a^{l-1} + b^l$ та $a^l = \sigma(z^l)$.
3) Помилка у останньому шарі: обчислити вектор $\delta^L = \nabla_a C \cdot \sigma'(z^L)$.
4) Розрахувати помилку на кожному з нейронів за рекурентною формулою: для будь-якого *l = L-1, L-2 … 2* обчислити $\delta^l = ((w^{l+1})^T \delta^{l+1}) \cdot \sigma'(z^l)$.

5) Компоненти градієнту обчислюються за формулами (8) і (9).

2.5 Висновки

У даному розділі показано, яким чином задача навчання нейронної мережі зводиться до задачі багатокритеріальної оптимізації. Були розглянуті деякі методи і техніки оптимізації нейронних мереж, серед яких пакетний і стохастичний градієнтний спуск, техніка моменту, метод Адам.

Під час оптимізації часто виникають проблеми, як то застрягання у локальному мінімумі, замалий або завеликий крок навчання тощо. Деякі з названих вище методів вирішують ці проблеми. Наразі метод Адам визнаний чи не найефективнішим методо оптимізації нейронних мереж. У даній роботі використовується реалізація методу Адам в бібліотеці Tensorflow.

Був розглянутий ключовий метод, що лежить в основі навчання нейронних мереж – метод зворотного поширення помилки. Він дозволяє обчислити компоненти градієнта функції втрат відносно параметрів моделі.

Таким чином, даний розділ дає уявлення про те, як відбувається навчання нейронних мереж і як вирішуються проблеми, що виникають під час навчання.

3 АНАЛІЗ ПІДХОДІВ ДО ПЕРЕДОБРОБКИ ДАНИХ І ЇХ ВПЛИВ НА РЕЗУЛЬТАТ КЛАСИФІКАЦІЇ

У даному розділі буде досліджено вплив різних технік передобробки даних на точність класифікації моделі. Архітектура початкової моделі описана в підрозділі 1.3. Основні кількісні характеристики датасету наведені в таблиці 2. Встановимо наступні гіперпараметри моделі: швидкість навчання рівна 0.001, кількість епох рівна 20. На початковій архітектурі зі швидкістю навчання рівній 0.001 і кількістю епох рівній 20 модель показує точність класифікації рівну 87%

3.1 Нормалізація зображень

Вхідні дані можуть бути нормалізовані таким чином, щоб вони мали математичне очікування рівне 0. На практиці доведено, що нормалізація датасету робить модель більш стійкою до ініціалізації параметрів мережі і пришвидшує збіжність методу оптимізації.

У даній роботі застосування нормалізації призвело до суттєвого покращення результату класифікації. Точність класифікації становить 90.8% проти початкових 87%

3.2 Генерація додаткових даних

У підпункті 1.2.1 було наведено гістограму частот дорожніх знаків, представлених у датасеті. З гістограми очевидно, що розподіл надто нерівномірний. Деякі класи недопредставлені в датасеті (250 екземплярів), а інші – перепредставлені (більше 2000 екземплярів). В такому випадку навчена модель може тяжіти до перепредставлених класів. Для вирішення цієї проблеми можна

згенерувати додаткові дані, що зробить розподіл більш рівним і розширить тренувальну вибірку.

3.2.1 Зміщення і поворот

Один із способів розширити датасет корисними даними – це генерація зображень під різними кутами огляду. OpenCV чи SciPy дозволяють легко це зробити, застосувавши афінні перетворення, такі як поворот і перенесення. Афінні перетворення – це такі перетворення лінійного простору, при яких паралельні прямі залишаються паралельними.

У даній роботі для зміщення і повороту зображення на деякі випадкові величини реалізовано функцію generate_example (рис. 13):

```
def generate_example(image):
    scipy.ndimage.interpolation.shift(image, [random.randrange(-5, 5), random.randrange(-5, 5), 0])
    scipy.ndimage.interpolation.rotate(image, random.randrange(-15, 15), reshape = False)
    return image
```

Рисунок 13 – Функція для зміщення і повороту зображення

Генерація сорока тисяч екземплярів зі зміщенням і поворотом допомогла підвищити точність класифікації до 92.5%

3.2.2 Яскравість і контраст

Інакше датасет можна розширити змінюючи яскравість чи контраст зображень. Генеруючи нові зображення з меншою і більшою яскравістю, меншим і більшим контрастом, модель стає стійкішою до можливих випадків вхідних даних.

Для зміни яскравості реалізована функція augment_brighness (рис. 14).

```
def augment_brightness(image):
    image = cv2.cvtColor(image, cv2.COLOR_RGB2HSV)
    random_bright = .25 + np.random.uniform()
    image[:,:,2] *= random_bright
    image = cv2.cvtColor(image, cv2.COLOR_HSV2RGB)
    return image
```

Рисунок 14 – Функція для зміни яскравості

Наприклад, для знаку «Ліворуч або прямо» маємо близько 1500 нових екземплярів. Деякі з них наведені на рис. 15.

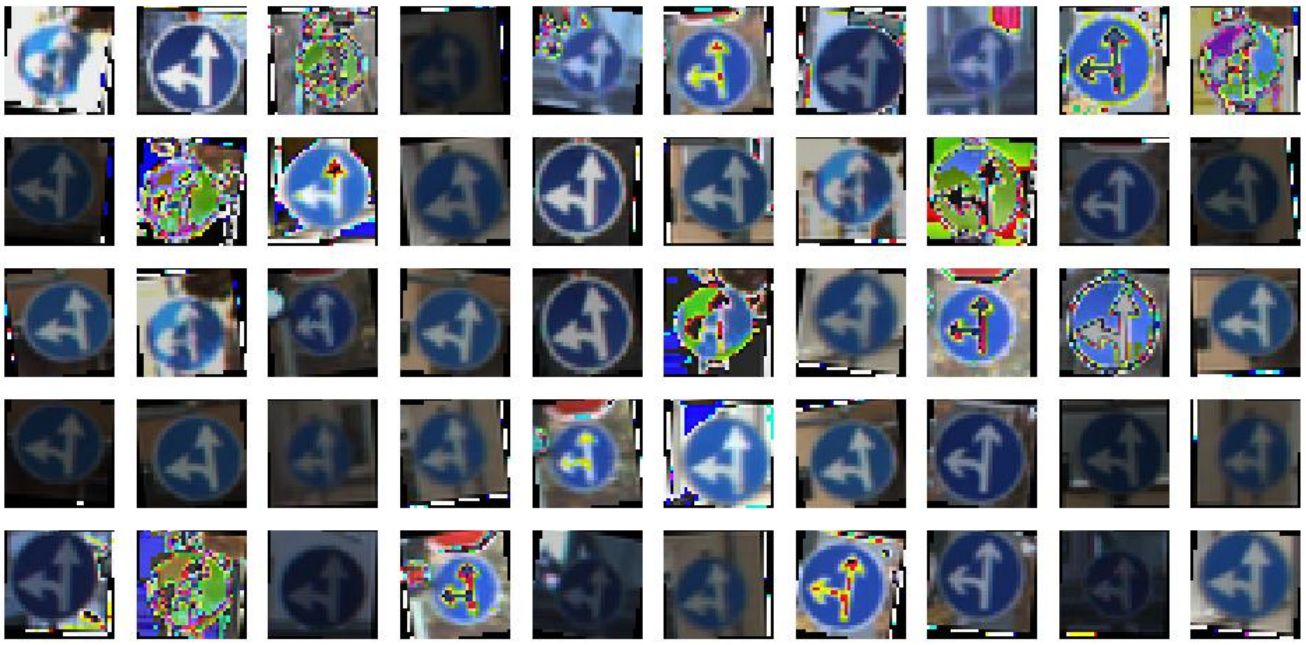


Рисунок 15 – Згенеровані екземпляри класу «Ліворуч або прямо»

Генерація сорока тисяч екземплярів зі зміщенням, поворотом і зміненою яскравістю (див. рис. 15) допомогла підвищити точність класифікації на 3.5%. Після застосування описаних вище перетворень маємо точність класифікації 93.5%. Через слабкі обчислювальні потужності машини не вдалося згенерувати більше екземплярів. Більше розширення датасету дало б ще кращий результат.

3.3 Висновки

У розділі описані декілька підходів до попередньої обробки навчального набору даних з метою підвищити точність класифікації. До датасету були застосовані нормалізація зображень, генерація нових екземплярів шляхом афінних перетворень і зміни яскравості.

Точність класифікації початкової моделі на сирому датасеті складає 87%. Застосування нормалізації зображень дозволило підвищити точність класифікації до 90%. Ще 3.5% точності були отримані завдяки генеруванню і аугментації зображень.

4 ДОСЛІДЖЕННЯ ВПЛИВУ ОСОБЛИВОСТЕЙ АРХІТЕКТУТРИ МЕРЕЖІ НА РЕЗУЛЬТАТ КЛАСИФІКАЦІЇ

Архітектура нейронної мережі – один з найважливіших аспектів, що визначає точність класифікації, а, відповідно, і загальну якість моделі. На жаль, не існує універсального рецепту проектування архітектури нейромережі. Кожна окрема задача вимагає особливого підходу до проектування, а відтак і власної архітектури.

Спираючись на деякі апріорні припущення і вдалий досвід вирішення схожих задач, необхідно провести ряд експериментів, аби зрозуміти, які архітектурні компоненти дійсно впливають на точність класифікації.

При цьому необхідно пам'ятати, що нейронна мережа – це обчислювальний граф з дуже великою кількістю параметрів, зберігання яких і маніпуляції над якими вимагають значних обчислювальних ресурсів.

У даному розділі будуть описані спроби відшукати оптимальну архітектуру нейронної мережі, зважаючи на наявні обчислювальні ресурси і особливості задачі, що вирішується.

4.1 Перехід від RGB до напівтонів сірого

Така маніпуляція над вхідними даними має ключову відмінність від попередніх. Вона вимагає зміни архітектури нейромережі. В підрозділі 1.3 була обрана початкова архітектура, що приймає на вхід зображення, представлене тривимірним масивом розмірністю 32x32x3. Тут 32 – це висота і ширина зображення, а 3 – позначає кількість каналів: R, G, B.

Припустимо, що колір зображення, що передається каналами R, G, B не несе важливої інформації для моделі. Тоді доцільним виглядає зменшення кількості

параметрів моделі шляхом перетворення зображення з RGB-простору у напівтони сірого.

Насправді, таке перетворення є переходом з RGB-простору у YUV-простір, де Y-компонента – яскравість, а дві інші – кольороворізносні. На рис. 16 наведений приклад зображення у YUV-просторі, розкладеного за компонентами.

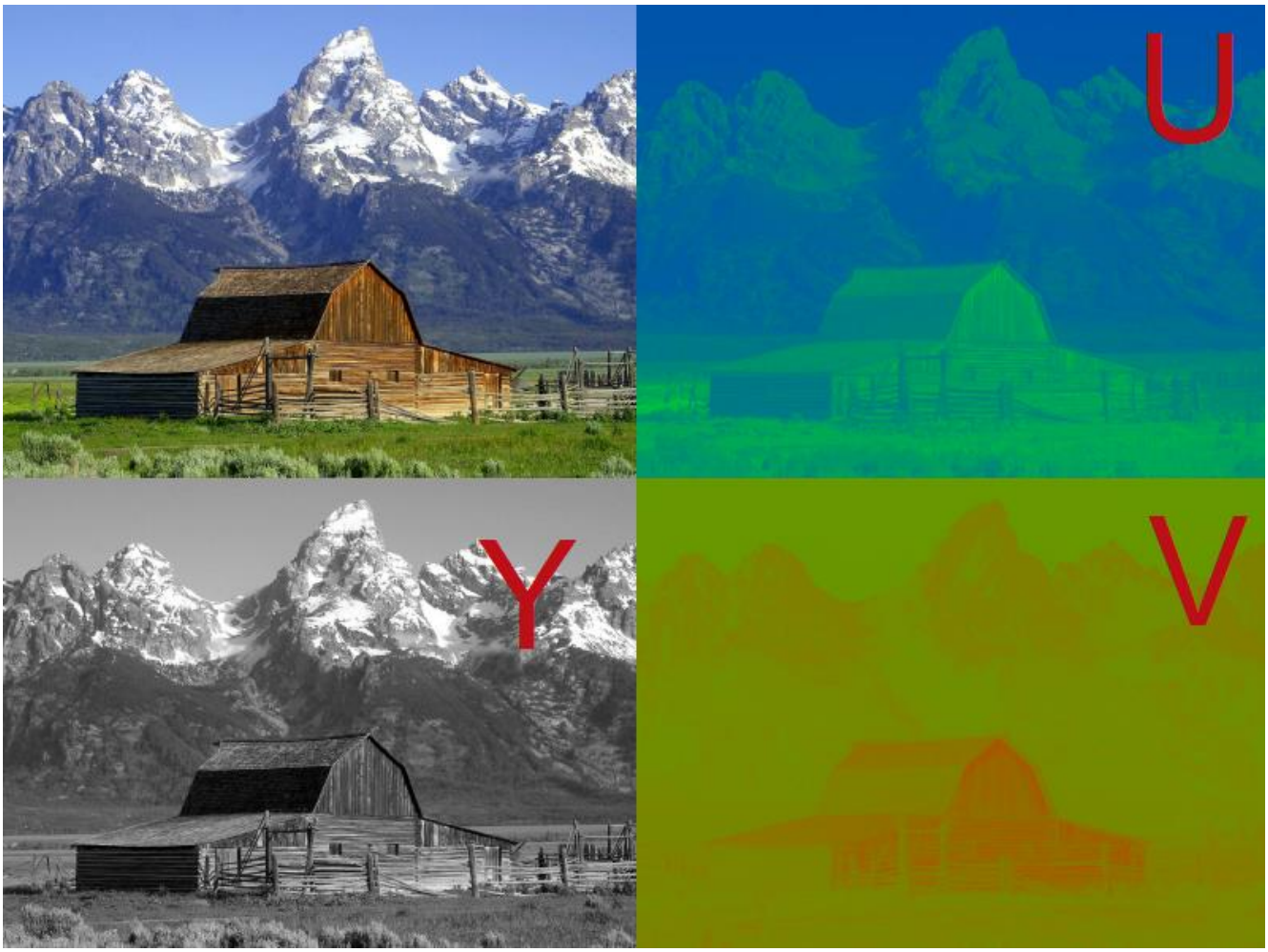


Рисунок 16 - Зображення у YUV-просторі, розкладене за компонентами

Таким чином, перехід до напівтонів сірого є по суті виділенням Y-компоненти. Перехід відбувається за наступною формулою:

$$\mathbf{Y = 0.299 * R + 0.587 * G + 0.114 * B}$$

Нижче наведено фрагмент згенерованого набору даних для класу «Ліворуч або прямо» (рис. 17).

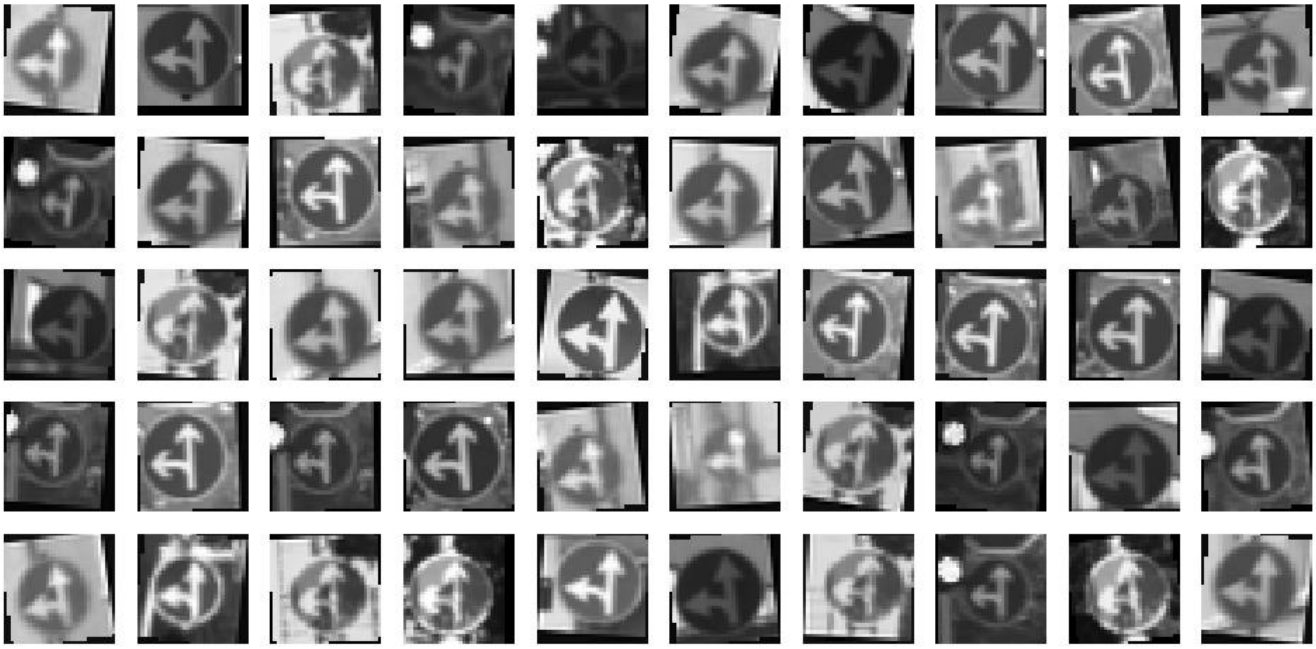


Рисунок 17 – Фрагмент згенерованого набору даних для класу «Ліворуч або прямо»

Кожне таке зображення представлене масивом, розмірність якого 32x32x1. Відповідно необхідно змінити початкову архітектуру нейромережі таким чином, щоб вона могла приймати на вхід таке зображення. Для цього достатньо використати фільтри розмірністю 5x5x1 замість фільтрів 5x5x3.

Точність класифікації складає 92.7% при тому, що розмірність вхідних даних зменшилась втричі. Незважаючи на те, що перетворення зображень у напіввідтінки сірого не призвело до бажаного покращення класифікації, обсяг задачі скоротився.

4.2 Зміна розміру шарів

В підрозділі 1.3 описана початкова архітектура нейронної мережі. На вхід мережа приймає зображення, представлене масивом дійсних чисел розмірністю

32x32x3. Далі розташовані два згорткових шари з шістьма і шістнадцятьма фільтрами 5x5 відповідно, за кожним з яких розташовано шар пулінгу 2x2. Далі знаходиться повнозв'язний класифікатор. Описана архітектура зображена на рис.18.

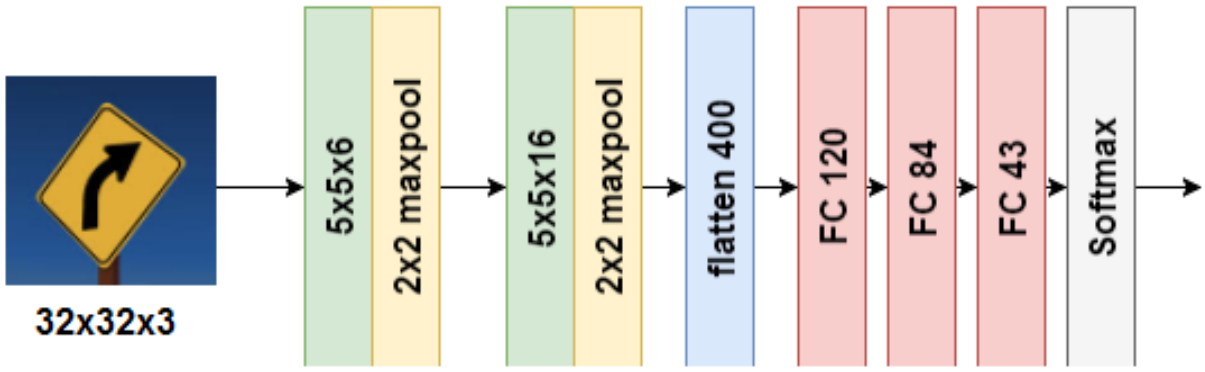


Рисунок 18 – Початкова архітектура нейронної мережі

Для покращення результату класифікації необхідно отримати якомога більше характеристик набору даних. Кількість характеристик, що їх виділяє згорткова нейронна мережа, залежить зокрема від кількості фільтрів. Збільшення кількості фільтрів призводить до стрімкого зростання кількості параметрів моделі. Тому необхідно розраховувати розмірність шарів, враховуючи характеристики обчислювальної машини, на якій відбувається навчання нейронної мережі.

Масштабуючи нейронну мережу вшир, отримуємо архітектуру (рис. 19), де спочатку йдуть два згорткових шари з шістнадцятьма і тридцять двома фільтрами 5x5 відповідно, за кожним з яких розташовано шар пулінгу 2x2. Далі знаходиться повнозв'язний класифікатор з шириною шарів 800, 400, 43.

Масштабування архітектури дає значний приріст точності класифікації – 96.3% після 20 епох. Проте підвищення точності вимагає більшого часу, що відводиться на навчання, і більше пам'яті для зберігання моделі.

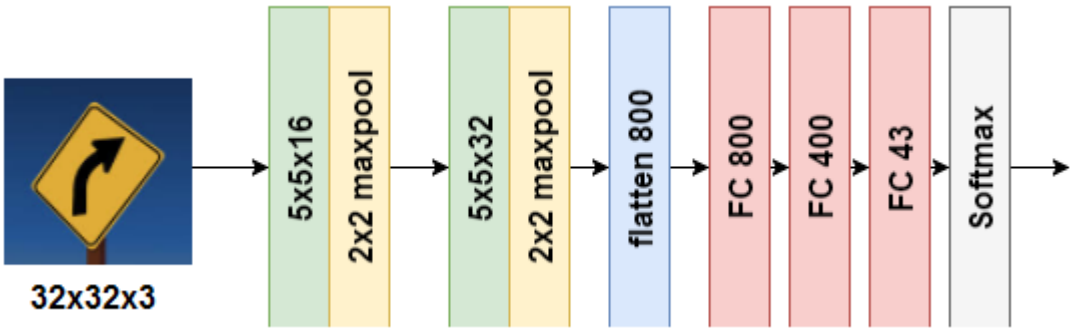


Рисунок 19 – Масштабована архітектура нейронної мережі

4.3 Multi-Scale архітектура

В традиційних згорткових нейронних мережах вихід останнього згорткового шару під'єднується до класифікатора. В роботі Пьера Серманета і Яна ЛеКуна [10], виходи всіх згорткових шарів подаються на вхід класифікатору. Такий підхід дозволяє використовувати при класифікації не тільки характеристика найвищого рівня, але й характеристики нижчого рівня, які точніше описують певні особливості зображень. Суть такого підходу полягає в тому, щоб дати класифікатору характеристики різних масштабів.

Перетворивши отриману в попередньому підрозділі архітектуру (див. рис. 19), маємо наступну (рис. 20):

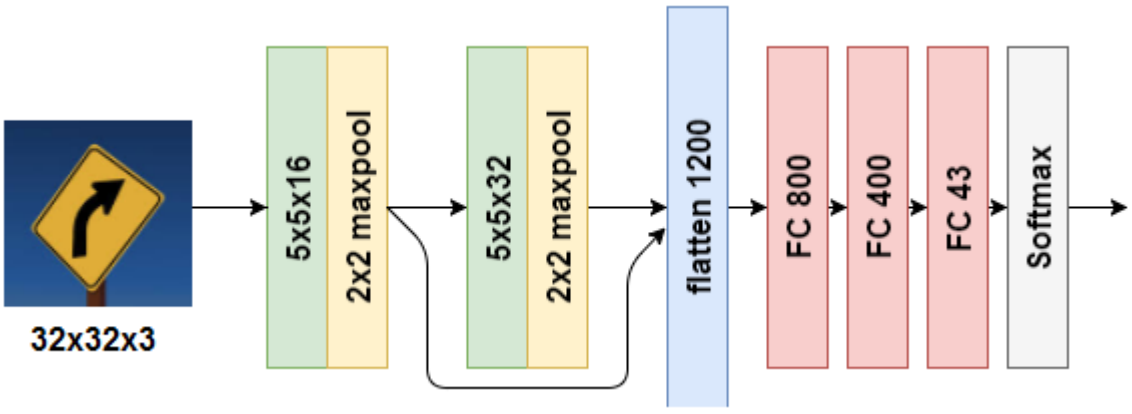


Рисунок 20 – Multi-Scale архітектура

Використання Multi-Scale архітектури дозволило підвищити точність класифікації на валідаційній вибірці до 97.4%

4.4 Висновки

Отже, перетворення зображень із RGB-простору в напівтони сірого не дало очікуваного покращення точності класифікації, хоча дозволило зменшити розмір вхідних даних втричі.

Масштабування нейронної мережі вшир за рахунок збільшення кількості фільтрів призвело до підвищення точності на 3% (96.5%).

Використання Multi-Scale архітектури дозволило покращити точність моделі ще на 1% (97.5%).

5 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проводиться оцінка основних характеристик програмного продукту, розробленого для вирішення задачі класифікації дорожніх знаків.

Програмний продукти призначено для використання на персональних комп’ютерах під управлінням будь-якої операційної системи.

Нижче наведено аналіз різних варіантів реалізації модулю з метою вибору оптимальної, з огляду при цьому як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. Як прямі, так і побічні витрати розподіляються по продуктам та послугам у залежності від потрібних на кожному етапі виробництва обсягів ресурсів. Виконані на цих етапах дії у контексті метода ФВА називаються функціями.

Мета ФВА полягає у забезпеченні правильного розподілу ресурсів, виділених на виробництво продукції або надання послуг, на прямі та непрямі витрати. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій.

Фактично цей метод працює за таким алгоритмом:

- визначається послідовність функцій, необхідних для виробництва продукту. Спочатку – всі можливі, потім вони розподіляються по двом групам: ті, що

впливають на вартість продукту і ті, що не впливають. На цьому ж етапі оптимізується сама послідовність скороченням кроків, що не впливають на цінність і відповідно витрат.

- для кожної функції визначаються повні річні витрати й кількість робочих часів.
- для кожної функції на основі оцінок попереднього пункту визначається кількісна характеристика джерел витрат.
- після того, як для кожної функції будуть визначені їх джерела витрат, проводиться кінцевий розрахунок витрат на виробництво продукту.

5.1 Постановка задачі техніко-економічного аналізу

У роботі застосовується метод ФВА для проведення техніко-економічний аналізу розробки системи аналізу нелінійних нестаціонарних процесів. Оскільки основні проектні рішення стосуються всієї системи, кожна окрема підсистема має їм задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу гетероскедастичних процесів в економіці та фінансах.

Відповідно цьому варто обирати і систему показників якості програмного продукту.

Технічні вимоги до продукту наступні:

- програмний продукт повинен функціонувати на персональних комп’ютерах із стандартним набором компонент;
- забезпечувати високу швидкість обробки даних та відклик користувачеві у реальному часі;
- забезпечувати зручність і простоту взаємодії з користувачем;

- забезпечувати можливість зручного масштабування та обслуговування;
- передбачати мінімальні витрати на впровадження програмного продукту.

5.1.1 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка програмного продукту, який вирішує задачу класифікації дорожніх знаків та будує відповідну модель.

Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

F_1 – вибір мови програмування;

F_2 – вибір фреймворка машинного навчання;

F_3 – вибір середовища розробки.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F_1:

а) Python;

б) C++;

Функція F_2:

а) Tensorflow

б) Caffe

Функція F_3:

а) Jupyter Notebook

б) Visual Studio

5.1.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 21). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (табл. 4).

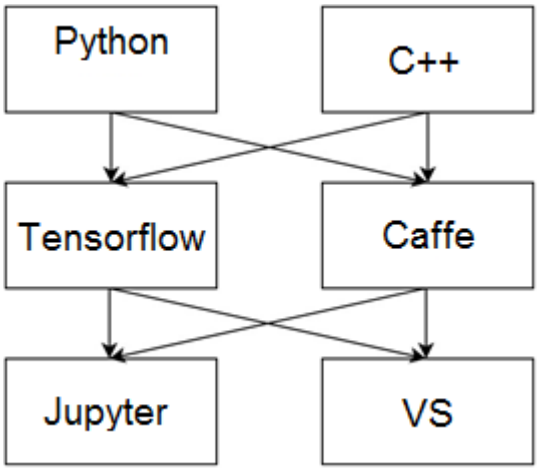


Рисунок 21 – Морфологічна карта

Морфологічна карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

Таблиця 4 – Позитивно-негативна матриця

Основні функції	Варіанти реалізації	Переваги	Недоліки
F1	*А*	Кросплатформений, Займає менше часу при написанні коду	Низька швидкодія, більший час на виконання операцій
	Б	Висока швидкодія, оптимізація пам'яті	Займає більше часу при написанні коду

Таблиця 4 (закінчення)

F2	А	Безкоштовність, чудова документація	Відсутність нативної реалізації під Windows
	Б	Надійність, швидкість, безкоштовність	Необхідність додаткової інсталяції, низький рівень користувацької підтримки
F3	*А*	Не займає багато пам’яті. Можно виконувати окремі ділянки програмного коду	Повільний
	Б	Висока інтегрованість з сервісами MS, дуже багато додаткових інструментів	Займає дуже багато пам'яті

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція *F1:*

Оскільки реалізація програмного коду для вирішення поставленої задачі є концептуально складною, необхідно обрати мову, що спрощує написання коду, тому варіант Б має бути відкинутий.

Функція *F2:*

Вибір фреймворку машинного навчання не відіграє велику роль у даному програмному продукту, тому вважаємо варіанти а) та б) гідними розгляду.

Функція *F3:*

Оскільки, програмний продукт реалізується мовою Python, використовуємо варіант А як найбільш сумісний з цією мовою.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

1. F1а – F2а – F3а

2. F1а – F2б – F3а

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

5.2 Обґрунтування системи параметрів ПП

5.2.1 Опис параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри: На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

- X1 – швидкодія мови програмування;
- X2 – об’єм пам’яті для збереження даних;
- X3 – час обробки даних;

- X4 – потенційний об’єм програмного коду.

X1: Відображає швидкодію операцій мови програмування залежно від обраної серверної технології.

X2: Відображає об’єм пам’яті в оперативній пам’яті персонального комп’ютера, необхідний для збереження та обробки даних під час виконання програми.

X3: Відображає час, який витрачається на дії.

X4: Показує розмір програмного коду який необхідно створити безпосередньо розробнику.

5.2.2 Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у таблиці 5.

Таблиця 5 – Основні параметри ПП

Назва Параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія мови програмування	X1	Оп/мс	19000	11000	2000
Об’єм пам’яті для збереження даних	X2	Мб	32	16	8
Час обробки запитів користувача	X3	мс	200	100	50
Потенційний об’єм програмного коду	X4	кількість строк коду	2000	1500	1000

За даними, наведеними у таблиці 5, будуються графічні характеристики параметрів – рис. 21-24.

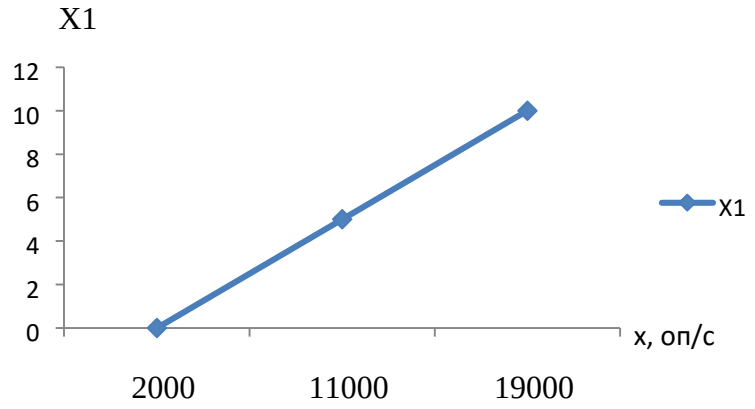


Рисунок 22 – швидкодія мови програмування

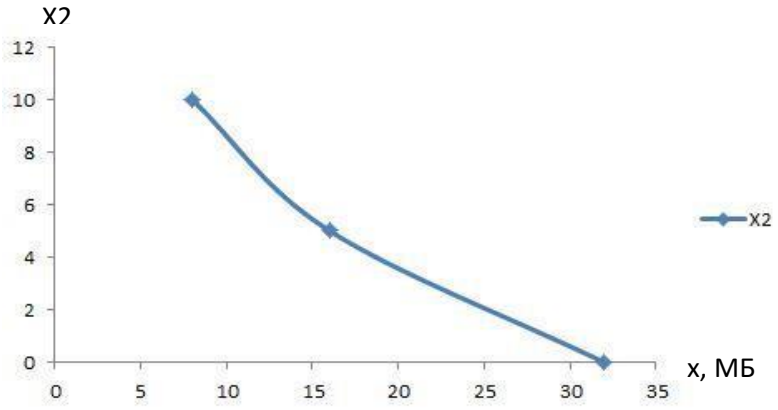


Рисунок 23 – об’єм пам’яті для збереження даних

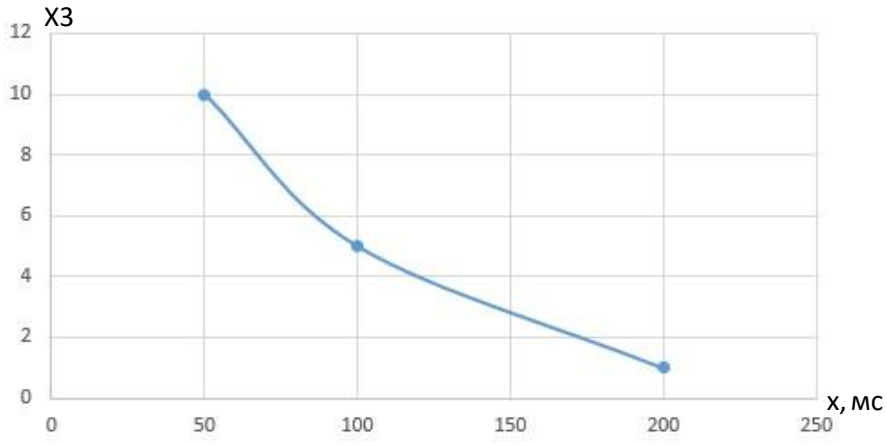


Рисунок 24 – час обробки запитів користувача

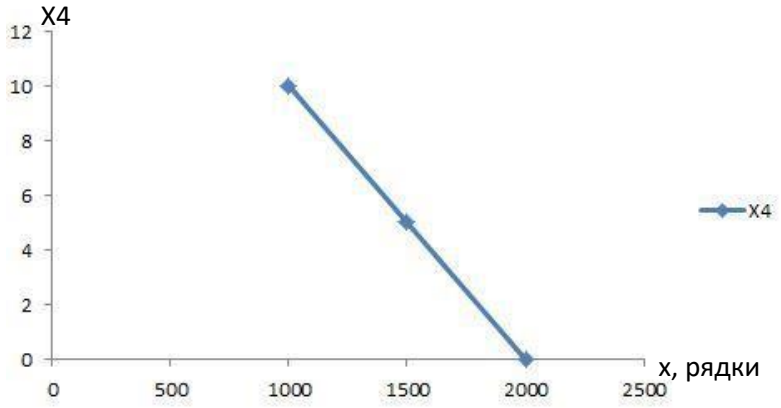


Рисунок – 25, потенційний об’єм програмного коду

5.2.3 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який має найбільш зручний інтерфейс та зрозумілу взаємодію з користувачем

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

- визначення рівня значимості параметра шляхом присвоєння різних рангів;
- перевірку придатності експертних оцінок для подальшого використання;
- визначення оцінки попарного пріоритету параметрів;
- обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 6.

Таблиця 6 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	$\Delta_i 2$
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	4	3	4	4	4	4	4	27	0,75	0,56
X2	Об’єм пам’яті для збереження даних	Мб	4	4	4	3	4	3	3	25	-1,25	1,56
X3	Час обробки запитів користувача	Мс	2	2	1	2	1	2	2	12	-14,25	203,06
X4	Потенційний об’єм програмного коду	кількість строк коду	5	6	6	6	6	6	6	41	14,75	217,56
	Разом		15	15	15	15	15	15	15	105	0	420,75

Для перевірки ступені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 105 ,$$

де $\boldsymbol{N}$ – число експертів,

$\boldsymbol{n}$ – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 26{,}25$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 420{,}75$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 420{,}75}{7^2(5^3 - 5)} = 1{,}03 > W_k = 0{,}67$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 7.

Таблиця 7 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	=	=	>	=	<	<	>	=	1
X1 і X3	<	<	>	<	>	<	=	<	0.5
X1 і X4	<	<	<	<	<	<	<	<	0.5
X2 і X3	<	<	=	>	=	=	>	=	1
X2 і X4	<	<	<	<	<	<	<	<	0.5
X3 і X4	<	<	<	<	<	<	<	<	0.5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*– тим, a_{ij} визначається по формулі:

$$a = \begin{cases} \textbf{1,5 при } X_i > X_j \\ \textbf{1 при } X_i = X_j \\ \textbf{0.5 при } X_i < X_j \end{cases}$$

З отриманих числових оцінок переваги складемо матрицю А=$\| a_{ij} \|$. Для кожного параметра зробимо розрахунок вагомості $K_{вi}$ за наступними формулами:

$$K_{\text{вi}} = \frac{b_i}{\sum_{i=1}^{n} b_i}, \text{де } b_i = \sum_{i=1}^{N} a_{ij}.$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%).На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{вi}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \text{де } b_i' = \sum_{i=1}^{N} a_{ij} b_j.$$

Як видно з таблиці 8, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 8 – Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{вi}}$	b_i^1	$K_{\text{вi}}^1$	b_i^2	$K_{\text{вi}}^2$
X1	1.0	1.0	0.5	0.5	3	0.1875	11.25	0.186	43	0.188
X2	1.0	1.0	1.0	0.5	3.5	0.2185	13.25	0.219	49.875	0.220
X3	1.5	1.0	1.0	0.5	4	0.25	14.75	0.244	55.5	0.243
X4	1.5	1.5	1.5	1.0	5.5	0.344	21.25	0.351	80.125	0.349
Всього:					16	1	60.5	1	228,5	1

5.3 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2*(об'єм пам'яті для збереження даних) та *X1* (швидкодія мови програмування) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X3 (*час обробки даних) обрано не найгіршим (не максимальним), тобто це значення відповідає або варіанту а) 1000 мс або варіанту б) 80мс.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 9):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j}$$

де n – кількість параметрів;$K_{вi}$ – коефіцієнт вагомості i–го параметра;B_i –

оцінка i–го параметра в балах.

Таблиця 9 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1(X1)	А	11000	5	0.188	0.93
F2(X3)	А, Б	3600	2.8	0.243	0.65
F2(X4)	А	9000	4	0.349	1.3
	Б	12000	2	0.349	0.71
F3(X2)	А	16	5	0.22	1.1

За даними з таблиці К за формулою

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}],$$

визначаємо рівень якості кожного з варіантів:

$K_{K1} = 0.93 + 0.65 + 1.3 + 1.1 = 3.98$

$K_{K2} = 0.94 + 0.68 + 0.7 + 1.1 = 3.39$

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

5.4 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як

$$T_O = T_P * K_П * K_{СК} * K_М * K_{СТ} * K_{СТ.М},$$

де T_P – трудомісткість розробки ПП;$K_П$ – поправочний коефіцієнт; $K_{СК}$ – коефіцієнт на складність вхідної інформації; $K_М$ – коефіцієнт рівня мови програмування; $K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм; $K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення.

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: T_P =90 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $K_П$ = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $K_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $K_{СТ}$ = 0.8. Тоді, за формулою 5.1, загальна трудомісткість програмування першого завдання дорівнює:

T_1 = 90 * 1.7 * 0.8 = 122.4 людино-днів.

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто T_P =27 людино-днів, $K_П$ =0.9,$K_{СК}$ = 1,$K_{СТ}$ =0.8:

T_2 = 27 * 0.9 * 0.8 = 19.44 людино-днів.

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

T_I = (122.4 + 19.44 + 4.8 + 19.44) . 8 = 1328,64 людино-годин;

T_{II} = (122.4 + 19.44 + 6.91 + 19.44) . 8 = 1345.52 людино-годин;

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 7000 грн., один фінансовий аналітик з окладом 9500грн. Визначимо зарплату за годину за формулою:

$$С_ч = \frac{М}{T_m \cdot t} грн.,$$

де М – місячний оклад працівників;T_m – кількість робочих днів тиждень;t – кількість робочих годин в день.

$$С_ч = \frac{7000 + 7000 + 9500}{3 \cdot 21 \cdot 8} = 46{,}62 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою

$$СЗП = С_ч \cdot Т_i \cdot КД,$$

де $С_ч$– величина погодинної оплати праці програміста;$Т_i$ – трудомісткість відповідного завдання; $К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить: I.

$С_{ЗП}$ = 46,62 * 1328.64 * 1.2 = 74340,57 грн.

II. $С_{ЗП}$ = 46,62 * 1345.52 * 1.2 = 75285,04 грн.

Відрахування на єдиний соціальний внесок становить 22%:

I. $С_{ВІД} = С_{ЗП} * 0.22$ = 74340,57 * 0.22 = 16354.93 грн.

II. $С_{ВІД} = С_{ЗП}$ * 0.22 = 75285,04 * 0.22 =16562.71 грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом 7000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$С_Г$ = 12 * М * $К_З$ = 12 * 7000 * 0,2 = 16800 грн.

З урахуванням додаткової заробітної плати:

$С_{ЗП} = С_Г * (1+ К_З) = 16800 * (1 + 0.2) = 20160$ грн.

Відрахування на єдиний соціальний внесок:

$С_{ВІД} = С_{ЗП} * 0.22 = 20160 * 0,22 = 4435.20$ грн.

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 10000 грн.

$С_А = К_{ТМ} * К_А * Ц_{ПР} = 1.15 * 0.25 * 10000 = 2875$ грн.,

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; $К_А$– річна норма амортизації; $Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$С_Р = К_{ТМ}. Ц_{ПР}. К_Р = 1.15 * 10000 * 0.05 = 575$ грн., де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$Т_{ЕФ} = (Д_К – (Д_В + Д_С) – Д_Р) . t_З. К_В = (365 – 116 – 4) * 8 * 0.9 = 1764$ годин,

де $Д_К$ – календарна кількість днів у році; $Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів; $Д_Р$ – кількість днів планових ремонтів устаткування; t –кількість робочих годин в день; $К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$С_{ЕЛ} = Т_{ЕФ} * N_С * Ц_{ЕН} = 1764 * 0,156 * 1,938 = 533.31$ грн.,

де $N_С$ – середньо-споживча потужність приладу; $Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$С_Н = Ц_{ПР}.0.67 = 10000. 0,67 =6700$ грн.

Тоді, річні експлуатаційні витрати будуть:

СЕКС = СЗП+ СВІД+ СА + СР+ СЕЛ + СН

$С_{ЕКС}$ = 20160 + 4435.20 + 2875 + 575 + 533.1 + 6700 = 35278.51 грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$С_{М-Г} = С_{ЕКС}/ Т_{ЕФ}$ = 35278.51 /1764 = 19.99 грн/час.

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$С_М = С_{М-Г} * Т$

I. $С_М$ = 19,99 * 1328,64 = 26559.51 грн.;

II. $С_М$ = 19,99 * 1345.52 = 26896.94 грн.;

Накладні витрати складають 67% від заробітної плати:

$С_Н = С_{ЗП}. 0,67$

I. $С_Н$ = 74340,57 * 0,67 = 49808,18 грн.;

II. $С_Н$ = 75285,04 * 0,67 = 50440,98 грн.;

Отже, вартість розробки ПП за варіантами становить:

СПП = СЗП+ СВІД+ СМ +СН

I. $С_{ПП}$ = 74340,57 + 16354.93 + 26559.51 + 49808,18 = 167063.19 грн.;

II. $С_{ПП}$ = 75285,04 + 16562.71 + 26896.94 50440,98 = 169185.67 грн.;

5.5 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$КТЕРj = ККj / СФj,$$

$К_{ТЕР1} = 3.98 / 167063.19 = 23,82 *10^{-6}$;

$К_{ТЕР2} = 3.39 / 169185.67 = 20,04 * 10^{-6}$;

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1}= 23,82 *10^{-6}$.

5.6 Висновки

В даному розділі проведено повний функціонально-вартісний аналіз ПП, який було розроблено в рамках дипломного проекту. Процес аналізу можна умовно розділити на дві частини.

В першій з них проведено дослідження ПП з технічної точки зору: було визначено основні функції ПП та сформовано множину варіантів їх реалізації; на основі обчислених значень параметрів, а також експертних оцінок їх важливості було обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій ПП.

Другу частину ФВА присвячено вибору із альтернативних варіантів реалізації найбільш економічно обґрунтованого. Порівняння запропонованих варіантів реалізації в рамках даної частини виконувалось за коефіцієнтом ефективності, для обчислення якого були обчислені такі допоміжні параметри, як трудомісткість, витрати на заробітну плату, накладні витрати.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 15,38 *10^{-6}$.

Цей варіант реалізації програмного продукту має такі параметри:

- мова програмування – Python;
- фреймворк машинного навчання - Tensorflow;
- середовище розробки – Jupyter Notebook.

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, функціонал і швидкодію.

ВИСНОВКИ

Метою роботи була розробка моделі для класифікації дорожних знаків на базі згорткової нейронної мережі, проведення досліджень з метою дізнатися, які чинники впливають на результат класифікації.

В першу чергу була спроектована архітектура штучної нейронної мережі на базі згорткової нейронної мережі і повнозв'язного перцептрона – класифікатора. Отримана нейронна мережа була навчена на наборі дорожних знаків і дала точність класифікації на валідаційному наборі даних 87%.

Ключове значення для якості моделі мають значення дані, на яких навчається нейронна мережа. З метою покращити результат класифікації мною було застосовано ряд технік передобробки даних. Так, нормалізація зображень дозволила підвищити точність класифікації на валідаційному наборі даних до 90%.

Початковий набір тренувальних даних був незбалансований, тобто деякі класи знаків були недопредставлені, а деякі – перепредставлені. Генерування нових екземплярів на базі існуючих за допомогою афінних перетворень і зміни яскравості дозволило вирівняти набір даних і призвело до підвищення точності класифікації на валідаційному наборі даних до 93.5%. Загалом завдяки передоброці даних, вдалося підвищити точність моделі з 87 до 93.5%.

Було проведено ряд експериментів над архітектурою моделі. Перетворення зображення з RGB-простору у напівтони сірого не підвищило точність класифікації (92.7%), хоча і дозволило зменшити об'єм вхідних даних і розмір моделі.

Масштабування архітектури вшир, тобто розширення шарів (в першу чергу – згорткових) за рахунок збільшення кількості фільтрів призвело до підвищення точності класифікації на валідаційному наборі даних до 96.5%.

Застосування Multi-Scale архітектури дозволило підвищити точність класифікації на валідаційному наборі даних до 97.5%.

Для запобігання перенавчання моделі застосовано техніку виключення (dropout) з коефіцієнтами від 0.5 до 0.7.

В результаті проведених досліджень розроблено модель, яка дозволяє класифікувати зображення дорожних знаків з точність більшою 90%. Розроблена модель може бути використана для вирішення більш глобальної задачі детектування і розпізнавання дорожних знаків у відеопотоці, що надходить з камер самокерованого автомобіля.

ПЕРЕЛІК ПОСИЛАНЬ

1. Goodfellow I. Deep Learning / Goodfellow I., Bengio Y. and Courville A.; Cambridge MA : MIT Press [2017] – 777 pages.
2. Springenberg, J. T. Striving for Simplicity: The All Convolutional Net / Springenberg, J. T. Dosovitskiy, A.; Brox, T. & Riedmiller, M. – Режим доступа: https://arxiv.org/abs/1412.6806.
3. Ruder S. (2016) An overview of gradient descent optimisation algorithms. – Режим доступу: https://arxiv.org/abs/1609.04747.
4. Robinds H. A stochastic approximation method // Annals of Mathematical Statistics / Robinds H. and Monro S. – 1951 – vol. 22 – pp. 400–407.
5. Darken, C. Learning rate schedules for faster stochastic gradient search / Darken, C., Chang, J. Moody, J. // Neural Networks for Signal Processing II Proceedings of the 1992 IEEE Workshop, September 1–11, 1992.
6. Dauphin, Y. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization / Dauphin, Y., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., Bengio, Y. – Режим доступу: http://arxiv.org/abs/1406.2572.
7. Sutton, R. S. Two problems with backpropagation and other steepest-descent learning procedures for networks. Proc. 8th Annual Conf. Cognitive Science Society – 1986.
8. Kingma, D. P. Adam: a Method for Stochastic Optimization / Kingma, D. P., Ba, J. L. // International Conference on Learning Representations, May 7-9, 2015, San-Diego, USA.
9. Nielsen M. Neural Networks And Deep Learning. – Режим доступу: http://neuralnetworksanddeeplearning.com/.
10. Sermanet, P. Traffic Sign Recognition with Multi-Scale Convolutional Networks / Sermanet, P., LeCun, Y. // The 2011 International Joint Conference on Neural Networks, September 2011.