

**ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА**

Факультет прикладної математики та інформатики

(повне найменування назва факультету)

Кафедра дискретного аналізу та інтелектуальних систем

(повна назва кафедри)

## **ДИПЛОМНА РОБОТА**

**ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ ДЛЯ ЗАДАЧІ КЛАСИФІКАЦІЇ  
ЗОБРАЖЕНЬ**

Виконав студент IV курсу, групи ПМІ-43

напряму підготовки (спеціальності)

122 - “Комп’ютерні науки та інформаційні технології”

(шифр і назва напрямку підготовки)

\_\_\_\_\_ Баранов М.В.

(підпис)

(прізвище та ініціали)

Керівник \_\_\_\_\_ ас. Квасниця Г. А.

(підпис)

(прізвище та ініціали)

Рецензент \_\_\_\_\_

(підпис)

(прізвище та ініціали)

## ЗМІСТ

ВСТУП.....	4
РОЗДІЛ 1. ПІДХОДИ РОБОТИ ІЗ ЗОБРАЖЕННЯМ.....	6
1.1 Розпізнавання зображень.....	6
1.1.1 Постановка задачі.....	6
1.1.2 Теоретичне підґрунтя роботи із зображенням.....	6
1.1.3 Класичні підходи класифікації зображень.....	6
1.2 Машинне навчання.....	7
1.2.1 Огляд машинного навчання.....	7
1.2.2 Машинне навчання без вчителя.....	8
1.2.3 Машинне навчання з вчителем.....	9
1.3 Нейронні мережі.....	11
1.3.1 Основа нейронної мережі.....	11
1.3.2 Багатошаровий персептрон.....	12
1.3.3 Функція активації.....	12
1.3.4 Типові функції активації.....	13
1.3.5 Тренування нейронної мережі.....	15
1.3.6 Методи градієнтного спуску для навчання нейронної мережі.....	16
РОЗДІЛ 2. ЗАСТОСУВАННЯ НЕЙРОННИХ МЕРЕЖ ДЛЯ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ.....	18
2.1.1 Вихід нейронної мережі.....	18
2.1.2 Цільова функція для оптимізації.....	18
2.1.3 Інтерпретація результату.....	19
2.2 Багатошаровий персептрон для класифікації зображень.....	19
2.2.1 Адаптація багатошарового персептрону для роботи із зображеннями..	19
2.3.1 Операція згортки.....	20
2.3.2 Згортковий шар нейронної мережі.....	20
2.3 Архітектура згорткової мережі для класифікації зображень.....	21

2.3.1 Використання згорткових шарів.....	21
2.3.2 Pooling прошарок.....	21
2.3.3 Класифікація результату згортки.....	22
РОЗДІЛ 3. АПРОБАЦІЯ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ.....	23
3.1 Аналіз тренувальної та тестової вибірки.....	23
3.1.1 Джерело даних.....	23
3.1.2 Аналіз даних.....	23
3.2 Побудова згорткової штучної нейронної мережі.....	24
3.2.1 Архітектура мережі.....	24
3.2.2 Процес тренування та аналіз результатів.....	25
3.2.3 Аналіз натренованої мережі.....	28
3.2.4 Аналіз рівня впевненості моделі.....	30
3.3 Технічне та програмне забезпечення.....	32
3.3.1 Мова програмування та додаткові бібліотеки.....	32
3.3.2 Фреймворк машинного навчання.....	32
3.3.3 Технічне забезпечення.....	34
ВИСНОВКИ.....	35
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ.....	37

## ВСТУП

Людський зір – один з найпотужніших органів чуттів. Існують різні оцінки кількості інформації, що надходить через органи зору. В середньому більше 80%-90% інформації людина отримує шляхом перетворення енергії електромагнітного випромінювання світлового діапазону та обробки цих сигналів мозком. За час еволюції зоровий апарат людини досягнув небачених висот: ми здатні розрізняти великі, дрібні предмети, відстані, положення, взаємне розташування; здатні знаходити подібності та відмінності між предметами, проводити узагальнення лише по одному екземпляру. Варто також відмітити швидкість роботи людського зору: мозок здатен обробляти 11 мільйонів біт сенсорної інформації в секунду, 10 мільйонів з яких припадає на зоровий апарат. Насправді, це не є захмарним числом. Якщо порівняти з кількістю інформації типового сучасного зображення формату FullHD — це лише одне зображення на хвилину! Проте людське око здатне опрацьовувати до 100 “зображень” в секунду, розрізняючи дрібніші деталі, які непомітні на цифрових фотографіях. На сьогодні досить детально досліджено процес роботи людського ока, але процес опрацювання отриманої досі приховує багато таємниць. В еру науково-технічної революції все більше попиту отримує завдання автоматичного аналізу зображень. Прикладів застосування машинного опрацювання візуальної інформації безліч: аналіз камер спостережень, системи автоматичного управління машиною, аналіз рентгенівських знімків у медицині тощо. Сфера цих завдань отримала назву — комп’ютерне бачення (англ. computer vision). Існує багато типових задач комп’ютерного бачення:

- Класифікація зображення
- Пошук відомих об’єктів на зображення
- Сегментація зображення
- Генерація нових зображень
- Покращення якості зображення
- Класифікація відео

- Локалізація об'єктів на відео в часовому та просторовому вимірах
- Реконструкція тривимірної сцени

Основними критеріями оцінки завдань є якість роботи, та швидкість (обмеженість обчислювальних ресурсів). Задачі комп'ютерного бачення зазвичай аналізують послідовність зображень (відео), тому важливим критерієм є швидкість задля отримання результату без затримки та уникнення втрати часової інформації.

Одним з найефективніших та виправданих застосувань комп'ютерного бачення початку 21-го століття — автономне водіння автомобіля. Є дуже багато критеріїв прийнятності цього проекту: надійність, точність, швидкість тощо. Слід також відмітити складність реалізації та велику кількість технічних деталей: розпізнавання автомобілів, дорожнього покриття, розмітки, перехресть, пішоходів, світлофорів, знаків. Однією з особливостей інформації — характерні спотворення внаслідок швидкості (розмиття як наслідок обмеженої витримки камер). Сучасні системи автономного водіння використовують багато додаткових приладів, робота яких базується не на аналізі візуальної інформації, проте це лише додаткове забезпечення.

## РОЗДІЛ 1. ПІДХОДИ РОБОТИ ІЗ ЗОБРАЖЕННЯМ

### 1.1 Розпізнавання зображень

#### 1.1.1 Постановка задачі

Робота має на меті дослідити сучасні підходи розпізнавання зображень та побудувати модель, яка здатна якісно класифікувати зображення дорожніх знаків низької якості. Вторинна вимога до проекту — швидкість роботи: проект повинен бути придатним до роботи в реальному часі, працювати без затримок.

#### 1.1.2 Теоретичне підґрунтя роботи із зображенням

Цифрове зображення — числова репрезентація двовимірного зображення. Зазвичай цифрове зображення дискретне. Дискретність зображення зумовлена обмеженнями технічного забезпечення камери. Бувай винятки, коли колір зображення є неперервним (колір задається довжиною хвилі). Проте, такий тип потрібен лише у вузькоспеціалізованих завданнях, та в повсякденному житті не використовується. В теорії зручно подавати зображення у вигляді матриці

$$I = \begin{pmatrix} i_{1,1} & i_{1,2} & \cdots & i_{1,n} \\ i_{2,1} & i_{2,2} & \cdots & i_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ i_{m,1} & i_{m,2} & \cdots & i_{m,n} \end{pmatrix}$$

Елементи матриці  $I_{i,j}$  несуть інформацію про інтенсивність кольору у позиції  $(i, j)$ . Для випадку чорно-білого зображення елементами матриці достатньо взяти скалярні величини. Для кольорового зображення необхідно вектор чисел. Кожен елемент відповідає за певний колір зображення. Цей вектори належить певному кольоровому простору. Зазвичай використовують кольоровий простір RGB — суміш червоного, зеленого та синього кольору. Математичний вигляд зображення відкриває можливості використання усіх потужностей математичного апарату для задач комп'ютерного зору.

#### 1.1.3 Класичні підходи класифікації зображень

Класичні способи класифікації зображень полягають у розробці алгоритмів людиною. Розробка таких алгоритмів вимагає досконалих знань предметної

області (розуміння задачі та всіх можливих випадків). Наприклад, завдання розпізнавання шахової дошки часто вирішується за допомогою обрахунку похідних зображення та пошуку максимумів градієнтів: перехід від білої клітинки до чорної зумовлює різке зростання похідної.

Перевагою такого підходу є цілковите розуміння алгоритму та можливість ґрунтовного пояснення його результатів. Ця деталь є важливою у медичній сфері. Багато рішень для медичного обслуговування (автоматичне діагностування пацієнтів, наприклад) було відкинуто медиками, оскільки не було можливості обґрунтування діагнозу, хоч здебільшого він і був правильним. Іншою перевагою є практично безмежні можливості реалізації альтернативних рішень підзадач, що може бути критичним на етапі покращення продуктивності проекту.

Великим недоліком класичних підходів розв’язування задач комп’ютерного зору є складність побудови таких алгоритмів для комплексних завдань. Пашука шахової дошки на зображення легко реалізується класичними засобами, оскільки шахова дошка складається з набору примітивних ознак. Проте, виділити такі ознаки для автомобілів є значно складнішим завданням з огляду на різноманітність кольору, форм, моделей автомобілів, різноманітних кутів огляду тощо. Зазвичай, рішення таких складних завдань досягається завдяки певним обмеженням (наприклад, статично зафіксована камера).

## 1.2 Машинне навчання

### 1.2.1 Огляд машинного навчання

Машинне навчання — клас методів штучного інтелекту, основним завданням яких не є пряме рішення задачі, а побудова алгоритму, який міг би вирішувати поставлене завдання. Побудова цього алгоритму базується на статистичних моделях. Припустимо, ми маємо набір даних

$$x = \{x_1, x_2, \dots, x_n\}$$

та анотації цих даних. Під анотаціями слід розуміти бажаний результат роботи алгоритму на наведених вище даних

$$y = \{y_1, y_2, \dots, y_n\}$$

Оскільки вхідні дані — математичні об'єкти, цілком логічно буде побудувати наш алгоритм у вигляді функції від вхідних даних

$$f(x', \bar{w}) = y'$$

де  $\bar{w}$  - внутрішні параметри алгоритму. Сама функція алгоритму зазвичай є сталою. Наприклад найпростіша одношарова нейронна мережа зводиться до матричного множення та поелементного застосування нелінійної функції активації. Тобто, функція матиме вигляд

$$f(x', \bar{w}) = \text{relu}(x' \cdot \bar{w})$$

функція *relu* - узагальнення функції  $\max(0, x)$  на багатовимірні об'єкти шляхом поелементного її застосування.

Машинне навчання поділяється на дві категорії: навчання з вчителем та без вчителя. Найскладнішою задачею є автоматичний пошук параметрів  $\bar{w}$ . Існує велика кількість алгоритмів здатних автоматично підбирати параметри, які б задовільняли певну задачу.

### 1.2.2 Машинне навчання без вчителя

Нехай функція нашої моделі залежить від вхідних даних та внутрішніх параметрів

$$f(x', \bar{w}) = y'$$

Особливістю навчання без вчителя є те, що модель не потребує ніякої апріорної інформації про очікуваний результат. Іншими словами під час тренування використовується лише множина  $x = \{x_1, x_2, \dots, x_n\}$ . До машинного навчання без вчителя відноситься:

- Кластеризація

Основним завданням кластерного аналізу є розбиття даних на певні групи (кластери) так, щоб покласова варіанса було якомога меншою, а міжкласова навпаки — більшою. Приклади алгоритмів кластеризації: K-means, C-means, DBSCAN, Affinity Propagation тощо.

- Розпізнавання аномалій

Розпізнавання аномалій вирішує завдання виділення з масиву даних таких елементів, які не підпорядковуються загальній структурі вхідних



параметрів. Найпоширенішою технікою розпізнавання аномалій є методи аналізу щільності.

- Зменшення розмірності

Зазвичай робота з даними вимагає роботи з багатовимірними об'єктами. Чотиривимірні об'єкти можна ще візуалізувати як рух проекції тривимірного об'єкта в часі. Дані вищих розмірностей важко репрезентувати графічно. Існують методи машинного навчання, які здатні апроксимувати багатовимірний простір іншим підпростором з меншої розмірності так, щоб мінімізувати втрату інформації про об'єкт спостереження

Є також змішаний тип — напіваавтоматичне навчання. Найширшого застосування такий тип навчання набув у вигляді навчання з підкріпленням: коли очікуваний результат формально є, але неможливо виділити якусь частину даних, які впливають на результат. Результат в такому випадку формується в результаті комплексного аналізу даних. На сьогоднішній час найбільше прикладів є для навчання ігрових ботів різноманітних ігор.

### 1.2.3 Машинне навчання з вчителем

Класичне навчання з вчителем вимагає анотацій до всіх вхідних даних. Кожна анотація — це очікуваний, правильний результат для певного входу моделі. На сьогодні анотовані дані є чи не найбільшою проблемою машинного навчання. Не існує способу автоматичного анотації даних. З огляду на необхідність великих обсягів даних для машинного навчання — процес анотації вимагає велику кількість часу та коштів. Найпоширенішою реалізацією навчання з вчителем є нейронні мережі.

Існують різноманітні вигляди абстрактної функції  $f$ . Наведемо деякі поширені приклади

- Методи К-найближчих сусідів (KNN)

Методи найближчих сусідів дозволяє кластеризувати простір предметної області, що дає змогу вирішити задачу класифікації. Кожен елемент тренувальної вибірки репрезентує точку в  $\mathbb{R}^n$ . Для кожного елементу тестової вибірки виконується пошук  $K$  найближчих сусідів, використовуючи певну метрику. Серед знайдених сусідів аналізуємо кількість екземплярів класів. Яких класів є найбільше — до такого класу найкраще віднести елемент тестової вибірки. Недоліком такого підходу є чутливість до аномалій тренувальної вибірки та неможливість розв'язування задачі регресії класичним підходом KNN.

- Метод опорних векторів

Оригінальний метод опорних векторів розрахований на завдання бінарної класифікації. Алгоритм дозволяє розділити два тренувальні дані на два класи завдяки побудови прямої, що розділяє ці класи. Ця пряма будується завдяки опорним векторам, обрахунок яких виконують на основі тренувальної вибірки. Для випадків, коли класи неможливо розділити лінійно застосовуються нелінійні ядра для переходу в іншу систему координат. Наприклад, двовимірні точки двох класів, розташовані у вигляді зовнішнього і внутрішнього кола переводять у тривимірний простір, де їх легко розділити площиною.

- Дерева рішень

Побудова дерева рішень полягає у побудові систем послідовних умовних запитань. Результатом серій відповідей на певні запитання є результат класифікації. Найкраще дерева рішень розв'язують задачу класифікації, проте існують модифікації дерев для регресії. Великим недоліком дерев рішень є властивість перенавчання, що зумовлює вкрай низьку точність класифікації або регресії на тестовій вибірці. Ця вразливість особливо притаманна багатовимірним даним. Покращенням дерева рішень — є “випадковий ліс” (англ. random forest).

Ідея цього класифікатора полягає в побудові певної кількості дерев рішень на підмножині тренувальної вибірки. Різноманітність підмножин зумовлює рівномірний розподіл помилок між всіма деревами. Ансамбль дерев (ліс) приймає рішення за допомогою голосування. Завдяки рівномірному розподілу помилок ймовірність помилитись усім деревам одночасно є меншою, що зумовлює більшу точність на тестовій вибірці.

### 1.3 Нейронні мережі

#### 1.3.1 Основа нейронної мережі

Найбільшого поширення набули нейронні мережі. Ідея побудови нейронної мережі дуже подібна до побудови мозку людини. Базовою одиницею нейронної мережі є нейрон. Подібно до нейрона людини, нейрон має певну сталу (для окремого нейрона) кількість входів та один вихід. Кожен нейрон в середині себе містить ваги та функцію активації. Математично роботу нейрона можна записати у вигляді функції

$$f(x) = \varphi\left(\sum_{i=1}^n w_i \cdot x_i + b\right)$$

$$x = \{x_1, x_2, \dots, x_n\}$$

де  $x$  - входні дані,  $w_i$  - внутрішні ваги,  $b$  - біас (зміщення),  $\varphi(\cdot)$  - функція активації. Наприклад, найпростіша функція активації — порогова функція

$$\varphi(x) = \begin{cases} 0, & x < t \\ 1, & x \geq t \end{cases}$$

$$t = \text{const}$$

Для зручності обчислень та компактнішої форми нейрона поширене подання входних даних у вигляді  $x = \{1, x_1, x_2, \dots, x_n\}$ , а ваги у вигляді  $w = \{b, w_1, w_2, \dots, w_n\}$

Таким чином формулу нейрона можна подати у вигляді

$$f(x) = \varphi\left(\sum_{i=1}^n w_i \cdot x_i\right)$$

Цю формулу можна подати з використанням одновимірної дискретної згортки

$$f(x) = \varphi(x * w)$$

Така форма подачі дозволяє спростити реалізацію штучного нейрона та

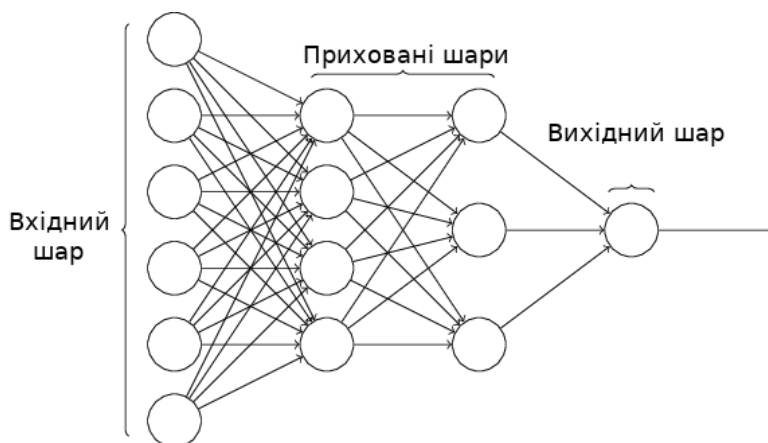
використовувати переваги розпаралелювання та специфічних обчислювальних ресурсів, які пристосовані до пришвидшення операції згортки.

Такий математичний об'єкт отримав назву “персептрон”. Він здатен розв'язувати задачу лінійної класифікації в  $n$ -вимірному просторі, оскільки його внутрішня реалізація будує уявну лінію, а функція активації визначає, по яку сторону лінії знаходиться вхідна точка. Проте, було доведено, що персептрон не придатний до задачі нелінійної класифікації. Це було показано на прикладі функції  $XOR$ , яку за допомогою персептрона реалізувати неможливо.

### 1.3.2 Багатошаровий персептрон

Для завдань нелінійної класифікації нам необхідно використовувати більше одного штучного нейрона.

Багатошаровий персептрон містить декілька шарів штучних нейронів. Виходом кожного шару вважається сукупність результатів роботи усіх нейронів цього шару. Це



формує вектор, який подається

Рис. 1 Багатошаровий персептрон

на вхід кожному нейрону наступного шару. Таким чином дані модифікуються та передаються від одного шару до іншого. Результат роботи останнього шару вважається результатом роботи всієї штучної нейронної мережі.

### 1.3.3 Функція активації

Багатошаровий персептрон має змогу вирішувати задачу нелінійної класифікації завдяки своїм нелінійним властивостям. Ці властивості досягаються шляхом функції активації. Важливою умовою функції активації є її нелінійність. Припустимо, що функція активації  $\varphi(\cdot)$  є лінійною, тобто

$$\varphi(x) = x$$

Тепер розглянемо роботу частини двохарового персептрону:

$$f(x) = \varphi(\varphi(x * w^1) * w^2)$$

Оскільки, функція  $\varphi(x * w^1) = x * w^1$ , то

$$\varphi(\varphi(x * w^1) * w^2) = \varphi(x * w^1 * w^2)$$

Зробимо заміну  $w^1 * w^2 = w'$ , отримаємо

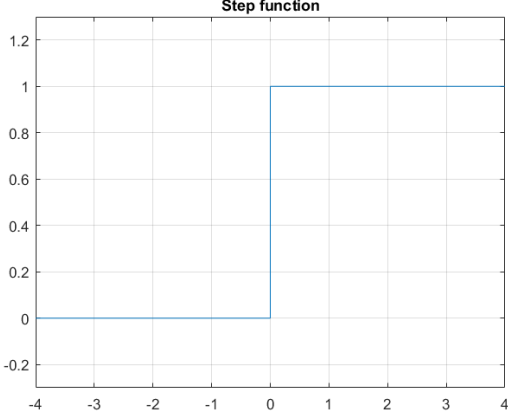
$$\varphi(\varphi(x * w^1) * w^2) = \varphi(x * w') = f(x)$$

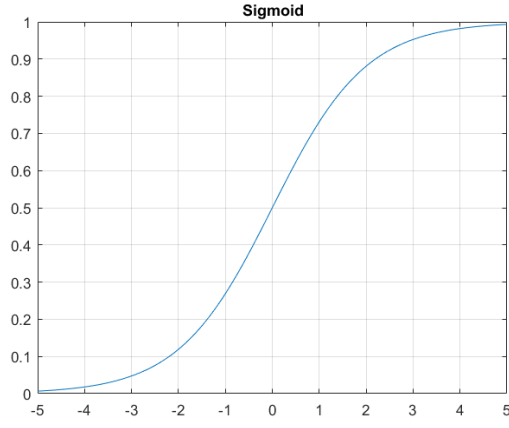
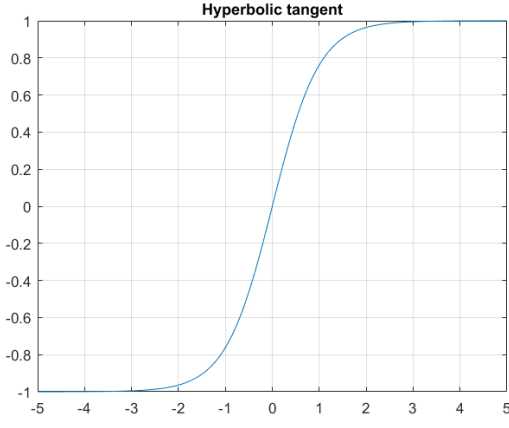
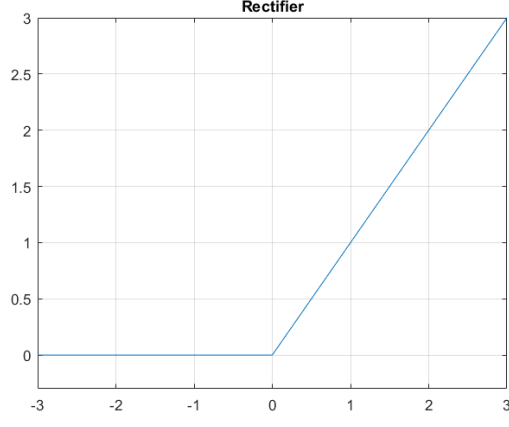
Тобто, використання багат шарового персептрону з нелінійною функцією активації еквівалентне використанню одношарового персептрону.

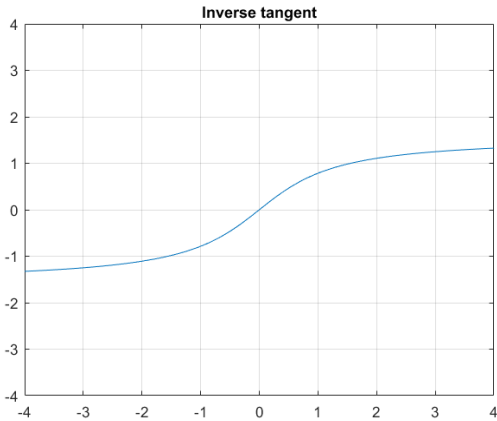
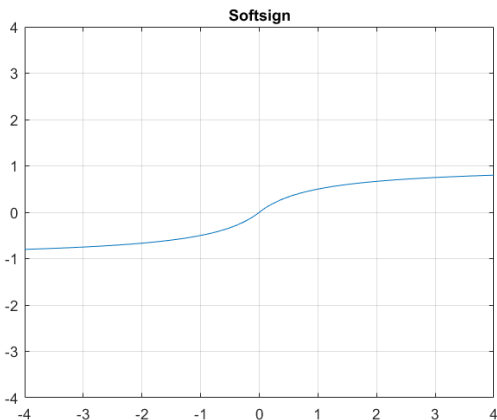
#### 1.3.4 Типові функції активації

В якості функції активації формально можна використовувати будь-яку нелінійну функцію. Проте, в практиці зазвичай застосовують відомі, поширені функції. Вибір функції активації впливає на ефективність роботи мережі, процес навчання та швидкість роботи. Оскільки функція активації застосовується велику кількість разів під час роботи або тренування мережі, то простота цієї функції є однією з складових швидкості роботи.

Таблиця 1

Функція активації	Графік
<p>Порогова функція</p> $\varphi(x) = \begin{cases} 0, & x < 0 \\ 1, & x > 0 \end{cases}$	 <p>The graph shows a step function titled 'Step function'. The horizontal axis (x) ranges from -4 to 4 with major ticks every 1 unit. The vertical axis (y) ranges from -0.2 to 1.2 with major ticks every 0.2 units. The function is plotted as a blue line that is at y=0 for x &lt; 0 and at y=1 for x &gt; 0. There is a vertical jump at x=0.</p>

Функція активації	Графік
<p>Сигмоїд</p> $\varphi(x) = \frac{1}{1+e^x}$	 <p>The graph shows the Sigmoid function, which is an S-shaped curve. It is plotted on a coordinate system with the x-axis ranging from -5 to 5 and the y-axis ranging from 0 to 1. The curve starts near 0 for negative x values and approaches 1 as x increases, passing through the point (0, 0.5).</p>
<p>Гіперболічний тангенс</p> $\varphi(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$	 <p>The graph shows the Hyperbolic tangent function, which is an S-shaped curve. It is plotted on a coordinate system with the x-axis ranging from -5 to 5 and the y-axis ranging from -1 to 1. The curve starts near -1 for negative x values and approaches 1 as x increases, passing through the point (0, 0).</p>
<p>Випрямляюча функція активації</p> $\varphi(x) = \max(0, x)$	 <p>The graph shows the Rectifier function, which is a piecewise linear function. It is plotted on a coordinate system with the x-axis ranging from -3 to 3 and the y-axis ranging from 0 to 3. The function is zero for negative x values and increases linearly for positive x values, passing through the point (0, 0).</p>

Функція активації	Графік
<p>Обернений тангенс</p> $\varphi(x) = \arctg(x)$	 <p>The graph shows the Inverse tangent function, which is an S-shaped curve passing through the origin (0,0). The x-axis ranges from -4 to 4, and the y-axis ranges from -4 to 4. The curve approaches horizontal asymptotes at y = ±π/2 ≈ ±1.57.</p>
<p>Гладкий сигмоїд</p> $\varphi(x) = \frac{x}{1 + \ x\ }$	 <p>The graph shows the Softsign function, which is an S-shaped curve passing through the origin (0,0). The x-axis ranges from -4 to 4, and the y-axis ranges from -4 to 4. The curve approaches horizontal asymptotes at y = ±1.</p>

### 1.3.5 Тренування нейронної мережі

Тренування мережі — процес пошуку ваг. Процес тренування базується на статистичних залежностях вхідних даних та очікуваних виходів. Природне бажання — отримати таку нейронну мережу, результат якої якомога краще відповідав очікуванням. Припустимо, ми маємо тренувальну вибірку  $\bar{x}$

$$\bar{x} = \{x_1, x_2, \dots, x_n\}$$

Та множину очікуваних виходів мережі  $\bar{y}$

$$\bar{y} = \{y_1, y_2, \dots, y_n\}$$

Нехай, наша штучна нейронна мережа задана функцією

$$f(x, \bar{w}) = y$$

Завдання тренування мережі полягає у пошуку таких параметрів  $\bar{w}$ , що

$$f(x', \bar{w}) - y' = 0$$

де,  $y'$ , бажаний вихід мережі при вхідних даних  $x'$ .

Очевидно, що в більшості випадків неможливо досягнути ідеальної відповідності виходу мережі та очікуваному виходу. Тому наведена вище умова зводиться до оптимізації величини

$$f(x', \bar{w}) - y'$$

Таким чином завдання тренування мережі зводиться до завдання оптимізації параметрів  $\bar{w}$ .

Зазвичай, для оптимізації мережі використовують певну метрику.

У наведеному вище прикладі, такою метрикою є  $d_{L_1}$  відстань

$$d_{L_1}(x, y) = \sum_{i=1}^n \|x_i - y_i\|$$

Проте, наведена метрика не завжди є ефективною, оскільки вона однаково аналізує як великі, так і малі відстані. Наприклад, для завдання класифікації зазвичай використовують метрику розрідженої перехресної ентропії

$$H(p, q) = - \sum_x p(x) \log(q(x))$$

В загальному, цільова функція оптимізації має вигляд

$$Q(p, q) = Q(f(x', \bar{w}), y')$$

де,  $p$  - очікуваний результат,  $q$  - реальний вихід моделі.

### 1.3.6 Методи градієнтного спуску для навчання нейронної мережі

Класичними методами оптимізації — є методи градієнтного спуску. Ідея цих методів базується на тому факті, що похідна функції в заданій точці завжди несе інформації про характер поведінки функції, а саме — напрям зростання. Таким чином, для оптимізації диференційованої функції достатньо обчислити похідну в заданій точці та змінювати її в напрямку спадання функції.

Нехай цільова функція  $F(x)$  визначена і диференційована в деякому околі точки  $x_0$ . Тоді функція  $F(x)$  спадає найшвидше, якщо точка  $x_0$  зміщується в напрямку  $-\nabla F(x)$ . Звідси випливає, якщо кожен наступну точку вибирати за правилом



$$x_{n+1} = x_n - \gamma \nabla F(x)$$

$$\gamma \in R_+$$

при чому,  $\gamma$  досить мала величина, то в загальному випадку

$$F(x_n) \geq F(x_{n+1})$$

А якщо  $x_n \neq x_{n+1}$ , то

$$F(x_n) > F(x_{n+1})$$

Недоліком класичних методів градієнтного спуску — є пошук найближчого локального мінімуму. Хоч в точці локального мінімуму будь-який напрямок зміни цільових параметрів призводить до зростання значення цільової функції, існують інші локальні мінімуми та глобальний мінімум, де це значення може бути ще менше. Як відомо, жадібні алгоритми не завжди дають оптимальний розв'язок задачі. Існує багато модифікованих алгоритмів, що базуються на методах градієнтного спуску. Найпоширеніші з них:

- Adam
- Adagrad
- Adadelat
- RMSProp

## РОЗДІЛ 2. ЗАСТОСУВАННЯ НЕЙРОННИХ МЕРЕЖ ДЛЯ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ

### 2.1 Нейронні мережі для завдання класифікації

#### 2.1.1 Вихід нейронної мережі

Природньо очікувані класи виходу мережі задавати індексами  $0, 1, \dots, n$ , проте цей спосіб немає перспектив застосування. Нейронна мережа найкраще працює з нормалізованими даними. По-перше, індекси класів не є нормалізованими значеннями. По-друге, послідовне задання індексів безумовно задає велику кореляцію між суміжними класами. Найкращий варіант задання очікуваного виходу в завданні класифікації — у вигляді унітарного коду. Наприклад, для завдання тернарної класифікації найкраще класи позначати так:

$$\begin{pmatrix} 1, 0, 0 \\ 0, 1, 0 \\ 0, 0, 1 \end{pmatrix}$$

Такий спосіб чітко розмежовує класи, дає змогу використовувати на вихідному прошарку штучної нейронної мережі таку кількість нейронів, яка задовільняє конкретній задачі.

Зазвичай, в завдання класифікації при унітарному заданні бажаних виходів використовується специфічна функція активації — softmax

$$\sigma(x)_j = \frac{e^{x_j}}{\sum_{i=1}^n e^{x_i}}$$

Така функція активації нормує значення вихідних нейронів, що дозволяє розглядати елементи виходу мережі, як ймовірності або впевненість у тому, чи іншому класі.

#### 2.1.2 Цільова функція для оптимізації

Оскільки вихід мережі та бажаний результат можна розглядати, як ймовірність, то цілком природньо як цільову функцію використовувати функцію перехресної ентропії. Ця функція досягає мінімуму, якщо очікуване та реальне значення збігаються. До того ж, логарифмічна складова активації дозволяє нерівномірно

оптимізовувати невеликі та великі відхилення відповідно.

$$H(p, q) = - \sum_x p(x) \log(q(x))$$

### 2.1.3 Інтерпретація результату

Під час роботи мережі передбаченням можна вважати клас, індекс якого обчислюється за формулою

$$\operatorname{argmax}(f(x', \bar{w}))$$

а рівень впевненості відповідно

$$f(x', \bar{w})_{\operatorname{argmax}(f(x', \bar{w}))}$$

## 2.2 Багатошаровий перцептрон для класифікації зображень

### 2.2.1 Адаптація багатошарового перцептрону для роботи із зображеннями

Реалізація штучного нейронну дозволяє

працювати з вхідними даними у вигляді

вектора. Проте зображення зазвичай

представлене у вигляді матриці.

Найпростіший варіант перетворення

матриці у вектор — послідовна

конкатенація усіх рядочків. Таким чином

ми сформували набір ознак, які

багатошаровий перцептрон намагатиметься

класифікувати.

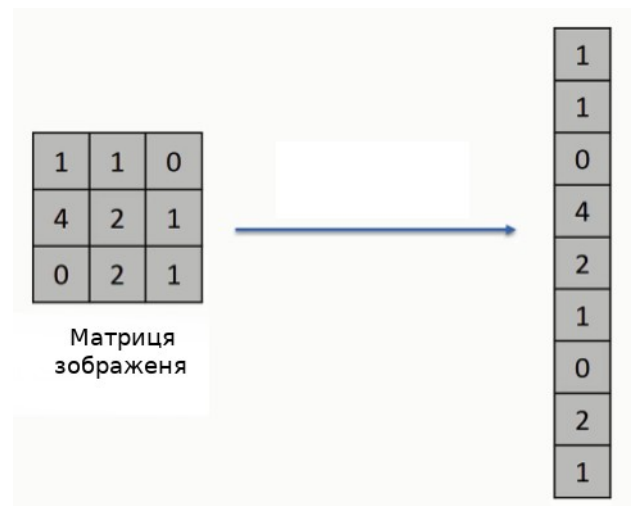


Рис. 2 Перетворення матриці у вектор

Недоліком використання багатошарового перцептрону для безпосередньої класифікації зображення — надлишковість. В реальних зображеннях чітко прослідковується кореляція сусідніх пікселів: кольори змінюються плавно, градієнтами. Багатошаровий перцептрон не має апіорної інформації про кореляції між суміжними пікселями. Для такої моделі залежність між сусідніми елементами матриці та діагонально протилежними — однакова. Це спричиняє проблеми перенавчання: штучна нейронна мережа може вивчити такі ознаки зображень, які не корелюють ніяк з предметною областю.

## 2.3 Згорткові нейронні мережі для класифікації зображень

### 2.3.1 Операція згортки

Згортка зображення  $I$  з ядром  $K$  та якорем  $\alpha$  являє собою заміну кожного пікселя  $p$  вихідної матриці на суму добутків відповідних пікселів вхідною матриці на

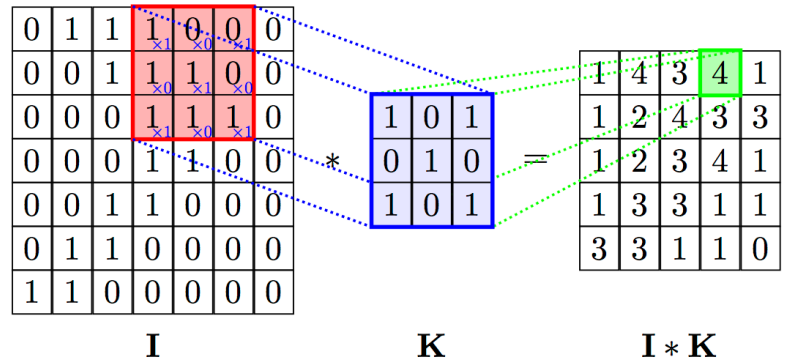


Рис. 3 Операція згортки

що положення елемента  $\alpha$  ядра  $K$  збігається з положенням вихідного пікселя  $p$ .

Операція згортки позначається

$$I * K = I'$$

$$(f * g)(t) = \int_{-\infty}^{+\infty} f(\tau) g(t - \tau) d\tau$$

Враховуючи дискретність зображення інтегральну форму згортки можна записати в простішому вигляді

$$I'(i, j) = \sum_{u=i-a_x}^{i+(m-a_x)} \sum_{v=j-a_y}^{j+(n-a_y)} I(i, j) K(u - (i - a_x), v - (j - a_y))$$

### 2.3.2 Згортковий шар нейронної мережі

Звичайна двовимірна згортка працює лише з матрицями. Якщо ми маємо кольорове зображення — то ми маємо три матриці. Для таких випадків згортка відбувається тривимірна, а ядро згортки має форму паралелепіпеда. Один згортковий шар нейронної містить одразу багато ядер тривимірних згортки. Причому, третій вимір (глибина) кожного ядра рівний кількості фільтрів (матриць) вхідного зображення. Застосування кожного такого фільтра продукує утворення нової матриці. Застосування  $N$  таких ядер утворює багатовимірне зображення. Особливість згорткових мереж в тому, що вони дозволяють в процесі тренування віднайти в автоматичному режимі такі значення згорткових ядер, які б виділяли з вхідних даних характерні ознаки, за якими було б значно легше класифікувати вхідне зображення.

## 2.3 Архітектура згорткової мережі для класифікації зображень

### 2.3.1 Використання згорткових шарів

Використання згорткових шарів штучної нейронної мережі дає змогу добувати характерні ознаки в тій чи іншій предметній області. Проте, варто зауважити, що операція згортки є досить дорогою. Складність двовимірної згортки оцінюється як  $O(W \cdot H \cdot M \cdot N)$ , де  $W, H$  — висота та ширина зображень,  $M, N$  - ширина та висота ядра згортки. Незважаючи на апаратне прискорення операцій з використанням відеокарт, велика кількість згорткових шарів та поступове збільшення кількості фільтрів суттєво сповільнюють роботу мережі.

### 2.3.2 Pooling прошарок

Зазвичай немає необхідності усі згорткові шари застосовувати до матриць великого розміру з огляду на кореляції суміжних елементів. З метою пришвидшення тренування мережі та уникнення перенавчання застосовують різноманітні варіації “pooling” прошарків. Основна ідея в основі цього шару схожа до ідеї згортки. Загальну формулу можна виразити у такому вигляді

$$I_{i,j} = \underset{a \in [0,m], b \in [0,n]}{G} \{ I_{i+a, j+b} \}$$

де,  $G: R^n \rightarrow R$  - функція, що оперує множиною елементів. Проте такий підхід не призведе до покращення роботи нейронної мережі. Є необхідність використовувати певне зміщення, що в результаті зменшує розміри вихідної матриці

$$I_{i,j} = \underset{a \in [0,m], b \in [0,n]}{G} \{ I_{s_w \cdot i + a, s_h \cdot j + b} \}$$

Такі прошарки не мають параметрів, які необхідно відшукати в процесі тренування мережі.

- Max-pooling прошарок

Результат роботи max-pooling прошарку можна виразити за допомогою формули

$$I_{i,j} = \underset{a \in [0,m], b \in [0,n]}{\max} \{ I_{s_h \cdot i + a, s_w \cdot j + b} \}$$

Таким чином, кожна множина елементів замінюється максимальним елементом з

цієї множини.

- Average-pooling прошарок

Average-pooling прошарок замінює множину елементів середнім арифметичним цієї множини

$$I_{i,j} = \frac{1}{m \cdot n} \sum_{a=0}^m \sum_{b=0}^n I_{s_w \cdot i + a, s_h \cdot j + b}$$

Мотивацією використання такого шару є ситуація, коли різноманітні фільтри на певному шарі нейронної мережі були обраховані незалежно, різними шарами. В такому випадку буде рівномірно врахована уся інформація на даному прошарку.

### 2.3.3 Класифікація результату згортки

З кожним наступним шаром дані, що обчислюються на прошарках менше репрезентують зображення, а все більше набувають вигляду чисельної характеристики цього зображення. Зазвичай, після певного шару згорткової мережі матрицю перетворюють у вектор. Цей вектор зручно класифікувати за допомогою багатощарового персептрону.

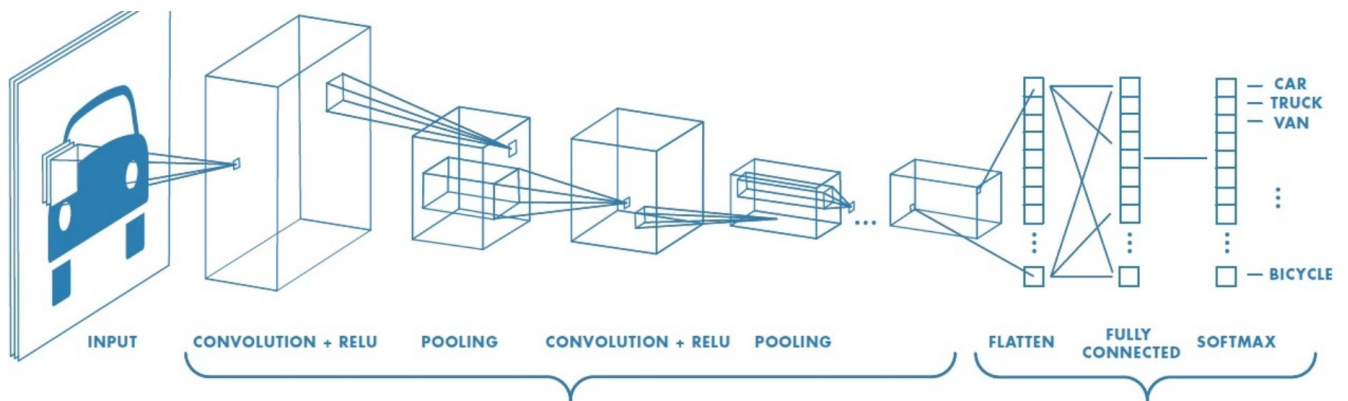


Рис. 4 Типова архітектура класифікаційної згорткової мережі

## РОЗДІЛ 3. АПРОБАЦІЯ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ

### 3.1 Аналіз тренувальної та тестової вибірки

#### 3.1.1 Джерело даних

Для апробації алгоритмів використано відкритий набір даних “The German Traffic Sign Recognition Benchmark”. Ці дані було опубліковано в рамках змагання по класифікації і розпізнаванню дорожніх знаків Німеччини у 2010-2011 роках. Набір даних містить 43 категорії дорожніх знаків. Обсяг тренувальної вибірки — близько 40000 елементів; тестової вибірки — 12600 елементів. Мотивація змагання — розробити новий підхід для тренування моделей задачі класифікації при роботі з незбалансованими, різноманітними даними.

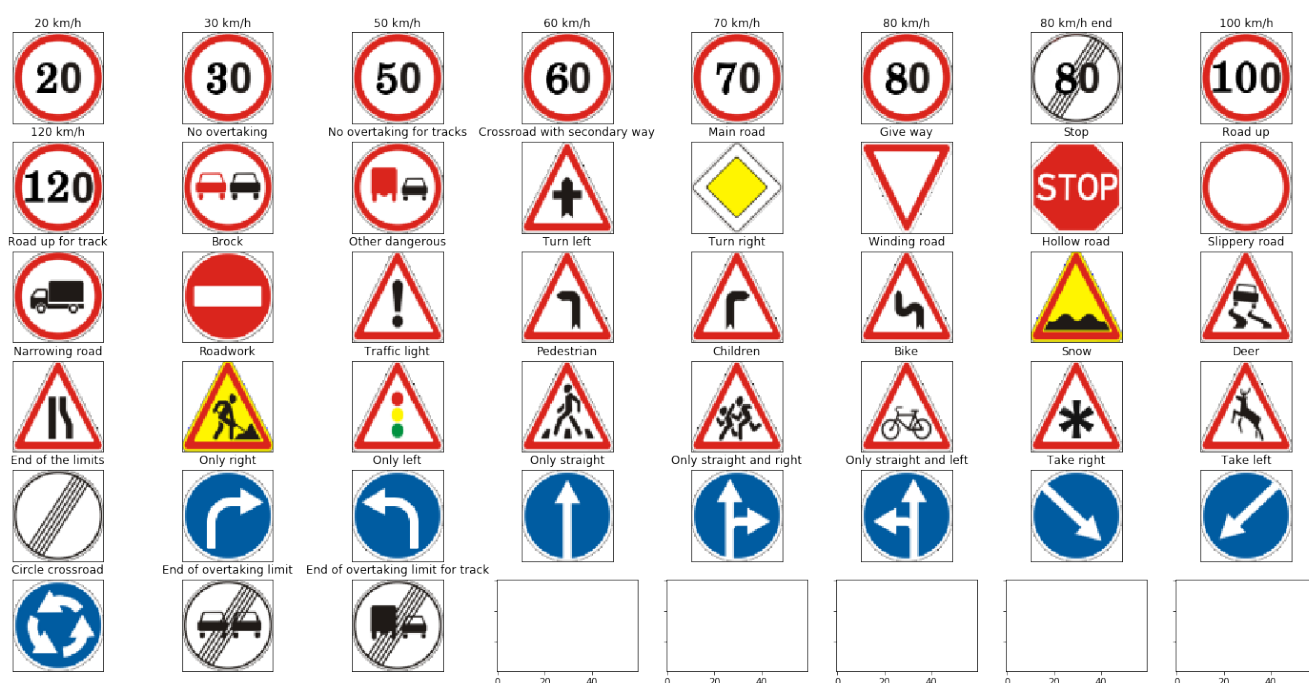


Рис. 5 Категорії дорожніх знаків

#### 3.1.2 Аналіз даних

Дані тренувальної та тестової вибірки є незбалансовані, проте розподіл покласовий розподіл кількості елементів має однакову природу для тренувальної і тестової вибірки відповідно.

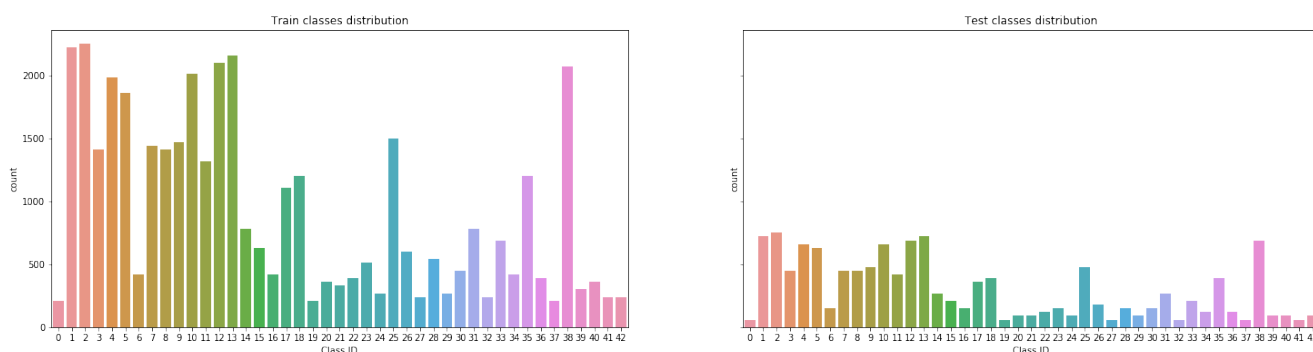


Рис. 6 Баланс тренувальної та тестової вибірки

В наборі даних присутні картинки з різноманітним освітленням, контрастом, зміщенням тощо.



Рис. 7: Приклад тренувальних елементів

Присутні зображення різноманітного розміру. Здебільшого картинки дотримуються квадратної форми. Медіана ширини та висоти — 35 пікселів.

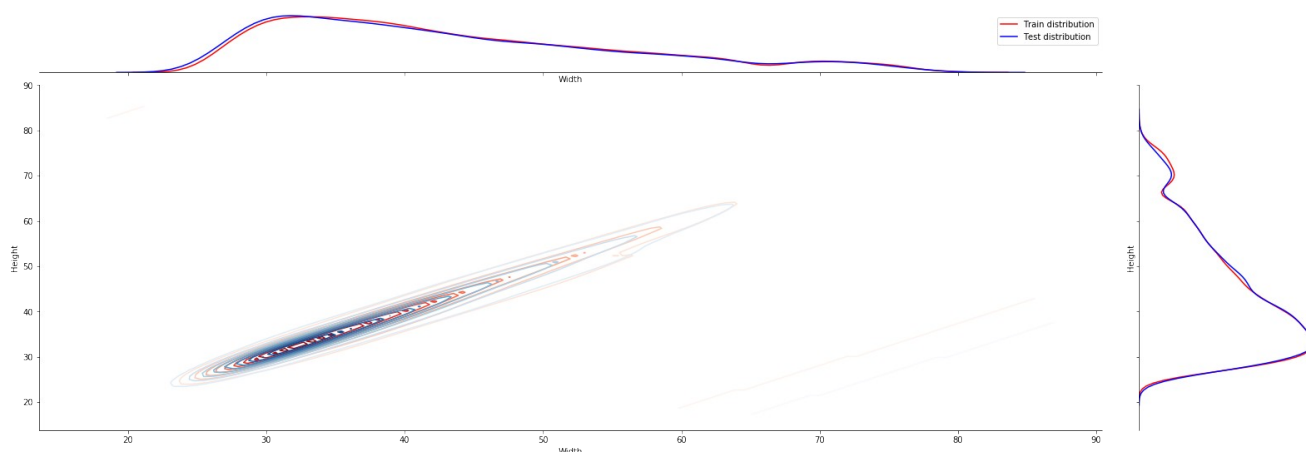


Рис. 8: Кореляція ширини та висоти зображень

## 3.2 Побудова згорткової штучної нейронної мережі

### 3.2.1 Архітектура мережі

В архітектурі згорткової мережі використано 8 згорткових прошарків, 1 max-pooling прошарок, та 2 щільних прошарки (двошаровий персептрон). З метою покращення та стабілізації тренування після кожного згорткового прошарку



використано нормалізацію девіації та середнього значенням даних. Також застосована техніка аугментації тренувальних даних. Детальна архітектура мережі наведена в таблиці 2.

Таблиця 2: Архітектура згорткової мережі

Операція	Параметри	Розмірність входу	Розмірність виходу
Згортковий	16 фільтрів	(60,60,3)	(58,58,16)
Згортковий	32 фільтрів	(58,58,16)	(56,56,16)
Згортковий	32 фільтрів	(56,56,16)	(54,54,32)
Згортковий	64 фільтрів	(54,54,32)	(52,52,64)
Max-pooling	-	(52,52,64)	(26,26,64)
Згортковий	64 фільтрів	(26,26,64)	(24,24,64)
Згортковий	128 фільтрів	(24,24,64)	(22,22,128)
Згортковий	256 фільтрів	(22,22,128)	(20,20,256)
Згортковий	400 фільтрів	(20,20,256)	(18,18,400)
Перетворюючи	-	(18,18,400)	(129600)
Щільний	256 нейронів	(129600)	(256)
Щільний	43 нейрони	(256)	(43)
Вихідний		(43)	-

### 3.2.2 Процес тренування та аналіз результатів

Розмір вибірки тренування обмежений 40000 елементів. Для покращення фінального результату було застосовано метод аугментації даних. Кожен елемент (картинку) тренувальної вибірки перед подачею на вхід штучній нейронній мережі було випадковим чином зміщено зображення по висоті та ширині максимум до 15 пікселів в одному з напрямків. Напрямок обирається випадково. Після цього до зображення застосовується операція повороту на кут, випадково обраний в межах  $[-0.3; 0.3]$  радіан.

Така техніка дозволяє штучно збільшити обсяг тренувальних даних. Це призводить до того, що модель змушена краще вивчити характерні ознаки вхідних даних для досягнення таких самих результатів.

Усі вхідні зображення були приведені до стандартного розміру 60x60 пікселів у висоту та ширину. Такий розмір оптимальний для даної вибірки, оскільки лише мала кількість елементів має розмір більший. Тобто, немає сенсу оперувати зображеннями більшого розміру, оскільки навряд чи це призведе до покращення результатів нейронної мережі.

Тренування моделі відбувалося протягом 100 епох. Одна епоха — етап тренування, за який усі елементи тренувальної вибірки один раз використовуються для оптимізації ваг штучної нейронної мережі.

Таблиця 3: Порівняння результатів

Модель	Точність
Згорткова	98.86%
Багатошаровий персептрон	20.96%

Згорткові нейронні мережі дають значно кращий результат в порівнянні з багатошаровим персептроном. В абсолютних величинах — це лише 150 неправильних класифікацій із майже 12600 елементів. На рис. 9 наведено

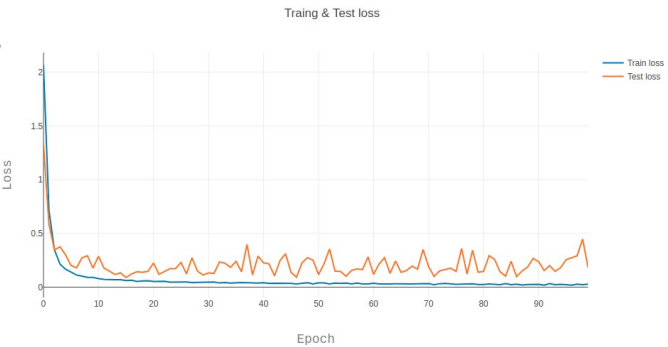


Рис. 9: Графік зміни значення цільової функції

перехресної ентропії на кожній епосі тренування. Точність тестової і тренувальної вибірки майже ідентичні, що

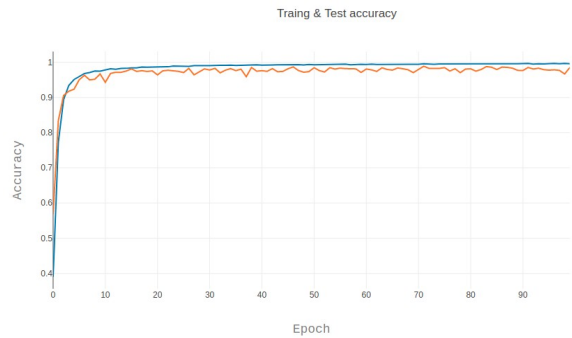


Рис. 10 Графік зміни точності моделі

свідчить про відсутність перенавчання нейронної мережі. Більшість похибок зумолені унікальними артефактами зображень: надлишкове розмиття, часткове перекриття знаку, екстремально низька роздільна здатність тощо. В деяких тестових прикладах

навіть людським зором важко правильно класифікувати зображення, проте

згорткова нейронна мережа правильно їх класифікувала.

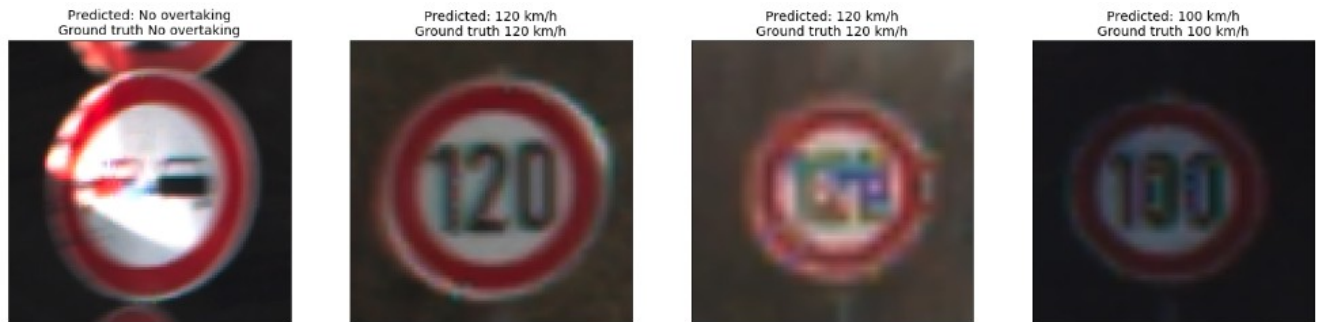


Рис. 11: Приклади правильної класифікації

На рис. 11 наведені приклади правильних класифікацій. Серед помилкових класифікацій варто відзначити рівень похибки: навіть помилкова класифікація має сенс, оскільки знаки обмеження швидкості 60 км\год на 80 км\год дуже схожі. На рис. 12 наведені приклади помилкової класифікації.

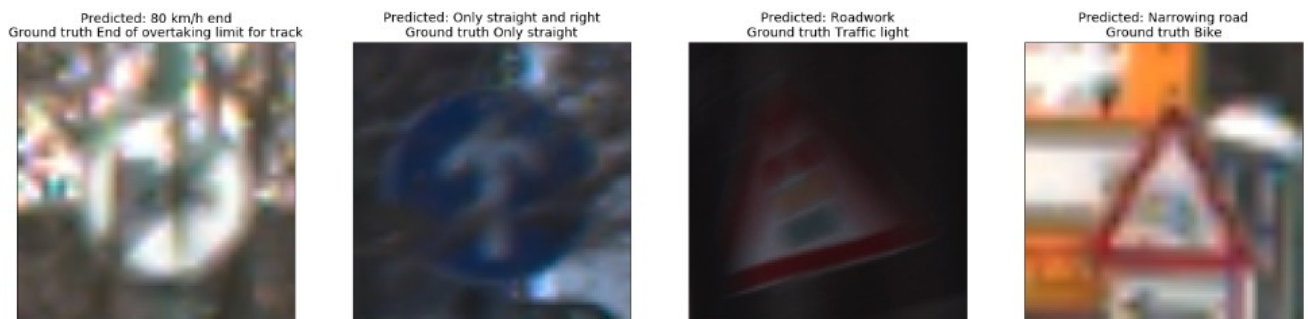


Рис. 12: Приклади помилкової класифікації

Для кращого розуміння роботи нейронної мережі варто дослідити природу похибок: в яких категоріях найчастіше стається похибка.

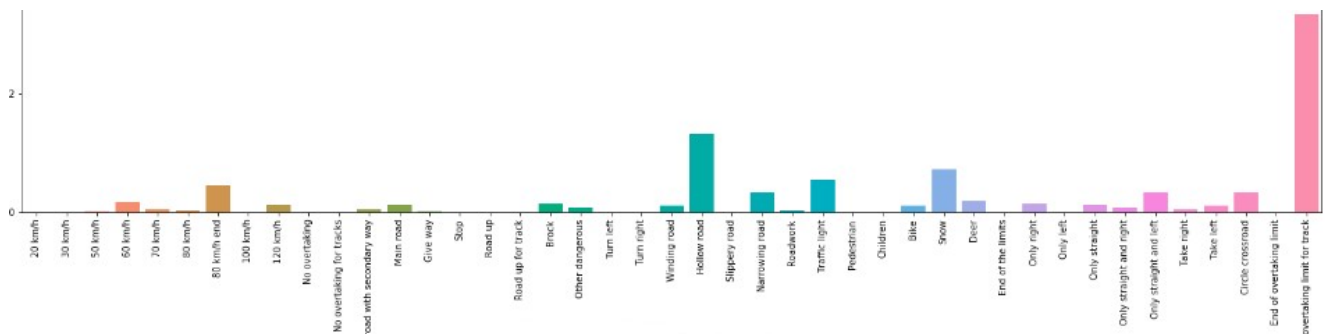


Рис. 13: Гістограма розподілу похибок по категоріях

Найбільший рівень похибок має категорія дорожніх знаків “Кінець обмеження обгону для вантажних автомобілів”. Очевидно, що в наборі даних є декілька категорій дорожніх знаків, візуально схожих до даного. Для розуміння, якими в

більшості розпізнаються неправильно прокалсифіковані знаки, слід побудувати матрицю неточностей.

Елемент матриці  $I_{i,j}$  показує,

з якою ймовірністю елемент категорії  $i$  розпізнається як

елемент категорії  $j$ . Ідеальний

класифікатор матиме

одиначну матрицю. Проте, у

випадку наведеної у

попередньому розділі моделі

найбільшою похибкою є

категорія “Кінець обмеження

обгону для вантажних

автомобілів”. Цей знак в

третині випадків

розпізнається як дорожній

знак “Кінець обмеження

швидкості 80 км\год”. Справді, ці знаки досить схожі, особливо в малих розмірах



Рис. 15: Подібність двох категорій

### 3.2.3 Аналіз натренованої мережі

Існує багато методів репрезентації знань згорткової мережі. Найпростіший спробі візуалізації — побудова зображень після згорткових шарів. Такий метод дає змогу зрозуміти, що саме відбувається в середині згорткових шарів, що вивчила модель.

На рис. 16-21 зображено результати роботи згорткових шарів.

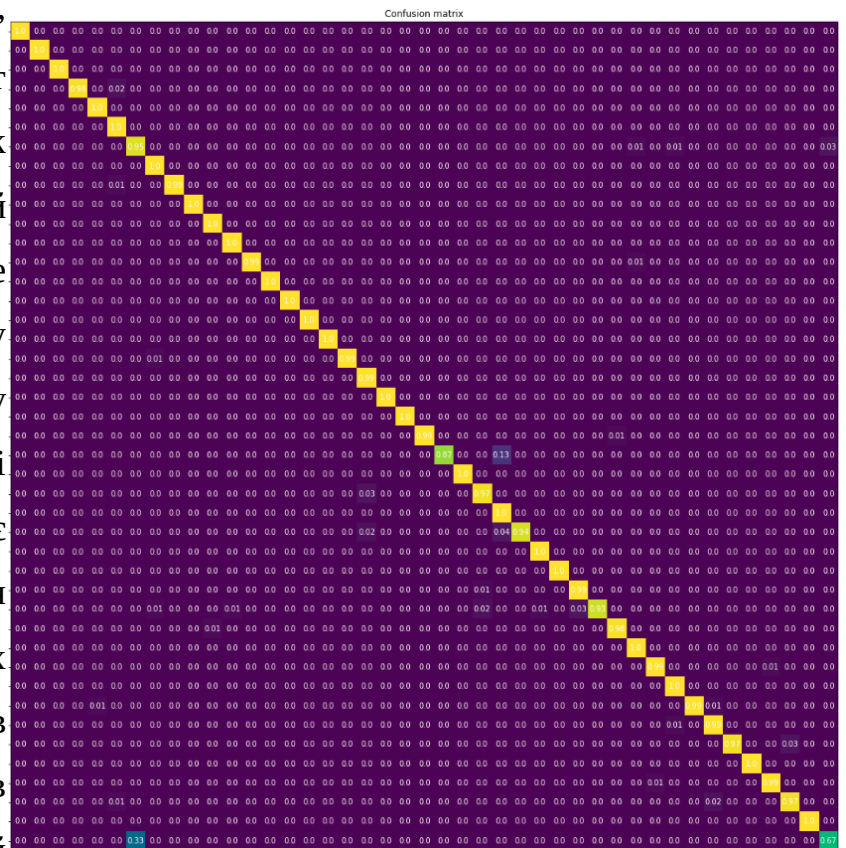


Рис. 14: Матриця неточностей

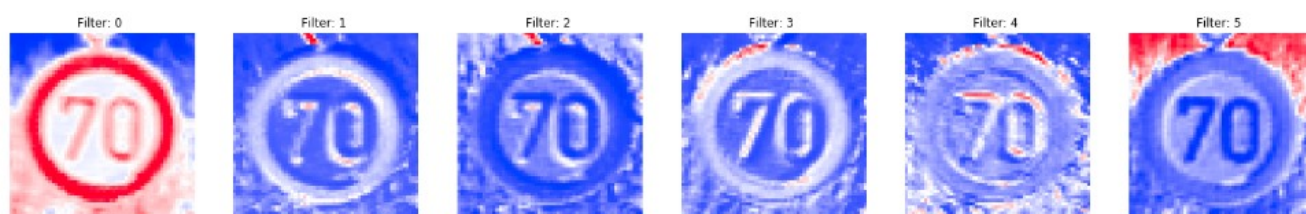


Рис. 16: Візуалізація згорткового шару №1

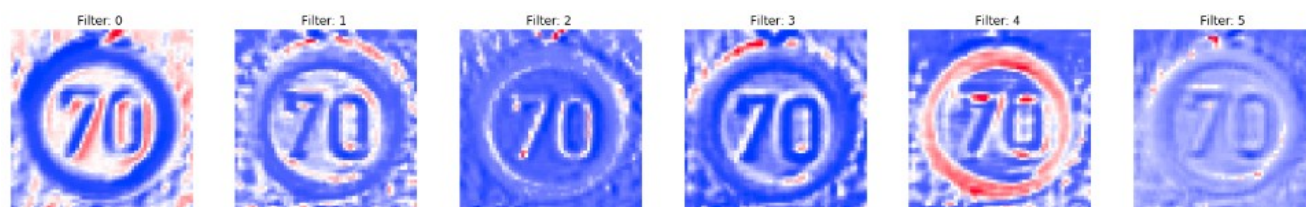


Рис. 17: Візуалізація згорткового шару №2

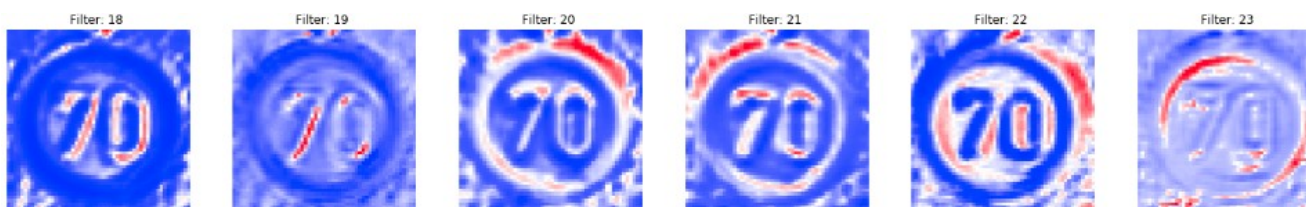


Рис. 18: Візуалізація згорткового шару №3

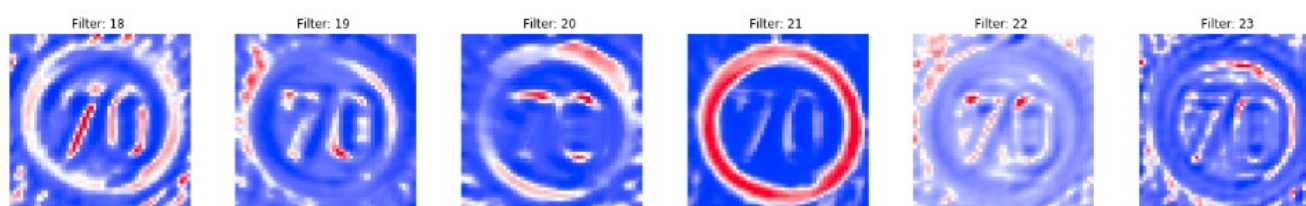


Рис. 19: Візуалізація згорткового шару №4

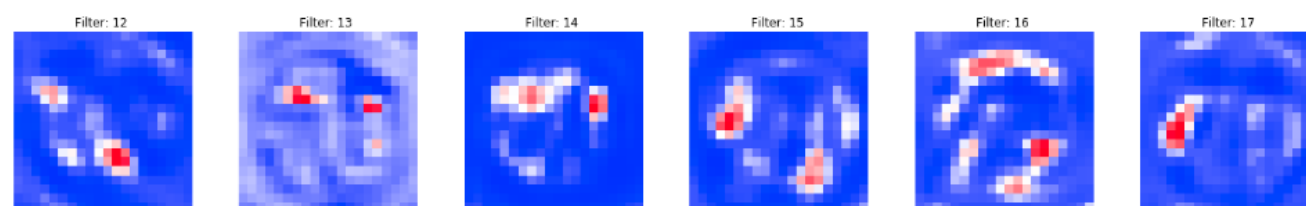


Рис. 20: Візуалізація згорткового шару №5



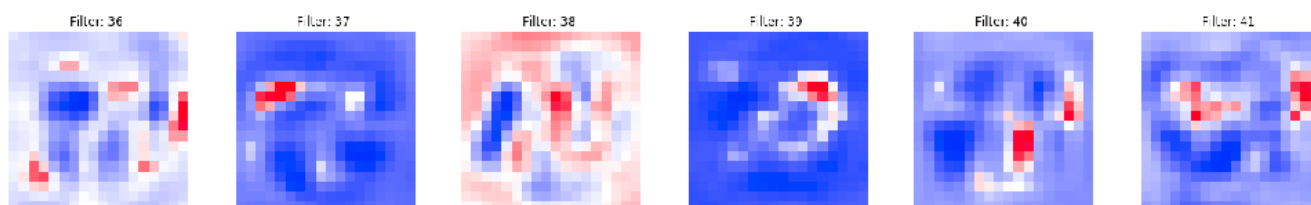


Рис. 21: Візуалізація згорткового шару №6

Недоліком такої візуалізації є те, що важко зрозуміти, яка саме частина зображення повпливала на результат. Метод Grad-CAM (Gradient base class activation map). Цей спосіб дозволяє побудувати матрицю важливості зображення. Цей метод працює завдяки зворотньому проходженню шарів нейронної мережі та обрахунку усіх градієнтів, включно до вхідних даних. Нормалізовані градієнти і становлять карту важливості. Для натренованої згорткової мережі було застосовано технологію Grad-CAM для декількох елементів тестової вибірки.

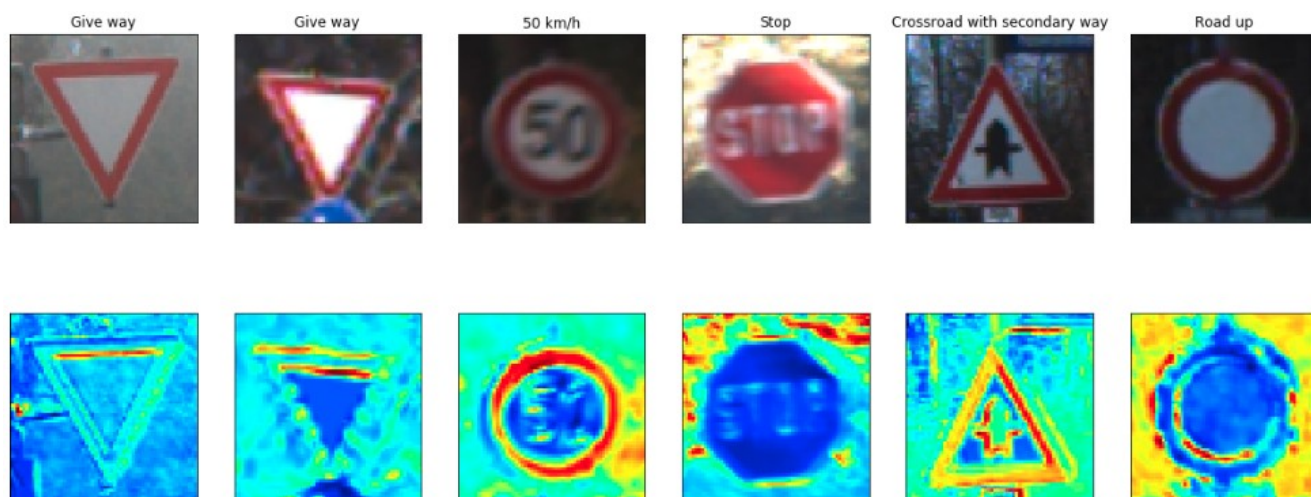


Рис. 22: Візуалізація результату Grad-CAM

Така візуалізація допомагає краще зрозуміти, які саме характеристики важливі для моделі. Так, наприклад, для розпізнавання категорії “Уступіть дорогу” важливою є рівна верхня грань знаку. Для знаку “STOP” практично неважливим є контент на тлі знаку — достатньо лише форми, оскільки вона унікальна в межах тренувальних даних.

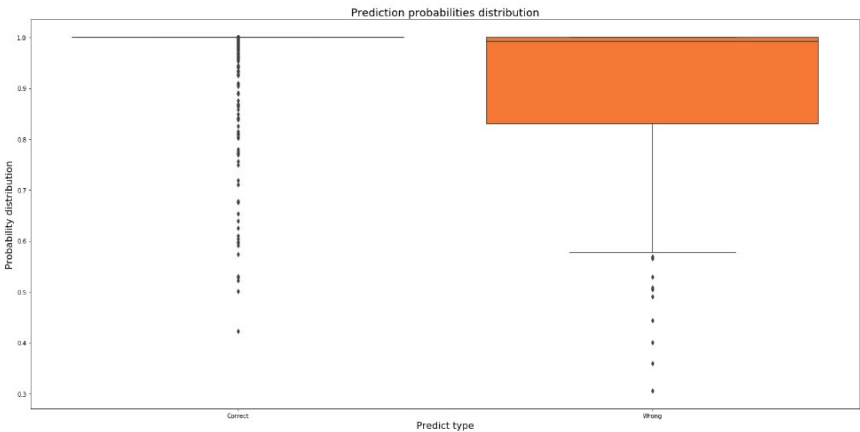
### 3.2.4 Аналіз рівня впевненості моделі

Очевидно, що без апіорної інформації про дані немає способів підвищити

точність лише аналізуючи результати моделі. Проте, є шанс зменшити кількість хибних класифікацій, аналізуючи рівень впевненості мережі. На рис. 23 представлений

розподіл рівнів впевненості в залежності від правильних та хибних класифікацій. Здебільшого, правильні класифікації мають рівень впевненості близький до 100%. Можна

спробувати ввести



порогове значення рівня впевненості, нижче якого вважати класифікацію не точною. При значенні порогу впевненості 80% вдалось зменшити кількість хибних

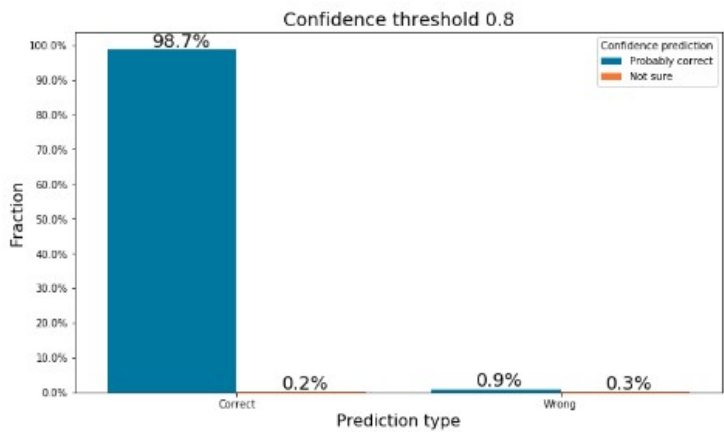


Рис. 24: Поріг впевненості 80%

класифікацій на 0.3%. За рахунок цього анульовано результат 0.2% правильних класифікацій. В залежності від конкретної задачі це може бути важливим. Для кращого розуміння поведінки порогового значення рівня впевненості на рис.

25 наведено графіки залежності кількості різних типів передбачень від значення рівня впевненості.

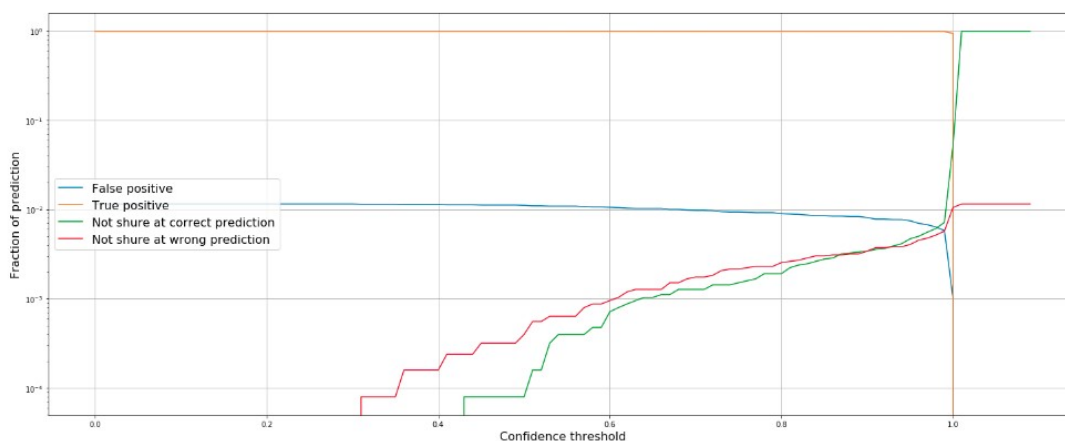


Рис. 25: Залежність кількості передбачень від порогу впевненості

### 3.3 Технічне та програмне забезпечення

#### 3.3.1 Мова програмування та додаткові бібліотеки

Уся технічна робота була реалізована на мові програмування Python версії 3.6. Як IDE — використано середовище Jupyter Notebook. Операційна система — Ubuntu 18.04 LTS.

Список додаткових бібліотек:

- NumPy — математичні обчислення з використанням ресурсів процесора  
Бібліотека NumPy — бібліотека з відкритим кодом, створена для математичних обчислень для мови Python. Математичні алгоритми, реалізовані на інтерпретованих мовах зазвичай працюють повільніше, ніж аналоги, реалізовані на компільованих мовах. Бібліотека NumPy надає інструменти для роботи з математичними обчисленнями у вигляді функцій і операторів, оптимізованих для роботи з багатовимірними даними. Самі алгоритми реалізовані на компільованих мовах, а на Python — лише обгортка. Таким чином, будь-який алгоритм може бути представлений у вигляді послідовності викликів функцій, що працюватиме так само швидко, як реалізація на компільованій мові.

- Pandas — зберігання та маніпуляції з даними  
Pandas - бібліотека з відкритим кодом, побудована з використанням бібліотеки NumPy. Надає інструментарій та структури даних для зберігання інформації та її обробки, без задіявання таких специфічних мов, як R, Octave тощо. Назва



бібліотеки походить від економетричного терміну “панельні дані” (англ. panel data).

- Matplotlib — візуалізація даних

Matplotlib — бібліотека з відкритим кодом, побудована з використанням бібліотеки NumPy. Основна ціль — візуалізація двовимірних та тривимірних графіків, які можуть використовуватись в інтерактивній графіці, наукових публікаціях, веб-аплікаціях тощо. В документації автор зізнається, що розробка бібліотеки починалась, як результат мімікрії графічним командам MATLAB, але є незалежним проектом. Перевагами бібліотеки є гнучкість, сумісність та інтеграція з IPython.

- Scikit-learn — фреймворк машинного навчання

Scikit-learn — бібліотека з відкритим кодом для мови програмування Python. Надає доступ до реалізації багатьох алгоритмів класифікації, кластеризації, регресії тощо. Архітектура бібліотеки розроблялась так, щоб було сумісність з іншими науковими пакетами: SciPy та NumPy

- Plotly — візуалізація інтерактивних графіків

Plotly — компанія, що розробляє інструменти для аналізу та візуалізації даних. Уже розроблено багато бібліотек для різноманітних мов: Python, R, MATLAB, Perl, Julia.

- Seaborn — високорівнева візуалізація даних

Seaborn — бібліотека з відкритим кодом, що дозволяє візуалізовувати двовимірні та тривимірні дані. Побудована з використанням бібліотеки matplotlib. Основною відмінністю є високорівневий інтерфейс, що дозволяє тратити мінімум зусиль для побудови візуалізацій; присутня суттєва інтеграція з бібліотекою pandas.

- tqdm — утиліта для візуалізація прогресу виконання коду

Невелика бібліотека з відкритим кодом, що дозволяє будувати інтерактивний таймер виконання коду, що є корисним при розробці алгоритмів, що опрацьовують велику кількість даних. Присутня реалізація для CLI (command line interface) та інтеграція з Jupyter Notebook (графічна візуалізація). Також є можливість використання без мови Python, з використанням pipe (Linux)

- Tensorflow — фреймворк машинного навчання

### 3.3.2 Фреймворк машинного навчання

Tensorflow — вільне, платформонезалежне програмне забезпечення (Apache 2.0) для програмування потоків даних (dataflow) та диференціального (differentiable) програмування. Розроблений командою Google Brain, опублікований 9 листопада, 2015 р. Бібліотека підтримує обчислення на різних пристроях (CPU, GPU, TPU) та різних платформах Linux, Windows, macOS, Android, iOS). Усі обчислення базуються на статичних обчислювальних графах. Корінь графа репрезентує результат роботи програми, листки — вхідні данні. Найважливішим засобом бібліотеки є автоматичне диференціювання функцій, яке застосовується безпосередньо до обчислювального графа. Для розробки нейронної мережі у Tensorflow є уже більшість типів шарів (CNN, RNN, FC, тощо), реалізованих як операція в графі, що є зручним та швидким у використанні. Зазвичай, розробка моделей відбувається з використанням Python. Є можливість розробки з використанням JavaScript, а з версії Tensorflow 2.0 також і Swift. Tensorflow є символьною мовою програмування, тобто під час розробки, написаний користувачем код виконається лише один раз, створивши обчислювальний граф і вся подальша робота буде виконуватись у цьому графі. Ця концепція має свої переваги і недоліки. Одним з найбільших недоліків є складність відлагодження, оскільки синтаксично правильний може створити обчислювальний граф з помилками. Для відлагодження процесу роботи обчислювального графу Tensorflow пропонує на вибір декілька варіантів відлагоджувачів: tfdbg (tensorflow debugger) та Tensorboard. tfdbg забезпечує CLI (command-line interface) і не завжди є зручним інструментом. Щоб краще зрозуміти, проаналізувати та відлагодити процес обчислення Tensorflow пропонує утиліту Tensorboard для візуалізації та відлагодження роботи програми. Цей інструмент дозволяє користувачеві будувати складні графіки, гістограми, матриці різноманітних обчислень в реальному часі, візуалізовувати обчислювальний граф. Також є можливість покроково відлагоджувати програму, перевіряючи значення кожного

тензора, пристрій виконання операцій, послідовність передачі даних тощо. Уніфікований вигляд обчислювального графу дає змогу легко експортувати модель на різні операційні та обчислювальні системи. Найбільшим обмеженням експорту моделі є обчислювальні ресурси пристрою. Сучасні нейронні мережі для пошуку об'єктів на зображенні чи розпізнавання мови вимагають великих затрат для обчислення з використанням потужних відеокарт. Tensorflow Lite пропонує спрощення моделі (з можливою втратою точності) —квантизацію. Квантизація дозволяє перейти від використання 32-бітових чисел до 8-бітових, що прискорює роботу моделі до чотирьох разів.

### 3.3.3 Технічне забезпечення

Процесор — Intel Core i9-9900K 3.6 GHz (5.0 GHz)

Відеокарти — NVIDIA GeForce GTX 1080ti, NVIDIA Tesla P100

## ВИСНОВКИ

Початок 21 століття став епохою відродження машинного навчання. Нові підходи, алгоритми, програмне та технічне забезпечення зумовило новий рівень розвитку науки про дані. Новий підхід побудови алгоритмів дозволяє розв'язувати завдання, які практично неможливо вирішити прямими алгоритмами. Це можливо завдяки статистичному аналізу та залежності великих об'ємів даних.

Найпоширенішим засобом керованого машинного навчання з вчителем є нейронні мережі. Завдяки математичному представлені штучної нейронної мережі відкривається безмежний простір можливостей застосування математичного апарату. Шляхом побудови та оптимізації цільової функції є можливість в автоматичному режимі відшукати усі необхідні параметри-ваги моделі. З огляду на необхідність використання великих обсягів даних постає проблема в обчислювальних ресурсах. Зазвичай, тренування нейронних мереж відбувається з використанням відеокарт. Недоліком відеокарт є відносно висока ціна та необхідність технічної підтримки. Існують ресурси, які дозволяють використовувати відеокарти (kaggle, google colab). Також є ряд платних ресурсів (Google Cloud Platform, Amazon Web Services), які дозволяють орендувати обчислювальні ресурси динамічно, в залежності від попиту.

Штучні нейронні мережі мають безліч сфер застосувань: аналіз тексту, робота з зображенням, відео, статистичними даними тощо. Існує багато уже реалізованих застосувань нейронних моделей: прогнозування погоди, розпізнавання номерних знаків, розпізнавання облич тощо. Наприклад, в Китаї техніку розпізнавання облич застосовують для пошуку злочинців в місцях скупчення людей. Відомий англійський репортер, фотографію якого завантажили у систему був розшуканий за 7 хвилин серед 4 мільярдів людей.

Незважаючи на шалений прогрес у галузях застосувань штучних нейронних мереж, досі є багато ще не розв'язаних проблем. Найбільшою з них є співвідношення швидкості роботи та якості алгоритму. В сучасному світі

важливою характеристикою алгоритму є здатність працювати в реальному часі.

У сфері обробки зображень згорткові нейронні мережі мають найкращу ефективність. Це пояснюється фактом статистичної залежності сусідніх пікселів зображення та особливостями операції згортки. Існує багато відомих архітектур для класифікації, сегментації, розпізнавання зображень тощо. Але для зображень невеликого розміру недоцільно використовувати великі архітектури, оскільки час роботи не відповідатиме бажаним очікуванням. Таким чином, найкращим рішенням є побудова та тренування згорткових нейронних мереж власноруч. З огляду на обмеженість наборів даних для тренування та тестування доцільно використовувати техніки аугментації для штучного збільшення об'ємів використовуваних даних.

Для власної реалізації нейронних мереж та інших алгоритмів машинного навчання існує декілька потужних фреймворків з підтримкою паралельного та розподіленого обчислення. Найпопулярніші з них: Tensorflow (Keras) та PyTorch (FastAI).

### СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

- [1] W. Lihua, K. Jo, Traffic sign recognition and classification with modified residual networks // Conference Paper, DOI: 10.1109/SII.2017.8279326, 2017
- [2] Leslie N. Smith, Cyclical Learning Rates for Training Neural Networks // U.S. Naval Research Laboratory, Code 5514 4555 Overlook Ave., SW., Washington, D.C. 20375
- [3] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, Antonio Torralba Computer Science and Artificial Intelligence Laboratory, MIT Learning Deep Features for Discriminative Localization // arXiv 2015
- [4] Y. Yang, H. Luo, H. Xu, and F. Wu, Towards real-time traffic sign detection and classification // IEEE Transactions on Intelligent Transportation Systems, vol. 17, pp. 2022–2031, July 2016.
- [5] S. Houben, A single target voting scheme for traffic sign detection // 2011 IEEE Intelligent Vehicles Symposium (IV), pp. 124–129, June 2011.
- [6] X. Yuan, X. Hao, H. Chen, and X. Wei, Robust traffic sign recognition based on color global and local oriented edge magnitude patterns // IEEE Transactions on Intelligent Transportation Systems, vol. 15, pp. 1466–1477, Aug 2014.
- [7] J. Jin, K. Fu, and C. Zhang, Traffic sign recognition with hinge loss trained convolutional neural networks // IEEE Transactions on Intelligent Transportation Systems, vol. 15, pp. 1991–2000, Oct 2014.
- [8] Google Brain Team, Going deeper with convolutions // arXiv, 2014
- [9] Баранов М, Бібліотека Tensorflow для задач машинного навчання // Дев'ята науково-практична конференція FOSS Lviv 2019: Збірник наукових праць/ Львів, 2019