НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ І
СПЕЦІАЛІЗОВАНИХ КОМП'ЮТЕРНИХ СИСТЕМ

«На правах рукопису»
УДК _____

«До захисту допущено»

Завідувач кафедри СПСКС

_____ В.П.Тарасенко_
(підпис) (ініціали, прізвище)

"___"_____2018р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 123 Комп'ютерна інженерія
(«Системне програмування»)

на тему: «СПОСОБИ ОРГАНІЗАЦІЇ ЗАСОБІВ НЕЙРОМЕРЕЖЕВОГО РОЗПІЗНАВАННЯ ОБ'ЄКТА НА ЗОБРАЖЕННІ»

Виконавла: студентка VI курсу, групи КВ-62м
(шифр групи)

Буц Вікторія Віталіївна _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник к.т.н., доцент Тарасенко-Клятченко О.В _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

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

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

Київ – 2018 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – другий (магістерський)
Спеціальність 123 Комп'ютерна інженерія
«Системне програмування»

ЗАТВЕРДЖУЮ
Завідувач кафедри СПСКС
_____ _В.П.Тарасенко__
(підпис) (ініціали, прізвище)

«___»_____2018р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Буц Вікторії Віталіївні
(прізвище, ім'я, по батькові)

1. Тема дисертації «Способи організації засобів нейромережевого розпізнавання об'єкта на зображенні»,
науковий керівник дисертації к.т.н., доцент Тарасенко-Клятченко О. В. ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «22» березня 2018 р. №986-с

2. Термін подання студентом дисертації 11 травня 2018 р.

3. Об'єктом дослідження є способи організації засобів нейромережевого розпізнавання об'єкта на зображенні з використанням графічних прискорювачів.

4. Предметом дослідження є методи і алгоритми реалізації архітектур згорткових мереж.

5. Перелік завдань, які потрібно розробити

- провести аналіз існуючих рішень розпізнавання об’єкта на зображенні методами глибинного машинного навчання.

- розглянути архітектури та основні принципи побудови побудови згорткових нейронних мереж

- дослідити алгоритми і методи навчання нейронних мереж, а також технології, що можуть бути застосовані для зменшення параметрів мережі та поліпшення значень критеріїв навчання.

- описати та обрати вхідні набори даних для навчання і тестування згорткової нейронної мережі.

- розробити та дослідити базову модель архітектури згорткової нейронної мережі, провести експерименти на запропонованих наборах даних застосовуючи графічні прискорювачі.

- запропонувати спосіб підвищення значень точності розпізнавання класифікатору базової моделі шляхом впровадження ієрархічного класифікатору. Створити нову модель.

- провести експерименти на запропонованих наборах для нової моделі згорткової мережі із використанням графічних прискорювачів. Порівняти та проаналізувати отримані результати.

6. Перелік ілюстративного матеріалу

- Структурна схема архітектури мережі. Нова модель

- Схема алгоритму навчання мережі методом зворотнього поширення помилки

- Схема алгоритму навчання мережі на графічних прискорювачах

- Демонстраційні гістограми роботи згорткової нейронної мережі

- Демонстраційні таблиці роботи згорткової нейронної мережі

- Структурна схема взаємозв’язків програмного забезпечення для реалізації глибинного навчання нейронних мереж

7. Перелік публікацій

- Стаття «Організація багатоетапного методу навчання згорткової нейронної мережі»

- Тези доповіді «Способи організації засобів нейромережевого розпізнавання об’єкта на зображенні з використанням графічних прискорювачів»

8. Дата видачі завдання 5 вересня 2016 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Ґрунтовне ознайомлення з предметною областю дослідження	17.12.2016	
2	Визначення структури магістерської дисертації; вивчення літератури, пошук додаткової літератури	04.03.2017	
3	Робота над першим розділом магістерської дисертації; проведення наукового дослідження	16.05.2017	
4	Проведення наукового дослідження; робота над другим розділом магістерської дисертації; розроблення програмного забезпечення	14.10.2017	
5	Проведення наукового дослідження; робота над статтею за результатами наукового дослідження	15.12.2017	
6	Проведення наукового дослідження; робота над третім розділом магістерської дисертації; підготовка матеріалів доповіді на конференції ПМК-2018.	20.02.2018	
7	Завершення роботи над основною частиною магістерської дисертації; підготовка ілюстративного матеріалу;	16.04.2018	
8	Оформлення текстової і графічної частини магістерської дисертації	20.04.2018	
9	Попередній розгляд магістерської дисертації на кафедрі	26.04.2018	

Студент _____ Буц В.В.
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Тарасенко-Клятченко О.В.
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Актуальність теми. З конжним роком зростає зацікавленість вирішення більш складних задач розпізнавання об’єктів, що обумовлена автоматизацією, необхідністю образних процесів комунікації в інтелектуальних системах. Тому удосконалення реалізації розпізнавання комютерними системами образів є актуальною. Один з перспективних напрямків вирішення даної проблеми ґрунтується на застосуванні штучних нейронних мереж і нейрокомп'ютерів, як найбільш прогресивних по відношенню проблем класифікації задач розпізнавання образів. У наш час запропоновано велику кількість архітектур нейромеж для застосування у розпізнаванні об’єктів. Аналіз запропонованих рішень показує, що й досі не існує такої моделі, яка б була кращою серед усіх результуючих показників роботи. Перспективу в удосконаленні архітектур вбачають у згорткових нейронних мережах. Переваги згорткових мереж над багатошаровими полягають у використанні спільної ваги у згорткових шарах, що означає, що для кожного пікселя шару використовується один і той же фільтр (банк ваги).

Об’єктом дослідження є способи організації засобів нейромережевого розпізнавання об’єкта на зображенні з використанням графічних прискорювачів.

Предметом дослідження є методи і алгоритми реалізації архітектур згорткових нейронних мереж.

Мета і задачі дослідження: створити нейромережеву систему розпізнавання об’єктів на зображеннях, використовуючи згорткову нейронну мережу власної архітектури із використанням ієрархічного класифікатору. Запропонувати архітектуру, яка призначатиметься для вирішення поставленної задачі найкращим чином — матиме вищі показники продуктивності, час навчання мережі, кількість параметрів при розпізнавання обєкта та вищий показник точності розпізнавання у

порівнянні з уже існуючими моделями згорткових нейронних мереж. Провести експерименти із навчання мережі використовуючи графічні прискорювачі (анлг. graphic processing unit, GPU) на семи наборах даних, а саме : CIFAR-10, CIFAR-100, GTSRB, MNIST, HASYv2, STL-10, та SVHN.

Наукова новизна полягає в наступному:

1. Вдосконалено метод розпізнавання об’єктів на зображеннях використовуючи згорткову нейронну мережу власної архітектури, що дозволило суттєво покращити метод класифікації та підвищити показник точності розпізнавання.
2. Застосовано точні налаштування та модернізований класифікатор мережі, а також перетворення вхідних даних, аби досягти низького рівня помилок та високого рівня точності класифікації у порівнянні із іншими засобами розпізнавання.

Практична цінність роботи полягає у можливості застосування отриманих результатів для ефективного використання згорткової нейронної мережі для задачі розпізнавання об’єктів.

Апробація роботи. Основні положення і результати роботи були представлені та обговорювалися на X науковій конференції магістрантів та аспірантів «Прикладна математика та комп’ютинг» ПМК-2018 (Київ, 21-23 березня 2018 р.) та опубліковані у збірнику „Наукові вісті НТУУ «КПІ»” № 10, 20118 р, а також у науковому журналі «Інтернаука», випуск 6 (березень 2018).

Структура та обсяг роботи. Магістерська дисертація складається з вступу, чотирьох розділів та висновків.

У вступі подано загальну характеристику роботи, обґрунтовано актуальність напрямку досліджень, сформульовано мету і задачі досліджень, показано наукову новизну отриманих результатів і практичну цінність роботи.

У першому розділі розглянуто основні принципи розпізнавання образів, зроблено короткий огляд існуючих методів вирішення задач комп'ютерного зору, представлено загальне поняття штучних комп'ютерних мереж та коротко описано методи їх навчання.

У другому розділі розглянуто шаблон архітектури згорткової нейронної мережі та показано, що є основою запропонованої системи розпізнавання об'єктів у вигляді базової моделі.

У третьому розділі описуються особливості наборів даних на яких буде навчатися мережа, а також принципи роботи ієрархічного класифікатору, що є інструментом для покращення точності розпізнавання. Описана основна методика навчання та тестування базової моделі мережі з використанням графічних прискорювачів.

У четвертому розділі запропоновано та впроваджено зміни щодо базової моделі для покращення точності класифікації, представлено нову модель згорткової нейронної мережі. Проведено навчання та тестування нової моделі мережі з використанням графічних прискорювачів та виконано аналіз отриманих результатів.

У висновках представлені результати проведеної роботи.

Робота представлена на 86 аркушах, містить посилання на список використаних літературних джерел.

Ключові слова: комп'ютерний зір, згорткова нейронна мережа, розпізнавання об'єкта, класифікатор, GPU.

ABSTRACT

The Relevance. Year by year, the interest in solving more complex tasks of object recognition is growing, due to automation needs for shaped communication processes in intelligent systems. Therefore, improving the implementation of recognition of computer image systems is relevant. One of the promising directions for solving this problem is based on the use of artificial neural networks and neurocomputers as the most progressive in relation to the problems of classification of pattern recognition tasks. In our time, a large number of neural network architectures are proposed for application in the recognition of objects. The analysis of the proposed solutions shows that there is still no such model that would be the best among all the resulting performance parametrs. Prospects for the improvement of architectures are seen in convolutional neural networks. The advantages of roller networks over multilayers are to use a common weight in the roller coasters, which means that for each pixel of the layer is used the same filter (weight).

The object of the study is the methods of organizing of neural network recognition of the object in the image using graphic processing units.

The subject of the study is the methods and algorithms for the implementation of architectures of convolutional neural networks.

The purpose and tasks: to create a neural network object recognition system on images, using a convolutional neural network of its own architecture using the hierarchical classifier. To propose an architecture that is best suited to solve a task - it will have higher performance, network learning time, number of parameters for object recognition and higher recognition accuracy compared to existing models of convolutional neural networks. Conduct network learning experiments using graphical processing units (GPUs) on seven data sets, namely: CIFAR-10, CIFAR-100, GTSRB, MNIST, HASYv2, STL-10, and SVHN.

The scientific novelty is as follows**:**

1. Improved method of object recognition in images using a convolutional neural network of its own architecture, which allowed to significantly improve the classification method and increase the accuracy of recognition.

2. The precise settings and upgraded network classifier, as well as the conversion of input data, have been applied to achieve low error rates and a high degree of accuracy of the classification compared with other means of recognition.

The practical value of the work is in the possibility of using the results obtained for the efficient use of the convolutional neural network for the task of object recognition.

Publications. The main provisions and results of the work were presented and discussed at the Xth Conference of Masters and Postgraduates "Applied Mathematics and Computing" of the AMC-2018 (Kyiv, March 21-23, 2018) and published in the collection "Scientific News of NTUU" KPI " № 10, 20118, as well as in the scientific journal "Internet Science", issue 6 (March 2018).

Structure and scope of work. The master's thesis consists of an introduction, four chapters and conclusions.

The *introduction* gives a general description of the work, substantiates the relevance of the research direction, formulates the purpose and objectives of the research, shows the scientific novelty of the results obtained and the practical value of the work.

In the *first* chapter the main principles of image recognition are considered, a brief overview of existing methods of solving computer vision problems is presented, the general concept of artificial computer networks is presented and the methods of their teaching are briefly described.

The *second* chapter discusses the architecture of the convolutional neural network architecture and shows that it is the basis of the proposed object recognition system as a base model.

The *third* chapter describes the peculiarities of the data sets on which the network will be studied, as well as the principles of the hierarchical classifier, which is a tool for improving the accuracy of the recognition. The basic method of training and testing of the basic model of the network using graphic accelerators is described.

The *fourth* chapter proposes and introduces changes to the basic model for improving the accuracy of the classification, introduces a new model of the convolutional neural network. The training and testing of a new model of the network using graphic accelerators has been carried out and the analysis of the obtained results has been performed.

The *conclusions* contains of the results of the work.

The work is presented on 86 pages, contains a link to the list of used literary sources.

Keywords: computer vision, convolutional neural network, object recognition, classifier, GPU.

Реферат

Актуальность темы. С каждым годом растет заинтересованность решения более сложных задач распознавания объектов, обусловленная автоматизацией, необходимостью образных процессов коммуникации в интеллектуальных системах. Поэтому совершенствование реализации распознавания комютернимы системами образов является актуальной. Одно из перспективных направлений решения данной проблемы основывается на применении искусственных нейронных сетей и нейрокомпьютеров, как наиболее прогрессивных по отношению проблем классификации задач распознавания образов. В наше время предложено большое количество архитектур нейросетей для применения в распознавании объектов. Анализ существующих решений показывает, что до сих пор не существует такой модели, которая была бы лучшей среди всех результирующих показателей работы. Перспективу в совершенствовании архитектур видят в сверточных нейронных сетях. Преимущества сверточных сетей над многослойными заключаются в использовании общих весов в сверточных слоях, что означает, что для каждого пикселя сверточного слоя используется один и тот же фильтр (банк веса).

Объектом исследования являются способы организации средств нейросетевого распознавания объекта на изображении с использованием графических ускорителей.

Предметом исследования являются методы и алгоритмы реализации архитектур сверточных нейронных сетей.

Цель и задачи исследования: создать нейросетевую систему распознавания объектов на изображениях, используя сверточную нейронную сеть собственной архитектуры с использованием иерархического классификатора. Предложить архитектуру, которая будет

предназначаться для решения поставленной задачи наилучшим образом – достигнуть более высоких показателей производительности, время обучения сети, количество параметров при распознавании объекта и высокого показателя точности распознавания по сравнению с уже существующими моделями сверточных нейронных сетей. Проделать эксперименты по обучению сети используя графические ускорители (анлг. Graphic processing unit, GPU) на семи наборах данных, а именно: CIFAR-10, CIFAR-100, GTSRB, MNIST, HASYv2, STL-10 и SVHN.

Научная новизна заключается в следующем:

1. Усовершенствован метод распознавания объектов на изображениях используя сверточную нейронную сеть собственной архитектуры, что позволило существенно улучшить метод классификации и повысить показатель точности распознавания.
2. Применены точные настройки и модернизированный классификатор сети, а также преобразования входных данных, чтобы достичь низкого уровня ошибок и высокого уровня точности классификации по сравнению с другими средствами распознавания.

Практическая ценность работы состоит в возможности применения полученных результатов для эффективного использования сверточной нейронной сети при решении задач распознавания объектов.

Апробация работы. Основные положения и результаты работы были представлены и обсуждались на X научной конференции магистрантов и аспирантов «Прикладная математика и компьютинг» ПМК-2018 (Киев, 21-23 марта 2018) и опубликованы в сборнике "Научные вести НТУУ« КПИ »" № 10 20118 г., а также в научном журнале «Интернаука», выпуск 6 (март 2018).Структура та обсяг роботи.

Магистерская диссертация состоит из введения, четырех разделов и выводов.

Во введении представлена общая характеристика работы, обоснована актуальность направления исследований, сформулированы цели и задачи исследований, показано научную новизну полученных результатов и практическую ценность работы.

В первом разделе рассмотрены основные принципы распознавания образов, сделано краткий обзор существующих методов решения задач компьютерного зрения, представлено общее понятие искусственных компьютерных сетей и коротко описаны методы их обучения.

Во втором разделе рассмотрены шаблон архитектуры сверточной нейронной сети и показано, что является основой предложенной системы распознавания объектов в виде базовой модели.

В третьем разделе описываются особенности наборов данных на которых будет учиться нейрронная сеть, а также принципы работы иерархического классификатора, который является инструментом для улучшения точности распознавания. Описана основная методика обучения и тестирования базовой модели сети с использованием графических ускорителей.

В четвертом разделе предложены и внедрены изменения относительно базовой модели для улучшения точности классификации, представлена новая модель сверточной нейронной сети. Проведено обучение и тестирование новой модели сети с использованием графических ускорителей и выполнен анализ полученных результатов.

В выводах представлены результаты проведенной работы.

Работа представлена на 86 листах, содержит ссылки на список использованных литературных источников.

Ключевые слова: компьютерное зрение, сверточная нейронная сеть, распознавание объекта, классификатор, GPU.

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

Графічні прискорювачі (анлг. graphic processing unit, GPU) - окремий пристрій персонального комп'ютера або ігрової приставки, що виконує графічний рендеринг.

ЗНМ – згорткова нейронна мережа.

Кластер (англ. cluster) — група однакових або подібних елементів, зібраних разом або близько розташованих один до одного.

Набір даних (англ. data set) — колекція однотипних даних, що застосовується в задачах машинної обробки даних.

Синапс нейрона (від грецького «synapsis» — з'єднання) — структура, яка дозволяє нейрону (нейронної мережі) проводити сигнал (у випадку штучного нейрону) до іншого нейрону.

Тензор (від лат. tendere, «тягнутись, простиратися») - математичний об'єкт, що узагальнює такі поняття як скаляр, вектор, ковектор, лінійний оператор і білінійна форма. Вивченням тензорів займається тензорне числення.

BN – Batch-нормалізація.

ВСТУП

Машинне навчання, разпізнавання об’єктів та комп’ютерний зір є одними з найперспективніших у подальшому розвитку серед галузей сучасної кібернетики. Прискорені темпи розвитку технологій, робототехніки, концепцій «розумного будинку» і «розумного міста», розвиток інтернет-речей і систем штучного інтелекту визначають цим напрямкам особливе місце у науці.

Існує багато методів та запропоновано багато алгоритмів рішення задачі розпізнавання об’єктів на зображенні, проте усі ці ідеї поступаються у точності результату, простоті та швидкодії штучним нейронним мережам. В основі сучасних глибоких нейронних мереж, як правило, лежать архітектури мереж згорткового типу, таких як когнітрон і неокогнітрон. Їх ефективність і стрімкий розвиток обумовлено гібридним підходом до архітектурних рішень, розвитком методів навчання, додаткових методів захисту від перенавчання. Внаслідок зростаючої популярності глибоких згорткових мереж досягаються суттєві успіхи у розпізнаванні об’єктів.

В даній роботі запропоновано архітектуру згорткової нейронної мережі, що створює нейромережеву систему розпізнавання об’єктів на зображеннях, використовуючи власний підхід до класифікації із застосуванням ієрархічного класифікатору. Архітектура призначатиметься для вирішення поставленної задачі найкращим чином — вона є універсальною для багатьох наборів даних забороражень і має вищі показники продуктивності, такі як затрати пам’яті, час навчання мережі, кількість параметрів при розпізнаванні, а головне краще значення точності розпізнавання об’єктів у порівнянні з уже існуючими моделями згорткових нейронних мереж. Основну увагу приділено підходу до навчання такої мережі та проведення есперименів на сформованих вибірках різних наборів даних, використовуючи графічні прискорювачі (анлг. graphic processing unit, GPU).

РОЗДІЛ 1. РОЗПІЗНАВАННЯ ОБ'ЄКТА МЕТОДАМИ МАШИННОГО НАВЧАННЯ

Машинне навчання є головною складовою теорії штучного інтелекту, глибокою математичною дисципліною, до якої відносяться науки математична статистика та теорія ймовірностей. Зазначимо два типи машинного навчання, що активно використовуються: індуктивне навчання та дедуктивне навчання. Індуктивне навчання базується на обробці емпіричних даних, а дедуктивне – на формалізації отриманих знань та подальшого їх упорядкування. Прийнято вважати, що область експертних систем включає дедуктивне навчання, тому і використовується в теорії і практиці машинного навчання загалом.

Розділ машинного навчання виник у результаті поділу науки про нейронні мережі в рамках науки про штучний інтелект на методи навчання мереж і види топологій архітектури мереж, увібравши в себе деякі інші області, такі як методи математичної статистики і теорію дискретного аналізу.

1.1 Теорія розпізнавання об'єкта

Теорія розпізнавання образів існує як розділ інформатики та суміжних дисциплін, що розвиває методи класифікації та ідентифікації об'єктів різної природи: сигналів, ситуацій, предметів, що характеризуються вичерпною кількістю деяких ознак [1]. Проблема розпізнавання об'єктів також виділена в розділ міждисциплінарних досліджень - в тому числі включаючи роботу зі створення штучного інтелекту, а також часто використовується при вирішенні практичних завдань у галузі комп'ютерного зору.

При постановці класичної задачі розпізнавання об'єктів заведено застосовувати математичну мову, ґрунтуючись на логічних міркуваннях і математичних принципах. В протилежність до цього підходу, існують методи розпізнавання об'єктів з використанням машинного навчання і

штучних нейронних мереж, сформовані не настільки формалізованих підходах до розпізнавання, і демонструють не гірший, а в деяких випадках і значно кращий результат.

Часто можна зустріти хибне зіствлення термінів “розпізнавання” та “класифікація” де вони розглядаються як синоніми, але не є повністю взаємозамінюваними. Кожен з цих двох термінів може мати свої сфери застосування, в залежності від поставленої задачі. Розглянемо загальні елементи моделі класифікації.

Клас - множина об'єктів, що мають спільні властивості. Для об'єктів одного класу передбачається наявність «схожості». Для задачі розпізнавання може бути визначено довільну кількість класів, більше одного. Кількість класів позначається числом S. Кожен клас має свою ідентифікуючу мітку класу [1].

Класифікація - процес призначення міток класу об'єктам, відповідно до певного опису властивостей цих об'єктів. Класифікатор – це цнструмент для присвоєння міток класам, який в якості вхідних даних отримує перелік ознак об'єкта. До одного з найпоширеніших способів класифікації можна віднести спосіб, що базується на описі об'єктів з використанням ознак, де кожен об'єкт характеризується набором числових або нечислових ознак. Проте існують типи даних для яких відкриті ознаки не дають високої точності класифікації, наприклад, колір точок зображень або цифровий звуковий сигнал. Загальна класифікація зображень собак і автомобілів є дуже простою для людини і водночас складною для машини. Причиною цього є можливість людини сприймати «приховані ознаки» недоступні для машини, такі як морда собаки або колеса автомобіля [1].

Верифікація - процес зіставлення досліджуваного об'єкта із однією моделлю об'єкта або описом класу. [1]

Під образом будемо розуміти найменування області в просторі ознак, в якій відображається безліч об'єктів або явищ матеріального світу.

Ознакою можна назвати опис тієї чи іншої властивості, що має пряме відношення до предмета або явища.

Простір ознак - це N-вимірний простір, визначений для даної задачі розпізнавання, де N - фіксоване число ознак, що були виміряні для будь-яких об'єктів. Вектор з простору ознак x, відповідний об'єкту задачі розпізнавання, це N-вимірний вектор з компонентами $(x_1, x_2, \dots, x_N)$, що утворюють значення ознак для даного об'єкта [1].

Іншими словами, розпізнавання образів можна визначити, як віднесення вихідних даних до певного класу за допомогою виділення істотних ознак або властивостей, які характеризують ці дані, із загальної маси несуттєвих деталей.

Прикладами завдань класифікації є розпізнавання символів, встановлення медичного діагнозу, прогноз погоди, розпізнавання осіб, класифікація документів, тощо.

Найчастіше вихідним матеріалом служить отримане із камери зображення. Постановку завдання можна сформулювати, як одержання векторів, що складаються з ознак для кожного класу на зображенні. Процес можна розглядати як процес кодування, що полягає в присвоєнні значення кожної ознаки із простору ознак для кожного класу.

Якщо розглянути 2 класи об'єктів: дорослі і діти. В якості значення ознак можна вибрати зріст і вагу (Рис. 1.1).

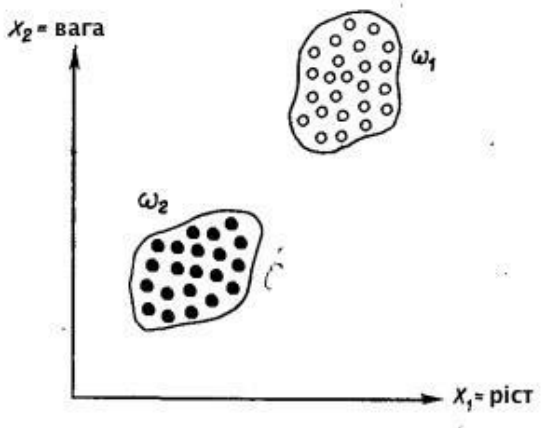


Рис 1.1 - Два непересічних класи

З рис. 1.1 видно, що ці два класи утворюють дві непересічні множини, що можна пояснити обраними ознаками. Проте не завжди є можливість вибрати правильні вимірювані параметри в якості ознак класів. Наприклад, вибрані параметри не підійдуть, щоб створити непересічні класи хокеїстів та волейболістів.

Іншим завданням розпізнавання є виділення із вихідних зображень характерних ознак та/або властивостей. Це завдання можна віднести до попередньої обробки. Ознака повинна являти собою характерну властивість конкретного класу, при цьому загальну для цього класу. Міжкласові ознаки – це ознаки, що визначають відмінності між класами. Загальні ознаки, що властиві усім класам, не несуть корисної інформації, тому для задачі розпізнавання об’єктів не розглядаються як характерні. Вибрати правильні ознаки - одна із важливих задач побудови систем розпізнавання.

1.2 Способи вирішення задач комп’ютерного зору

Комп'ютерний зір, у свою чергу, розвивається як теорія і технологія створення машин, які можуть виявляти, відстежувати та класифікувати об'єкти. Як наукова дисципліна, комп'ютерний зір тісно пов'язане з машинним навчанням та теорією розпізнавання образів, але відноситься до більш спеціалізованої технології створення штучних систем, які отримують інформацію та оперують інформацією з зображень. Автоматичне планування або прийняття рішень на основі підсистем комп'ютерного зору так само займає важливу частину в області штучного інтелекту, оскільки автономні системи досить складного рівня організації, які виконують деякі механічні дії (наприклад, переміщення робота), потребують високорівневих даних, що представляють інформацію про середовища, в яких вони функціонують.

Класичні методи комп'ютерного зору, розпізнавання об’єктів і машинного навчання можна умовно розділити на три групи: методи фільтрації, методи логічного аналізу, методи навчання.

1.2.1 Методи фільтрації

У завданнях комп'ютерного зору фільтрація найчастіше використовується для попередньої обробки зображення перед аналізом його внутрішніх морфологічних ознак, але зустрічаються і завдання, в яких достатнім і бажаним буде використання тільки фільтрації (наприклад, в задачах машинного зору).

Бінаризація по порогу [2], вибір області гістограми. Для RGB зображень і зображень в градаціях сірого - порогом є значення кольору. Встановлення порогу, за яким і відбудеться бінаризація, багато в чому визначає процес бінаризації. Зазвичай бінаризація здійснюється за допомогою адаптивного алгоритму, що вибирає поріг. Таким алгоритмом може бути вибір математичного очікування, моди або піків гістограми. При роботі з гістограмою бінаризація ефективна для сегментації кольорів.

Перетворення Фур'є майже не використовується при обробці зображень в чистому вигляді, оскільки для аналізу зображень одновимірного перетворення зазвичай не вистачає і виникає необхідність використання куди більш ресурсного двовимірного перетворення. Цей метод застосовується тільки в разі якщо необхідний аналіз спектра, оскільки використання згортки цікавить області з уже готовим фільтром. Проте, одномірне перетворення Фур'є застосовується при компресії зображень.

Фільтри частот. Найпростіший приклад фільтра низьких частот - фільтр Гаусса, фільтра високих частот - фільтр Габора. Для кожної точки зображення вибирається вікно, в рамках якого виконується множення вихідних даних з фільтром того ж розміру (згортка). Такий підхід отримав досить широке поширення в практиці, дозволяючи виділяти на зображенні

необхідну інформацію, відсіюючи зайву інформацію. Зокрема, підхід поширений як одна з реалізацій швидкого шумозаглушення в багатьох областях науки і техніки [2].

Вейвлет-перетворення [2]. Для згортки з сигналом (областю зображення) використовується якась характеристична функція, названа вейвлетом, що визначається як вейвлет-перетворення. Вейвлети - це сімейства функцій, локальних за часом і по частоті, в яких всі функції виходять з однієї з функцій, за допомогою її змін положення і розтягування по осі, що відображає час. Існує набір класичних функцій, залучених в вейвлет-аналізі. До них відносяться вейвлет Хаара, вейвлет Морлі, вейвлет MHat, вейвлет Добеши. На практиці вейвлет-аналізом називається пошук довільного патерну на зображенні за допомогою згортки зображення з моделлю цього патерну. Класичні вейвлети використовуються для стиснення або класифікації зображень.

Метод обчислення кореляції [2], що лежить в основі вейвлет-перетворення, саме по собі є незамінним інструментом в системах комп'ютерного зору і часто використовується в своєму природному вигляді, наприклад, для знаходження зрушень або оптичних потоків (кореляція відеопотоку). На основі обрахованої корелятором різниці реалізується найпростіший детектор зсуву.

Метод фільтрів функцій [2]. У цьому підході використовуються математичні фільтри, що дозволяють виявляти прості математичні функції на зображенні, для чого формується акумулююче зображення (накопичувальний простір) в якому для кожної точки вихідного зображення будується безліч породжуючих її функцій. Класичним прикладом є узагальнене перетворення Хафа, що застосовується до бінаризованих зображень, що дозволяє знаходити на зображенні функції. Модифіковане перетворення Хафа дозволяє шукати будь-які фігури, але в обробці зображень його використання пов'язане з недостатньою

стабільністю: висока чутливість до якості бинаризації і низька швидкість роботи змушують шукати більш ефективні методи.

1.2.2 Методи логічного аналізу

За допомогою фільтрації можна отримати набір даних, що буде придатний для обробки, але більшість завдань галузі комп'ютерного зору вимагають аналізу внутрішньої структури зображення і морфологічних ознак зображених об'єктів, для цього досліджуються і впроваджуються методи логічного аналізу зображень.

Методи математичної морфології є результатом теорії та техніки аналізу й обробки геометричних структур, заснованих на теорії множин, топології та випадкових функціях. Ці методи реалізуються базовими операціями. Операції двійкової морфології є деяким перетворенням впорядкованої множини або підмножини (області зображення) за допомогою структурного елементу. Структурним елементом є двійкове зображення довільного розміру та довільної структури, але найчастіше використовуються симетричні елементи, такі як прямокутник фіксованого розміру або коло фіксованого діаметру. Результатом перетворення також є двійкове зображення [2].

Методи математичної морфології дозволяють видаляти шуми з довійкових зображень, а також реалізують алгоритми пошуку контурів, але на практиці ці методи використовуються в поєднанні з іншими алгоритмами.

Контурний аналіз - це потужний математичний апарат, що дозволяє описувати, зберігати і знаходити об'єкти які знаходяться в формі зовнішніх контурів. Вище розглядалися фільтри контурів, результатом застосування яких природним чином є контури об'єктів на зображенні без застосування додаткової бінаризації. У контурному аналізі попередні стадії фільтрації і додаткової бінаризации є обов'язковими етапами завдання. Передбачається, що контур містить необхідну інформацію про форму

об'єкта, а внутрішні точки до уваги не приймаються, що обмежує область застосування алгоритмів контурного аналізу. Проте отримані з його допомогою контури дозволяють перейти від двовимірного простору образу до простору контурів, в деяких завданнях це значно зменшує складність алгоритму. Методи контурного аналізу інваріантні щодо перенесення, повороту і масштабування зображення об'єкта. Серед методів контурного аналізу можна виділити алгоритми, в яких контур об'єкта відстежується і векторизуется; скануючі алгоритми, засновані на перегляді всього зображення і виділення контурних точок без відстеження; алгоритми відстеження контурів на напівтонових зображеннях, дослідження кривизни функцій.

На практиці методи контурного аналізу досить чутливі до умов середовища, що може викликати складність їх використання в реальних умовах для більшості завдань комп'ютерного зору, проте вони корисні в задачах машинного зору, коли умови середовища досить строго визначені. У таких ситуаціях методи контурного аналізу є неперевершеними лідерами по швидкодії, володіючи зрозумілою і простою логікою, що обумовлює зручність їх використання в спеціалізованих областях.

Пошук особливих точок (feature detection). Для вирішення допоміжної прикладної задачі в даній роботі буде застосований один з алгоритмів *feature detection* [2], тому на цій темі варто зупинитися докладніше. Пошук особливих точок є одним з найбільш поширених методів в класичному комп'ютерному зорі. Особливі точки надають унікальні характеристики об'єкта, дозволяючи порівнювати різні зображення одного об'єкта або одного класу об'єктів між собою. Тому на практиці особливі точки найбільш актуальні в задачах, в яких можливим рішенням є обробка серії зображень або відео потоку і подальший аналіз отриманих масивів особливих точок. Алгоритми feature detection можна умовно класифікувати за ступенем стабільності точок при переходах від одного зображення (кадру) об'єкта або класу об'єктів до іншого.

Складність алгоритмів пошуку зростає з необхідним рівнем стабільності шуканих точок.

Результатом роботи таких алгоритмів є безліч особливих точок, зокрема, кутів, для яких необхідно побудувати математичний опис. Формування математичного опису - це завдання дескриптора. Багато дескрипторів одночасно вирішують завдання пошуку особливих точок і побудови описів цих точок за допомогою вбудованих алгоритмів або оригінальним власним способом. Ознаки (описи) будуються на основі інформації про інтенсивність, про кольори та текстуру особливих точок.

1.3 Штучна нейронна мережа

Штучна нейронна мережа є концептуальною моделлю біологічної нейронної мережі і складається з пов'язаних між собою різним чином шарів штучних нейронів, які організовують загальну активну структуру і функціонально впливають на роботу один одного. У більшості архітектур штучних нейромереж активність нейрона визначається перетворенням зовнішнього сумарного впливу інших нейронів на даний нейрон.

З моменту свого зародження технології штучних нейронних мереж розвивалися досить відокремлено від класичних методів, нерідко докорінно змінюючи уявлення про предмет і проблематику теорії машинного навчання і розпізнавання об’єктів, залишаючи значний вплив на теоретичний, термінологічний і методологічний апарати цих дисциплін. Через деякий час після розвитку базових моделей штучних нейронних мереж, відбувся значний поділ науки про нейромережі на види топологій архітектури мереж і методи навчання мереж.

У більшості архітектур штучних нейронних мереж функції активації нейронів фіксовані, а ваги синапсів є параметрами мережі. Деякі входи нейронів є зовнішніми входами сукупної мережі, а деякі виходи нейронів - виходами сукупної мережі. Завдання нейромережі полягає в перетворенні

вхідного вектора у вихідний вектор, що здійснюється вагою і топологією мережі.

1.3.1 Модель штучного нейрона та його функції активації

Штучний нейрон характеризується своїм поточним станом. Тут можна провести аналогію з нервовими клітинами головного мозку, які можуть бути пошкоджені або неправильно правцювати. Він володіє групою синапсів - односпрямованих вхідних зв'язків, з'єднаних з виходами інших нейронів, присутня така частина, як аксон – вихідний зв'язок штучного нейрона, з якого сигнал (збудження або гальмування) надходить на синапси наступних нейронів. Загальний вигляд штучного нейрона наведено на рисунку 1.2. Штучний нейрон спочатку імітує властивості, що дані біологічному нейрону. Тут безліч вхідних сигналів, позначених $x_1, x_2, \dots, x_N$, надходить на штучний нейрон. Ці вхідні сигнали, в сукупності позначаються вектором X, приходять до синапсів біологічного нейрона. Кожен синапс характеризується величиною сінапсичного зв'язку або його вагою w_i [3].

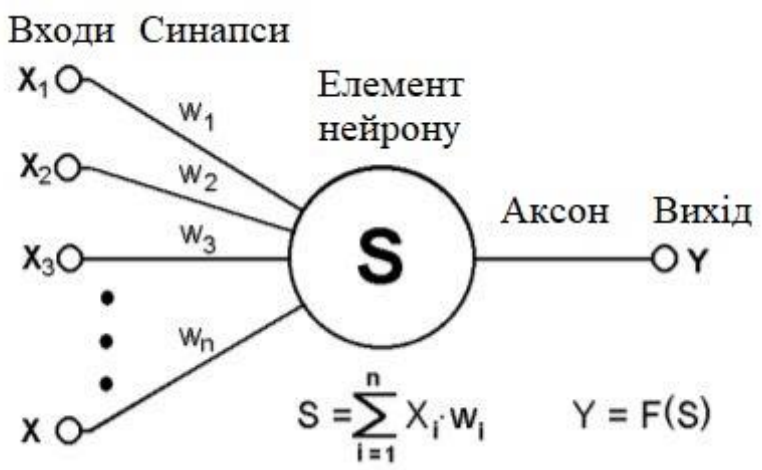


Рис. 1.2 - Модель штучного нейрона

Для реалізації нелінійності при активації нейрона, його активність, крім різних видів суматорів і систем ваг на входах, визначається функцією одного аргументу - функцією активації. Нейрон в цілому реалізує скалярну функцію векторного аргументу, а вихідний сигнал нейрона визначається видом функції активації і може бути дійсним або цілим. Функція активації

застосовується до зваженої суми постсинаптичних сигналів на вході нейрона. Таким чином, активність нейрона повністю визначається його параметрами - вагами і його функцією активації. Існує безліч передавальних функцій, що застосовуються на практиці при розробці нейронних мереж, деякі з них служать для реалізації нелінійності системи. Вибір тієї чи іншої функції активації часто залежить від умов завдання і структури мережі. Деякі з розглянутих нижче функцій застосовуються тільки в застарілих системах або в навчанні, але вважаються класичними і згадуються щоразу при вивченні штучних нейронних мереж (Рис. 1.3).

Порогова функція Хевісайда є найпростішою кусочно-лінійною передавальною функцією. Ця функція використовувалася в класичному персептроні, в даний час використовується в основному з метою навчання теорії нейронних мереж.

Лінійна функція активації. При використанні нескладної кусочно-лінійної функції сигнал на виході нейрона лінійно пов'язаний зі зваженою сумою сигналів на його вході. В даний момент лінійна функція на практиці також використовується дуже рідко.

Сигмоїдальна функція активації. Сигмоїд є монотонно зростаючою всюди диференційованою S-образною нелінійною функцією з насиченням. Забезпечує посилення слабких сигналів і запобігає насиченню сильних сигналів. Є однією з найбільш поширених передавальних функцій, часто використовується в нейронних мережах і сьогодні. Введення сигмоїдальних функцій було зумовлено недостатньою гнучкістю класифікаторів на основі порогових передавальних функцій і дозволило перейти від жорсткої однорозрядної логіки до більш гнучкої поведінки і адаптивної параметризації нейронних мереж. Прикладом сигмоидальной функції є логістична передавальна функція [3].

Функція гіперболічного тангенса відрізняється від логістичної кривої тим, що її область значень лежить в інтервалі (-1; 1), що в деяких випадках може спростити завдання навчання нейромерж.

ReLU (Rectified Linear Unit). У неглибоких нейромережах використовуються нелінійні функції активації. Часто зустрічаються різновиди сигмоїдальних і тангенціальних функцій є нелінійними, але на практиці при навчанні глибоких нейромереж такі функції можуть привести до проблем із загасанням або збільшенням градієнтів. Функція ReLU є випрямленою лінійною функцією і на даний момент вважається набагато більш простим і ефективним з точки зору обчислювальної складності варіантом передавальної функції. Похідна цієї функції дорівнює або 0, або 1, від чого її застосування запобігає розростанню і загасанню градієнтів, і призводить до зменшення ваг, що позитивно позначається на обчислювальної здатності нейромережі. Передавальна функція ReLU (1.1) є одним з останніх успіхів в області методів налаштувань глибоких нейронних мереж.

$$f\,(x) = \max\,(0,\, x) \tag{1.1}$$

У формулі (1.1) x - вхід нейрона.

Сьогодні існує сімейство різних модифікацій ReLU, які вирішують проблеми надійності цієї функції при проходженні через нейрон великих градієнтів: Leaky ReLU, Parametric ReLU, Randomized ReLU [3].

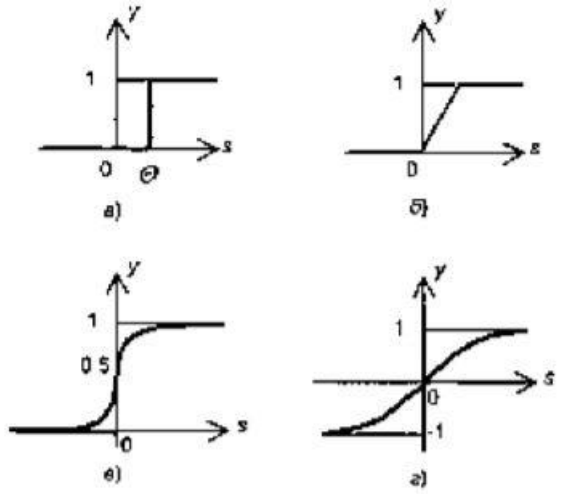


Рис. 1.3 - Приклади функцій активації:
а - функція одиничного стрибка; б - лінійний поріг;
в - логістична функція; г - гіперболічний тангенс

1.3.2 Одношарові штучні нейронні мережі

Можливість простого розпізнавання доступна одному нейрону, проте з'єднання тих же нейронів в мережі - заропука кращого результату. Одношаровою є мережа, що складається з сукупності нейронів, утворення яких формує шар, як зображено на Рис. 1.4. Ліві вершини схеми зображеного нейрону служать для розподілу сигналів, що подаються на вхід. За цими вершинами не закріплено жодних обчислень, тому вони не вважаються шаром і позначені колами, щоб відрізняти їх від обчислювальних нейронів, що позначаються квадратами. Існує сполучення окремою вагою кожен елементу з множини входів X з кожним штучним нейроном. За нейроном закріплена робота подачі зваженої суми входів в мережу. З метою спільності штучні та біологічні мережі мають відсутні з'єднання. Існують з'єднання між виходами і входами елементів в самому шарі. Ваги представлені елементами матриці W. Матриця має n рядків і m стовпців, де n - число входів, а m - число нейронів. Наприклад, w_{12} - це вага, що зв'язує перший вхід з другим нейроном. Таким чином, обчислення вихідного вектора Y, зводиться до матричному множенню Y = XW [3].

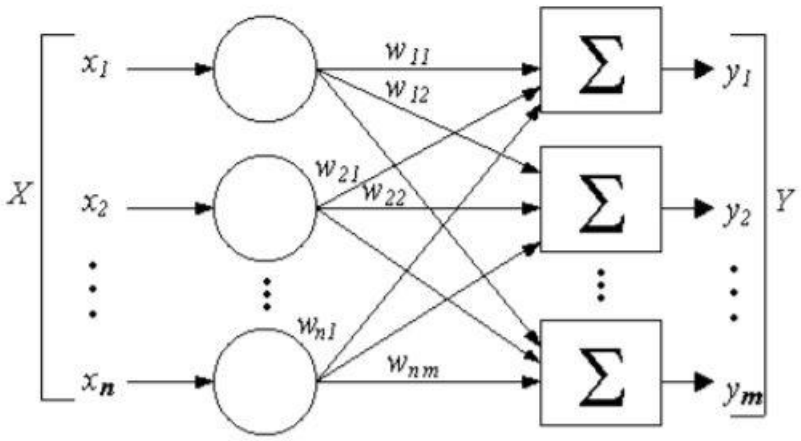


Рис 1.4 - Проста одношарова штучна нейронна мережа

1.3.3 Багатошарові штучні нейронні мережі

Відмінності обчислювальних процесів в нейронних мережах часто обумовлені способом взаємозв'язків нейронів. За сукупністю критеріїв на сьогоднішній день багатошарові архітектури можна розділити на статичні і динамічні. Кожен з класів архітектур нейронних мереж може включати безліч підкласів, реалізуючи різні підходи, нижче будуть наведені основні з них.

До статичних архітектур відносять мережі прямого поширення, в яких реалізована однонаправлений зв'язок між шарами, відсутні динамічні елементи і зворотній зв'язок, а вихід навченої нейромережі однозначно визначається входом і не залежить від попередніх станів мережі.

Статичні штучні нейронні мережі прямого поширення:

- Персептрон
- Нейронна мережа Кохонена
- Когнітрон та неокогнітрон
- Сучасна згорткова нейронна мережа

На противагу статичним архітектурам, існують динамічні архітектури штучних нейромереж, що реалізують рекурентну структуру з використанням зворотніх зв'язків, завдяки чому стан мережі в кожний момент часу залежить від попереднього стану. Рекурентні нейромережі як правило базуються на багатошаровому персептроні.

1.3.3.1 Персептрон, як модель розпізнавання

Елементарний персептрон організовується на основі сенсорних даних на вході - S-елементів, асоціативних елементів - A-елементів, і реагуючих елементів на виході - R-елементів. Набір S-елементів, пов'язаний з A-елементом утворює асоціацію, і A-елемент активізується після того як досягнуто певне число сигналів від S-елементів. A-елемент передає зважений сигнал на R-елемент обрахунку суми, і в залежності від

того, чи перевищує зважена сума деякий поріг, R-елемент видає результат роботи персептрону, показано на рисунку 1.5 [3].

Багатошаровий персептрон організовується з додатковими прихованими шарами А-елементів, розташованими між S-елементами і R-елементами. Принципова складність завдань, що вирішуються багатошаровим персептроном, є найвищою для класу персептронів. Навчання елементарного і багатошарового персептрона полягає в зміні вагових коефіцієнт зв’язків А – R. Персептрон здатний працювати в режимі розпізнавання або узагальнення.

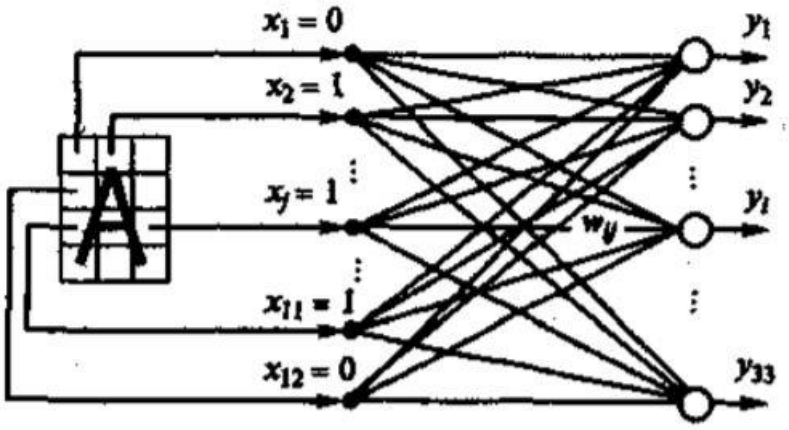


Рис 1.5 - Персептрон

В одношаровому персептроны вхідні елементи безпосередньо пов'язані з вихідними за допомогою системи ваг, зв'язки S - A організовані за принципом однозначної відповідності. Одношаровий персептрон є окремим випадком класичного елементарного персептрону, найпростішою мережею прямого поширення - лінійним класифікатором, і має безліч принципових обмежень, таких як неможливість реалізації функції XOR.

1.3.3.2 Когнітрон та неокогнітрон, як основа згорткових нейромереж

Когніторон був розроблений на основі будови біологічної зорової кори, має ієрархічну принципово багатошарову архітектуру. Нейрони між шарами когнітрону пов'язані тільки локально, і кожен шар реалізує різні

рівні узагальнення: вхідні шари сприймають прості образи, такі як лінії, великі однорідні ділянки, їх орієнтацію і локалізацію в просторі вхідних даних, в той час як глибокі шари сприймають складніші абстрактні структури, незалежні від локалізації та інших простих ознак образу.

Когнітрон [3] організовується з ієрархічно пов'язаних збудливих і гальмуючих шарів. Співвідношення збуджуючих і гальмуючих сигналів на вході нейрона визначає його стан збудження. Існують спрощені моделі когнітрону, що будуються з одновимірних шарів, але спочатку когнітрон конструювався як каскад двовимірних шарів. Пресинаптичний простір сигналів визначає виходи попереднього шару, постсинаптичний простір - входи наступного шару або площини. Нейрон когнітрону сприймає не весь постсинаптичний простір сигналів, а тільки його частину, реалізується принцип локальної зв'язності. Область пресинаптичного простору сигналів, що утворюють постсинаптичний простір сигналів, що впливають на стан даного нейрона, називається його локальним рецептивних полем. Рецептивні поля близьких один до одного постсинаптичних нейронів, звані зонами конкуренції, перекриваються, тому активність даного пресинаптичного нейрона позначається на все більше поширення області постсинаптичних нейронів наступних шарів ієрархії. Розміри зон конкуренції обумовлюють кількість сприймання в просторі області ознак.

Неокогнітрон є прямим розвитком ідеї, що лежать в основі когнітрону і точніше моделює структуру зорової кори головного мозку і є класифікатором, здатним до кращого розпізнавання об’єктів. Кожен шар неокогнітрона складається з площини простих S-нейронів і площини складних С-нейронів, що також організують локальну зв'язність.

Локальне рецептивне поле на площині S-нейронів наступного шару формується пресинаптичними сигналами площини С-нейронів попереднього шару. Локальні ознаки способу сприймаються S-нейронами, а спотворення локальних ознак компенсуються С-нейронами. В результаті

цього процесу кожен шар після вхідного має своїм входом все більш узагальнену картину, утворену С-нейронами попередніх шарів.

З кожним рівнем глибини первинні прості ознаки визначаються у більш складних з’єднаннях. Площину S-нейронів можна розглядати як один нейрон, ваги якого визначають ядро згортки, що застосовуються до попереднього шару у всіх можливих позиціях. Всі С-нейрони реагують на образ, відповідний ядру згортки, в їх рецептивному полі, тому він визначається інваріантно до його локалізації.

Сучасні глибокі згорткові нейронні мережі засновані на ідеях, що лежать в основі неокогнітрона, і сьогодні застосовуються для вирішення широкого кола завдань: від промислових, корпоративних та дослідницьких до повсякденно побутових, включаючи завдання, які вирішуються мобільними пристроями.

1.4 Основні принципи навчання нейронних мереж

Процес навчання штучний нейронних мереж розглядається як налаштування архітектури і ваг зв'язків між нейронами (параметрів) для ефективного виконання поставлених перед мережею завдань. Існує два великих класи навчання: клас детерменірованних методів і клас стохастичних методів.

До класу детерменірованих методи входять методи, в основі яких лежить ітеративна корекція параметрів мережі, в ході поточної ітерації яка грунтується на поточних параметрах. Основним детерменірованним методом і найпоширенішим методом навчання мереж сьогодні є метод зворотнього поширення помилки.

До класу стохастичних методів входять методи, що змінюють параметри мережі випадковим чином і зберігають тільки ті зміни параметрів, які призвели до поліпшення результатів роботи. Стохастичні алгоритми навчання реалізуються за допомогою порівняння помилок.

Навчання з учителем. Під час навчання мережі з учителем кожному прикладу з навчальної вибірки відповідає вектор, що характеризує однозначну правильну відповідь, що подається відразу на вихід мережі в обхід всієї її архітектури.

Після отримання власного результату мережі, алгоритм порівнює результуючий вектор з правильною відповіддю, на основі чого відбувається корекція подальших помилок. Правило корекції помилки, на якому заснований метод зворотного поширення помилки, є класичним прикладом навчання з учителем.

Навчання без вчителя у застосуванні до штучний нейромереж реалізується природним чином в процесі навчання, коли автоматичне налаштування параметрів мережею призводить до появи однакових результатів її функціонування при досить близьких вхідних значеннях, що на практиці можна порівняти зі зниженням розмірності даних в результаті ітераційного методу головних компонент. Правило Хеба, засноване на гіпотезі про посилення зв'язків між біологічними нейронами в разі їх одночасного збудження, і методи навчання при змаганні нейронів шляхом порівняння інтенсивності їх реакції є класичними прикладами методів навчання без учителя.

Метод зворотного поширення помилки (Backprop). Метод є класичним методом навчання з учителем, заснований на правилі корекції помилок і був розроблений як метод навчання багатошарового персептрона. Основна ідея полягає в поширенні сигналів помилки після її обчислення на виході мережі в напрямку, зворотньому прямому поширенні сигналів під час звичайного обчислювального процесу, від виходів мережі до її входів. При зворотньому проході синаптичні ваги налаштовуються з метою мінімізації помилки. Фактично, реалізується стохастичний градієнтний спуск, відбувається рух в багатовимірному просторі ваг до мінімуму помилки в сторону, протилежну градієнту. В процесі навчання циклічно знаходяться рішення на одно-критеріальні завдання оптимізації.

Для можливості реалізації методу передавальна функція нейронів повинна бути диференційована. Оскільки метод є модифікацією класичного методу градієнтного спуску, то може розглядатися як градієнтний спуск по поверхні помилки.

1.5 Висновок до першого розділу

Технологія створення машин, що мають здатність бачити лежить в основі комп'ютерного зору. Ця дисципліна відноситься до штучних систем, котрі можуть виконувати завдання розпізнавання. У вищенаведеному розділі наведена основна теорія розпінавання об'єктів, що лежить в основі класифікації.

Огляд методів вирішення задач комп'ютерного зору за групами обґрунтував переваги нейромережевого методу над усіма запронованими. Принцип роботи нейронів штучної мережі дозволяє виявити його активність, що визначається його параметрами - вагами і його функцією активації.

Труднощі при спотвореннях, такі як зміна образу у розмірах і поворот зображення, вирішуються при моделюванні неокогнітрона, що використовує якісно нову архітектуру. В основу архітектури неокогнітрона покладена організація зорової системи живих істот. Розглянуті основні принципи навчання нейронних мереж з їх особливостями показали, що ефективного навчання розпізнавання об'єктів можна досягти саме методом зворотнього поширення помилки.

РОЗДІЛ 2. ЗГОРТКОВА НЕЙРОННА МЕРЕЖА

Архітектура згорткових нейронних мереж призначена і використовується для ефективного розпізнавання зображень, де чергуються згорткові шари (англ. convolutions) з нелінійними функціями

активації (ReLU або гіперболічний тангенс tanh) і шари об'єднання або підвибірки (pooling layers).

На відміну від мережі прямого поширення, де кожен вхідний нейрон з'єднується з вихідним нейроном в наступному шарі, в згорткових мережах для отримання вихідних значень застосовуються згортки над кожним вхідним шаром. В операції згортки використовується матриця ваг невеликого розміру, яка проходить по всьому поточному шарі, формуючи після кожного зсуву сигнал активації для нейрона наступного шару з аналогічною позицією. Ця матриця називається ядром згортки; вона використовується для різних нейронів вихідного шару. Детально про згортковий шар описано у пункті 2.2.1 «Шари згортки» цієї роботи.

2.1 Гібридні архітектури нейронних мереж

Результат роботи нейронної мережі залежить від декількох факторів:

- якості і кількості навчальних даних;
- типу архітектури нейронної мережі;
- кількості параметрів обраної мережі.

Існує величезна кількість різновидів архітектур нейронних мереж, в рамках даної роботи розглянуті тільки деякі з них: персептрон (див. Розділ 1), мережі радиальньо-базисних функцій, рекурентні мережі та згорткові нейронні мережі.

Мережі радіально-базисних функцій (РБФ) - це нейронні мережі з радіально-базисної функцією активації $\rho(\gamma)$. Мережі радіально-базисних функцій містять тільки 3 шари: вхідний, прихований шар з РБФ як функцію актівації і вихідний шар з лінійною функцією активації [3].

Найчастіше в якості радіально-базисної функції ρ застосовується функція Гаусса (2.1):

$$\rho(\gamma) = \exp(-\gamma^2) \tag{2.1}$$

Рекурентні нейронні мережі. Нейронна мережа називається рекурентною, якщо в ній є зворотні зв'язки між нейронами, тобто вихідний сигнал нейрона передається на вхід іншого нейрона, розташованого в шарі з меншим індексом. Наявність зворотного зв'язку забезпечує отримання інформації не тільки з попереднього шару, але і від попередніх проходів навчання, що гарантує збереження тимчасової інформації. Це означає, що результат буде залежати ще й від послідовності передачі навчальних даних.

Згорткові нейронні мережі схожі на мережі прямого поширення типу персептрон: нейрони даного типу також мають ті, яких навчають ваги і параметр зсуву. Однак ЗНМ беруть до уваги те, що вхідні дані становлять собою зображення, володіючи даною інформацією, можна закодувати певні властивості в архітектурі мережі, що дозволяє значно зменшити кількість параметрів нейронної мережі.

Поруч із рекурентними нейронними мережами часто згадують згорткові нейронні мережі, тому варто розуміти їх відмінності.

Згорткова нейронна мережа:

- мережа має вхід фіксованого розміру і генерує виходи фіксованого розміру;
- відноситься до типу штучної нейронної мережі зі зворотним зв'язком - варіації багатошарових персептронів, які призначені для використання мінімальних обсягів попередньої обробки;
- використовують схему взаємодії між нейронами подібною до організації зорової кори тварин;
- підходять у застосуванні для обробки зображень і відео.

Рекурентна нейронна мережа:

- мережа може обробляти довільні розміри введення / виведення;

- на відміну від родинних нейронних мереж - може використовувати свою внутрішню пам'ять для обробки довільних послідовностей входів;
- рекурентні нейронні мережі використовують інформацію про часові ряди. Те, що я говорив в останній раз, вплине на те, що я буду говорити далі;
- підходять у застосуванні для аналізу тексту й мови.

Як правило будь-яка архітектура мережі має свої певні недоліки та переваги. Одним з методів вирішення проблеми недоліків конкретної архітектури – гібридизація мереж. Тут мається на увазі суміщення деяких архітектур в одну аби покращити ті чи інші якості. Також можна зустріти і гібридизацію з не нейромережевими моделями, тобто зі зовсім іншими методами розпізнавання образів. Головним принципом при побудові гібридної архітектури являється використання сильної сторони тієї чи іншої моделі та повного уникання слабких сторін.

Іншими словами гібридні нейронні мережі являють собою об'єднання декількох нейронних мереж для вирішення певних завдань. Це дозволяє виконати розбиття задачі з високою складністю на більш прості підзадачі, а архітектура нейронної мережі може бути оптимізована під конкретну задачу. Ще однією перевагою гібридних нейронних мереж є те, що при програмній реалізації швидкодія може варіюватися в широких межах залежно від обраної структури, що дозволяє застосовувати їх для вирішення завдань реального часу. Гібридні нейронні мережі об'єднують кілька різних за структурою і своїй природі «експертів», тому кожен з них може генерувати різні відповіді для одного і того ж вхідного набору даних. Для формування загальної відповіді гібридної нейронної системи найчастіше використовуються зважене підсумовування результатів або динамічний вибір однієї нейронної мережі, і використання отриманого нею результату в якості вихідного значення [3].

Відомі алгоритми, які дозволяють підібрати оптимальне число нейронних мереж, правильно їх організувати і навчити для вирішення завдань із заданою точністю.

2.2 Шаблон архітектури згорткової нейронної мережі

Як відомо, багатошарові штучні нейронні мережі отримують вхідні дані (наприклад один вектор), після чого трансформують інформацію, проводячи її через ряд прихованих шарів. Кожен прихований шар складається з безлічі нейронів, де кожний нейрон має сильний зв'язок з усіма нейронами в попередньому шарі і де нейрони в якості одного шару повністю незалежні один від одного і не мають спільних з'єднань. Останній повнозв'язну шар називається вихідним шаром, і в налаштуваннях класифікації він демонструє число класів [4].

Звичайні штучні нейронні мережі (Рис. 2.1) погано масштабуються у випадку з зображеннями великих розмірів. Так, в системі комп'ютерного зору CIFAR-10, картинки становить [32 × 32 × 3] (32 - ширина, 32 - висота, 3 - канали кольорів), тому один повністю підключений нейрон в першому прихованому шарі звичайної нейронної мережі має вагу 32 * 32 * 3 = 3 072. Здається, що це значення можна змінювати, але повнозв’язна структура не масштабується для великих зображень. Картинка обсягом більше, наприклад, [200 × 200 × 3], приведе до того, що повністю підключений нейрон буде важити 120 000. Крім того, потрібно залучити кілька таких нейронів, що призведе до додавання параметрів. Недоліком повнозв’язності буде величезна кількість параметрів, що швидко привести до перенавчання.

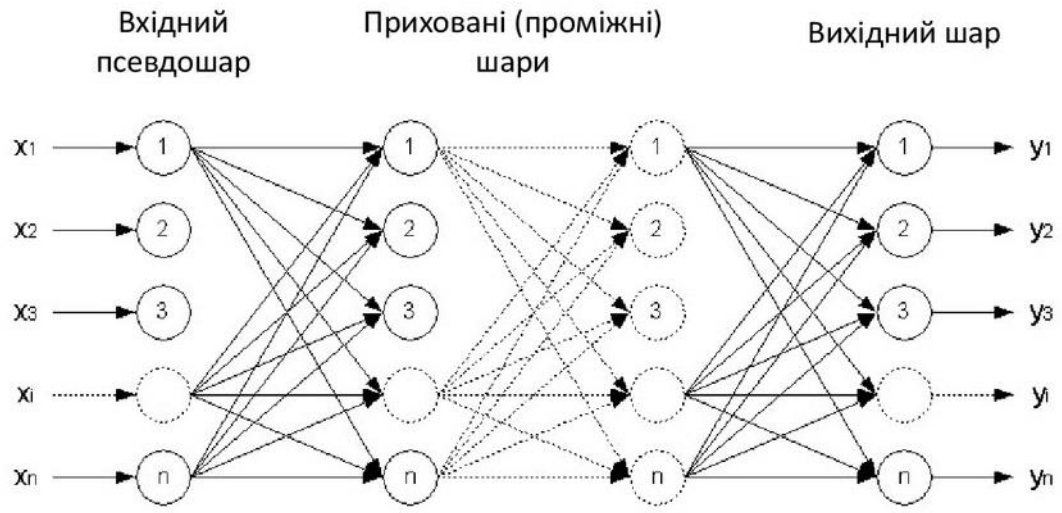


Рис. 2.1 - Багатошарова нейронна марежа

Згорткові нейронні мережі (Рис 2.2) користуються тим, що ввідні дані складаються з зображень, і вони обмежують побудову мережі більш розумним шляхом. На відміну від звичайної нейронної мережі, шари ЗНМ складаються з нейронів, розташованих в 3-х вимірах: ширині, висоті і глибині, тобто у вимірах, які формують об’єм. Наприклад, зображення на вході CIFAR-10 є вхідними активаційними об’ємами, а об’єм сформований вимірами 32 × 32 × 3. Як буде описано далі, нейрони будуть підключені тільки до невеликої області шару. Крім того, результуючий вихідний шар для даної системи комп'ютерного зору складе 1x1x10, оскільки до кінця побудови ЗНМ зображення перетвориться в єдиний вектор оцінок класу, розташованих по вимірюванню глибини [4].

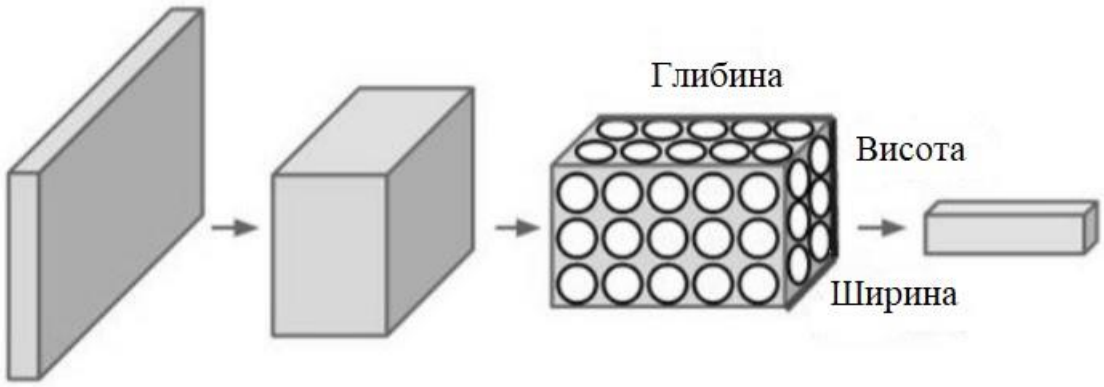


Рис. 2.2 Згорткова нейронна мережа

Отже основу згорткових нейромереж складають шари. Кожен шар характеризується простим набором задач: він перетворює вхідні дані у вигляді 3D-об'єму у вихідний 3D-об'єм з деякою диференційованою функцією, яка може мати або не мати параметри.

2.2.1 Шари згортки

Головним шаром згорткової нейронної мережі беззаперечно можна вважати шар згортки [4], робота якого є основою даної мережі. Параметри шару згортки складаються з набору фільтрів для навчання (також можна зустріти назву - ядра Кернела). Кожен фільтр має невеликі просторові габарити (ширину і висоту), але проходить по всій глибині вхідного об’єму. Наприклад, стандартний фільтр першого шару згорткової нейронної мережі може бути розміром [5x5x3]. Під час проходу вперед, виконується ніби ковзання кожного фільтру по ширині і висоті вхідних даних і обчислюється скалярний добуток між записами фільтра і входом у будь-яке положення. У міру проходження фільтра по ширині і висоті зображення, складається двомірну активаційна карта, яка надає відгук цього фільтра на кожній просторовій позиції (Рис. 2.3). Мережа навчає фільтри, що активуються при виявленні певної візуальної особливості. Це може бути грань деякої спрямованості, плямистість конкретного кольору на першому шарі або кільцеподібні візерунки на більш високих рівнях мережі. Після мережа працює з цілим набором фільтрів в кожному шарі згортки, і кожен з цих фільтрів буде формувати окрему двомірну карту ативації або карту ознак. Карти ознак складаються вздовж «глибини» вхідного об’єму і будуть формувати вихідний об’єм.

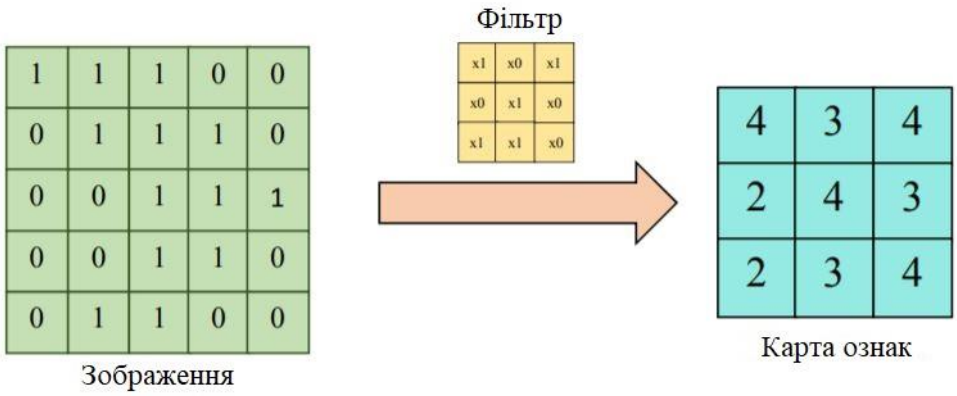


Рис 2.3 - Формування карти ознак. Операція згортки.

При роботі з вхідною інформацією високою розмірністю, установка зв'язку між нейронами і всіма нейронами з попереднім об’ємом є недоцільною. Замість цього потрібно застосовувати кожен нейрон тільки до локальної області вхідного об’єму. Просторова протяжність його зв'язку є гіперпараметром і називається рецептивним полем (полем сприйнятливості). Ідея рецептивного поля полягає в тому, щоб об'єднати нейрони прихованого шару лише з такими нейронами попереднього шару, які входять в деяку маленьку область n × n. Причому для кожного нейрона вибирається своя окрема область. Дана область і називається локальним рецептивним полем. Рецептивні поля виявляють елементарні візуальні ознаки такі, як кут або край, а їх комбінації дають можливість отримати складніші ознаки [4].

Легко здогадатися, що площа рецептивного поля дорівнює площі фільтра. Просторова протяжність уздовж осі глибини завжди еквівалента глибині вхідного об’єму. Слід ще раз підкреслити асиметрію в тому, як ми розглядаємо просторові виміри (ширину і висоту), а як - глибину: з’єднання є локальними по ширині і висоті, але завжди проходять через всю глибину вхідного об’єму.

Приклад вхідного об’єму (прямокутник, параметри [32 × 32 × 3], зображення CIFAR-10) і можливий обсяг нейронів в першому шарі згорткової нейронної мережі показано на Рис 2.4. Кожен нейрон в згортковому шарі з'єднаний тільки з локальною вхідною областю, де

просторова протяжність визначається шириною і висотою, але нейрон проходить через всю глибину, охоплює всі кольорові канали.

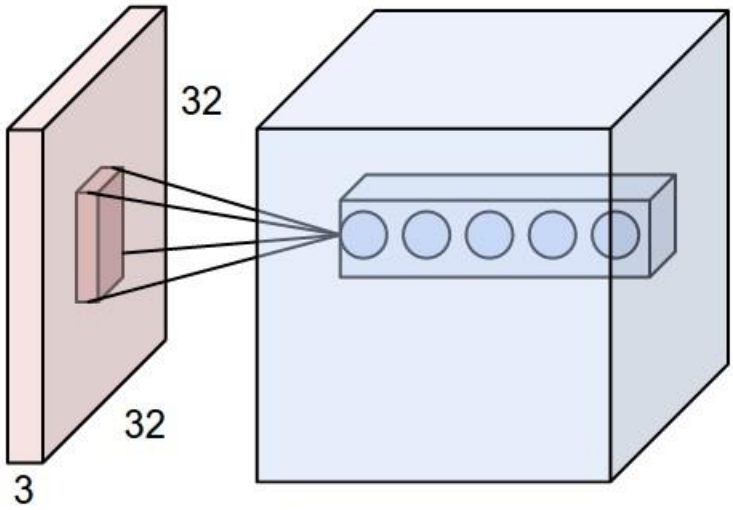


Рис 2.4 - З'єднання нейронів шару згортки з локальним вхідним об’ємом (частиною вхідного зображення)

Нейрони згорткових нейромереж залишаються незмінними: вони як і звичайні нейрони мереж обчислюють скалярний добутог їх ваги і вхідних даних з подальшою нелінійністю, але їх підключення обмежується локальними просторовими вимірами.

2.2.2 Параметри та гіперпараметри мережі

Існують 3 гіперпараметри мережі, які контролюють розмір вихідної інформації: глибина, крок і доповнення нулями [5].

Першим гіперпараметром вихідного об’єму є глибина. Гіперпараметр еквівалентний числу фільтрів, які використовуються; кожен з фільтрів навчається пошуку різних ознак на вході. Наприклад, перший згортковий шар в якості вхідної інформації бере необроблене зображення, де різні нейрони, розташовані уздовж глибини, можуть активуватися в разі наявності, наприклад, згустків певного кольору. Позначимо безліч нейронів, які «дивляться» на одну ділянку вхідного об’єму як стовпець глибини.

Другим важливим гіперпараметром є крок, з яким фільтр виконує прохід. Розмір кроку (англ. stride): Розмір кроку визначає величину зсуву фільтру на кожному кроці. Чим більше крок, тим менше фільтр застосовується і тим менше розмір вихідної матриці. Зазвичай використовується крок, що дорівнює одиниці, проте більший крок може дозволити побудувати модель, поведінка якої буде нагадувати рекурсивну нейронну мережу (тобто згорткова мережа з великим кроком буде виглядати як дерево); це дозволить формувати менші вихідні об’єми просторових вимірів.

При проході фільтру виникають ситуації, коли зручно заповнювати кордон вхідного зображення нулями. Розмір цього доповнення нулями і є гіперпараметром. Ключовою особливістю доповнення нулями є те, що воно дозволяє контролювати просторовий розмір вихідних об’ємів (найчастіше ми використовуємо дану властивість, щоб зберегти просторовий розмір вхідної інформації з метою збереження вхідних та вихідних значень висоти і ширини однаковими). Без використання доповнення нулями отримуємо вузьку згортку.

Просторовий розмір вихідного об’єму можна обчислити як функцію від вхідного об’єму, розміру рецептивного поля нейронів згорткового шару, кроку, з яким вони переміщуються, і кількості заповнення нулями на кордоні вхідної картинки. Параметри згорткового шару [5]:

K - кількість фільтрів;

F - розмір фільтра;

S - крок;

P - кількість заповнень нулями.

Розмір (об’єм) вихідних даних W2 × H2 × D2 обчислюється таким чином (2.2, 2.3,2.4) [5]:

$$W_2 = {(W_1 - F + 2 \times P)}/{S} + 1, \quad (2.2)$$

$$H_2 = {(H_1 - F + 2 \times P)}/{S} + 1, \quad (2.3)$$

$$D_2 = K, \quad (2.4)$$

де W1 - ширина, H1 - висота, D1 - глибина вхідних даних.

Спільне використання параметрів використовується в шарах згортки з метою контролю кількості параметрів.

Ще одним параметром виключно згорткових мереж є шари об’єднання. Шари об'єднання допомагають скоротити розмірність вихідної інформації, при цьому зберігаючи саму помітну інформацію. Наприклад, якщо фільтр визначає, чи міститься явна ознака. Якщо десь присутня явна ознака, то результат застосування фільтра до цієї області зображення дасть велике значення і невелике для інших частин зображення.

Канали кольору (англ. channels): канали - це різні "погляди" на вхідні дані. Наприклад, в розпізнаванні зображень зазвичай три каналу - RGB.

2.2.3 Максимальні та усереднені шари підвибірки

В архітектурі згорткових нейронних мереж застосовують шар підвибірки між послідовностями згорткових шарів. Його функція полягає в поступовому зменшенні просторових габаритів зображення з метою зменшення кількості параметрів мережі та обчислень загалом, а також контролю перенавчання. Шар підвибірки працює незалежно від зрізу глибини вхідних даних і масштабує обсяг просторово, використовуючи функцію максимуму. Найчастіше використовується шар з фільтрами розміру 2 × 2 з кроком 2; подібний шар знижує дискретизацію кожного зрізу глибини входу в 2 рази як по ширині, так і по висоті. Кожна операція максимуму в цьому випадку буде вибирати максимальне значення з чотирьох чисел. Розмір по глибині при цьому залишається незмінним. У

загальному випадку шар підвибірки отримує на вхід об’єм даних розміром W2 × H2 × D2 та потребує два гіперпараметри: протяжність та крок.

У кожному шарі згортки для кожного каналу свій фільтр, ядро згортки якого обробляє попередній шар за фрагментами. Результат застосування різних фільтрів об'єднується. Так виходять шари об'єднання. Операція субдискредитації виконує зменшення розмірності сформованих карт ознак. У даній архітектурі мережі вважається, що інформація про факт наявності шуканого ознаки важливіше точного знання його координат, тому з кількох сусідніх нейронів карти ознак вибирається максимальний і приймається за один нейрон ущільненої карти ознак меншої розмірності. Шляхом цієї операції, крім прискорення подальших обчислень, мережа стає гнучною до масштабу вхідного зображення.

На додаток до підвибірки максимуму, дані шари можуть виконувати інші функції, представлені у таблиці 2.1, наприклад, підвибірку усереднення або навіть L2-нормовану підвибірку. Історично підвибірка усереднення використовувалася досить часто, але останнім часом вона відійшла на другий план у порівнянні з максімізаціонной підвибіркою, яка на практиці працює краще.

Таблиця 2.1 - Типи функцій активації *a* шарів підвибірки згорткових нейронних мереж множини *A*

Назва функції	Визначення функції
Максимимальна підвибірка	$\max\{a \in A\}$
Усереднена підвибірка	$\frac{1}{\lvert A\rvert}\sum_{a \in A} a$
l_2 підвибірка	$\sqrt{\sum_{a \in A} a^2}$
Стохастична підвибірка	Ймовірність $p_i = \frac{a_i}{\sum_{a_j \in A} a_j}$

На Рис 2.5 зображено операцію підвибірки із застосуванням максимізаційної та усередненої функції.

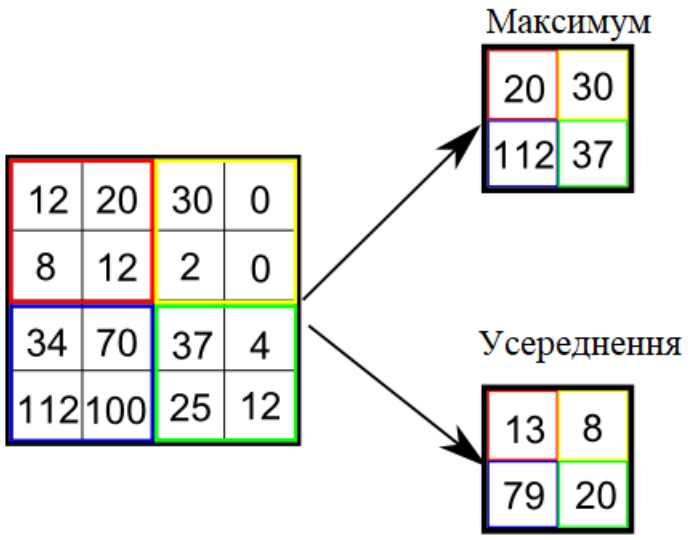


Рис 2.5 - Операція підвибірки

2.2.4 Техніка dropout

Одна з головних проблем використання глибоких нейронних мереж, чи то багатошарових звичайних, чи то згорткових – це перенавчання (англ. overfitting) [6]. Перенавчання проявляється у тому, що створена модель починає пояснювати тільки приклади з навчальної вибірки, адаптуючись до навчальних прикладів, замість того щоб вчитися класифікувати приклади, які не брали участі в навчанні (втрачаючи здатність до узагальнення). За останні роки було запропоновано безліч рішень проблеми перенавчання, але одне з них перевершило інші, завдяки своїй простоті і досягненням у практичних результатах.

Техніку Dropout можна описати як метод, який використовується для запобігання перенавчання та певної адаптації нейронів до вхідних даних шляхом встановлення вихідного сигналу будь-якого нейрона у значення нуля з ймовірністю p. Виключені нейрони не вносять свій внесок в процес навчання ні на одному з етапів алгоритму зворотнього поширення помилки (англ. backpropagation), який часто застосовується до навчань мереж; тому

виключення хоча б одного з нейронів рівносильно навчанню ніби нової нейронної мережі.

Графічне представлення техніки Dropout (Рис 2.6), взяте зі статті [6], в якій техніка була вперше представлена.

Головна ідея техніки Dropout - замість навчання однієї нейромережі навчити ансамбль декількох нейромереж, а потім усереднити отримані результати.

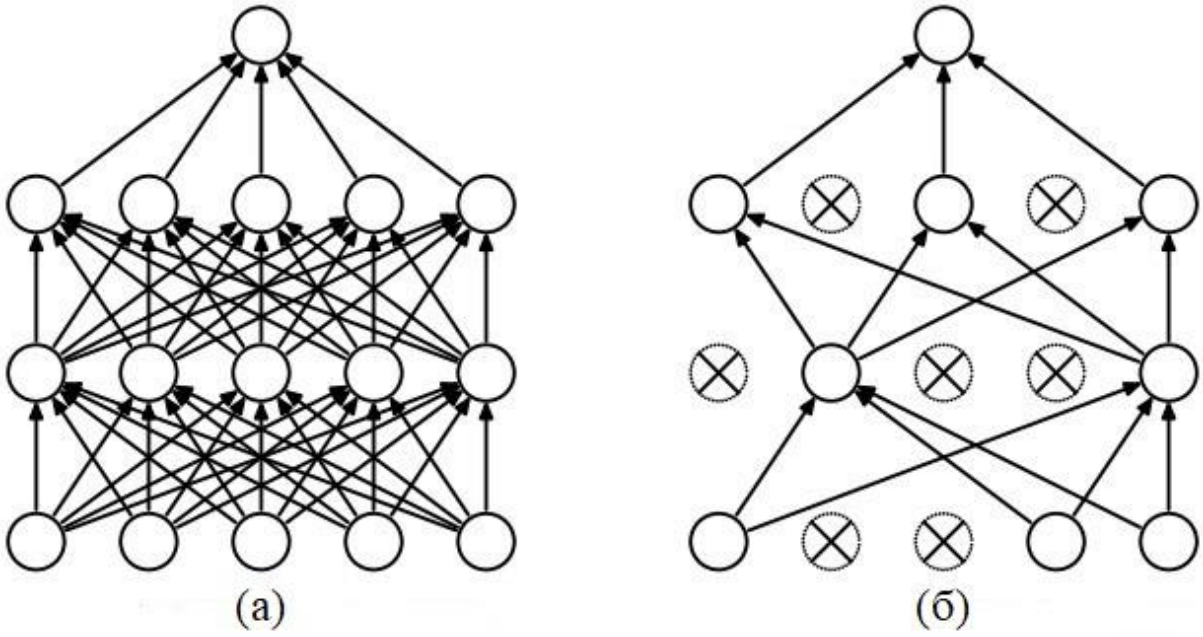


Рис. 2.6 - Техніка Dropout. Зліва (а) - нейронна мережа до того, до застосування Dropout, а праворуч (б) – та сама мережа після застосування техніки Dropout. Мережа, зображена зліва, використовується при тестуванні, після навчання параметрами.

За словами авторів техніки у стандартній нейронної мережі похідна, отримана кожним параметром, повідомляє цьому ж параметру, як він повинен змінитися, щоб, з огляду на діяльність інших блоків, мінімізувати функцію втрат. Тому блоки можуть змінюватися, виправляючи при цьому помилки інших блоків. Це може привести до надмірної спільної адаптації (англ. co-adaptation), що, призводить до перенавчання, оскільки ці спільні адаптації неможливо узагальнити для даних, які не брали участі в навчанні. Автори техніки зауважили, що Dropout запобігає спільній

адаптації для кожного прихованого блоку, утворюючи невеликий відсоток надійності існування інших прихованих блоків. Тому прихований блок не може покладатися на інші блоки в виправленні власних помилок.

Імовірність виключення кожного нейрона однакова. Застосування техніки Dropout на етапі навчання можна представити як змінену функцію активації (2.5).

$$out = D \cdot a(h), \tag{2.5}$$

тут $a(h)$ - функція активації, а символ $\cdot$ добуток Адамара [7].

2.2.5 Техніка batch-normalization

У багатошарових мережах кожен шар приймає дані з попереднього шару. Дані можуть бути представлені у вигляді будь-якого тензору, координати якого попередньо визначені. Це призводить до того, що параметри в подальших шарах мережі гіршають. Стає необхідним вибрати дуже низьку швидкість навчання, щоб відрегулювати той факт, що функції введення можуть різко змінюватися з часом.

Для шару, що приймає дані, будо б дуже зручним отримувати їх у вигляді тензорів саме з координатами із фіксованого розподілу, одного для всіх шарів, тоді навчання перетворенню до параметрів розподілу стає непотрібним. Стає доречним між усіма шарами встановити обрахунки, які б оптимально нормалізували виходи попереднього шару, і знижували б тиск на наступний шар, що значно покращило б його роботу в мережі. Отже спосіб вирішення цієї проблеми полягає в нормалізації міні-партій (англ. batches).

Популярними на сьогоднішній день є дві техніки нормалізації : local response normalization та batch-normalization. Обидві нормалізації використовуються для запобігання зниженню швидкості навчання параметрів мережі. Локальна нормалізація (англ. local response

normalization) відповідності є самостійним шаром мережі. Дана операція нормалізує кожне значення вхідної матриці по каналу.

Пакетна нормалізація (англ. batch-normalization), вперше введена в [8], застосовує стандартну нормалізацію до отриманих значень, а потім лінійно трансформує їх (2.6):

$$y = \frac{x - \mu}{\sqrt{\sigma^2 - \epsilon}} \cdot \gamma + \beta, \tag{2.6}$$

тут ϵ, γ та β є параметрами, що підлягають налаштованню. μ та σ, середнє значення та дисперсія залежать від того на якому етапі знаходиться мережа. Якщо відбувається навчання мережі, то ці значення беруться від значень, отриманих тільки в поточний момент. Під час тестування мережі значення беруться з усієї зібраної статистики.

Питання коли саме застосовувати нормалізацію (шар нормалізації) й досі залишаться відкритим. Для виконання dropout не має значення чи нормалізація застосовується до або після функції активації. З огляду на це, можливі варіанти застосування:

- Шар згортки / Класифікатор (повнозв'язний шар) → BN (нормалізація) → функція активації → Dropout → . . .
- Шар згортки / Класифікатор (повнозв'язний шар) → функці активації → BN → Dropout → . . .
- Шар згортки / Класифікатор (повнозв'язний шар) → функція активації → Dropout → BN → . . .
- Шар згортки / Класифікатор (повнозв'язний шар) → Dropout → BN → функція активації → . . .

Для побудови майбутньої мережі було вирішено виконувати BN одразу після застосування операції згортки і до застосування операції нелінійності (функції активації).

2.2.6 Класифікатор

Як і в звичайних нейронних мережах, в згорткових мережах в повнозв'язну шари (класифікатор) [9] нейрони мають зв’язок з усіма функціями активації з попереднього шаріу. Активації ж шарів класифікації можуть бути обчислені за допомогою множення матриць, що супроводжується зміщенням.

Відмінність між шаром класифікації і шаром згортки полягає у тому, що нейрони шару згортки з'єднані тільки з локальною областю на вході, і що нейрони цього шару можуть спільно використовувати параметри. Однак нейрони в обох шарах, незважаючи на свої особливості, підраховують скалярний добуток, тому їх функціональна форма ідентична. Більш того, можна виконати конвертацію між повнозв'язним і згортковим шарами.

Класифікатор повинен запам'ятовувати всі навчальні дані та зберігати їх для подальших порівнянь із даними з тестового запуску. Це дуже ресурсозатратно, оскільки набори даних можуть бути розміром у гігабайтах.

Лінійний класифікатор (Рис 2.7) дає оцінку класу як зважену суму всіх значень пікселів у трьох його кольорових каналах. Залежно від того, які значення встановлено для цих ваг, функція класифікатору має здатність позитивно або негативно оцінювати (залежно від знаку кожної ваги) певні кольори у певних положеннях зображення. Наприклад, людина стверджуватиме, що клас "корабель" може бути більш імовірним, якщо з боків на зображенні є багато синього кольору (що може відповідати воді).

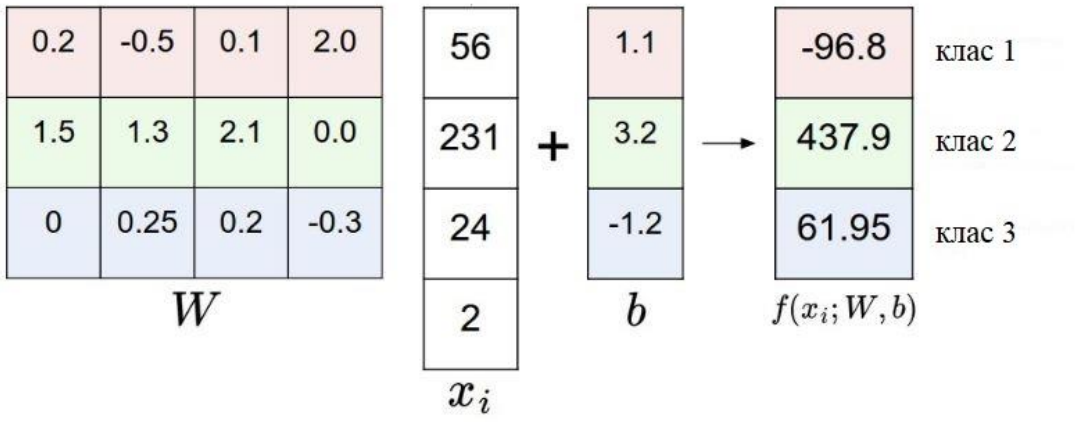


Рис 2.7 - Лінійний класифікатор

Ще один спосіб організації класифікатору – шаблонізація. Особливість способу стосується ваг W і полягає в тому, що кожен рядок W відповідає шаблону (або іноді також називається прототипом) для одного з класів. Оцінка кожного класу для зображення отримується шляхом порівняння кожного шаблону, один за одним, із зображенням за допомогою внутрішньої ознаки (або точки-ознаки), щоб знайти те, що "підходить" найкраще. За допомогою такого порівняння лінійний класифікатор знаходить відповідність шаблону.

2.2.7 Softmax

Популярним вибором класифікатору є класифікатор Softmax, який має зовсім іншу функцію втрат та в основу якого покладено нормовану експоненційну функцію. Цей класифікатор по своїй суті є узагальненим класифікатором двійкової логістичної регресії. На відміну від методу опорних векторів, який обчислює результати вектору $f(x_i, W)$ як оцінки для кожного класу, класифікатор Softmax керований трохи більш інтуїтивним підходом, а також має імовірнісну інтерпретацію. У класифікаторі Softmax функція втрат (2.7) відображення $f(x_i, W) = Wx_i$залишається незмінною, підключається функція крос-ентропії (2.8) [10].

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right), \tag{2.7}$$

$$L(X,Y) = -\frac{1}{n}\sum_{n}^{i=1} y^i \ln a\left(x^i\right) + \left(1-y^i\right)\ln\left(1-a\left(x^i\right)\right), \tag{2.8}$$

де X={x(1),…,x(n)} — це набір вхідних прикладів для навчання у вхідному наборі. Y={y(1),…,y(n)} — це відповідний набір міток для вхідного набору. Функція a(x) — вихідні значення нейронної мережі з вхідним значенням x.

2.3 Висновок до другого розділу

Згорткові нейронні мережі знайшли своє застосування можна при обробці зображень. Якщо уявити, що зображення - двовимірні функції, то різні перетворення зображень - це операція згортка функції зображення з локальною функцією, яка називається ядром згортки. Операція згортки чергується з операцією підвибірки.

Підвибірка застосовується для зменшення загального розміру зображення і збільшення ступеня інваріантності застосовуваних до нього фільтрів. При розгляді архітектури мережі спираються на факт, що наявність якої-небудь ознаки на зображенні набагато важливіше, ніж точне знання її координат. Таким чином, суттю підвибірки є вибір максимального нейрона з декількох сусідніх. Потім, цей нейрон приймаємо за елемент наступної, але вже зменшеної карти ознак. Дана операція допомагає збільшити інваріантність до масштабу вхідного зображення.

Після проходження декількох шарів карта ознак вироджується в вектор або скаляр, і таких карт ознак стає сотні. На виході згорткових

шарів мережі додатково встановлюють кілька шарів повнозв’язної нейронної мережі, що виконують роботу класифікатору.

У розділі 2 було розглянуто загальну топологію згорткових нейронних мереж, також описано усі складові їх архітектури та значення параметрів і гіперпараметрів згорткових мереж. Також було описано суть техніки dropout, що має широке застосування при запобіганні перенавчань мереж.

У розділі пояснено суть нормалізації вхідних даних для шарів та важливість запобігати зниженню швидкості навчання мережі та кількості її параметрів.

РОЗДІЛ 3. МЕТОДИКА НАВЧАННЯ ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ

3.1 Методи навчання нейронних мереж

Здатність до навчання виступає як фундаментальна особливість людського мозку може розвиватися. Якщо говорити про навчання в контексті нейронних мереж, то тут варто розглядати налаштування архітектури мережі, ваги нейронів мережі, їх зв’язків. Все це має свій вплив на ефективне виконання поставленої задачі перед мережею. Зазвичай нейронна мережа повинна налаштувати свої параметри (такі як вага кожного нейрону) за поданими навчальними прикладами. Властивість мереж навчатися за поданими прикладами робить їх більш досконалими у порівнянні з системами, що працюють за прописаними заздалегідь правилами. Серед усіх чинних методів навчання мереж можна виділити два класи: детермінований та стохастичний.

Детермінований клас методів інтерактивно корегує параметри мережі, спираються на стан поточних параметрів, величини входів, фактичних та бажаних виходів. Ілюстрацією такого методу являється метод зворотнього поширення помилки.

Стохастичний клас методів навчання мереж змінює параметри мережі випадковим чином. При цьому збереженими залишаються лише ті зміни, що призвели до покращення. В якості прикладу стохастичного метожу можна привести такий алгоритм [11]:

1. Обрати параметри мережі випадковим чином та підкорегувати їх на невелику випадкову величину. Обрахувати отримані значення виходів за запропонованими значеннями входів.
2. Порівняти ці виходи з бажаним результатом та обчислити різницю значень. Ця різниця називається помилкою. Ціль навчання поляє у тому, щоб мінімізувати значення помилки.
3. Значеннння помилки зменшено, корегування збережно. У негативному випадку значення корегування відкидається і вибирається заново.

Кроки 2 та 3 повторюють поки мережу не буде навчено.

Необхідно зауважиит, що стохастичні методи навчання можуть потрапити до «пастки» локального мінімуму (Рис 3.1).

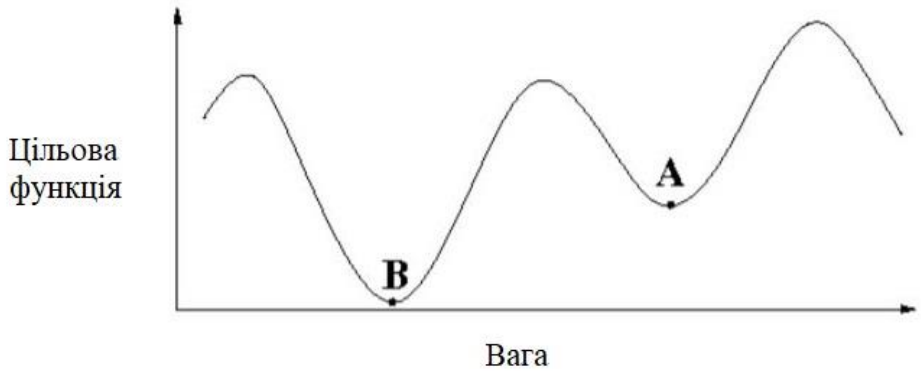


Рис 3.1 - Проблема локального мінімуму

Розглянемо зображення рисунку 3.1. Нехай початкове значення помилки рівно чи близьке до значення у точці А. Якщо випадкові кроки

корегування дуже малі, то будь-які відхилення від точки А збільшать помилку та не будуть братися до уваги. Таким чином найменше значення у точці В ніколи не буде знайдено. При дуже великих випадкових кроках корегування параметрів мережі помилка буде піддаватися зміні так різко, що ніколи не закріпиться за одним із мінімумів.

Щоб уникати проблеми локального мінімуму [11] можна поступово зменшити випадковий середній розмір кроків корегування. Коли середній розмір кроків завеликий, значення помилки буде приймати усі значення з рівною імовірністю. При плавному зменшенні розміру кроків буде дотримано умови при якому значення помилки буде на деякий час «застрягати» в точці В. Коли значення кроку буде ще більше зменшено, значення помилки буде «зупинятися» на короткий проміжок часу і в точці А і в точці В. При зменшенні кроку безперервно то у кінці буде досягнуто величину кроку, що буде достатньою для подолання локального мінімуму А, але не локального мінімуму В.

Математичне представлення навчання мереж зводиться до наступного вигляду:

Нехай ми маємо рівняння $Y = F(X)$ яке вирішує задачу і набір вхідних-вихідних даних (X^1, Y^1), $(X^2, Y^2) \dots (X^n, Y^n)$. При навчанні мережі потрібно знайти таку функцію, що буде близькою до функції з помилкою. Тобто знаючи всі ці параметри завдання зводиться до задачі багатовимірної оптимізації функції. А для її рішення застосовуються такі алгоритми:

1. Алгоритми локальної оптимізації з обчисленням часткових похідних першого порядку [11]:
    - градієнтний алгоритм (метод найшвидшого спуску);
    - методи з одновимірноїю і двовимірною оптимізацією цільової функції в напрямку антіградієнта;

- методи, що враховують напрямок антіградієнта на декількох кроках алгоритму.

2. Алгоритми локальної оптимізації з обчисленням часткових похідних першого і другого порядку:
   - метод Ньоютона;
   - методи оптимізації з розрідженими матрицями Гессе;
   - метод Левенберга-Марквардта та ін.
3. Стохастичні алгоритми оптимізації:
   - пошук у випадковому напрямку;
   - метод Монте-Карло та ін.
4. Алгоритми глобальної оптимізації (задачі глобальної оптимізації вирішуються за допомогою перебору значень змінних, від яких залежить цільова функція).

3.1.1 Навчання з учителем

Визначення методу навчання з учителем можна сформулювати так: задача машинного навчання у реалізації функції виведення на підставі відомих навчальних даних.

Структура навчальних даних складається з набору прикладів для виконання навчання. В даному методі навчання кожен приклад представляє собою пару, що складається з вхідного об'єкта (зазвичай вектора) і бажаного вихідного значення (контрольний сигнал). Необхідно прорахувати значення функції на момент входу даних, потім порівняти значені із значенням очікуваного резузльту, вирахувати помилку і скорегувати параметри (вага нейронів) мережі на цю помилку.

Алгоритм навчання з учителем [12]:

1. Взяті дані для навчання мережі розділити на дві частини. Першою половиною буде навчальна вибірка, другою - тестова вибірка. (можливо розділити дані 70/30).

2. Прорахувати значення функції для навчальної вибірки та знайти значення функції помилки.
3. Провести корегування ваг синапсів у мережі.
4. Повторювати пункти 2 і 3 до тих пір поки значення функції помилки не буде мінімальним. Повторювати ці дії можна як для кожного екземпляра навчальної вибірки так і для всієї вибірки в цілому. У першому випадку навчання буде повільніше, але з більшою точність. У другому випадку, показник точності занепадає, але навчання буде відбуватися швидше.
5. Перевірити всі значення за тестовою вибіркою, щоб зрозуміти наскільки добре система змогла узагальнити дані (навчитися).

Якщо нейронна мережа навчається з використанням заздалегідь відомих правильних відповідей, то такий алгоритм навчання називається - навчання з учителем. Необхідно відзначити, що при навчанні з учителем потрібна велика вибірка, щоб в достатній мірі сформувати робочу і гнучку нейронну мережу.

3.1.2 Навчання без учителя

Метод навчання без учителя, також називають методом неконтрольованого навчання, який має змогу знайти структуру або відносини між різними входами. Головна відмінність від методу навчання "з вчителем", це те що в наявності є тільки вхідні дані. Алгоритм навчання без вчителя застосовується по суті тоді, коли відомі тільки вхідні дані. На основі їх мережа вчиться видавати найкращі вихідні результати. Поняттям «найкращих результатів» визначається самим алгоритмом навчання. Зазвичай алгоритм підлаштовує параметри так, щоб мережа видавала однакові результати для достатньо близьких вхідних значень [12].

Найбільш важливим неконтрольованим навчанням є кластеризація, яка створює різні кластери введення і зможе вносити будь-які нові дані до

відповідного кластеру. Крім кластеризації є інші методи неконтрольованого навчання: виявлення аномалій, навчання і навчання в теорії Хебба. Приховані змінні моделі, такі як алгоритм максимізації очікувань, метод моментів і методи поділу сліпих сигналів.

Хоче даний метод і часто використовується у прикладних задачах, він часто піддається критиці вчених через свою біологічну схильність. Важко уявити, що у мозку присутній механізм порівняння отриманих результатів з бажаними.

3.1.3 Навчання засноване на корекції помилок

Цей метод навчання перцептрона, запропонований Френком Розенблатом ще у далекому 1957 році [12], являє собою такий метод навчання, при якому вага зв'язку зміні не піддається, поки перциптрон видає правильний рузьтат. Параметри впроваджують зміни у випадку неспівпадання вихідних значень з вхідними. Для обчислення величини корекції використовується різниця між реальним і бажаним значенням виходу мережі.

Дана модель використовує навчання з учителем, тобто навчальна множина складається із сукупності вхідних векторів для кожного з яких вказано вихідний вектор. Не зважаючи на деякі обмеження модель стала основою для багатьох сучасних більш складних алгоритмів навчання.

3.1.4 Конкурентне навчання

Метод конкурентного навчання полягає у змаганні кожного з вихідних нейронів мережі за активацію [12]. З цього можна зробити висновок що з усіх вихідних нейронів до роботи приступає лише нейрон із самим великим виходом. Такий алгоритм має нагадування біологічних нейронних мереж. За допомогою методу конкуренції можна класифікувати вихідні дані де схожі приклади групуються до одного класу і подаються як один зразковий елемент. При цьому кожни нейрон із множини нейронів

несе відповідаль лише за один клас. Загальне число класів з якими працює мережа дорівнюватиме число вихідних нейронів.

Конкурентне навчання дуже зручно використовуваних в задачах класифікації вхідних образів. У цьому випадку кожен нейрон вихідного шару відповідає за один образ.

3.1.5 Навчання Хебба

Навчання Хебба засновано на фізіологічних і психологічних дослідженнях.

Алгоритм навчання можна представити у вигляді правил з 2 частин [12]. Де перша частина звучить так: якщо два нейрона по обидві сторони синапсу активізуються одночасно, то міцність цього з'єднання зростає. І друга частина - якщо два нейрона по обидві сторони синапсу активізуються асинхронно, то такий синапс послаблюється або взагалі відмирає.

3.2 Алгоритм зворотнього поширення помилки

Згорткові мережі за своїм типом поширення активаційного сигналу між нейронами є прямими, тому застосування алгоритму зворотного поширення помилки є цілком доречним до таких мереж. Даний алгоритм ще називають алгоритмом градієнтного спуску через те, що стратегія підбору такого важливого параметру, як вага для кожного нейрону багатошарової мережі базується на градієтному методі.

Безперервна цільова функція як показник успішності мережі в загальному випадку визначається як квадратична різниця суми між фактичним результатом і очікуваним вихідним значенням.

Алгоритм зворотного поширення помилки при навчанні використовує два поширення мережею – пряме та зворотнє [4]. На самому початку алгоритму відбувається саме прямий прохід де вхідні дані у вигляді вектору реалізують поширення між шарами, від початкових до

останніх. В результаті прямого проходу генерується набір вихідних сигналів, який і визначає реакцією мережі на вхідні дані. Під час прямого проходу усі синаптичні ваги мережі є фіксованими. Другим етапом алгоритму є зворотній прохід де параметри (усі синаптичні ваги) налаштовуються відповідно за правилами корекції помилок. Суть згаданого правила така: від очікуваних вихідних значень віднімається отримане (результуюче) значення фактичного виходу і в результаті такої операції формується сигнал помилки. Сигнал помилки поширюється, як відлуння, в протилежному синаптичних зв'язків, тому алгоритм і отримав таку назву. А синаптичні ваги у свою чергу підлаштовуються з метою максимального наближення вихідного сигналу мережі очікуваного результату.

Недоліком алгоритму зворотного поширення помилки є те, що він не дозволяє в загальному випадку досягти глобального мінімуму. Тому і постають основні труднощі навчання нейронних мереж, що полягають в методах виходу з локальних мінімумів. Головними недоліками градієнтного спуску або алгоритму алгоритму зворотнього поширення помилки при навчанні мережі є [4]:

- «Параліч» мережі. Значення ваг мережі в результаті корекції можуть досягти дуже великих величин. Оскільки помилка, що посилається назад в процесі навчання, пропорційна похідній стискаючій функції, процес навчання може майже зупинитися. Цьому можна запобігти, зменшуючи крок, але процес навчання при цьому буде відбуватися довше.
- Розмір кроку. Якщо значення кроку не змінюється і воно досить мале, то метод сходитися занадто повільно. Якщо ж крок занадто великий, то може виникнути параліч мережі. Необхідно змінювати значення кроку: збільшувати до тих пір,

поки не припиниться поліпшення оцінки в напрямок антиградієнта і зменшувати, якщо оцінка не поліпшується.

3.3 Показники нейронної мережі: точність, втрата та якість

Епоха - прямий і зворотний прохід по всім тренувальним прикладам.

Розмір серії (batch) - кількість тренувальних прикладів для однієї ітерації прямого і зворотного проходів. Кількість ітерацій - кількість проходів: кожен прохід використовує приклади (batch). Один прохід дорівнює прямому проходу та зворотньому проході. Тобто маючи 1000 прикладів, batch = 500, нам буде потрібно зробити дві ітерації, щоб завершити одну епоху.

З математичної точки зору, навчання нейронних мереж - багатопараметрична задача нелінійної оптимізації.

Процес навчання мереж описують за допомогою графіку кривої, де горизонтальна вісь відображає кількість навчальних зразків, що отримує мережі, а вертикальна вісь відображає значення помилки, що отримано при класифікації вхідних даних.

На рисунку 3.1 зображено два криві: криву значень помилок на навчальному наборі (величина якої задана горизонтально) та криву значень помилок на тестовому або перевірочному наборі (яка має фіксований розмір). Чим більше даних використовується для навчання, тим більше помилок видасть архітектура мережі, що відповідатимуть навчальним даним. У той же час навчальні дані стають більш схожими на справжній розподіл даних, які слід зафіксувати за навчальними даними. У певний момент часу помилка на навчальному та тестовому наборах повинна бути приблизно однаковою [12].

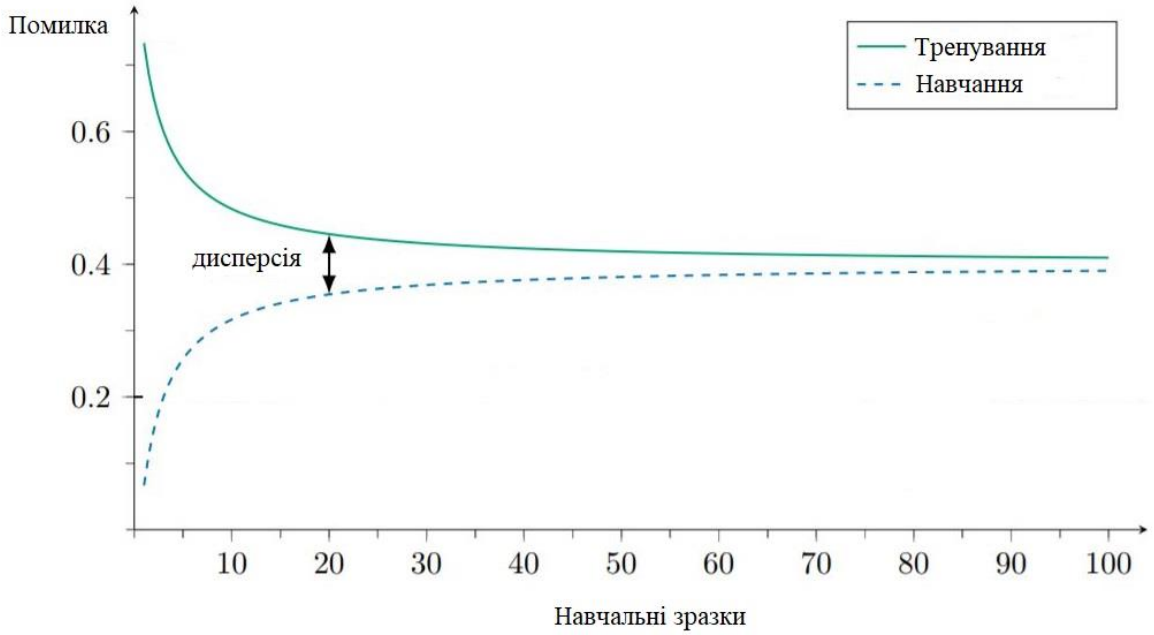


Рис 3.1 - Процес навчання мережі

Крива процесу навчання для встановлення є індикатором правильності роботи мережі, якщо велика кількість навчальних даних без будь-яких змін, що в свою чергу покращує ефективність мережі. Побудувавши криву навчального набору, можна оцінити, чи здатна архітектурна модель мережі відповідати даним для потрібної класифікації помилки. Значення помилки при тестовому наборі ніколи не повинне бути значно нижчим, ніж значення помилки навчального набору. Якщо помилка на навчальному наборі занадто висока, то збільшення даних не допоможе вирішити проблему. Натомість стане зрозуміло, що сама архітектура моделі або алгоритм її навчання потребують коригування. У випадку, коли на графіку крива навчального набору значно перевищує криву тестового (перевірочного) набору даних необхідно або зменшити роздільну здатність зображень, або ввести додаткові навчальні зразки даних, або збільшити регуляризацію для вирішення даної проблеми.

Показники мережі слугують для обчислення та відображення гіперпараметрів (наприклад, навчальної епохи). Показники можна подати у вигляді графіків гіперпараметрів на горизонтальній осі та значення якості

на вертикальній осі. Точність розпізнавання, представлена як помилка = (1 - точність) або втрата є типовими показниками якості. У випадку, коли кількість навчальних епох розглядається як головний гіперпараметр, показники діють як індикатор довгого часу тренування та продуктивності мережі. Побудувавши графік значень помилки на наборі тренувань, а також значень помилки на наборі перевірки, можна також оцінити, чи є загрозливим перенавчання [6] (Рис 3.2).

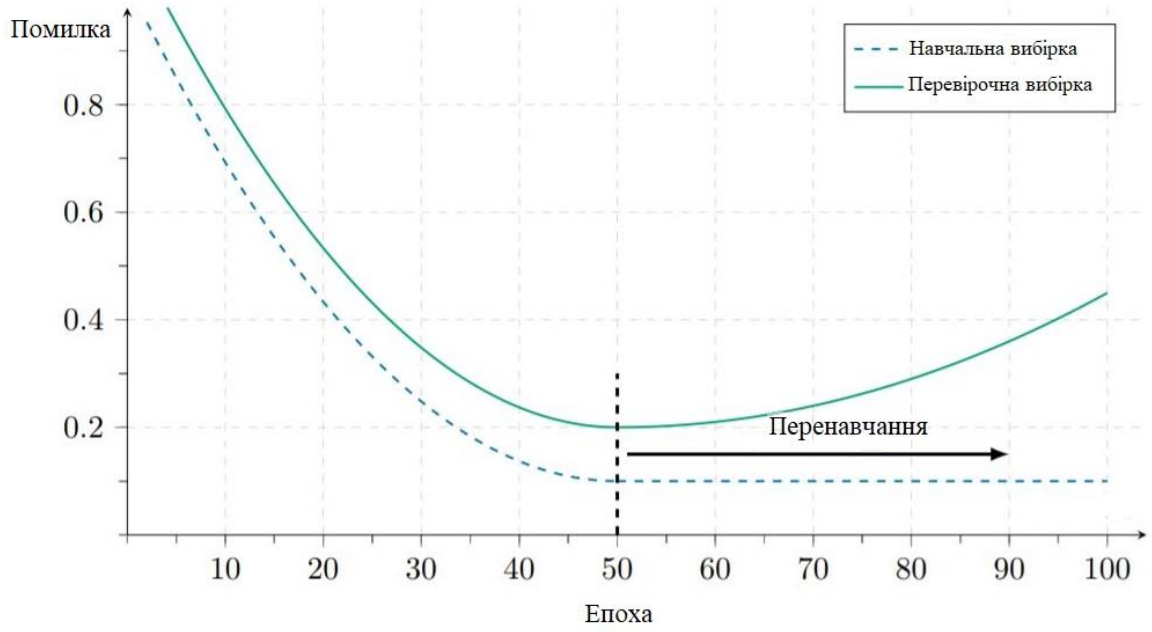


Рис. 3.2 - Проблема перенавчання мережі

Розглянемо рисунок 3.1 детальніше де зображено типову ситуацію перенавчання. Кількість епох навчання виступає як гіперпараметр мережі, а оцінка якості розпізнавання виражено через значення помилки. Чим довше буде навчатися мережа, тим краще вона звикне до тренувального набору даних. У якийсь момент мережа добре розпізнаватиме лише для навчальні дані і втратить здатність до узагальнення. На цьому етапі криві якості навчальної та перевірочної виборок на графіку розходяться. Поки класифікатор продовжує вдосконалювати показник якості на тренувальному наборі, він навпаки погіршує якість розпізнавання при застосуванні тестового набору. При ситуації, коли показник якості не поліпшується при достатній кількості епох варто змінити значення

ініціалізованих ваг нейронів на початку мережі, або застосувати нормалізацію моделі.

3.3.1 Функція втрат

Застосовані у лінійних класифікаторах функції визначення оцінки приналежності об’єкту до певного класу не надають певного контролю за вхідними даними, адже вони параметризовані значенням набором ваги нейронів W. Тому варто встановити контроль над цими вагами нейронів, і потрібно зробити це їх таким чином, щоб прогнозовані оцінки класів відповідали значенню з навчальних виборок.

Практичне застосування класифікаторів показало, що не завжди оцінка класу, що дійсно відповідає правильній ідентифікації об’єкта є вірною, тому рішенням стане обчислення не позитивного результату розпізнавання, а навпаки – негативного. Для цього і використовується функція втрат або цільова функція (функція вартості). Інтуїтивно, значення функції втрат буде високим, якщо роботу з класифікації навчальних даних завершилася помилками, і значення буде низьким, якщо все працюватиме вірно.

Функція втрат (англ. loss function) [13], також називається функцією помилки або функцією вартості - це функція, яка вираховує дійсне значення для складної події, як-от передбачення приналежності до класу вхідного вектора. Вона використовується для визначення цільової функції. У пролематиці класифікації функція втрат подається як перехресна ентропія з L_1або L_2 регуляризацією (3.1). С точки зору теорії інформації, навчанням є мінімізація значень перехресної ентропії стосовно реальних класів і гіпотезами моделі [13].

$$\mathrm{E_{CE}(W)} = \underbrace{\sum_{x\in X}\sum_{k=1}^{K}[t_k^x \log(o_k^x) + (1 - t_k^x)\log(1 - o_k^x)]}_{\text{Перехресна ентропія}} + \underbrace{\lambda_1 \cdot \overbrace{\sum_{w\in W}|w|}^{L_1} + \lambda_2 \cdot \overbrace{\sum_{w\in W} w^2}^{L_2}}_{\text{Значення втрат мережі}} \tag{3.1}$$

У формулі 3.1 параметр W - вага, X - це набір навчальних даних, $K \in N \geq 0$ це число класів, t_k^x вказує чи x є прикладом з класу k. o_k^x є вихідним значенням алгоритму класифікації, який залежить від ваг. λ1, λ2 ∈ [0, ∞) вага регуляризації, зазвичай це значення менше за 0,1.

Значенні втрати даних є позитивним, коли класифікація є неправильною, але значення втрат є числом більшим для більш складних моделей мережі. Якщо дві моделі мереж розпізнають однаковий набір даних, то кращою визнається модель архітектура якої є простішою.

Приводом для зображення саме функції втрат методом кривої на графіку замість інших показників якості є те, що тоді графік міститиме більше інформації про якість мережі. Головним недоліком функції втрати є те, що вона не має верхньої межі, як точність, і її важко інтерпретувати. Функція втрати лише показує відносний прогрес навчання, тоді як точність показує абсолютний прогрес.

Існують критерії покращення використання функції втрат, які можуть допомогти у вдосконаленні мережі [13]:

Якщо втрати протягом кількох епох не зменшуються, то темп навчання може бути занизьким. Процес оптимізації також може бути зафіксований у місцевому мінімумі.

Неправильне значення функції втрат у вигляді числа, що не існує, може бути пов'язана із надто високими темпами навчання. Іншою причиною може бути поділ на нуль або логарифм нуля. В обох випадках додавання невеликої константи, як от 10^{-7}, вирішує проблему.

Якщо крива функції втрат в залежності від кількості епох напочатку досягла межі, погані значення ініціалізації ваги може бути цьому причиною.

3.3.2 Точність, як критерій якості навчання нейронної мережі

Серед архітектурних моделей нейронних мереж, робота яких спрямована на класифікацію зображень, існують декілька критеріїв оцінки якості даної роботи. Більшість критеріїв базуються на так званій матриці передбачень, що позначається c_{ij}, діагональ якої містить кількість правильних прогнозів. Припустимо нехай $t_i = \sum_{j=1}^{k} c_{ij}$ буде числом навчальних зразків для класу i, тоді найбільш узагальненим критерієм якості є точність (англ. accuracy) виражена такою формулою (3.2):

$$accuracy(\text{c}) = \frac{\sum_{j=1}^{k} c_{ii}}{\sum_{j=1}^{k} t_i} \in [0,1] \tag{3.2}$$

Одна з проблем точності, як критерію якості - це відхилення від оцінки класу. Якщо один клас є більш узагальненим за інші, то найпростішим способом оцінення будь-якого іншого класу високим балом буде його постійна класифікація як узагальненого. Для вирішення цієї проблеми можна використовувати усереднене значення точності.

Однак окрім такого критерію як точність класифікації, на практиці важливими є також і інші критерії якості:

- швидкість оцінки та аналізу нових зображень, що надаються мережі;
- затримка часу навчання;
- ресурсозатратність;
- стійкість до (не) випадкових відхилень у навчальних даних;
- розмірність архітектури мережі.

Помітно, що подання значення числа точності з плаваючою комою дозволяє обробляти більше даних на одному пристрої.

3.4 Огляд бібліотек глибинного навчання

На сьогоднішній день у реалізації нейронних мереж важливу роль реалізації посталених архітектур на налаштувань відіграють, як правило, спеціалізовані пакети, що надають вже повністю реалізовані базові структури та алгоритми для зручної розробки. Найбільш широко використовуються такі популярні бібліотеки, як Theano, Tensorflow та Torch та ін. Дані пакети реалізують програмний інтерфейс для реалізації низькорівневих розрахунків, тому прийняття рішень, в першу чергу, повинно базуватися на суперечностях про продуктивність та зручність розробки. Варто відзначити, що Torch реалізує інтерфейс для мови Lua, не дуже популярної серед програмістів, що істотно знижує зручність і швидкість розробки. Згідно дослідженням Theano and Torch показують приблизно однакову продуктивність на одну й тому ж наборі даних.

Окрім описаної вище незручності використання бібліотеки Torch, існує ще одна, також пов'язана зі зручністю розробки. Інші два згаданих вище інструменти здатні працювати з бібліотекою більш високого рівня Keras [14], що надає загальні, для Tensorflow [15] і Theano, заздалегідь визначені алгоритми та структури. За результатами порівняння та аналізу документації бібліотек було прийнято рішення використовувати для програмної реалізації згорткової нейронної мережі зв'язок бібліотек Tensorflow і Keras.

Бібліотека Keras [14] - програмний продукт, написаний мовою python, що надає високорівневе API для роботи зі штучними нейронними мережами. Даний продукт створювався з метою запобіганню труднощів в розробці нейронних мереж, пов'язаних із синтаксичними особливостями конкретних пакетів. В січні 2016 року даний інструмент було придбано

компанією Google і адаптовано як основне високорівневе надлаштування над бібліотекою Tensorflow.

Бібліотека Theano була написана в першу чергу як розширення мови Python, яка дозволяє ефективно розрахувати математичні вирази, що містять багатомірні масиви. В бібліотеці реалізується базовий набір інструментів для побудови нейронних мереж. Процес створення моделі та визначення її параметрів вимагає написання об'ємного коду, що включає реалізацію класу моделі, самостійного визначення її параметрів, реалізації методів, що визначають функцію помилок, правило обрахування градієнтів, способи зміни ваги нейронів.

Бібліотека Caffe реалізована на мові програмування C++. Топологія нейромереж, вихідні дані та спосіб навчання задаються за допомогою конфігураційних протоколів (застосовується технологія і протокол передачі даних) у прототиповому форматі. Побудова структури мережі виконується з простотою та зручністю. Бібліотека Caffe підтримується досить великою спільнотою розробників та користувачів і на сьогоднішній день є самою розповсюдженою бібліотекою глибинних навчальних програм широкого кола застосування.

Через те, що в останні роки методи машинного навчання отримали велику популярність і застосовуються в безлічі різних областей, існує великий вибір бібліотек для реалізації моделей машинного навчання. В таблиці 3.1 представлені найбільш відомі та поширені бібліотеки.

Python – гнучка мова програмування високого рівня, створена в 1991 році, що ставить за свої цілі низький «рівень входу» та легку читаємість результуючих програм. Мова програмування Python має кілька версий. В даний час широко використовуються версії 2.7 і 3.5.

Таблиця 3.1 - Порівняння бібліотек глибинного навчання

Назва бібліотеки	*Theano*	*Caffe*	*Torch*	*TensorFlow*
Мова програмування	Python	Python, C++	Lua	Python
Спеціалізація на методах машинного навчання	Різноманітні види регресій, нейронні мережі	Нейронні мережі	-	Різноманітні види регресій, нейронні мережі
Можливість глибинного навчаня	+	+	+	+
Швидкість роботи	Середня	Висока	Низька	Висока
Якість позпаралелювання між декількома пристроями	-	-	-	+
Допоміжі функції (автоматичне обрахування градієнта та ін.)	-	+ (зарахунок допоміжних бібліотек Python)	+ (зарахунок допоміжних бібліотек Python)	+

3.4.1 TensorFlow

TensorFlow [15] - це бібліотека програмного забезпечення з відкритим програмним кодом для задач машинного навчання, розробленого компанією Google. Вона дозволяє створювати та навчати нейронні мережі різної архітектури, що знаходять своє застосування у виявленні та розпізнаванні образів, пошуку взаємозв'язків. TensorFlow

також включає в себе TensorBoard, який представляє собою засоби візуалізації в браузері для оцінки ефективності навчання та мережевих параметрів моделі.

TensorFlow досягає своєї продуктивності завдяки паралельній обробці задач між центральними та графічними процесорами GPU (у тому числі декількох одночасно) і навіть горизонтальне масштабування за допомогою gRPC. Tensorflow підтримує мови програмування Python і C++.

Кожне обчислення в TensorFlow представляється у вигляді графу потоку даних, графу обчислень. Граф обчислень є моделлю, яка описує як будуть виконуватися обчислення. Важливо зауважити, що складання графа обчислень і виконання операцій в заданій структурі - два різних процеси. Граф складається з плейсхолдеров (tf.Placeholder), змінних (tf.Variable) і операцій. У ньому виробляється обчислення тензорів - багатовимірних масивів, які можуть бути числом або вектором.

Графи виконуються в сесіях (tf.Session). Існують два типи сесій: звичайні та інтерактивні (tf.InteractiveSession); інтерактивна сесія підходить для виконання в консолі. Сесія зберігає стан змінних (Variables) і черг (queues). Явне створення сесій і графів гарантує належне звільнення ресурсів пам'яті. У графі кожна вершина має 0 або більше входів і 0 або більше виходів, і являє собою реалізацію операції. Тензорами є ребра графа, а саме масиви довільного розміру (тип масиву вказується під час побудови графа). Особливі вершини, що управляють залежності (control dependencies), також можуть бути в графі: вони вказують, що вихідний вузол для контрольної залежності повинен закінчити виконання до того, як вузол одержувача контрольної залежності почне виконуватися.

Кожна операція має назву і являє собою абстрактне обчислення (наприклад, операція суми). У операції можуть бути атрибути. Ядро - специфічна реалізація операції, яка може бути виконана на певному типі пристрою (центральний або графічний процесор).

Змінна - особливий вид операції, який повертає лічильник на постійно мінливий тензор: така змінна не зникає після одиничного використання графа. Лічильники на подібні тензори передаються чисельним операціями, які потім змінюють вказаний тензор. У завданнях машинного навчання, параметри моделі зазвичай зберігають тензори в змінних, які оновлюються на кожному кроці навчання.

3.5 Набори даних. Етап попередньої підготовки даних

Для оцінки роботи мережі за певними критеріями використовується набір даних, у нашому випадку будуть використовуватися набори даних у вигляді сховищ класифікованих зображень:

- CIFAR-10 (англ. Canadian Institute for Advanced Research 10) сховище являє собою 10-класовий набір кольорових зображень розміром 32 × 32 пікселів. Усі зображення поділено такі на класи: літак, автомобіль, птах, кіт, олень, собака, жаба, кінь, корабель, вантажівка. Досягнуто значення точності розпізнавання 96.54 % [16].
- CIFAR-100 - це набір зі 100 класів кольорових зображень розміром 32 × 32 пікселів. Запропоновані сховищем 100 класів згруповані між собою до 20 суперкласів. Групи включають в себе тварин, людей, рослин, зовнішніх сюжетів, транспортних засобів та інших предметів. CIFAR-100 не є суперсетом CIFAR-10, оскільки CIFAR-100 не містить класу літака. Досягнуто значення точності розпізнавання 82.82 % [17].
- GTSRB (англ. German Traffic Sign Recognition Benchmark) це набір даних із 43-х знаків дорожнього руху. 51 839 зображень є кольоровими та мають мінімальний розмір від 25 × 25 пікселів до 266 x 232 пікселів. Досягнуто значення точності розпізнавання 99.46 % [17].

- HASYv2 (англ. Handwritten Symbols version 2) цей набір складається з 369 класів чорно-білих зображень розміром 32 × 32 пікселі. 369 класів містять латинські та грецькі букви, стрілки, математичні символи. Досягнуто значення точності розпізнавання 82 % [18].
- STL-10 (англ. self-taught learning 10) це 10-класовий набір кольорових зображень розміром 96 × 96 пікселів. Представляє десять класів: літак, птах, машина, кіт, олень, собака, кінь, мавпа, корабель, вантажівка. Досягнуто значення точності розпізнавання 74.80 % [19].
- SVHN (англ. Street View House Numbers) набір існує у двох форматах. Для наступних експериментів використовувався об'ємний формат цифр. Він містить 10 цифр, обрізаних з фотографій Google Street View. Усі зображення кольорові, розміром 32 × 32 пікселів. Досягнуто значення точності розпізнавання 98.41 % [20].
- MNIST (англ. Mixed National Institute of Standards and Technology) набір складається з рукописних цифр, а саме навчальний набір складається із 60 000 прикладів та тестовий набір з 10 000 прикладів. Це підмножина більшого набору, що доступна у NIST. Цифри є нормовані за розміром та центровані у зображенні фіксованого розміру. Набір є гарним прикладом для спроб методі навчання та методі розпізнавання образів на реальних даних у дії, витрачаючи при цьому мінімальні зусилля на попередню обробку та форматування зображень. Досягнуто значення точності розпізнавання 98.7 % [21].

Як етап попередньої обробки, нсі зображення з наборів даних перед поданням на вхід до згорткової мережі повинні бути змінені. Незважаючи

на малий розмір зображення співвідношення сторін у перетворенні дотримано оригінальним для того, щоб уникнути спотворення картинки. Значення пікселів для червоного, синього, зеленого каналів, що знаходяться у діапазоні від 0 до 255 було нормалізовано шляхом поділу кожного зі значень на 255. Результатом стане набір коефіцієнтів, що приймають значення у діапазоні від 0 до 1. Початковий формат значень був цілочисленним, його було замінено на формат з плаваючою комою для найкращого виконання цієї нормалізації. Для вирішення проблеми класифікації значення пікселів подаються у вигляді векторів, що відповідають кожному з 10 класів, які трансформуються до бінарної матриці. Аби досягти кращого розпізнавання обєктів у різних положеннях на зображенні набір даних було доповнено модифікованими вхідними зображеннями. Під модифікацією зображень розуміється для наборів CIFAR-10, CIFAR-100, STL-10, MNIST та HASYv2 горизонтальне відображення зображення, вертикальне відображення зображення, зміни висоти або ширини картинки. Для GTSRB набору даних зображення для проведення навчання та тестування були масштабовані до розміру 32 × 32 пікселі. Лише для набору SVHN ніяких змін даних не застосовується.

3.6 Ієрархічна модель класифікатору

Установка принципів роботи класифікатору для окремих наборів даних становить певну складність у розробників. Багато прикладів, що ілюструють структури класифікаторів не є найкращим зразком серед показників якості, та й оцінка кожного з прикладів може тривати досить великий проміжок часу, оскільки навчання згорткових нейронних мереж може займати дні або навіть і тижні за тривалістю. Крім того, деякі методи аналізу вхідного набору даних стають гіршими у використанні зі збільшенням кількості класів або доповненими навчальними зразками. Запропоноване рішення цієї проблеми - це побудова ієрархії класифікаторів. Основний класифікатор (або корневий) виділяє кластери

класів (підкласи), котрі у свою чергу виділяють одиничні класи, тобто вина є спеціалізованими. Основний класифікатор повинен виділити 6 явних класів (це може бути класи транспортних засобів, дорожніх знаків, цифр та ін.) або 17 неявних класів. Якщо основний класифікатор передбачає транспортний засіб, то інший класифікатор повинен передбачити чи є він двоколісним (велосипед) або чотирьох колісним (автомобіль). Наприклад, якщо основний класифікатор прогнозує дорожній знак, то інший класифікатор повинен передбачити, чи це знак обмеження швидкості, чи це знак попередження про небезпеку або щось інше. Проте, якщо головний класифікатор класифікує дорогу, то інший класифікатор не стане активним. Ієрархічний підхід класифікатору виділяє 7 кластерів класів і, таким чином, використовує 8 класифікаторів. Така ієрархічна модель класифікаторів потребує сформовані кластери класів.

Існує два способи кластеризації класів: за подібністю або семантикою. Якщо для семантичної кластеризації потрібні або додаткові відомості, або обробка вручну, то класифікація за подібністю може автоматично братися із даних. Іноді семантично подібні класи часто також є візуально подібними. Наприклад, у наборі даних ImageNet більшість собак семантично і візуально схожі на один одного, ніж на інші об’єкти. Прикладом неочевидної ситуації класифікації можуть стати символи: символ суми зовнішнім виглядом для грецької букви, але семантично символ значно ближчий до оператора додавання.

Одним з підходів до кластеризаціх класів за подібністю є навчання класифікатора та дослідження значень його прогнозів. Кожен клас представляється у матриці прогнозів одним рядком. Ці рядки можуть бути представлені за допомогою стандартних кластерних алгоритмів, наприклад, таких як k-значення.

Матриця прогнозів класифікаторів $c_{ij} \in N^{k \times k}$ будується на тому, як часто клас і був класифікований та клас j був передбачений. На основі

матриці прогнозів класи можуть бути кластеризовані за алгоритмом, головні ідеї якого представлені так:

- Порядок стовпців та рядків у матриці прогнозів є довільним. Це означає, що можна переставляти рядки та стовпці. Якщо рядки i та j переставлені, то стовпці i та j також повинні бути переставлені, щоб зберегти структуру матриці прогнозів.
- Якщо два класи часто прогнозуються, то вони схожі на класифікатор.

Навчання класифікатору проводилося на 100 класах набору даних CIFAR-100. Кожний клас має 100 зразів для тесту. Точність встановлення приналежності класів до кластерів наведена в таблиці 3.2. Той факт, що основний класифікатор досягає кращих результатів у межах кластерів, ніж спеціалізовані класифікатори у 13 з 14 випадків (як видно з таблиці 3.2), може бути пов'язаний з обмеженими характеристиками навчальних даних, перенавчанням або невеликим розміром 32 × 32 пікселі зображень вхідного набору. Експеримент також показує, що більшість помилок пов’язана з невизначенням правильного кластера. Отже, в даному випадку необхідно зробити щось більше, ніж поліпшити ущільнення класів в кластері, щоб покращити основний класифікатор.

Хоча класи всередині кластера фіксують більшість класифікацій, багато неправильних класифікацій виникають поза кластерами.

Таблиця 3.2 - Точність основного класифікатора під час навчання на повному наборі зі 100 класів із розподілом у 14 кластерів.

Клстер	Класи	Точність визначення класифікатору приналежності класу до кластеру	
		Основний класифікатор	Спеціалізований класифікатор
1	3	83.27 %	71.98 %
2	5	58.55 %	43.42 %
3	2	91.13 %	83.66 %
4	2	87.38 %	81.71 %
5	3	79.31 %	71.11 %
6	2	78.65 %	72.18 %
7	2	91.11 %	87.52 %
8	2	87.23 %	81.89 %
9	2	90.88 %	87.99 %
10	2	84.52 %	74.10 %
11	2	87.09 %	75.22 %
12	2	94.67 %	76.17 %
13	2	83.58 %	86.33 %
14	2	89.95 %	89.50 %

3.6.1 Вибір архітектури згорткової нейронної мережі. Базова модель

Існує багато варіантив обрахунку кількості параметрів для кожного шару мережі, зокрема, число параметрів залежне від наластувань розміру карт ознак. Якщо n_i (i = 0, ..., n) дорівнює кількості вихідних карт озназ шару i, де при i = 0 це вхідний шар та усі фільтри мають розмір 3 x 3, тоді розрахунок числа параметрів для наступних шарів мережі виглядає так (3.3):

Число параметрів (3.3)

$$= \sum_{i=1}^{k}((n_{i-1} \cdot 3^2 + 1) \cdot n_i)$$

Для повнозв'язного шару з n вузлами та k входами матиме $n \cdot (k + 1)$ параметрів, де + 1 - це нейрон зміщення. Шар згортки і з k_i фільтрами розміром n x m буде застосовано до k_{i-1} карти ознак і матиме $k_i \cdot k_{i-1}(n \cdot m + 1)$ параметрів, де + 1 - це нейрон зміщення. Для повнозв'язного шару з n вузлами після застосованих k карт ознак розміром $m_1 \times m_2$ матимемо $n \cdot (k \cdot m_1 \cdot m_2 + 1)$ параметрів. Щільний блок глибиною L швидкості росту n та карт ознак розміром 3 × 3 матиме $L + n \cdot 3^2 + 3^2 \cdot n^2 \sum_{i=0}^{L} L - i = L + 9n + 9n^2 \frac{L^2 - L}{2}$ параметрів.

Базова архітектура моделі згорткової мережі може бути описана за таким шаблоном :

$$\text{Блок згортки } (n) = (\text{Згортка} - \text{Пакетна нормалізація} - \text{Функція активації})^2 - \text{Підвибірка}$$

Використовується функція активації ReLU в усіх згорткових шарах, окрім останнього повнозв'язного, тут застосовано функцію softmax. Перед останнім шаром згортки застосовано шари виключення із dropout імовірність якого дорівнює значенню 0.5. Детально архітектуру базової моделі представлено у таблиці 3.3. На рисунку 3.3 графічно зображено архітектуру мережі з таблиці 3.

Таблиця 3.3 - Архітектура базової моделі згорткової мережі

N	*Тип*	*Канали @ фільтр / крок*	*Параметри*	*Вихідний розмір*
	Вхідні дані		0	3 @ 32 x 32
1	Згортковий шар	32 @ 3 x 3 x 3 / 1	896	32 @ 32 x 32
2	BN + ReLu		64	32 @ 32 x 32
3	Згортковий шар	32 @ 3 x 3 x 32 / 1	9248	32 @ 32 x 32
4	BN + ReLu		64	32 @ 32 x 32
	Шар підвибірки (max)	2 x 2 / 2	0	32 @ 16 x 16
5	Згортковий шар	64 @ 3 x 3 x 32 / 1	18 496	64 @ 16 x 16
6	BN + ReLu		128	64 @ 16 x 16
7	Згортковий шар	64 @ 3 x 3 x 64 / 1	36 928	64 @ 16 x 16
8	BN + ReLu		128	64 @ 16 x 16
	Шар підвибірки (max)	2 x 2 / 2	0	64 @ 8 x 8
9	Згортковий шар	64 @ 3 x 3 x 64 / 1	36 928	64 @ 8 x 8
10	BN + ReLu		128	64 @ 8 x 8
	Шар підвибірки (max)	2 x 2 / 2	0	64 @ 4 x 4
11	Згортковий шар (v)	512 @ 4 x 4 x 64 / 1	**524 800**	512 @ 1 x 1
12	BN + ReLu		1024	512 @ 1 x 1
	Dropout 0.5		0	512 @ 1 x 1
13	Згортковий шар	512 @ 1 x 1 x 512 / 1	262 656	512 @ 1 x 1
14	BN + ReLu		1024	512 @ 1 x 1
	Dropout 0.5		0	512 @ 1 x 1

Таблиця 3.3 (продовження)

N	*Тип*	*Канали @ фільтр / крок*	*Параметри*	*Вихідний розмір*
15	Згортковий шар	k @ 1 x 1 x 512 / 1	k*(521 + 1)	k @ 1 x 1
	Шар підвибірки (avg)	1 x 1	0	k @ 1 x 1
16	BN + softmax		2k	k @ 1 x 1
$\sum =$			515k + 892 512	103 424+2k

Базова архітектура мережі (таблиця 3.3) приймає вхідні дані трьох каналів розміром 32 x 32. Усі згорткові шари використовують такий параметр мережі як доповнення нулями, окрім згорткового шару під номером 11, оскільки тут постає завдання зменшити розмір карти ознак до розміру 1 x 1. Якщо вхідна карта ознак перевищує 32 × 32, потужність розділяється між двома блоками шарів згортки, нормалізації та функції активації, а також додається один шар максимальної підвибірки.

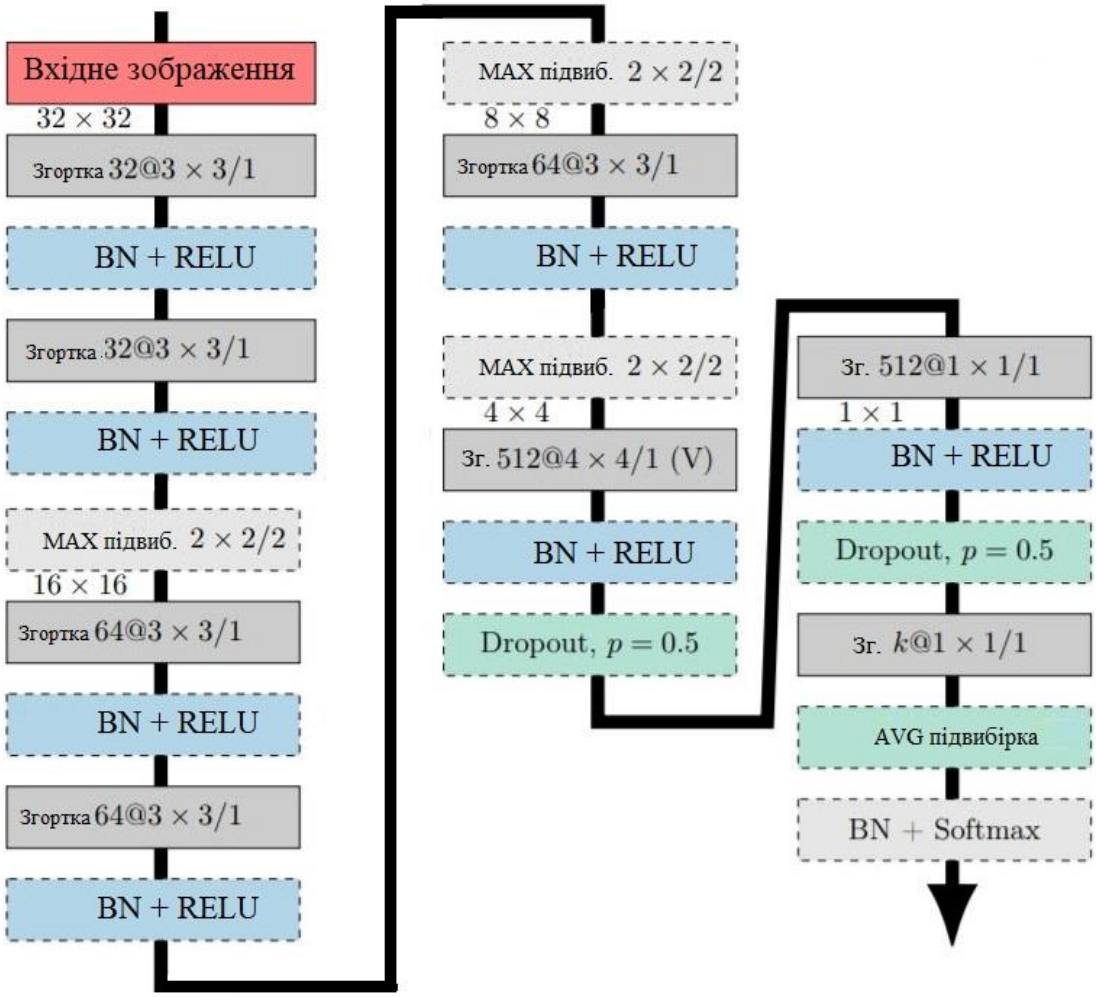


Рис 3.3 - Графічне представлення архітектури базової згорткової нейронної мережі

Як видно з таблиці 2 перший згортковий шар базової моделі має 896 параметрів. Якщо припустити, що небажано застосовувати менш як три фільтри для кожного шару, то всі шари використовуватимуть лише фільтри розміром 3 × 3, тоді максимальна глибина дорівнюватиме 1. Крім того, передбачається використання щонайменше 800 параметрів, тоді існує ще 120 комбінацій можливих шарів.

Було проведено навчання базової моделі з установкою таких параметрів:

- Алгоритм навчання – зворонтнього поширення помилки [4].

- Темп навчання - 10^{-4}.
- Розмір серії (англ. batch size) – 64.
- Кількість епох – 1000.

Вид підготовки даних для кожного з наборів див. підрозділ 3.5. Якщо набір даних не визначає частку навчального / тестового наборів, застосовується розподіл 67% / 33%. Якщо набір даних не включає в себе перевірочний набір, то набір для навчання розподіляється по на 90% навчального набору / 10% тестового набору. Налаштування кількості епох (циклів) за допомогою процедури, відомої як рання зупинка - просто зупиняється процес навчання, якщо за задану кількість епох (англ. patience) втрати не починають зменшуватися. Так як наш набір даних відносно невеликий і насичується швидко, ми встановимо patience рівним 10 епохам, а максимальну кількість епох становитиме 1000. Вага ядра ініціалізуються рівномірною ініціалізацією за схемою [HZRS15b].

Всі експерименти реалізовані за допомогою бібліотеки Keras 2.0 та Tensorflow 1.0, а також для застосування графічних прискорювачів використано бібліотеку cuDNN 5.1 (бекенд) [22]. Експерименти проводились на графічному процесорі GeForce 940MX та подані у таблиці 3.4. Як видно з результатів, наведених у таблиці 3.4, значення відсотків точності для наборів даних не є вищими за представлені результати інших архітектур, тому варто замислитися над оптимізацією ієрархічного класифікатору та самої архітектури загалом аби покращити існуючи цифри та перевершити існуючі рішення.

Таблиця 3.4 - Показник точності розпізнавання базової моделі при різних наборах даних

N	Набір даних	Навчання мережі		Тестування мережі	
		Точність	Похибка	Точність	Похибка
1	CIFAR-10	91.25 %	σ = 1.10	85.90 %	σ = 0.87

Таблиця 3.4 (продовження)

N	Набір даних	Навчання мережі		Тестування мережі	
		Точність	Похибка	Точність	Похибка
2	CIFAR-100	76.64%	σ = 1.41	82.82%	σ = 0.55
3	GTSRB	100%	σ = 0.01	99.46%	σ = 0.11
4	HASYv2	88.48 %	σ = 0.41	81.00%	σ = 0.10
5	MNIST	99.93 %	σ = 0.07	98.70%	σ = 0.06
6	STL-10	94.12 %	σ = 0.87	74.80%	σ = 0.34
7	SVHN	99.02 %	σ = 0.07	98.41%	σ = 0.10

Завдяки процедурі ранньої зупинки, різниться кількість епох під час навчання мережі, наприклад, з набором даних, де дані були змінені чи доповнені, число епох коливається від 133 до 182, зі стандартним відхиленням у 17.3 епохи при наборі CIFAR-100. На рис 3.4 показано графік відображення змін значень функції втрат за епохами на прикладі набору CIFAR-100.

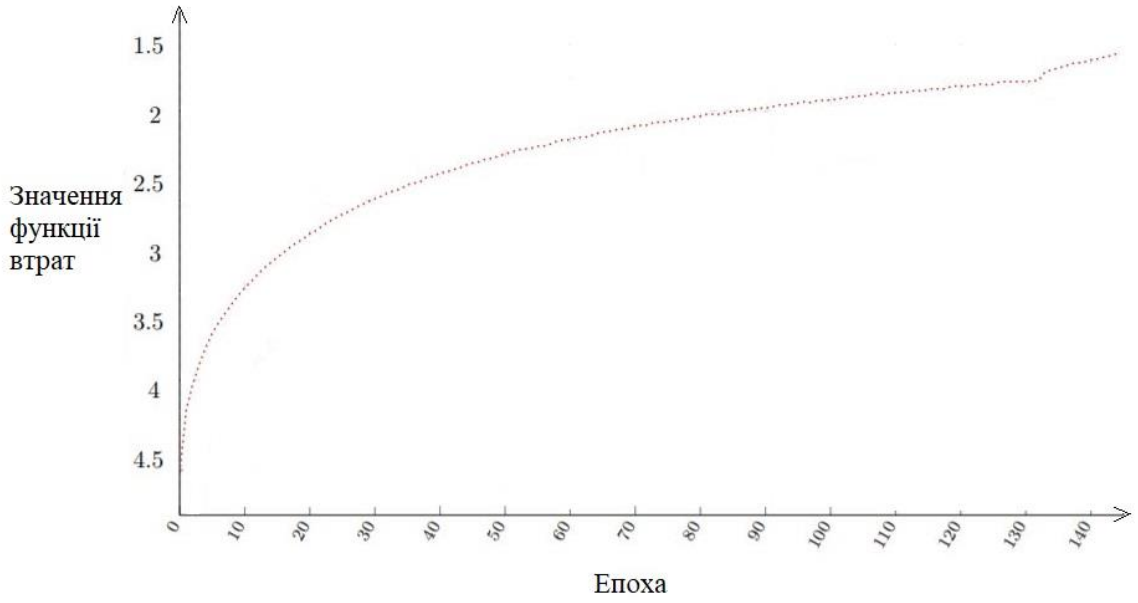


Рис 3.4 - Крива значення функції втрат по епохам

3.7 Навчання згорткової нейронної мережі з використанням графічних прискорювачів

Для навчання нейронних мереж з використанням графічних прискорювачів (англ. GPU) використовують бібліотеки, що спеціалізуються на паралельному процесу роботі мережі. Наприклад, для мережі, яка розпізнає об'єкти з набором даних CIFAR-10, застосування GPU дозволило прискорити навчання більше ніж у 7 разів, порівняно з CPU. Однак межу прискоренню не встановлено, і навчання можна зробити ще швидше, якщо використовувати бібліотеку NVIDIA cuDNN.

На відміну від популярного аналогу cuDNN [22] - бібліотеки NVIDIA CUDA, яка забезпечує можливість виконання на GPU будь-яких обчислень, бібліотека cuDNN спеціально розроблена для навчання глибоких нейронних мереж, особливо таких як згорткові. Вона містить оптимізовані для GPU реалізації згорткових і рекурентних мереж, різних функцій активації (напівлінійна, сигмоїдальна, гіперболічний тангенс, softmax), алгоритму зворотнього поширення помилки та інше. Бібліотека cuDNN дозволяє навчати нейронні мережі на GPU в кілька разів швидше, ніж просто CUDA. Також варто зазначити, що бібліотека є цілком безкоштовною та її можна завантажити на сайті розробника.

Розглянемо саме у який спосіб бібліотекам для навчання мереж на графічних прискорювачах вдається досягти зниження обчислюваної складності та повноцінно повести навчання глибокої мережі. Умовно процес можна розділити на такі етапи [15]:

- GPU приймають графічні дані, що відображають стан згортальної нейронної мережі і містять одну або більше текстур, що представляють одну або більше змінних нейронної мережі, причому згадані текстури містять текстуру з двовимірної адресацією, щонайменше одна зі згаданих текстур представляє змінну нейронної мережі з адресацією , що має

розмірність більш як двоє, яка була сплощена в двовимірну адресацію;

- на GPU виконують одну або більше програм для здійснення прямого проходу в згортковій нейронншй мережі, одну або більше програм для здійснення зворотнього проходу в згортковій нейронної мережі, причому виконання включає в себе виконання згорткових операцій;
- використовують одну або більше програм для зміни карт ознак в згортковій нейронній мережі за допомогою зміни графічних даних на підставі результатів зворотнього проходу;
- повторно виконують одну або більше програм для здійснення прямих проходів, зворотніх проходів і для зміни графічних даних, поки згорткова нейронна мережа повністю не навчиться.

Проведемо порівняння часу навчання згорткової нейронної мережі базової моделі використанням графічного прискорювача і без (таблиця 3.5), а також зробимо заміри часу навчання для архітектури DenseNet-40-12, на тому ж графічному прискорювачі, використовуючи набір даних CIFAR-10, як одного з найменш складних наборів для розпізнавання.

Таблиця 3.5 - Час навчання згорткової нейронної мережі на CPU та GPU

Мережа	Tensorflow	Час навчання		
		1 зображення	128 зображень	Епоха
Базова модель	Intel i7-4930K	3 мс	245 мс	230.0 с
Базова модель	GeForce 940MX	4 мс	120 мс	144.6 с
DenseNet-40-12	GeForce 940MX	27 мс	2403 мс	-

Есперимент проведено на графічному прискорювачі GeForce 940MX та процесорі марки Intel моделі i7-4930K. GeForce 940MX –графічний

прискорювач або відеокарта об’ємом 2 Гб та таковою частотою у 1176 МГц, в основі якого лежить графічний чіп GM108 архітектури Maxwell, який має 384 ядра CUDA і шину пам'яті 64 біта.

Оптимізатор Tensorflow використовує інструкції SSE4.X, AVX, AVX2 та FMA. Результати есперименту наведені в таблиці 3.5.

3.8 Висновок до третього розділу

У задачах класифікації або розпізнавання величезною перевагою нейронних мереж над спеціалізованими алгоритмами постає їх здатність до навчання, тому у розділі 3 розглянуті основні ідеї та методи реалізації цього складного процесу. Щоб покращити та пришвидшити процес навчання, а також уникнути такої події, як перенавчання, мереж звертаються до математичних засобів, таких як алгоритм зворотнього поширення помилки або процедур нормалізації.

Оцінити якість мереж допомагають такі показники, як точність та втрата (обраховується функцією втрат), побудувавши графік яких, чи подавши у вигляді таблиць, можна провести аналітику напрацьованих результатів.

У розділі 3 було запропоновано архітектуру згорткової мережі із ієрархічним класифікатором (базової моделі), а також проведено навчання та встановлено показник точності для кожного з поданих наборів даних за допомогою сучасних бібліотек глибинного навчання. З огляду результатів, які подані у таблиці 3.4 варто зазначити, що базова модель мережі потребує значного покращення аби досягти рівня показника точності вищого за уже існуючі.

РОЗДІЛ 4. СПОСІБ ПОКРАЩЕННЯ ТОЧНОСТІ КЛАСИФІКАЦІЇ БАЗОВОЇ МОДЕЛІ

4.1 Нейрон зміщення

Нейрон зміщення або просто зміщення (англ. bias) - це третій вид нейронів серед існуючих, що використовується в більшості нейромереж. Особливість цього типу нейронів полягає в тому, що його вхід і вихід завжди дорівнюють одиниці і вони ніколи не мають вхідних синапсів. Нейрони зміщення можуть, або присутні в нейронній мережі для кожного з шарів, або повністю відсутні, 50/50 бути не . З'єднання у нейронів зсувів такі ж, як у звичайних нейронів - з усіма нейронами наступного рівня, за винятком того, що синапсів між двома нейронами зміщення бути не може. Відповідно, їх можна розмістити на вхідному шарі всіх схованих шарів, але ніяк не на вихідному шарі, так як їм просто не буде з чим формувати зв'язок [23].

Нейрон зміщення потрібен для того, щоб мати можливість отримати вихідний результат, шляхом зсуву графіку функції активації праворуч або ліворуч. Коли при навчанні регулюється вага прихованих і вихідних нейронів, то змінюється нахил функцій активації на графіку. Однак, регулювання ваги нейронів зміщення може дати нам можливість змінити функцію активації по осі абсцис і захопити нові ділянки. Іншими словами, якщо точка, відповідальна за вирішення, буде знаходитися на графіку зліва, то нейронна мережа ніколи не зможе вирішити завдання без використання нейронів зміщення. Тому рідко можна зустріти нейронні мережі без нейронів відхилення.

Також нейрони зміщення допомагають у тому випадку, коли всі вхідні нейрони отримують на вході 0 і незалежно від того, яка у них вага, всі вони передаються на наступний шар 0, але не у випадку присутності нейрона зміщення. Наявність або відсутність нейронів зміщення – це також гіперпарамерт нейронної мережі.

4.2 Модернізація класифікатору базової моделі. Нова модель

З огляду на результати роботи архітектури базової моделі згорткової нейронної мережі (див. розділ 3) можна зауважити, що шляхи її оптимізації полягають у наступному:

- Видалення нейронів зміщення для останніх шарів мережі: для всіх шарів, на виході роботи яких отримуємо карти ознак розміром 1 × 1, зміщення вилучається.
- Збільшення розмірності шару максимізаційної підвибірки до розміру 3 × 3.
- Збільшення числа фільтрів для вхідного шару мережі.

Ваховуючи усі запропоновані зміни до базової архітектури отримаємо нову, модернізовану, архітектуру згорткової нейронної мережі, що представлена у таблиці 4.1. На рисунку 4.1 графічно зображено архітектуру оптимізованої моделі мережі з таблиці 4.1.

Таблиця 4.1 - Архітектура нової моделі згорткової мережі

N	*Тип*	*Канали @ фільтр / крок*	*Параметри*	*Вихідний розмір*
	Вхідні дані		0	3 @ 32 x 32
1	Згортковий шар	69 @ 3 x 3 x 3 / 1	1932	69 @ 32 x 32
2	BN + ReLu		138	69 @ 32 x 32
3	Згортковий шар	69 @ 3 x 3 x 32 / 1	42 918	69 @ 32 x 32
4	BN + ReLu		138	69 @ 32 x 32
	Шар підвибірки (max)	3 x 3 / 2	0	32 @ 16 x 16
5	Згортковий шар	64 @ 3 x 3 x 32 / 1	39 808	64 @ 16 x 16
6	BN + ReLu		128	64 @ 16 x 16
7	Згортковий шар	64 @ 3 x 3 x 64 / 1	36 928	64 @ 16 x 16
8	BN + ReLu		128	64 @ 16 x 16

Таблиця 4.1 (продовження)

N	*Тип*	*Канали @ фільтр / крок*	*Параметри*	*Вихідний розмір*
	Шар підвибірки (max)	3 x 3 / 2	0	64 @ 8 x 8
9	Згортковий шар	64 @ 3 x 3 x 64 / 1	36 928	64 @ 8 x 8
10	BN + ReLu		128	64 @ 8 x 8
11	Згортковий шар (v)	512 @ 4 x 4 x 64 / 1	**524 288**	512 @ 1 x 1
12	BN + ReLu		1024	512 @ 1 x 1
	Dropout 0.5		0	512 @ 1 x 1
13	Згортковий шар	512 @ 1 x 1 x 512 / 1	262 144	512 @ 1 x 1
14	BN + ReLu		1024	512 @ 1 x 1
	Dropout 0.5		0	512 @ 1 x 1
15	Згортковий шар	k @ 1 x 1 x 512 / 1	k * 521	k @ 1 x 1
	Шар підвибірки (avg)	1 x 1	0	k @ 1 x 1
16	BN + softmax		2k	k @ 1 x 1
	$\sum =$		514k +947 654	179 200 + 2k

Покращена модель (архітектура) мережі (таблиця 4.1) приймає вхідні дані трьох каналів розміром 32 x 32. Усі згорткові шари використовують такий параметр мережі як доповнення нулями, окрім згорткового шару під номером 11, оскільки тут постає завдання зменшити розмір карти ознак до розміру 1 x 1. Якщо вхідна карта ознак перевищує 32 × 32, потужність розділяється між двома блоками шарів згортки, нормалізації та функції активації, а також додається один шар

максимальної підвибірки. У даній мережі відсутні нейрони зміщення, що вплинуло на зменшення кількості параметрів.

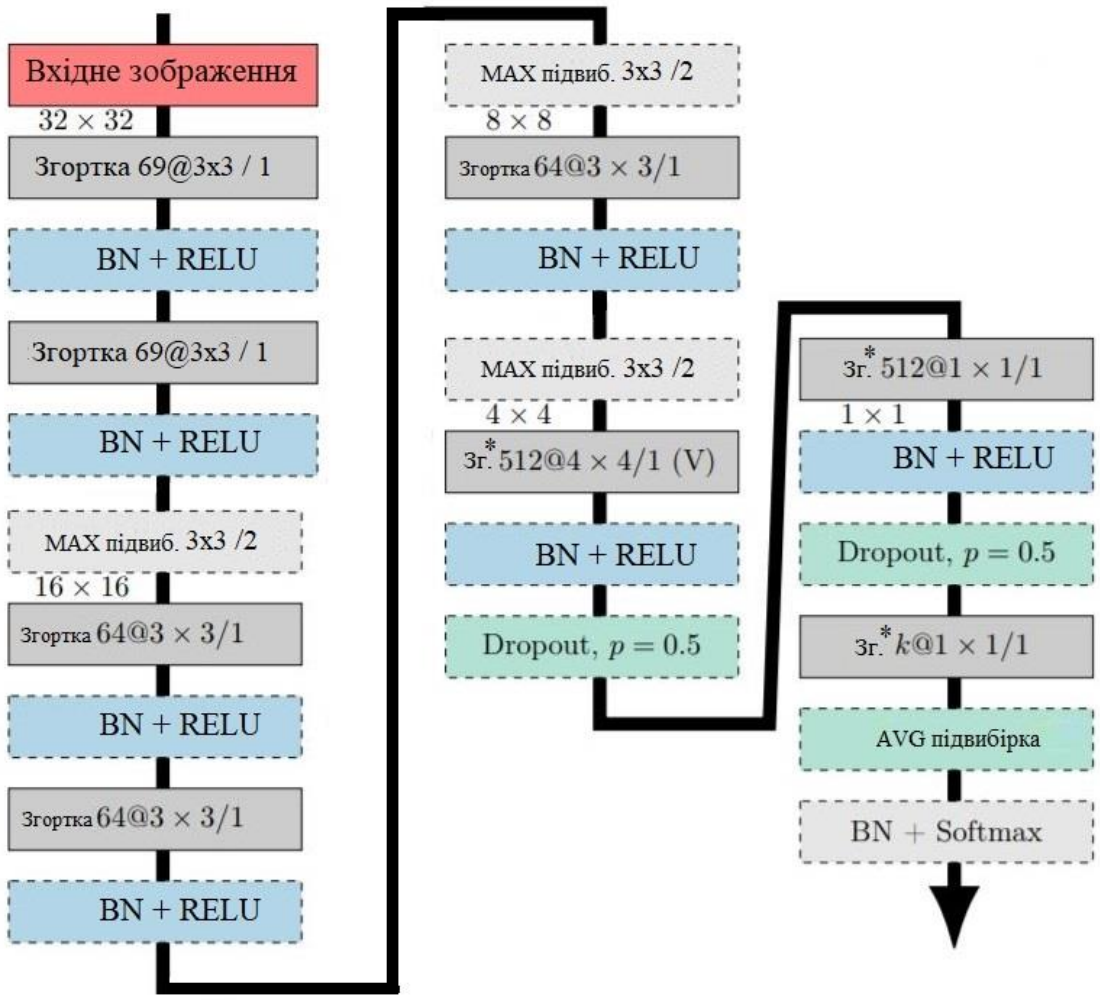


Рис 4.1 - Графічне представлення архітектури нової згорткової нейронної мережі (* позначені шари де відсутнє зміщення)

Всі експерименти навчанння оптимізованої моделі реалізовані за допомогою бібліотеки Keras 2.0 та Tensorflow 1.0, а ткож ждя застосування графічних прискорювачів використано бібліотеку cuDNN 5.1 (бекенд). Експерименти проводились на графічному процесорі GeForce 940MX та подані у таблиці 4.2

Есперимент проведено на графічному прискорювачі GeForce 940MX та процесорі марки Intel моделі i7-4930K.

Оптимізатор Tensorflow використовує інструкції SSE4.X, AVX, AVX2 та FMA. Результати есперименту наведені в таблиці 4.3.

Таблиця 4.2 - Показник точності розпізнавання нової моделі при різних наборах даних

N	Набір даних	Навчання мережі		Тестування мережі	
		Точність	Похибка	Точність	Похибка
1	CIFAR-10	90.6 %	σ = 0.71	84.95 %	σ = 0.45
2	CIFAR-100	85.99 %	σ = 1.99	82.91 %	σ = 0.73
3	GTSRB	100.00 %	σ = 0.00	99.61 %	σ = 0.10
4	HASYv2	88.59 %	σ = 0.45	85.76 %	σ = 0.15
5	MNIST	99.98 %	σ = 0.10	99.58 %	σ = 0.13
6	STL-10	95.43 %	σ = 2.51	75.99 %	σ = 1.98
7	SVHN	99.08 %	σ = 0.07	98.88 %	σ = 0.12

Таблиця 4.3 - Час навчання нової моделі згорткової нейронної мережі на CPU та GPU

Мережа	Tensorflow	Час навчання		
		1 зображення	128 зображень	Епоха
Нова модель	Intel i7-4930K	5 мс	432 мс	386.0 с
Нова модель	GeForce 940MX	4 мс	205 мс	192.2 с

Спираючись на отриані результати обрахунку часу навчання оптимізованої моделі згорткової мережі можна сказати, що було досягнуто значного покращення швидкодії роботи самої мережі у порівнянні із базовою моделлю, детальний розгляд описано у настопному підрозділі 4.3.

4.3 Аналіз отриманих результатів

Таблиця оцінки точності мережі на наборах (таблиця 4.2) продемонструвала, що показник точності значно вищий для 6 наборів із 7, що навчалися на новій покращеній моделі. Де значення точності класифікації на наборі STL-10 було підвищено з 74.80% до 75.99% без змін набору даних взагалі; на наборі HASYv2 значення точності було підвищено з 81% до 85.76 %; на наборі GTSRB відсоток точності було підвищено з 99.46% до 99.61%; на наборі MNIST значення точності покращено з 98.7% до 99.58% ; на наборі SVHN значення точності покращено з 98.41% до 98.88% ; на наборі CIFAR-100 значення точності покращено з 82.82% до 82.591% . В основному таких задовільних результатів було досягнуто шляхом поєднання функції активації ReLU, техніки Dropout, збільшеної кількості навчальних даних та навчання мережі прискорених алгоритмах навчання. Видалення змішення в шарах, близьких до виходу мережі, і збільшення параметрів у шарах, близьких до входу мережі, а також використання підвибірки розміром 3 × 3 замість розміру 2 × 2, поліпшило базову модель архітектури мережі.

Ієрархічний класифікатор не виправдав сподівань бути кращим лише на наборі CIFAR-10 і такий результат можна пояснити так:

- вплив значно невеликої розмірності оригінальної картинки набору 32 x 32;
- не вистачає класів для виконання кластеризації (підхід ієрархічного класифікатору);
- Більшу кількість класів легше класифікувати, якщо кожен новий клас подається з більшою кількістю даних. Це можна пояснити тим, що відмінність об'єкта від фону має аналогічні властивості навіть для різних класів.

Навчання та тестування нової моделі мережі на GPU проходило повільніше, якщо порівнювати з результатами часу базової моделі, пов’язано це з тим, що модифікована архітектура базової моделі отримала

більше фільтрів вхідного шару та збільшений розмір підвибірки максимуму, що призвело до незначного збільшення параметрів в мережі. Також застосування Batch-нормалізації значно збільшило час тренування мережі: чим менший розмір серії використовується (англ. batch), тим більшим стає час навчання для кожної епохи . Висока точність за нижчих розмірів серії була емпірично підтверджена. Однак розмір серії під час проведених експериментів можливо був занадто низьким, тому і отримано значну затримку у часі навчання мережі.

4.4 **Висновок до четвертого розділу**

Згорткова нейронна мережа навчається шляхом прямого та зворотнього проходів і при цьому значення згорткових ядер та матриці зміщення зазнають змін. Тонке налаштування такого гіперпараметра мережі, як нейрону зміщення може значно вплинути на результуючий обрахунок кількості параметрів.

Було реалізовано оновлену архітектуру базової моделі із внесенням таких змін, як збільшення кількості фільтрів вхідного шару, збільшення розмірності шару максимізаційної підвибірки та вилучення нейрону зміщення у завершальних шарах. Вилучення нейронів зміщень у певних шарах не дуже вплинуло на загальну кількість параметрів в мережі, а також застосування фільтру більшої розмірності зменшило число параметрів у шарі під номером 11 (таблиця 4.1) де карта ознак мінімізується до найменшого розміру і мережа на основі набору таких карт виконує задачу класифікації. Також наведено короткий аналіз отриманих даних у підрозділі 4.3 даного розділу.

ВИСНОВКИ

Важливою властивістю нейронних мереж є їх здатність до навчання на основі даних навколишнього середовища та у результаті навчання підвищити свою продуктивність. Посилення продуктивності відбувається з часом у відповідності до певних правил. Загалом навчання нейронної мережі відбувається за допомогою інтерактивного процесу корекції синаптичних ваг нейронів мережі. В ідеальному випадку нейронна мережа отримує знання із навколишнього середовища на кожній ітерації процесу свого навчання. У даній роботі розглянуто особливості основних методів, а також технік навчання нейронних мереж, детально описано складові частини згорткових мереж та їх взаємозв’язок.

Серед запропонованої кількості архітектур згорткових мереж критерії оцінки якості навчання виражаються через показники, а саме такі, як точність, число втрат (обраховує функція втрат) та інші.

Програмна реалізація згорткових нейронних мереж базується на використанні технологій глибинного навчання представлених у вигляді бібліотек та спеціальних пакетів, що дозволяють програмістам проектувати архітектури мереж та налаштовувати параметри.

Головною ідеєю вдосконалення способу класифікації згорткової нейронної мережі є ієрархічний класифікатор, котрий стане універсальним інструментом для обрахунку точності застосованим на багатьох вхідних наборах даних. Основа ієрархії класифікатору лежить у кластеризації поданих класів.

Важливим етапом покращення дії класифікації є також вибір базової моделі згорткової нейронної мережі та обрахунок її показників точності, параметрів в залежності від розміру кроку навчання та використаного алгоритму, аби надалі мати розуміння які саме архітектурні рішення вносити, що досягти оптимізації моделі та покращення роботи. Програмно модель згорткової нейронної мережі було реалізовано за допомогою

бібліотеки Keras мовою Phyton, а для виконання складних математичних обчислень використовувалась бібліотека TensorFlow. Обрану базову модель було навчено на семи вхідних наборах даних.

Помітно значне зменшення часу навчання нейронних мереж із використанням графічних прискорювачів. Графічні процесори (GPU) краще підходять для машинного навчання, ніж центральні процесори (CPU). Технічні особливості GPU допомагають виконати одночасно безліч матричних операцій, які використовуються саме для навчання нейронних мереж. Для прискорення роботи за зменшення обчислюваної складності при навчанні базової моделі згорткової нейронної мережі використовувалась бібліотека cuDNN.

Результатом роботи стала розроблена модернізована модель, на основі базової архітектури, згорткової мережі для розпізнавання об’єкта, що може застосовуватися одразу до семи наборів даних завдяки такому універсальному інструменту, як ієрархічних класифікатор. Нову модель представлено з урахуванням таких змін, щодо базової моделі, як збільшення числа фільтрів вхідного шару, збільшення розміру підвибірки максимуму та вилучення нейронів зміщення у вихідних шарах. Значне підвищення значення відсотку точності розпізнавання майже для усіх запропонованих для тестування наборах даних, а також зменшення числа параметрів мережі було досягнуто завдяки нової моделі архітектури згорткової мережі у поєднанні з ієрархічним класифікатором.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Дж.Ту, Р.Гонсалес. Принципы распознавания образов. М., Мир, 1978.
2. Гонсалес Р., Вудс Р. Цифровая обработка изображений. М.:Техносфера, 2005. - 1072 с.
3. Каллан Р. Основные концепции нейронных сетей = The Essence of Neural Networks First Edition. — 1-ше. — «Вильямс», 2001. — С. 288. — ISBN 5-8459-0210-X. (рос.)
4. Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel: Backpropagation Applied to Handwritten Zip Code Recognition, Neural Computation, 1(4):541-551, Winter 1989.
5. J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” Journal of Machine Learning Research, vol. 13, no. Feb, pp. 281–305, Feb. 2012. [Online]. Available: http://jmlr.csail.mit.edu/papers/volume13/ bergstra12a/bergstra12a.pdf
6. N. Srivastava, G. E. Hinton et al., “Dropout: a simple way to prevent neural networks from overfitting.” Journal of Machine Learning Research, vol. 15, no. 1, pp. 1929–1958, 2014. [Online]. Available: https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf
7. Hiai, Fumio; Lin, Minghua (February 2017). "On an eigenvalue inequality involving the Hadamard product".
8. S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” arXiv preprint arXiv:1502.03167, Feb. 2015. [Online]. Available: https://arxiv.org/abs/1502.03167
9. A. G. Howard, “Some improvements on deep convolutional neural network based image classification,” arXiv preprint arXiv:1312.5402, Dec. 2013. [Online]. Available: https://arxiv.org/abs/1312.5402

10. S. J. Nowlan and G. E. Hinton, “Simplifying neural networks by soft weight-sharing,” Neural computation, vol. 4, no. 4, pp. 473–493, 1992. [Online]. Available:https://www.cs.toronto.edu/~hinton/absps/sunspots.pdf

11. Кудрявцев Л. Д. Математический анализ. — 2-е изд. — М.: Высшая школа, 1973. — Т. 1.

12. T. Xiao, J. Zhang et al., “Error-driven incremental learning in deep convolutional neural network for large-scale image classification,” in International Conference on Multimedia, no. 22. ACM, 2014, pp. 177–186.

13. J. Ortigosa-Hernández, I. Inza, and J. A. Lozano, “Towards competitive classifiers for unbalanced classification problems: A study on the performance scores,” arXiv preprint arXiv:1608.08984, Aug. 2016. [Online]. Available: https://arxiv.org/abs/1608.08984

14. F. Chollet, “Keras,” https://github.com/fchollet/keras, 2015

15. M. Abadi, A. Agarwal et al., “Tensorflow: Large-scale machine learning on heterogeneous distributed systems,” arXiv preprint arXiv:1603.04467, Mar. 2016. [Online]. Available: https://arxiv.org/abs/1603.04467

16. G. Huang, Z. Liu, and K. Q. Weinberger, “Densely connected convolutional networks,” arXiv preprint arXiv:1608.06993, Aug. 2016. [Online]. Available: https://arxiv.org/abs/1608.06993v1

17. P. Sermanet and Y. LeCun, “Traffic sign recognition with multi-scale convolutional networks,” in International Joint Conference on Neural Networks (IJCNN), Jul. 2011, pp. 2809–2813. [Online]. Available: http://ieeexplore.ieee.org/document/6033589/

18. M. Thoma, “The HASYv2 dataset,” arXiv preprint arXiv:1701.08380, Jan. 2017. [Online]. Available: https://arxiv.org/abs/1701.08380

19. J. Zhao, M. Mathieu et al., “Stacked what-where auto-encoders,” arXiv preprint arXiv:1506.02351, Jun. 2015. [Online]. Available: https: //arxiv.org/abs/1506.02351v1

20. G. Huang, Z. Liu, and K. Q. Weinberger, “Densely connected convolutional networks,” arXiv preprint arXiv:1608.06993, Aug. 2016. [Online]. Available: https://arxiv.org/abs/1608.06993v1

21. M. Hardt and T. Ma, “Identity matters in deep learning,” arXiv preprint arXiv:1611.04231, Nov. 2016. [Online]. Available: https: //arxiv.org/abs/1611.04231

22. S. Chetlur, C. Woolley et al., “cuDNN: Efficient primitives for deep learning,” arXiv preprint arXiv:1410.0759, Oct. 2014. [Online]. Available: https://arxiv.org/abs/1410.0759

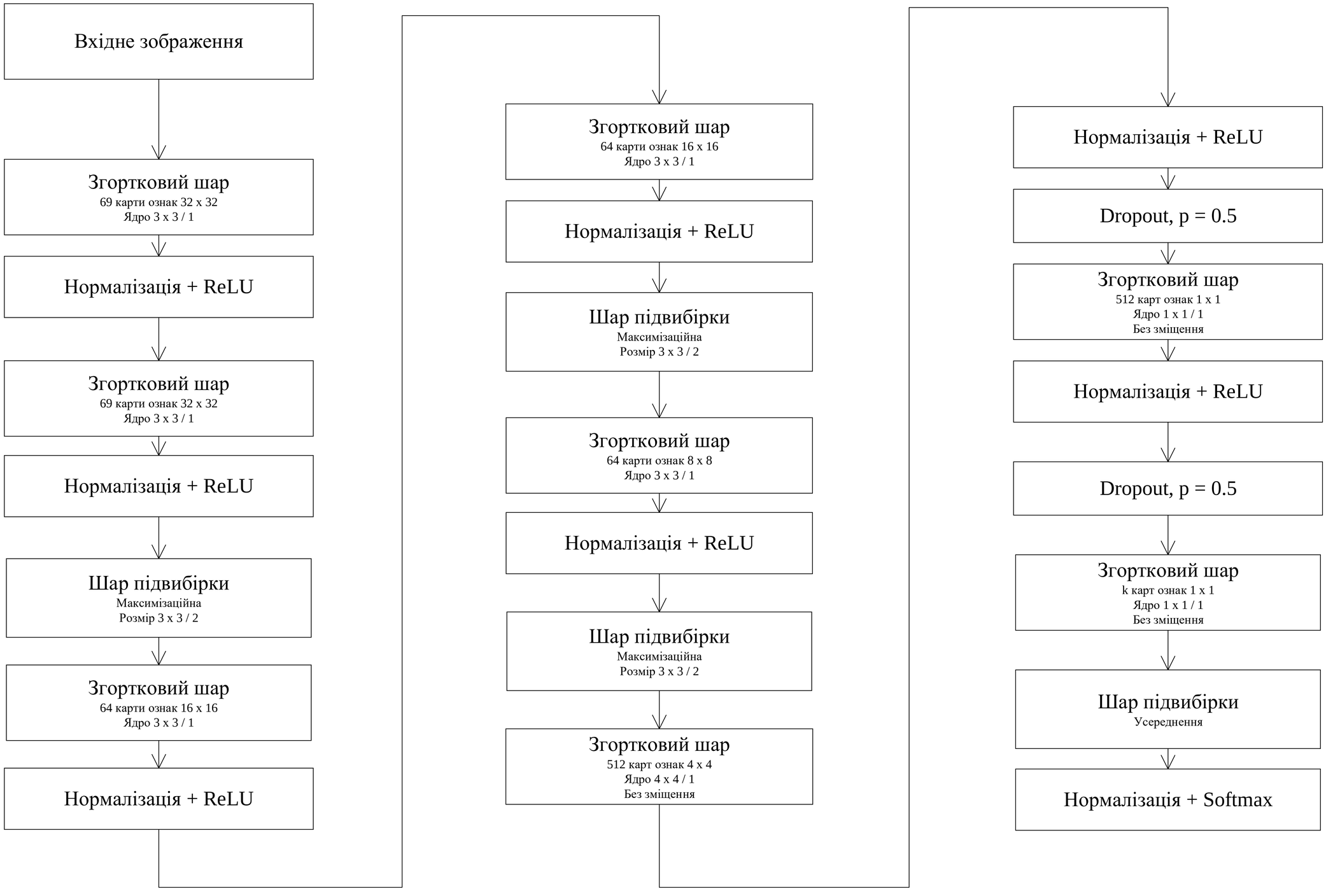
23. N. McLaughlin, J. M. D. Rincon, and P. Miller, “Data-augmentation for reducing dataset bias in person re-identification,” in International Conference on Advanced Video and Signal Based Surveillance (AVSS), no. 12, Aug. 2015, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/ 7301739/

24. Тарасенко-Клятченко О.В., Буц В.В. стаття «Організація багатоетапного методу навчання згорткової нейронної мережі». Журнал «Інтернаука» випуск 6 (березень 2018). [Інренет-ресурс] Посилання: https://www.inter-nauka.com/issues/2018/6/3624/

Додаток 1

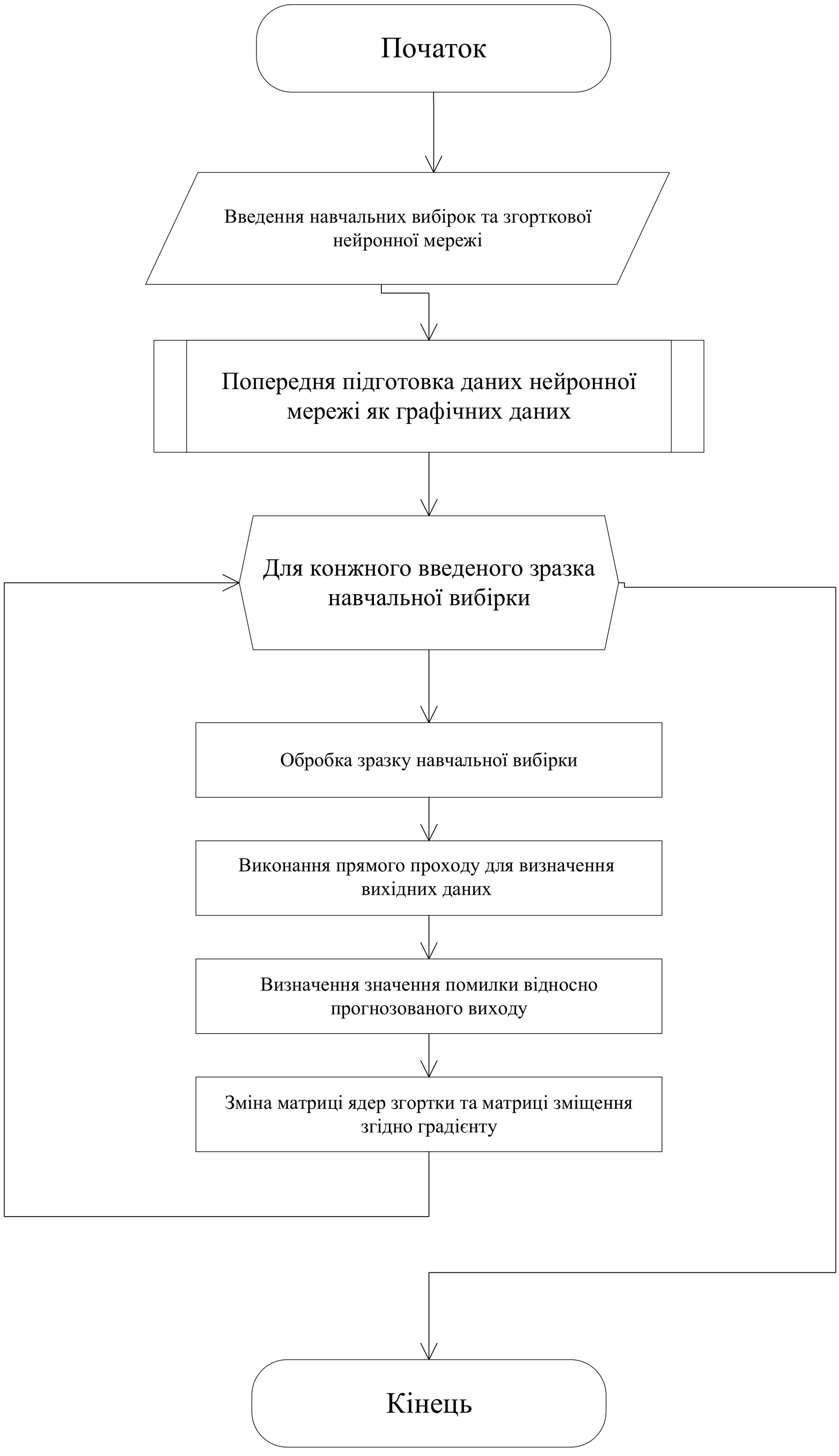
Копії графічних матеріалів

Структурна схема архітектури мережі. Нова модель



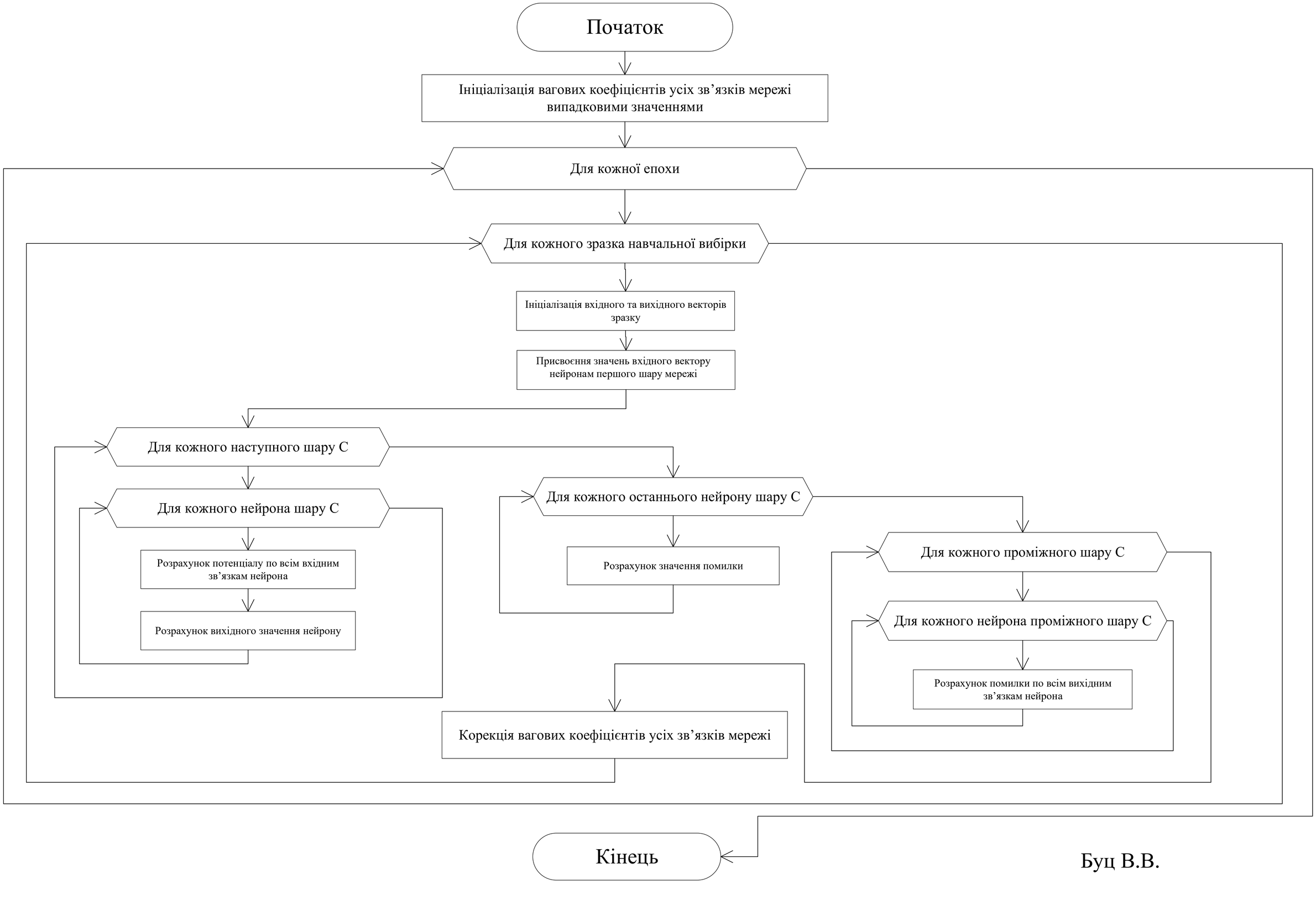
Буц В.В.

Схема алгоритму навчання мережі на графічних прискорювачах



Буц В.В.

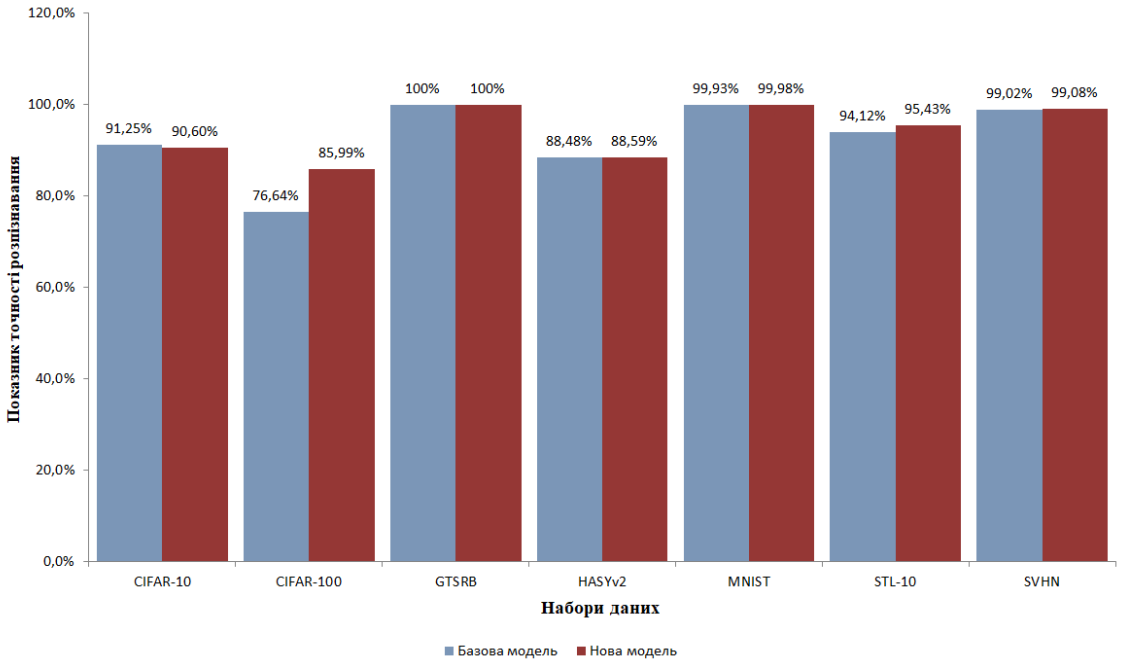
Схема алгоритму навчання мережі методом звортнього поширення помилки



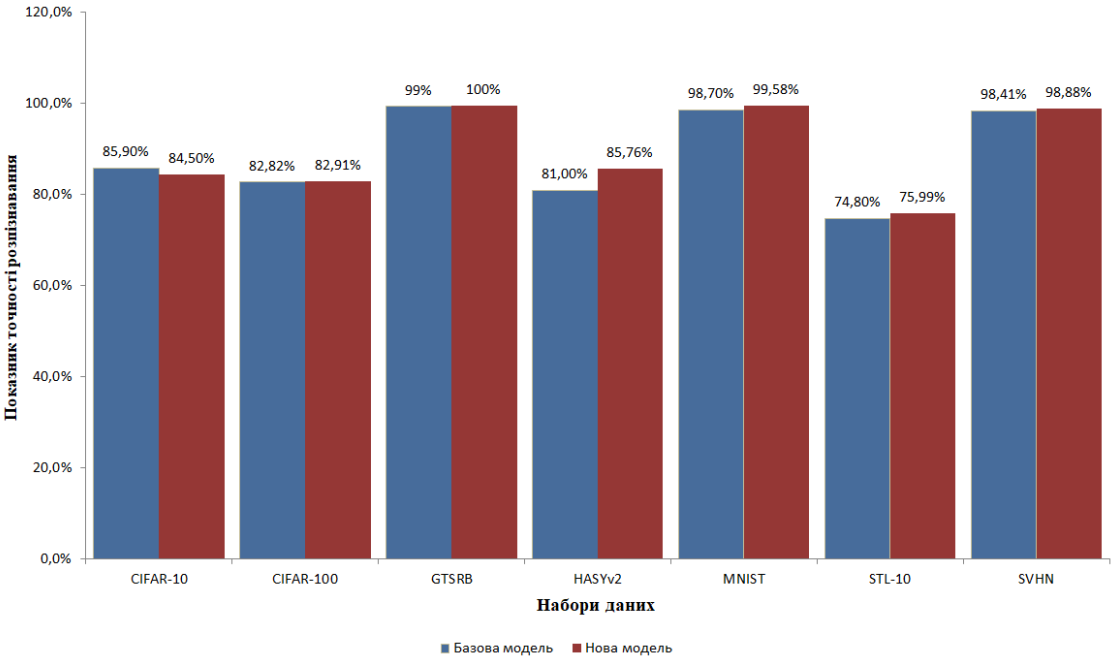
Буц В.В.

Демонтраційні гістограми результатів роботи згорткової нейронної мережі

Показник точності розпізнавання базової та нової моделей згорткових нейронних мереж при навчанні



Показник точності розпізнавання базової та нової моделей згорткових нейронних мереж при тестуванні



Буц В.В.

Демонтраційні таблиці результатів роботи згорткової нейронної мережі

Показник точності розпізнавання базової та нової моделей згорткових нейронних мереж при різних наборах даних

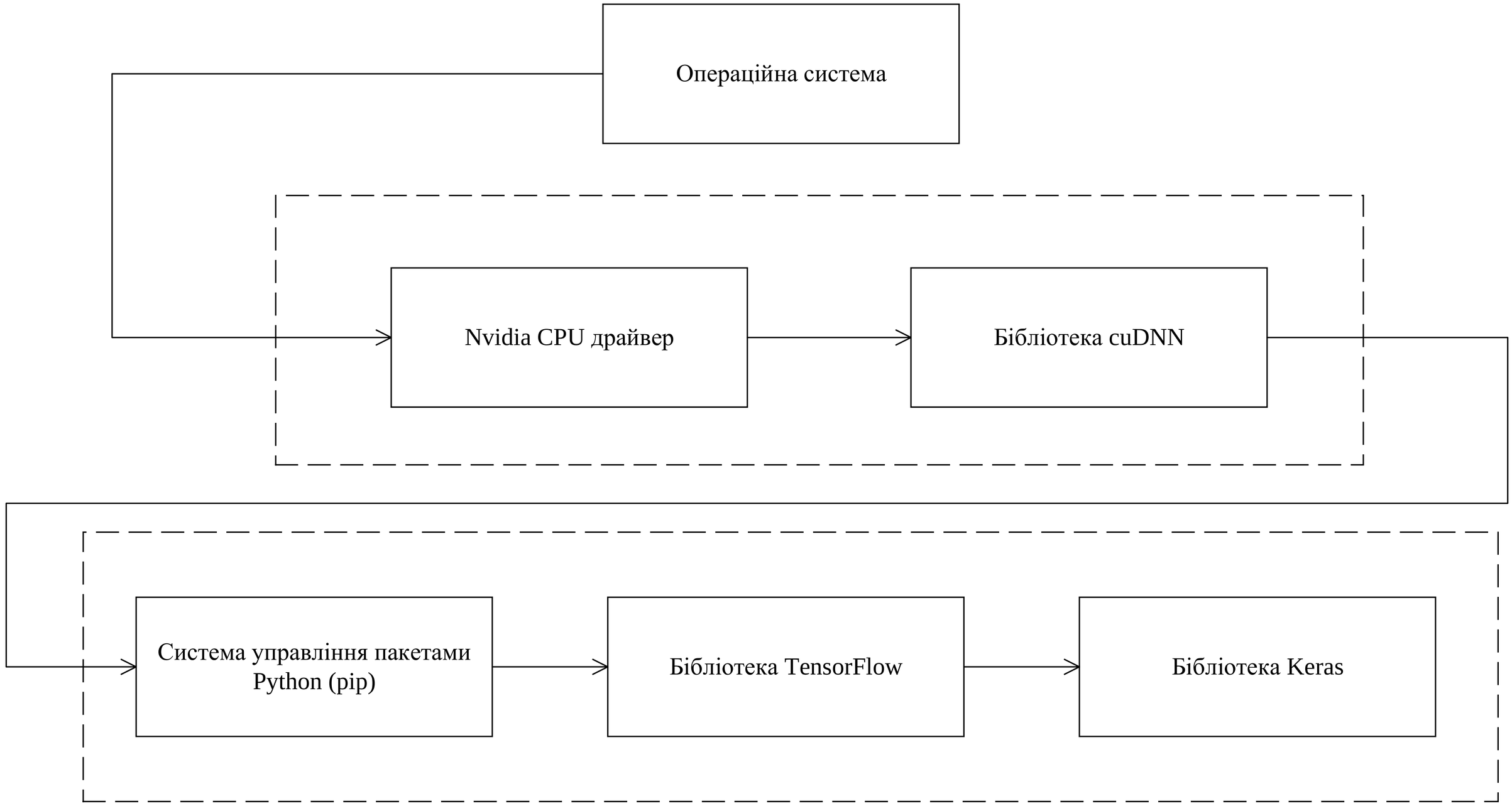
N	Набір даних	Навчання моделі				Тестування моделі			
		Базова модель		Нова модель		Базова модель		Нова модель	
		Точність	Похибка	Точність	Похибка	Точність	Похибка	Точність	Похибка
1	CIFAR-10	91.25 %	σ = 1.10	90.6 %	σ = 0.71	85.90%	σ = 0.87	84.95 %	σ = 0.45
2	CIFAR-100	76.64%	σ = 1.41	85.99 %	σ = 1.99	82.82%	σ = 0.55	82.91 %	σ = 0.73
3	GTSRB	100%	σ = 0.01	100.00 %	σ = 0.00	99.46%	σ = 0.11	99.61 %	σ = 0.10
4	HASYv2	88.48 %	σ = 0.41	88.59 %	σ = 0.45	81.00%	σ = 0.10	85.76 %	σ = 0.15
5	MNIST	99.93 %	σ = 0.07	99.98 %	σ = 0.10	98.70%	σ = 0.06	99.58 %	σ = 0.13
6	STL-10	94.12 %	σ = 0.87	95.43 %	σ = 2.51	74.80%	σ = 0.34	75.99 %	σ = 1.98
7	SVHN	99.02 %	σ = 0.07	99.08 %	σ = 0.07	98.41%	σ = 0.10	98.88 %	σ = 0.12

Час навчання нової моделі нейронної мережі на CPU та GPU

Мережа	Tensorflow	Час навчання		
		1 зображення	128 зображень	Епоха
Нова модель	Intel i7-4930K	5 мс	432 мс	386.0 с
Нова модель	GeForce 940MX	4 мс	205 мс	192.2 с

Буц В.В.

Структурна схема взаємозв’язків програмного забезпечення для реалізації глибинного навчання нейронних мереж



Буц В.В.

Додаток 2

Копії публікацій по темі магистерської дисертації

УДК 004.932.72

К.т.н, доцент Тарасенко-Клятченко О.В., магістрант Буц В.В.

Національний технічний університету України
«Київський політехнічний інститут імені Ігоря Сікорського»

СПОСОБИ ОРГАНІЗАЦІЇ ЗАСОБІВ НЕЙРОМЕРЕЖЕВОГО РОЗПІЗНАВАННЯ ОБ'ЄКТА НА ЗОБРАЖЕННІ З ВИКОРИСТАННЯМ ГРАФІЧНИХ ПРИСКОРЮВАЧІВ

Abstract

Oxana V. Tarasenko-Klyatchenko, assoc. prof., PhD; Viktoriia Buts, student
Facilities for object recognition from image using neural network on GPU

This paper concerns the task of facilities for object recognition from image using neural network in accordance to specified configuration. Most attention in this paper is paid to architecture selection and performance issues. It was important to choose a well adjusted dataset, a feasible architectural design built upon suitable parameters and required hardware resources to fit the training process on GPU.

Вступ

Машинне навчання характерне для багатошарових нейронних мереж, проте використання таких мереж для аналізу зображень та комп'ютерного зору має два основні недоліки:

- виникає потреба у великій кільності характеристик нейронів для навчання самої мережі (наприклад вага нейрону);
- дані для обробки представлені у вигляді одномірного масиву або вектора і через це втрачається топологія, адже пікселі аналізуються лише вертикально.

Метод стохастичного градієнтного спуску [1] дозволяє проводити ефективне навчання розпізнавання зображень глибоких згорткових нейронних мереж. Переваги згорткових мереж над багатошаровими полягають у використанні спільної ваги у згорткових шарах, що означає, що для кожного пікселя шару використовується один і той же фільтр (банк ваги).

Термінологія

Нейронні мережі (англ. neural networks) — біологічно подібні моделі візуальних систем, що застосовуються для ефективного вирішення проблем класифікації об'єктів.

Згорткова нейронна мережа (англ. convolutional neural network, CNN, ConvNet) в машинному навчанні — це такий тип штучної нейронної мережі,

в якому схему з'єднання нейронів запозичено з організації зорової кори тварин, окремі нейрони якої впорядковано таким чином, що вони реагують на області, які покривають зорове поле, частково перекриваючись.

Постановка задачі

Створити нейромережеву систему розпізнавання об’єктів на кольорових зображеннях, використовуючи згорткову нейронну мережу власної архітектури. Запропонувати архітектуру, яка призначатиметься для вирішення поставленної задачі найкращим чином — матиме вищі показники продуктивності, такі як затрати пам’яті, час навчання мережі, кількість параметрів при розпізнавання обєктів у порівнянні з уже існуючими моделями згорткових нейронних мереж. Провести експерименти із навчання мережі використовуючи графічні прискорювачі (анлг. graphic processing unit, GPU).

Підготовка даних

Вхідний набір даних CIFAR-10 [2] включає загальні об'єкти, такі як літаки, автомобілі, птахи тощо. Усі зображення є кольоровими, розміром 32x32 пікселів. Незважаючи на малий розмір зображення співвідношення сторін у перетворенні дотримано оригінальним для того, щоб уникнути спотворення картинки. Значення пікселів для червоного, синього, зеленого каналів, що знаходяться у діапазоні від 0 до 255 було нормалізовано шляхом поділу кожного зі значень на 255. Результатом стане набір коефіцієнтів, що приймають значення у діапазоні від 0 до 1. Початковий формат значень був цілочисленним, його було замінено на формат з плаваючою комою для найкращого виконання цієї нормалізації [3]. Для вирішення проблеми класифікації значення пікселів подаються у вигляді векторів, що відповідають кожному з 10 класів, які трансформуються до бінарної матриці. Аби досягти кращого розпізнавання обєктів у різних положеннях на зображенні набір даних було доповнено модифікованими вхідними зображеннями. Під модифікацією зображень розуміється горизонтальне відображення зображення, вертикальне відображення зображення, зміни висоти або ширини картинки.

Архітектура мережі

Архітектура згорткових мереж складається з різних шарів, що виконують перетворення вхідного об'єму даних. Кожен шар згорткової мережі складається з нейронів, з'єднаних з вузлами попередніх шарів, таким чином, вихід у даному вузлі для шару L є вихідною функцією вузлів у шарі $L-1$. На прикладі п’яти основних шарів даної мережі наведемо архітектуру, що пропонується для розгляду:

1. Згортковий шар (англ. convolutional layer). Його параметри містять набори фільтрів (або ядер) для навчання мережі, які мають невелике рецептивне поле, що простягаються на всю глибину вхідного об'єму. Кожен фільтр здійснює згортку по ширині та висоті вхідного об’єму під час прямого проходу. Виконується обчислення скалярного добутку даних фільтру та входу, і формується двовимірна карта активації цього фільтру. Перші згорткові шари вивчають низькорівневі особливості, такі як краї, лінії та кути. Наступні шари вивчають більш складні особливості (наприклад, частини та моделі).

2. Шари підвибірки (англ. pooling layers). Вони реалізовані нелінійними функціями для реалізації підвибірки, серед яких найпоширенішою є максимізаційна підвибірка. Вона розділяє вхідне зображення на набір прямокутників без перекриття, і для кожної такої підобласті виводить її максимум. Ідея створення цього шару полягає в тому, що якщо ознаку було знайдено, то її точне положення не так важливе, як її положення відносно інших ознак. Функцією підвибіркового шару є поступове скорочення просторового розміру представлення для зменшення об'єму параметрів (наприклад вага нейрону) та обчислень у мережі.

3. Шари активації (англ. activation layers) - функції активації імітують поведінку аксона нейрона, що запускає сигнал при контакті з подразником. Ключовою особливістю згорткових мереж є функція активації ReLU [4].

4. Шари виключення (англ. dropout layers) які генерують випадковий набір активацій, обнуляючи їх значення. Це означає, що мережа повинна надавати правильну класифікацію, навіть якщо деякий набір активацій виключено.

5. Повнозв’язні шари (англ. fully-connected layers) слугують як провідники між шарами, де всі виходи попереднього шару пов’язані з усіма входами повнозв’язних шарів.

На рис. 1 наведено варіант архітектури, що пропонується.

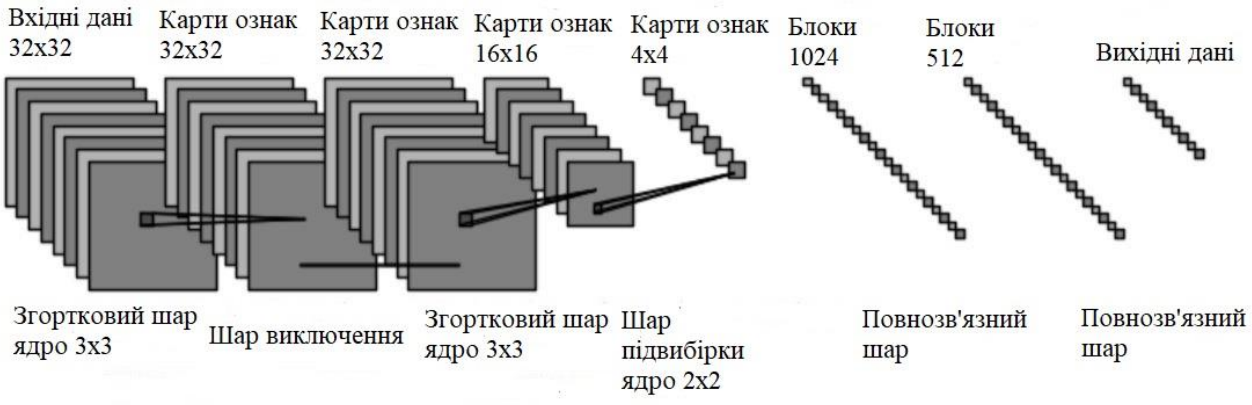


Рис. 1. Архітектура запропонованої згорткової нейроної мережі

Результати експерименту

Для трансляції великої кількості карт ознак у запропонованій архітектурі слід застосувати додаткові більш щільні шари, де використано функції активації ReLU. Функція softmax [4] на вихідному рівні аналізує значення для вибору певного класу. В результаті реалізації запропонованої моделі згорткова мережа була навчена за допомогою оптимізатору ефективного стохастичного градієнтного спуску і функції логарифмічної втрати (1) , що представлена нище [4].

$$L\,(X,Y) = \; -\frac{1}{n}\sum_{n}^{i=1} y^i \ln a\left(x^i\right) + \left(1-y^i\right)\ln(1-a(x^i)) \qquad (1)$$

Де $X = \{x(1), \dots, x(n)\}$ — це набір вхідних прикладів для навчання у вхідному наборі. $Y = \{y(1), \dots, y(n)\}$ — це відповідний набір міток для вхідного набору. Функція $a(x)$ — вихідні значення нейронної мережі з вхідним значенням x.

В результаті експерименту навчання запропонованої моделі на п’ятидесяти епохах закінчилося приблизно за 2 години, що дало замогу обрахувати точність класифікації та значення втрат функціональності в навчальному наборі даних через кожну епоху, де найкраща точність класифікації складала 94%, а значення функції втрат склали 0,2. Навчання проводилося на персональному комп’ютері з такими характеристиками: процесор Intel Core i3 (2.1 GHz), 4GB RAM, відеоадаптер NVIDIA GeForce 920M GPU (954 MHz). Навчальна модель була розроблена в Keras, бібліотеці Python для глибокого навчання, використовуючи брандмауер Theano та графічні прискорювачі (англ. graphic processing unit, GPU). Результати навчань запропонованої згорткової мережі на кольорових зображеннях розміром 32x32 наведено у таблиці 1.

Таблиця 1

Результати навчань запропонованої згорткової мережі на кольорових зображеннях розміром 32x32

Назва шару	Ядра /Нейрони	Функція активації	Канали	Вихідний розмір	Параметри
Згортковий шар 1	3x3	ReLU	32	32x32	890
Шар виключення 1 (20%)			32	32x32	0
Згортковий шар 2	3x3	ReLU	32	32x32	9230
Шар підвибірки 1	2x2		32	16x16	0
Згортковий шар 3	3x3	ReLU	64	16x16	18501
Шар виключення 2 (20%)			64	16x16	0
Згортковий шар 4	3x3	ReLU	64	16x16	36898
Шар підвибірки 2	2x2		64	8x8	0
Згортковий шар 5	3x3	ReLU	128	8x8	73802
Шар виключення 3 (20%)			128	8x8	0
Згортковий шар 6	3x3	ReLU	128	8x8	146765
Шар підвибірки 3	2x2		128	4x4	0
Розгалудження	2048				0
Шар виключення 4 (20%)	2048				0
Повноз’язний шар 1	1024	ReLU			2086859
Шар виключення 5 (20%)	1024				0
Повноз’язний шар 2	512	ReLU			534500
Шар виключення 6 (20%)	512				0
Повноз’язний шар 3	10	softmax			5130

Висновки

Згорткова нейромережа розпізнавання об’єктів на кольорових зображеннях запропонованої архітектури, завдяки застосуванню точних налаштувань та додаткових (спеціальних) перетворень вхідних даних, досягла низького рівня помилок та високого рівня точності класифікації у порівнянні із іншими засобами розпізнавання [5]. Така мережа дозволяє суттєво зменшити витрати ресурсів пам’яті, проста у реалізації та має покращений метод класифікації за рахунок нормалізації вхідних данних та застосуванню додаткових шарів обробки даних.

Література

1. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016. Mode of access : http://www.deeplearningbook.org.

2. Сховище зображень для навчання нейронних мереж [Електронний ресурс]. «The CIFAR-10 dataset». Режим доступу: https://www.cs.toronto.edu/~kriz/cifar.html.

3. *Brownlee, J.* (2016, July 1). Object Recognition with Convolutional Neural Networks in the Keras Deep Learning Library. From Machine Learning Mastery: http://machinelearn-ingmastery.com/object-recognition-convolutional-neural-networks-keras-deeplearning-library.

4. *Eleonora Cagli1*, *Cecile Dumas*, *Emmanuel Prouff*. Convolutional Neural Networks with Data Augmentation against Jitter-Based Countermeasure International on Conference on Multimedia, pages 6-7. ACM, 2014.
5. Rodrigo Benenson. [Electronic resource] // What is the class of this image?, Last updated on 2016-02-22: – Mode of access: WWW.URL: http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html#43494641522d3130.

УДК 004.85

Тарасенко-Клятченко Оксана Володимирівна
кандидат технічних наук, доцент
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Тарасенко-Клятченко Оксана Владимировна
кандидат технических наук, доцент
Национальный технический университет Украины
«Киевский политехнический институт имени Игоря Сикорского»
Tarasenko-Klyatchenko Oxana
Candidate of Technical Sciences, Associate Professor
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"

Буц Вікторія Віталіївна
магістрант
Національного технічного університету України
«Київський політехнічний інститут імені Ігоря Сікорського»
Буц Виктория Витальевна
магистрант
Национального технического университета Украины
«Киевский политехнический институт имени Игоря Сикорского»
Buts Viktoriia
Graduating Student of the
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"

Організація багатоетапного методу навчання згорткової нейронної мережі
Организация многоэтапного метода обучения сверточной нейронной сети
The multi-stage training method for a convolutional neural network

Анотація. *В даній роботі описано методи та алгоритми навчання нейронних мереж. Також запропоновано організацію поетапного навчання нейронної мережі, на основі адаптивного та генетичного методів, що була успішно застосована до згорткової нейронної мережі для вирішення задачі класифікації об'єкта на зображенні.*

Ключові слова: *навчання, згорткова нейронна мережа, генетичний метод навчання, класифікація.*

Аннотация. *В данной работе описаны методы и алгоритмы обучения нейронных сетей. Также предложено организацию поэтапного обучения нейронной сети на основе адаптивного и генетического методов, которая была успешно применена к сверточной нейронной сети для решения задачи классификации объекта на изображении.*

Ключевые слова: *обучение, сверточная нейронная сеть, генетический метод обучения, классификация.*

Summary. *This paper describes methods and algorithms for training neural networks. Also, the organization of multi-stage training of the neural network, based on adaptive and genetic methods, which has been successfully applied to the convergent neural network to solve the problem of object classification*.

Key words: *training, convolutional neural network, genetic method, classification.*

Вступ. Машинне навчання є головною складовою теорії штучного інтелекту, глибокою математичною дисципліною, до якої відносяться науки математична статистика та теорія ймовірностей. Зазначимо два типи машинного навчання, що активно використовуються: індуктивне навчання та дедуктивне навчання. Індуктивне навчання базується на обробці емпіричних даних, а дедуктивне – на формалізації отриманих знань та подальшого їх упорядкування. Прийнято вважати, що область експертних систем включає

дедуктивное навчання, тому і використовується в теорії і практиці машинного навчання загалом.

Причиною виникнення розділу машинного навчання, як наукової галузі, стало поділ науки про нейронні мережі, що досліджує методи навчання та топології архітектур мереж, в рамках науки про штучний інтелект. Таким чином, вирішальним фактором задачі оптимізації стає вибір методу навчання у реалізації мережі не залежно від її типу.

Згорткова нейронна мережа. Успішним у застосуванні до вирішення задач комп'ютерного зору можна назвати такий тип біологічно подібних моделей візуальних систем, як згорткова нейронна мережа. Переваги згорткових мереж над багатошаровими полягають у використанні спільної ваги у згорткових шарах, що означає, що для кожного пікселя шару використовується один і той же фільтр (банк ваги). Кожен фільтр здійснює згортку по ширині та висоті вхідного об'єму під час прямого проходу. Виконується обчислення скалярного добутку даних фільтру та входу, і формується двовимірна карта активації цього фільтру [1]. Архітектура згорткових мереж побудована з різних шарів, не тільки згорткового, що виконують перетворення вхідного об'єму даних. Кожен шар згорткової мережі складається з нейронів, з'єднаних з вузлами попередніх шарів. Важливи і наступним шаром є шари максимізаціїної підвибірки. Він реалізований за допомогою нелінійних функцій для реалізації операції підвибірки. Операція підвибірки розділяє вхідне зображення на набір прямокутників без перекриття, і для кожної такої підобласті виводить її максимум [1]. Допоміжними можна назвати шар активації, що реалізує функцію активації, імітуючи поведінку аксона нейрона, що запускає сигнал при контакті з подразником та шар виключання який генерує випадковий набір активацій, обнуляючи їх значення. Після чередування вищевказаних шарів, кількість яких залежить від архітектури мережі, вихідним шаром є шар, що утворюють повнозв'язні шари – класифікатор [1].

Особливості вхідного набору даних. Набору даних для навчаня згорткової нейронної мережі варто обирати згідно з поставленої задачі. У нашому випадку згорткова нейронна мережа повинна навчитися розпізнавати об’єкти на кольорових зображеннях. Вхідний набір даних CIFAR-10 [2] включає загальні об'єкти, такі як літаки, автомобілі, птахи тощо. Усі зображення є кольоровими, розміром 32x32 пікселів. Незважаючи на малий розмір зображення співвідношення сторін у перетворенні потрібно дотримати оригінальним для того, щоб уникнути спотворення картинки. Значення пікселів для червоного, синього, зеленого каналів, що знаходяться у діапазоні від 0 до 255 було варто нормалізовувати шляхом поділу кожного зі значень на 255. Результатом стане набір коефіцієнтів, що приймають значення у діапазоні від 0 до 1. Початковий формат значень був цілочисленним, його треба замінити на формат з плаваючою комою для найкращого виконання цієї нормалізації. Для вирішення проблеми класифікації значення пікселів подаються у вигляді векторів, що відповідають кожному з 10 класів, які трансформуються до бінарної матриці. Аби досягти кращого розпізнавання обєктів у різних положеннях на зображенні набір даних доповнено модифікованими вхідними зображеннями. Під модифікацією зображень розуміється горизонтальне відображення зображення, вертикальне відображення зображення, зміни висоти або ширини картинки.

Методи та алгоритми навчання нейронних мереж. Вагомим аргументом на користь застосування градієнтних методів до навчання мереж є можливість вираження цільової функції через диференціальну функцію. Метод стохастичного градієнтного спуску [3] дозволяє проводити ефективне навчання розпізнавання зображень саме для глибоких згорткових нейронних мереж.

А от для вирішення задач оптимізації з певними критеріями використовуються методи із зістосуванням генетичних алгоритмів. Симбіоз в

одному алгоритмі навчання градієнтних і генетичних методів може значно перевищити очікуваний результат.

Огляд загального аналізу градієнтних методів навчання глибоких нейронних мереж приводить до твердження, що будь-який з існуючих методів, можна представити як окремий випадок адаптивного алгоритму.

Загальна формула зміни ваги нейрону:

$$\vec{w}_{k+1} = \vec{w}_k + step_k \vec{p}_k, \quad (1)$$

де $\overrightarrow{p_k}$ - напрямок руху, $step_k$ - розмір кроку на k –й ітерації.

Розрахунок напрямку руху має наступний вигляд:

$$\vec{p}_k = \vec{g}_k + \sum_{i=1}^{\min(k-1,m)} \beta_i \cdot \vec{g}_{k-i}, \quad (2)$$

де $\vec{p}_k$ – напрямок руху, $\vec{g}_k$ – напрямок антиградієнта на ітерації, β – коефіцієнт визначення ваги градієнта, m – кількість градієнтів, k – номер поточної ітерації.

Розглянемо адаптивний алгоритм мінімізіції функції помилки [4].

1. Спочатку встановлюємо початкове значення для параметрів w_0, p_0 та $step_0$.
2. З навчальної вибірки обраємо черговий вектор та подаємо його на вхід до мережі.
3. Визначаємо напрямок руху по формулі (2).
4. Визначаємо критерій зупинки, наприклад середньоквадратичку помилку.
5. При виконанні умови зупинки переходимо до кроку 6, інакше – до кроку 2.
6. Кінець алгоритму.

Результатом виконання алгоритму є навчена мережа. До основного недоліку даного алгоритму відноситься стан мережі так званий «параліч» мережі, потрапляння в локальні мінімуми, багаторазове представлення всієї навчальної множини.

Генетичний алгоритм [5] є ітераційним, здатним до обчислень в деякій околиці глобального мінімуму. Завдяки цьому алгоритм може застосовуватися в підборі ваг штучного нейрону при навчанні мережі. При навчанні мережі в генетичному алгоритмі використовують наступні визначення : ген (коефіцієнт ваги), хромосома (набір генів, або набір коефіцієнтів ваги), популяція (множина наборів хромосом), епоха (ітерація). Таким чином, до параметрів генетичного алгоритму відносяться розмір популяції, число хромосом для мутацій, ймовірності вибору хромосоми і ймовірність мутації гена. Відповідний підбір параметрів дозволяє виділити генетичний алгоритм з широкого класу алгоритмів.

Основними сутностями, над якими в певному порядку в межах однієї епохи проводяться операції є хромосоми. Операція схрещування - створення з певною ймовірністю (Pc) нової хромосоми з генів двох інших і додавання її до популяції. Операція мутації - зміна з ймовірністю (Pm) значення довільного гена будь-якої хромосоми і додавання її до популяції.

Поетапна організація навчання. Розглянемо поетапно процес навчання згорткової нейронної мережі, що буде заснований на принципах генетичного та адаптивного методів навчання:

Eman 1. Початок. Створення згорткової нейронної мережі з визначеними вхідними коефіцієнтами ваги.

Eman 2. Навчання за адаптивним алгоритмом. Проводиться навчання за адаптивним алгоритмом мінімізіції функції помилки поки не буде досягнуто критерій переходу до генетичного методу навчання.

Eman 3. Додавання навченої мережі до популяції. За генетичним методом навчання створити популяцію обсягом N – 1 особин. В першу популяцію додати вже навчену за адаптивним алгоритмом мережу.

Eman 4. Схрещування. Виконується схрещуванняння особи з ймовірністю вибору пари P_c. Від кожної пари отримаємо S нащадків.

Eman 5. Вибір нащадків. Вибір з нової популяції N найкращих нащадків.

Eman 6. Вибір представника. Якщо кращий представник особини відповідає заданій якості навчання, переходимо до етапу 9.

Eman 7. Мутація. Проводиться мутація для особин, вибраних з ймовірністю P_m. Для кожного гена вибраної особини проведемо мутацію.

Eman 8. Вибір представника. Якщо кращий представник особини відповідає заданій якості навчання, перейти до етапу 9. У випадку невідповідності повернутися до етапу 4.

Eman 9. Завершення. Мережа навчена (обраний кращий представник особини).

Описані вище етапи навчання є узагальненими і можуть застосовуватися до різних топологій мереж.

Висновки. В якості задачі для навчання обрано задачу класифікації об’єктів на кольорових зображеннях попередньо підготовлених для подання на вхід мережі. Навчальна вибірка складалася з 5000 прикладів для кожного з 10 класів вхідного набору CIFAR-10.

Проведені есперименти з навчання мережі топології згорткової нейронної мережі показали, що поетапне навчання моделі на 50 епохах закінчилося приблизно за 2 години, з витратами часу на обчислення однієї єпохи у 145 секунд. У свою чергу навчання популярним градієнтним методом виконувалося майже у 3,5 години. Отже застосування поетапного навчання, як методу, найкращим чином підходить для задачі пришвидшення часу навчання. Поетапне навчання дало замогу обрахувати точність класифікації та функцію втрати, значення якої становило 0.7 в навчальному наборі даних через кожну епоху.

Література

1. Згорткова нейронна мережа [Електронний ресурс].. Режим доступу: http://uk.wikipedia.org/wiki/Згорткова_нейронна_мережа.
2. Сховище зображень для навчання нейронних мереж [Електронний ресурс]. «The CIFAR-10 dataset». Режим доступу: https://www.cs.toronto.edu/~kriz/cifar.html.
3. Іван Гудфелов, Йоша Бенгіо, Арон Коурвілле. «Машинне навчання» MIT Press, 2016. Режим доступу: http://www.deeplearningbook.org.
4. Ліла В.Б. Алгоритм та программна реализація адаптивного метода навчання штучних нейронних мереж// Инженерний вестн. Дона, 2012.
5. Девід Е. Голберг «Дзен та мистецтво генетичного алгоритму». 3rd International Conference on Genetic Algorithms, pp. 80-85.

Додаток 3

Фрагменти програмного коду

Файл analize_model.py

```
import logging
import sys
import keras.layers.convolutional
import keras.layers.normalization
from keras import backend as K
from keras.models import load_model
import seaborn as sns
sns.set_style("whitegrid")
import matplotlib.pyplot as plt
import operator
from functools import reduce
import numpy as np
import yaml
import imp
from run_training import make_paths_absolute
import os
import pprint
import scipy.misc
import scipy.stats
import glob
logging.basicConfig(format='%(asctime)s %(levelname)s %(message)s',
                    level=logging.DEBUG,
                    stream=sys.stdout)
from msthesis_utils import make_mosaic
def get_activations(model, layer_index, X_batch):
    """Get activation of one layer for one input."""
    get_activations = K.function([model.layers[0].input, K.learning_phase()],
                                 [model.layers[layer_index].output, ])
    activations = get_activations([X_batch, 0])
    return activations
def show_conv_act_distrib(model, X, show_feature_maps=False):
    """Show the distribution of convolutional layers for one input."""
    X_train = np.array([X])
    layer_index = 0
    activations_by_layer = []
    labels = []
    for layer_index in range(len(model.layers)):
        act = get_activations(model, layer_index, X_train)[0]
        if show_feature_maps:
            scipy.misc.imshow(X_train[0])
            mosaik = make_mosaic(act[0], 8, 4)
            scipy.misc.imshow(mosaik)
        data = act[0].flatten()
        if isinstance(model.layers[layer_index],
                      keras.layers.convolutional.Conv2D):
            print("\tlayer {}: len(data)={}".format(layer_index, len(data)))
```

```
                activations_by_layer.append(data)
                labels.append(layer_index)
            layer_index += 1

        # Activations
        for label, fw in enumerate(activations_by_layer):
            print("99% filter weight interval of layer {}: [{:.2f}, {:.2f}]"
                  .format(label, np.percentile(fw, 0.5), np.percentile(fw, 99.5)))

        f, ax1 = plt.subplots(1, 1)
        p = sns.violinplot(data=activations_by_layer, orient="v",
                           palette=sns.color_palette(palette="RdBu", n_colors=1),
                           ax=ax1)
        p.tick_params(labelsize=16)
        ax1.set_xticklabels(labels)
        ax1.set_title('Convolution activations by layer')
        sns.plt.show()
    def show_conv_weight_dist(model, small_thres=10**-6):
        """Show the distribution of conv weights of model."""
        layer_index = 0
        filter_weights = []
        bias_weights = []
        filter_weight_ranges = []
        labels = []
        for layer in model.layers:
            if not isinstance(layer, keras.layers.convolutional.Conv2D):
                layer_index += 1
                continue
            weights = layer.get_weights()

            # Filter
            print("{}: {} filter weights in {}th layer"
                  .format(weights[0].shape,
                          reduce(operator.mul, weights[0].shape, 1),
                          layer_index))

            # Measure distribution (replacement by 1x1 filters)
            ranges = []
            w, h, range_i, range_j = weights[0].shape
            if w > 1 or h > 1:
                for i in range(range_i):  # one filter, but different channel
                    for j in range(range_j):  # different filters
                        elements = weights[0][:, :, i, j].flatten()
                        ranges.append(elements.max() - elements.min())
                ranges = np.array(ranges)
                filter_weight_ranges.append(ranges)
```

```
    # measure distribution
    data = weights[0].flatten()
    labels.append(layer_index)
    filter_weights.append(data)
    data_small = np.array([el for el in data if abs(el) < small_thres])
    print("< {}: {}".format(small_thres, len(data_small)))
    # Bias
    if len(weights) > 1:
        print("{}: {} bias weights in {}th layer"
              .format(weights[1].shape,
                      reduce(operator.mul, weights[1].shape, 1),
                      layer_index))
        data = weights[1].flatten()
        bias_weights.append(data)
        data_small = np.array([el for el in data if abs(el) < small_thres])
        print("< {}: {}".format(small_thres, len(data_small)))
    else:
        print("No bias in layer {}".format(layer_index))
    layer_index += 1

labels = [1, 3, 5, 7, 9, 11, 13, 15]  # baseline

# Filter weight ranges
f, ax1 = plt.subplots(1, 1)
p = sns.violinplot(data=filter_weight_ranges, orient="v",
                   palette=sns.color_palette(palette="RdBu", n_colors=1),
                   ax=ax1)
p.tick_params(labelsize=16)
p.set_xlabel('Layer', fontsize=20)
p.set_ylabel('Weight range', fontsize=20)
ax1.set_xticklabels(labels)
ax1.set_title('Filter weight ranges by layer')
sns.plt.show()

# Filter weights
for label, fw in enumerate(filter_weights):
    print("99% filter weight interval of layer {}: [{:.2f}, {:.2f}]"
          .format(label, np.percentile(fw, 0.5), np.percentile(fw, 99.5)))

f, ax1 = plt.subplots(1, 1)
p = sns.violinplot(data=filter_weights, orient="v",
                   palette=sns.color_palette(palette="RdBu", n_colors=1),
                   ax=ax1)
p.tick_params(labelsize=16)
p.set_xlabel('Layer', fontsize=20)
ax1.set_xticklabels(labels)
ax1.set_title('Filter weight distribution by layer')
```

```
    sns.plt.show()
    # Bias weights
    for label, fw in enumerate(bias_weights):
        print("99% bias weight interval of layer {}: [{:.2f}, {:.2f}]"
              .format(label, np.percentile(fw, 0.5), np.percentile(fw, 99.5)))

    f, ax1 = plt.subplots(1, 1)
    p = sns.violinplot(data=bias_weights[:], orient="v",
                       palette=sns.color_palette(palette="RdBu", n_colors=1),
                       ax=ax1)
    p.tick_params(labelsize=16)
    p.set_xlabel('Layer', fontsize=20)
    ax1.set_xticklabels(labels[:])
    ax1.set_title('Bias weight distribution by layer')
    sns.plt.show()


def show_batchnorm_weight_dist(model):
    """Show the distribution of batch norm weighs for one model."""
    # analyze
    gamma_weights = []
    beta_weights = []
    layer_index = 0
    labels = []
    for layer in model.layers:
        if isinstance(layer, keras.layers.normalization.BatchNormalization):
            labels.append(layer_index)
            weights = layer.get_weights()
            data = weights[0].flatten()
            gamma_weights.append(data)
            data = weights[1].flatten()
            beta_weights.append(data)
        layer_index += 1

    labels = [2, 4, 6, 8, 10, 12, 14, 16]  # baseline

    # Gamma weights
    if len(gamma_weights) > 0:
        for label, fw in zip(labels, gamma_weights):
            print("99% gamma interval of layer {}: [{:.2f}, {:.2f}]"
                  .format(label,
                          np.percentile(fw, 0.5),
                          np.percentile(fw, 99.5)))

        f, ax1 = plt.subplots(1, 1)
        p = sns.violinplot(data=gamma_weights, orient="v",
                           palette=sns.color_palette(palette="RdBu",
```

```
                                                    n_colors=1),
                        ax=ax1)
        p.tick_params(labelsize=16)
        p.set_xlabel('Layer', fontsize=20)
        ax1.set_xticklabels(labels)
        ax1.set_title('Gamma distribution by layer')
        sns.plt.show()

    # beta weights
    if len(beta_weights) > 0:
        for label, fw in zip(labels, beta_weights):
            print("99% beta interval of layer {}: [{:.2f}, {:.2f}]"
                  .format(label,
                          np.percentile(fw, 0.5),
                          np.percentile(fw, 99.5)))

        f, ax1 = plt.subplots(1, 1)
        p = sns.violinplot(data=beta_weights, orient="v",
                           palette=sns.color_palette(palette="RdBu",
                                                     n_colors=1),
                           ax=ax1)
        p.tick_params(labelsize=16)
        p.set_xlabel('Layer', fontsize=20)
        ax1.set_xticklabels(labels)
        ax1.set_title('Beta distribution by layer')
        sns.plt.show()


def main(config, data_module, model_path, image_fname):
    """Run analysis."""
    model = load_model(model_path)

    print("## Activation analyzation")
    if image_fname is not None:
        image = scipy.misc.imread(image_fname)
    else:
        data = data_module.load_data(config)
        X_train = data['x_train']
        X_test = data['x_test']
        # y_train = data['y_train']
        # y_test = data['y_test']

        X_train = data_module.preprocess(X_train)
        X_test = data_module.preprocess(X_test)
        image = X_train[0]

    show_conv_act_distrib(model, image)
```

```
    print("## Weight analyzation")
    show_conv_weight_dist(model)
    print("## BN analyzation")
    show_batchnorm_weight_dist(model)


def get_parser():
    """Get parser object for script xy.py."""
    from argparse import ArgumentParser, ArgumentDefaultsHelpFormatter
    parser = ArgumentParser(description=__doc__,
                            formatter_class=ArgumentDefaultsHelpFormatter)
    parser.add_argument("-f", "--file",
                        dest="filename",
                        help="Experiment yaml file",
                        required=True,
                        metavar="FILE")
    parser.add_argument("--model",
                        dest="model_path",
                        help="Model h5 file",
                        metavar="FILE")
    parser.add_argument("--image",
                        dest="image_fname",
                        help="A single image",
                        metavar="FILE")
    return parser


if __name__ == "__main__":
    args = get_parser().parse_args()
    # Read YAML experiment definition file
    with open(args.filename, 'r') as stream:
        config = yaml.load(stream)
    # Make paths absolute
    config = make_paths_absolute(os.path.dirname(args.filename),
                                 config)

    # Print experiment file
    pp = pprint.PrettyPrinter(indent=4)
    pp.pprint(config)

    # Load data module
    dpath = config['dataset']['script_path']
    sys.path.insert(1, os.path.dirname(dpath))
    data = imp.load_source('data', config['dataset']['script_path'])

    # Load model
```

```
        if args.model_path is not None:
            model_path = args.model_path
        else:
            artifacts_path = config['train']['artifacts_path']
            model_path = os.path.basename(config['train']['artifacts_path'])
            model_paths = glob.glob("{}/*.h5".format(artifacts_path))
            model_paths = [m for m in model_paths if "_chk.h5" not in m]
            model_path = model_paths[0]
        logging.info("Take {}".format(model_path))
        main(config, data, model_path, args.image_fname)
```

Файл adam_keras.py

```
from keras.optimizers import Adam

def get_optimizer(config):
    lr = config['optimizer']['initial_lr']
    optimizer = Adam(lr=lr)  # Using Adam instead of SGD to speed up training
    return optimizer
```

Файл sgd.py

```
from keras.optimizers import SGD

def get_optimizer(config):
    lr = config['optimizer']['initial_lr']
    optimizer = SGD(lr=lr)  # Using Adam instead of SGD to speed up training
    return optimizer
```