Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційних радіотехнологій та технічного захисту інформацій

Кафедра Радіотехнологій інформаційно-комунікаційних систем

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти другий (магістерський)

ГЮІК.ХХХХХХ.000ПЗ
(позначення документа)

РОЗРОБКА ТА ДОСЛІДЖЕННЯ СИСТЕМИ РОЗПІЗНАВАННЯ ТРИВИМІРНИХ ОБ'ЄКТІВ ЗА ХМАРАМИ ЗАДАНИХ КОНТРОЛЬНИХ ТОЧОК ІЗ ЗАСТОСУВАННЯМ НАВЧЕНОЇ НЕЙРОМЕРЕЖІ
(тема)

Виконав:
студент ІІ курсу, групи РПСКм -19-1
Кавун Б. О.
(прізвище, ініціали)
Спеціальність
172 Телекомунікації та радіотехніка
(код і повна назва спеціальності)
Тип програми
освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма
Радіоелектронні пристрої, системи та комплекси
(повна назва освітньої програми)
Керівник професор Цопа О.І.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри РТІКС _____ Цопа О.І.
(підпис) (прізвище, ініціали)

2020 р.

Не містить відомостей заборонених до відкритого публікування.

Студент Б.О. Кавун

Керівник О.І. Цопа

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет Інформаційних радіотехнологій та технічного захисту інформацій
Кафедра Радіотехнологій інформаційно-комунікаційних систем
Рівень вищої освіти другий (магістерський)
Спеціальність 172 Телекомунікації та радіотехніка
Тип програми Освітньо-професійна
Освітня програма Радіоелектронні пристрої, системи та комплекси

ЗАТВЕРДЖУЮ:
Зав.кафедри_____
(підпис)
«_____»_____ 2020 р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Кавуну Богдану Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи **РОЗРОБКА ТА ДОСЛІДЖЕННЯ СИСТЕМИ РОЗПІЗНАВАННЯ ТРИВИМІРНИХ ОБ'ЄКТІВ ЗА ХМАРАМИ ЗАДАНИХ КОНТРОЛЬНИХ ТОЧОК ІЗ ЗАСТОСУВАННЯМ НАВЧЕНОЇ НЕЙРОМЕРЕЖІ**

Затверджена наказом по університету від **2 листопада 2020 р. № 1507Ст**

2. Термін подання студентом проекту (роботи) **12 грудня 2020 р.**

3. Вихідні дані до проекту (роботи)
3.1 Система виконана у вигляді мобільного додатка (для платформи Android).
3.2 Система використовує дворівневу нейромережу.
3.3 В якості пристрою вводу первинної інформації застосовується камера мобільного пристрою.
3.4 Розпізнавання не менше ніж 10 різних предметів.
3.5 Швидкість розпізнавання одного об'єкта не більше ніж 3 сек.

4. Перелік питань, що потрібно опрацювати в роботі
Реферат. Перелік умовних позначень, символів, одиниць, скорочень і термінів, Вступ. 4.1 Огляд і аналіз навчених нейромереж. 4.2 Огляд аналогічних рішень. 4.3 Розробка структурної схеми. 4.4 Розробка програмного забезпечення. Висновки. Перелік джерел, посилання. Додатки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)

Комп'ютерна презентація

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	Проф. Цопа Олександр Іванович		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Вступ	5.09 - 10.09	Виконано
2	Огляд і аналіз навчених нейромереж.	11.09 - 16.09	Виконано
3	Огляд аналогічних рішень.	17.09 - 21.09	Виконано
4	Розробка структурної схеми.	22.09 - 30.09	Виконано
5	Розробка програмного забезпечення.	1.10 -20.10	Виконано
6	Реферат	21.10 - 11.11	Виконано
7	Перелік умовних позначень, символів, одиниць, скорочень і термінів	12.11 - 15.11	Виконано
8	Висновки	16.11 - 20.11	Виконано
9	Оформлення пояснювальної записки	21.11 - 30.11	Виконано
10	Оформлення презентації	1.12 - 11.12	Виконано
11	Подання роботи на кафедру	12.12.2020	Виконано

Дата видачі завдання **4 вересня 2020 р.**

Студент_____
(підпис)

Керівник роботи _____ проф. Цопа О.І.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка атестаційної роботи містить 98 сторінок тексту, 16 рисунків, 10 джерел посилання та 3 додатки.

ПРОГРАМА, НЕЙРОМЕРЕЖА, КОРИСТУВАЧ, АЛГОРИТМ, ANDROID, ДОДАТОК, РОЗПІЗНАВАННЯ

Мета роботи: Розробка та навчання нейромережі, запрограмованої на самостійне розпізнавання предметів у форматі Android-додатка для спрощення побутових задач людей з вадами зору.

У даній кваліфікаційній роботі розроблено програмне забезпечення у вигляді нейромережі, яка розпізнає предмети побуту, використовуючи камеру мобільного пристрою.

Нейромережа побудована з двома прихованими рівнями, кожен з них має 18 нейронів, має 14 814 параметрів.

Нейромережа має ряд вже закладених предметів побуду, які розпізнаються з великим (близьким до 100%) процентом точності.

Користувач також має можливість увімкнути режим озвучування назви розпізнаного предмета.

ABSTRACT

The explanatory note of the attestation work contains 98 pages of text, 16 figures, 10 reference sources and 3 appendices.

PROGRAM, NEURAL NETWORK, USER, ALGORITHM, ANDROID, APPLICATION, RECOGNITION

Purpose: Development and training of a neural network programmed for self-recognition of objects in the format of an Android application to simplify the household tasks of visually impaired people.

In this qualification work, software was developed in the form of a neural network that recognizes household items using the camera of a mobile device.

The neural network is built with two hidden levels, each of which has 18 neurons, has 14,814 parameters.

The neural network has a number of already laid objects of construction, which are recognized with a large (close to 100%) percentage of accuracy.

The user also has the option to enable the sound mode of the name of the recognized object.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Ш І - штучний інтелект
ШНМ - нейромережа
ННМ - навчена нейромережа
ПЗ - програмне забезпечення
ОС - операційна система
Android-додаток - додаток для мобільних пристроїв під управлінням ОС “Android”
GAN - генераційно-змагальна ШНМ
ЛМ - метод Левенберга-Марквардта

ВСТУП

В наш час ми досить тісно пов’язані з таким явищем в електроніці та програмуванні, як Ш І. Ш І використовується в найрізноманітніших сферах промисловості та побуту, та призначений полегшити життя людини, зробити його більш комфортним. Так, наприклад, ми вже маємо досить розвинуті системи керування автомобілями на базі Ш І, реакція яких значно перевершує реакцію людини. Також під керування Ш І людство довірило і поміч у запуску космічних ракетоносіїв, що підвищило точність та знизило ризики.

Але найбільше Ш І проявляє себе у вигляді ШНМ - створена людиною, вона здатна до самонавчання та вдосконалення з кожним її використанням. Ми часто зустрічаємо такі ШНМ коли, наприклад, слухаємо музику за вподобаннями, читаємо книги онлайн, навіть коли просто шукаємо інформацію в Інтернеті. ННМ, зокрема, використовується в області розпізнавання об’єктів. Це і автопілоти в електрокарах, і регулювання дорожнього руху, і допомога людям з певними вадами здоров’я.

В рамках даної кваліфікаційної роботи розробляється ПЗ у вигляді Android-додатку для розпізнавання об’єктів у просторі.

Метою даної роботи є розробка та навчання ШНМ на мові програмування Java, для спрощення побутових задач людей з вадами зору.

Для досягнення поставленої мети в роботі були сформовані наступні задачі:

- на основі відомої інформації розробити ШНМ;
- навчити ШНМ розпізнавати об’єкти побуту;
- використовувати в процесі навчання хмари контрольних точок для кожного об’єкта;
- робота системи в режимі “реального часу”;

- зберегти швидкість розпізнавання окремого об’єкта в рамках не більше 3 сек;

- реалізувати доступ до камери мобільного пристрою;

- реалізувати масштабування хмари контрольних точок в залежності від відстані до об’єкта;

- реалізувати функціонал самонавчання в процесі використання ПЗ.

1 ОГЛЯД І АНАЛІЗ ННМ

1.1 Що таке нейромережа

ШНМ - це математична модель у вигляді програмного і апаратного втілення, що будується на принципах функціювання біологічних нейромереж. ШНМ є одним із видів машинного навчання. На рисунку 1.1 схематично зображено структуру одношарової нейромережі. Основою будь-якої ШНМ є дві складові: нейрони за з’єднання. Нейрони представляють собою деякий найпростіший елемент графу, що має певне числове значення. Кожен нейрон має функцію, яка обчислює *значення збудження.* Для кожного нейрона така функція своя. Нейрони пов’язані між собою *з’єднаннями.* Кожне з’єднання має своє числове значення, яке називають *вагою.* Мережа утворюється з'єднанням виходів певних нейронів зі входами інших нейронів з утворенням орієнтованого графу. Функції, які обчислюють збудження, та ваги, змінюються в процесі *навчання.* Такий процес керується *правилами навчання,* заданими людиною.

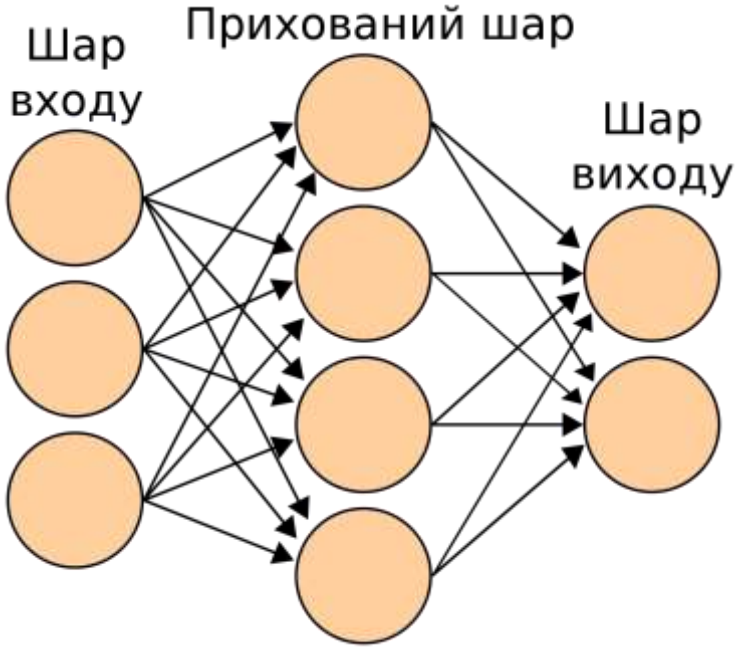

Рисунок 1.1 - Схематичне зображення одношарової нейромережі.

1.2 Класифікація нейромереж

1.2.1 ШНМ прямого поширення

ШНМ прямого поширення (feed forward neural networks, FF або FFNN) і *персептрони* (perceptrons, P) дуже прямолінійні, вони передають інформацію від входу до виходу. На рисунку 1.2 представлено схематичне зображення ШНМ типу персептрон (а) та прямого поширення (б). Нейронні мережі часто описуються у вигляді листкового торта, де кожен шар складається з вхідних, прихованих або вихідних клітин. Клітини одного шару не пов'язані між собою, а сусідні шари зазвичай повністю пов'язані. Найпростіша нейронна мережа має дві вхідних клітини і одну вихідну, і може використовуватися в якості моделі логічних вентилів. FFNN зазвичай навчається за методом зворотного поширення помилки, в якому мережа отримує безлічі вхідних і вихідних даних. Цей процес називається навчанням з учителем, і він відрізняється від навчання без учителя тим, що в другому випадку безліч вихідних даних мережу становить самостійно. Вищезазначена помилка є різницею між введенням і висновком. Якщо у мережі є достатня кількість прихованих нейронів, вона теоретично здатна змоделювати взаємодію між вхідним і вихідними даними. Практично такі мережі використовуються рідко, але їх часто комбінують з іншими типами для отримання нових.

а) б)

Рисунок 1.2 - ШНМ типу: а) персептрон, б) прямого поширення

1.2.2 ШНМ Хопфілда

ШНМ Хопфілда (Hopfield network, HN) - це повнозв’язна нейронна мережа із симетричною матрицею зв'язків. На рисунку 1.3 представлено схематичне зображення ШНМ Хопфілда. Під час отримання вхідних даних кожен вузол є входом, в процесі навчання він стає прихованим, а потім стає виходом. Мережа навчається так: значення нейронів встановлюються відповідно до бажаного шаблоном, після чого обчислюються ваги, які в подальшому не змінюються. Після того, як мережа навчилася на одному або декількох шаблонах, вона завжди буде зводитися до одного з них (але не завжди - до бажаного). Вона стабілізується в залежності від загальної «енергії» і «температури» мережі. У кожного нейрона є свій поріг активації, що залежить від температури, при проходженні якого нейрон приймає одне з двох значень (зазвичай -1 або 1, іноді 0 або 1). Така мережа часто називається мережею з асоціативною пам'яттю; як людина, бачачи половину таблиці, може представити другу половину таблиці, так і ця мережа, отримуючи таблицю, наполовину зашумлену, відновлює її до повної.

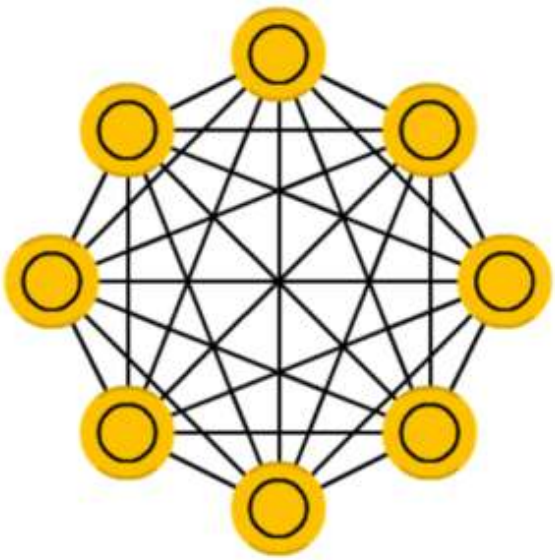

Рисунок 1.3 - ШНМ Хопфілда

1.2.3 Ланцюг Маркова

Ланцюги Маркова (Markov chains, MC або discrete time Markov Chains, DTMC) - це попередники машин Больцмана (BM) і мереж Хопфілда (HN). На рисунку 1.4 представлено схематичне зображення Ланцюга Маркова. Їхній зміст можна пояснити так: які мої шанси потрапити в один з наступних вузлів, якщо я перебуваю в даному? Кожне наступне стан залежить тільки від попереднього. Хоча насправді ланцюга Маркова не є ШНМ, вони дуже схожі. Також ланцюг Маркова не обов'язково повнозв'язний.

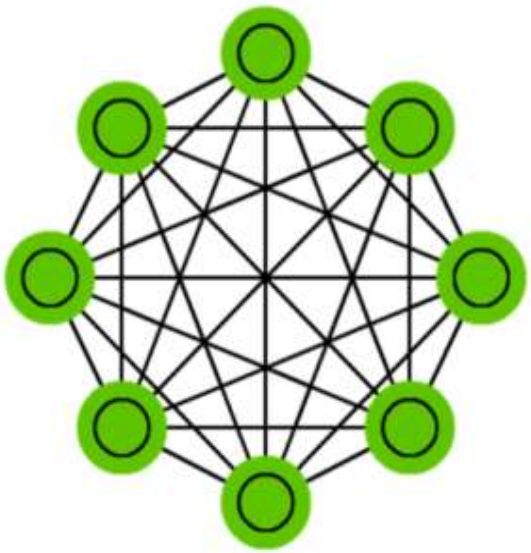

Рисунок 1.4 - Ланцюг Маркова

1.2.4 Машина Больцмана

Машина Больцмана (Boltzmann machine, BM) дуже схожа на мережу Хопфілда, але в ній деякі нейрони позначені як вхідні, а деякі - як приховані. На рисунку 1.5 представлено схематичне зображення Машини Больцмана. Вхідні нейрони в подальшому стають вихідними. Машина Больцмана - це стохастична мережа. Навчання проходить за методом зворотного поширення помилки або за алгоритмом порівняльної розбіжність. В цілому процес навчання дуже схожий на такий, що застосовується у мережі Хопфілда.

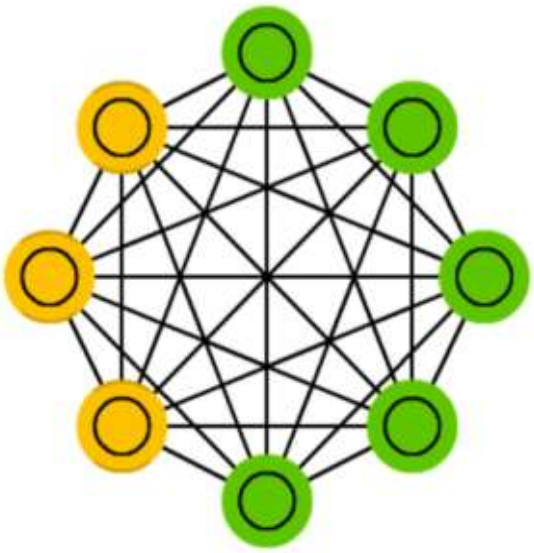

Рисунок 1.5 - Машина Больцмана

1.2.5 Обмежена машина Больцмана

Обмежена машина Больцмана (restricted Boltzmann machine, RBM) дуже схожа на машину Больцмана і, отже, на мережу Хопфілда. На рисунку 1.6 представлено схематичне зображення обмеженої машини Больцмана. Єдиною різницею є її обмеженість. У ній нейрони одного типу не пов'язані між собою. Обмежену машину Больцмана можна навчати як FFNN, але з одним нюансом: замість прямої передачі даних і зворотного поширення помилки потрібно передавати дані спершу в прямому напрямку, потім в зворотному. Після цього проходить навчання за методом прямого і зворотного поширення помилки.

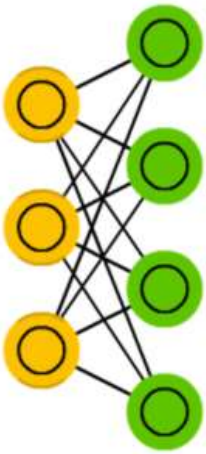

Рисунок 1.6 - Обмежена машина Больцмана

1.2.6 Автокодіровщик

Автокодіровщик (autoencoder, AE) чимось схожий на FFNN (ШНМ прямого поширення), адже це, швидше, інший спосіб використання FFNN, ніж фундаментально інша архітектура. На рисунку 1.7 представлено схематичне зображення автокодіровщика. Основною ідеєю є автоматичне кодування (в сенсі стиснення, не шифрування) інформації. Сама мережа по формі нагадує пісочний годинник, в ній приховані шари менше вхідного і вихідного, причому вона симетрична. Цю ШНМ можна навчити методом зворотного поширення помилки, подаючи вхідні дані і прирівнюючи помилку до різниці між входом і виходом.

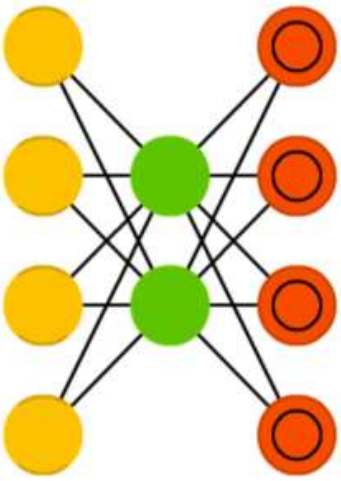

Рисунок 1.7 - Автокодіровщик

1.2.7 Розріджений автокодіровщик

Розріджений автокодіровщик (sparse autoencoder, SAE) - в якомусь сенсі протилежність звичайного. На рисунку 1.8 представлено схематичне зображення розрідженого автокодіровщика. Замість того, щоб навчати мережу відображати інформацію в меншому «обсязі» вузлів, ми збільшуємо їх кількість. Замість того, щоб звужуватися до центру, мережа там роздувається. Мережі такого типу корисні для роботи з великою кількістю дрібних властивостей набору даних. Якщо навчати мережу як звичайний автокодіровщик, нічого корисного не вийде. Тому крім вхідних даних

подається ще і спеціальний фільтр розрідженості, який пропускає тільки певні помилки.

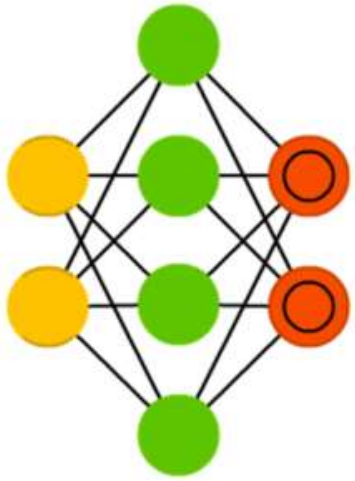

Рисунок 1.8 - Розріджений автокодіровщик

1.2.8 Варіаційний автокодіровщик

Варіаційні автокодіровщіки (variational autoencoder, VAE) мають схожу з AE архітектуру, але навчають їх іншому: наближенню імовірнісного розподілу вхідних зразків. На рисунку 1.9 представлено схематичне зображення варіаційного автокодіровщика. У цьому вони беруть початок від машин Больцмана. Проте, вони спираються на Байєсову математику, коли мова йде про імовірнісні висновки і незалежності, які інтуїтивно зрозумілі, але складні в реалізації. Якщо узагальнити, то можна сказати що ця мережа приймає до уваги вплив нейронів. Якщо щось одне відбувається в одному місці, а щось інше - в іншому, то ці події не обов'язково пов'язані, і це повинно враховуватися.

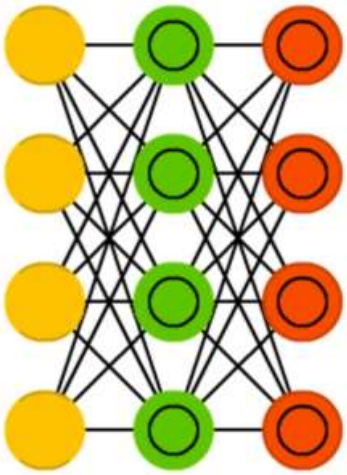

Рисунок 1.9 Варіаційний автокодіровщик

1.2.9 Шумоподавляючий автокодіровщик

Шумоподавляючі автокодіровщіки (denoising autoencoder, DAE) - це AE, в які вхідні дані подаються в зашумленому стані. На рисунку 1.10 представлено схематичне зображення шумоподавляючого автокодіровщика. Помилку ми обчислюємо так само, і вихідні дані порівнюються з зашумленими. Завдяки цьому мережа вчиться звертати увагу на більш широкі властивості, оскільки маленькі властивості можуть змінюватися разом з шумом.

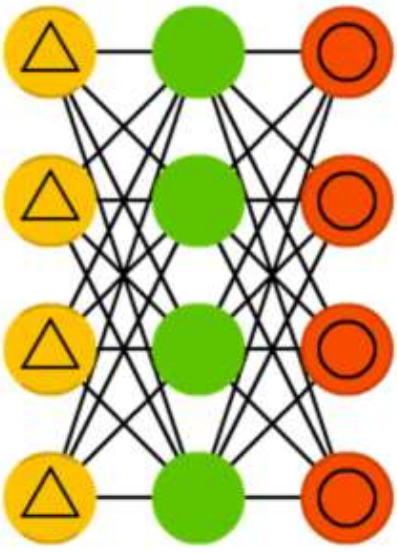

Рисунок 1.10 - Шумоподавляючий автокодіровщик

1.2.10 ШНМ типу “deep belief”

ШНМ типу «deep belief» (deep belief networks, DBN) - це назва, яку отримав тип архітектури, в якій мережа складається з декількох з'єднаних RBM або VAE. На рисунку 1.11 представлено схематичне зображення ШНМ типу “deep belief”. Такі мережі навчаються по блоках, причому кожному блоку потрібно лише вміти закодувати попередній. Така техніка називається «жадібним навчанням», яка полягає у виборі локальних оптимальних рішень, що не гарантують оптимальний кінцевий результат. Також мережу можна навчити (методом зворотного поширення помилки) відображати дані у вигляді ймовірнісної моделі. Якщо використовувати навчання без учителя, стабілізовану модель можна використовувати для генерації нових даних.

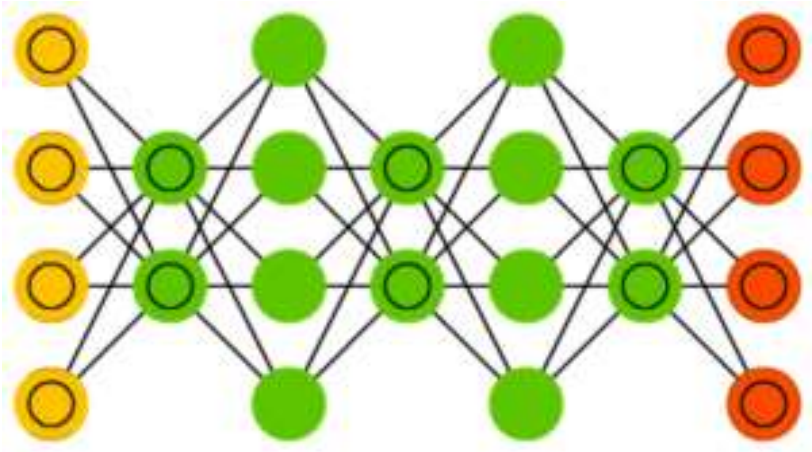

Рисунок 1.11 - ШНМ типу “deep belief”

1.2.11 Згорткові ШНМ

Згорткові ШНМ (convolutional neural networks, CNN) і *глибинні згорткові ШНМ* (deep convolutional neural networks, DCNN) сильно відрізняються від інших видів мереж. На рисунку 1.12 представлено схематичне зображення згорткової ШНМ. Зазвичай вони використовуються для обробки зображень, рідше для аудіо. Типовим способом застосування CNN є класифікація зображень: якщо на зображенні є кішка, мережа видасть «кішка», якщо є собака - «собака». Такі мережі зазвичай використовують «сканер», який не парсить усі дані за один раз. Наприклад, якщо у вас є зображення 200 × 200, ви не будете відразу обробляти всі 40 тисяч пікселів. Замість цього мережа розраховує квадрат розміру 20 x 20 (зазвичай з лівого

верхнього кута), потім зрушиться на 1 піксель і розраховує новий квадрат, і т.д. Ці вхідні дані потім передаються через згорткові шари, в яких не всі вузли з'єднані між собою. Ці шари мають властивість стискуватися з глибиною, причому часто використовуються ступеня двійки: 32, 16, 8, 4, 2, 1. На практиці до кінця CNN прикріплюють FFNN для подальшої обробки даних. Такі мережі називаються глибинними (DCNN).

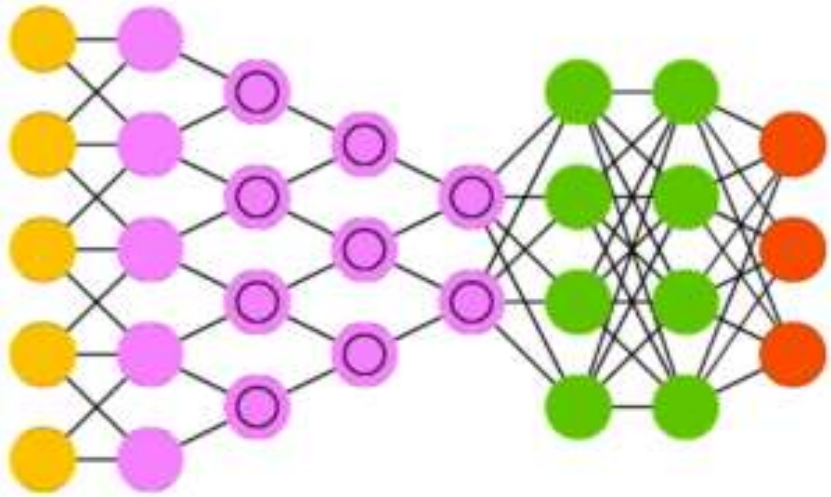

Рисунок 1.12 - Згорткові ШНМ

1.2.12 Розгорткові ШНМ

Розгорткові ШНМ (deconvolutional networks, DN), також звані зворотними графічними мережами, є зворотним до згорткових нейронних мереж. На рисунку 1.13 представлено схематичне зображення розгорткової ШНМ. Уявіть, що ви передаєте мережі слово «кішка», а вона генерує картинки з кішками, схожі на реальні зображення котів. DNN теж можна об'єднувати з FFNN. Варто зауважити, що в більшості випадків мережі передається не рядок, а який бінарний вектор: наприклад, <0, 1> - це кішка, <1, 0> - собака, а <1, 1> - і кішка, і собака.

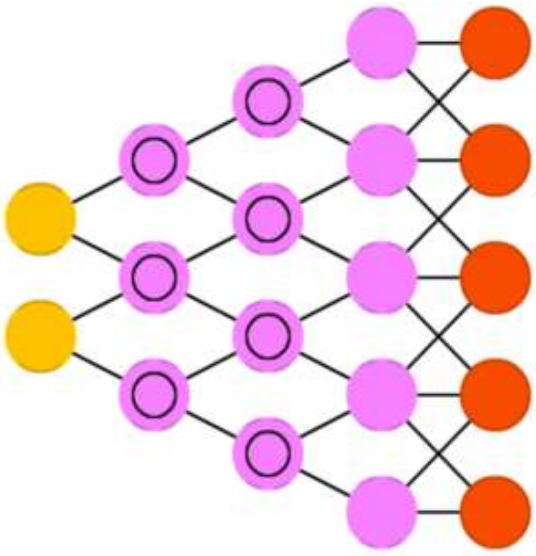

Рисунок 1.13 - Розгорткові ШНМ

1.3 Способи навчання нейромережі

1.3.1 Навчання без вчителя

Ідеально розмічені і чисті дані дістати нелегко. Тому іноді перед алгоритмом стоїть завдання знайти заздалегідь не відомі відповіді. Ось де потрібно навчання без учителя.

У навчанні без вчителя (unsupervised learning) у моделі є набір даних, і немає явних вказівок, що з ним робити. Нейронна мережа намагається самостійно знайти кореляції в даних, витягуючи корисні ознаки і аналізуючи їх. Залежно від завдання модель систематизує дані по-різному.

Кластеризація. Навіть без спеціальних знань експерта-орнітолога можна подивитися на колекцію фотографій і розділити їх на групи за видами птахів, спираючись на колір пера, розмір або форму дзьоба. Саме в цьому полягає кластеризація - найбільш поширена завдання для навчання без учителя. Алгоритм підбирає схожі дані, знаходячи спільні ознаки, і групують їх разом.

Виявлення аномалій. Банки можуть виявити шахрайські операції, виявляючи незвичайні дії в купівельному поводженні клієнтів. Наприклад, підозріло, якщо одна кредитна карта використовується в Каліфорнії і Данії в один і той же день. Схожим чином, навчання без вчителя використовують для знаходження викидів в даних.

Асоціації. Виберете в онлайн-магазині дитячу іграшку, яблучне пюре і дитячу чашку-непроливайку і сайт порекомендує вам додати нагрудник і радіо няню до замовлення. Це приклад асоціацій: деякі характеристики об'єкта корелюють з іншими ознаками. Розглядаючи пару ключових ознак об'єкта, модель може передбачити інші, з якими існує зв'язок.

Автоенкодери. Автоенкодери приймають вхідні дані, кодують їх, а потім намагаються відтворити початкові дані з отриманого коду. Не так багато реальних ситуацій, коли використовують простий автоенкодер. Але варто додати шари і можливості розширяться: використовуючи зашумлені і вихідні версії зображень для навчання, автоенкодери можуть видаляти шум з відеоданих, зображень або медичних сканів, щоб підвищити якість даних.

У навчанні без вчителя складно обчислити точність алгоритму, тому що в цих відсутні «правильні відповіді» або мітки. Але розмічені дані часто ненадійні або їх занадто дорого отримати. У таких випадках, надаючи моделі свободу дій для пошуку залежностей, можна отримати хороші результати.

1.3.2 Навчання з вчителем

Навчання з вчителем (supervised learning) передбачає наявність повного набору розмічених даних для тренування моделі на всіх етапах її побудови.

Наявність повністю розмічені датасету означає, що кожному наприклад в навчальному наборі відповідає відповідь, який алгоритм і повинен отримати. Таким чином, розмічений датасет з фотографій квітів навчить нейронну мережу, де зображені троянди, ромашки або нарциси. Коли мережу отримає нове фото, вона порівняє його з прикладами з навчального датасету, щоб передбачити відповідь.

В основному навчання з вчителем застосовується для вирішення двох типів задач: класифікації і регресії.

У задачах *класифікації* алгоритм пророкує дискретні значення, що відповідають номерам класів, до яких належать об'єкти. У навчальному датасеті з фотографіями тварин кожне зображення буде мати відповідну мітку - «кішка», «собака» або «єнот». Якість алгоритму оцінюється тим, наскільки точно він може правильно класифікувати нові фото з котами, єнотами і собаками.

А ось завдання *регресії* пов'язані з безперервними даними. Один із прикладів, лінійна регресія, обчислює очікуване значення змінної *y*, враховуючи конкретні значення *x*. Більш утилітарні завдання машинного навчання використовують велику кількість змінних. Як приклад, нейронна мережа, передбачає ціну квартири в Харкові на основі її площі, місця розташування і доступності громадського транспорту. Алгоритм виконує роботу експерта, який розраховує ціну квартири виходячи з тих же даних.

Таким чином, навчання з учителем найбільше підходить для задач, коли є значний набір достовірних даних для навчання алгоритму. Але так буває далеко не завжди. Мала кількість вихідних даних - дуже часта проблема в машинному навчанні на 2019-2020 роки.

1.3.3 Навчання з частковим залученням вчителя

Це золота середина.

Навчання з частковим залученням вчителя (semi-supervised learning) характеризується своєю назвою: навчальний датасет містить як розмічені, так і розділеного дані. Цей метод особливо корисний, коли важко отримати з даних важливі ознаки або розмітити всі об'єкти - трудомістке завдання.

Цей метод машинного навчання поширений для аналізу медичних зображень, таких як скани комп'ютерної томографії або МРТ. Досвідчений рентгенолог може розмітити невелику кількість сканів, на яких виявлені пухлини і захворювання. Але вручну розмічати всі скани - занадто трудомістка і дорога задача. Проте нейронна мережа може отримати інформацію з невеликої частки розмічених даних і поліпшити точність прогнозів в порівнянні з моделлю, яка навчається виключно на нерозмічену даних.

Популярний метод навчання, для якого потрібно невеликий набір розмічених даних, полягає в використанні GAN. Уявіть собі змагання двох нейронних мереж, де кожна намагається перехитрити іншу. Це GAN. Одна з мереж, генератор, намагається створити нові об'єкти даних, які імітують навчальну вибірку. Інша мережа, дискримінатор, оцінює, чи є ці згенеровані дані реальними або підробленими. Мережі взаємодіють і циклічно удосконалюються, оскільки дискримінатор намагається краще відокремлювати

підробки від оригіналів, а генератор намагається створювати переконливі підробки.

1.3.4 Навчання з підкріпленням

Відеоігри засновані на системі стимулів. Завершіть рівень і отримаєте нагороду. Переможете всіх монстрів і заробите бонус. Потрапили в пастку - кінець гри, не потрапляйте. Ці стимули допомагають гравцям зрозуміти, як краще діяти в наступному раунді гри. Без зворотного зв'язку люди б просто брали випадкові рішення і сподівалися перейти на наступний ігровий рівень. Навчання з підкріпленням (reinforcement learning) діє за тим же принципом. Відеоігри - популярна тестова середовище для досліджень.

Агенти ШНМ намагаються знайти оптимальний спосіб досягнення мети або поліпшення продуктивності для конкретного середовища. Коли агент робить дії, що сприяють досягненню мети, він отримує нагороду. Глобальна мета - передбачати такі кроки, щоб заробити максимальну нагороду в кінцевому підсумку.

При прийнятті рішення агент вивчає зворотний зв'язок, нові тактики і рішення здатні привести до більшого виграшу. Цей підхід використовує довгострокову стратегію - так само як в шахах: наступний найкращий хід може не допомогти виграти в кінцевому рахунку. Тому агент намагається максимізувати сумарну нагороду.

Це ітеративний процес. Чим більше рівнів з зворотного зв'язку, тим краще стає стратегія агента. Такий підхід особливо корисний для навчання роботів, які керують автономними транспортними засобами або інвентарем на складі.

Так само, як і учні в школі, кожен алгоритм вчиться по-різному. Але завдяки різноманітності доступних методів, питання в тому, щоб вибрати підходящий і навчити вашу нейронну мережу розбиратися в середовищі.

1.4 Задачі, для яких застосовуються нейромережі

Рішення завдання класифікації є одним з найважливіших застосувань ШНМ. Завдання класифікації представляє собою завдання віднесення зразка до одного з декількох множин, що попарно не перетинаються. Прикладом таких завдань може бути, наприклад, завдання визначення кредитоспроможності клієнта банку, медичні завдання, в яких необхідно визначити, наприклад, результат захворювання, рішення задач управління портфелем цінних паперів (продати купити або "притримати" акції в залежності від ситуації на ринку), завдання визначення життєздатних і схильних до банкрутства фірм.

При вирішенні задач класифікації необхідно віднести наявні статичні зразки (характеристики ситуації на ринку, дані медогляду, інформація про клієнта) до певних класів. Можливо кілька способів подання даних. Найбільш поширеним є спосіб, при якому зразок видається вектором. Компоненти цього вектора є різні характеристики зразка, які впливають на прийняття рішення про те, до якого класу можна віднести даний зразок. Наприклад, для медичних завдань в якості компонентів цього вектора можуть бути дані з медичної карти хворого. Таким чином, на підставі деякої інформації про приклад, необхідно визначити, до якого класу його можна віднести. Класифікатор таким чином відносить об'єкт до одного з класів відповідно до визначеного розбиттям N-мірного простору, яке називається простором входів.

Перш за все, потрібно визначити рівень складності системи. В реальних задачах часто виникає ситуація, коли кількість зразків обмежена, що ускладнює визначення складності завдання. Можливо виділити три основні рівні складності. Перший (найпростіший) - коли класи можна розділити прямими лініями (або гіперплощинами, якщо простір входів має розмірність більше двох) - так звана лінійна роздільність. У другому випадку класи неможливо розділити лініями (площинами), але їх можна виділити за допомогою більш складного поділу - нелінійна роздільність. У третьому випадку класи перетинаються і можна говорити тільки про ймовірнісної роздільності.

В ідеальному варіанті після попередньої обробки ми повинні отримати лінійно роздільне завдання, так як після цього значно спрощується побудова класифікатора. На жаль, при вирішенні реальних завдань ми маємо обмежену

кількість зразків, на підставі яких і проводиться побудова класифікатора. При цьому ми не можемо провести таку передобробку даних, при якій буде досягнута лінійна роздільність зразків.

1.5 Алгоритм побудови класифікатора на основі ШНМ

1. Робота з даними

- Скласти базу даних із прикладів, характерних для даного завдання.
- Розбити всю сукупність даних на два множини: навчальне і тестове (можливо розбивка на 3 множини: навчальне, тестове і підтверджуюче).

2. Попередня обробка

- Вибрати систему ознак, характерних для даного завдання, і перетворити дані відповідним чином для подачі на вхід мережі (нормування, стандартизація і т.д.). В результаті бажано отримати лінійно відокремлюваний простір множини зразків.
- Вибрати систему кодування вихідних значень (класичне кодування, 2 на 2 кодування і т.д.)

3. Конструювання, навчання і оцінка якості мережі

- Вибрати топологію мережі: кількість шарів, число нейронів в шарах і т.д.
- Вибрати функцію активації нейронів (наприклад "сигмоїда")
- Вибрати алгоритм навчання мережі
- Оцінити якість роботи мережі на основі підтверджуючої множини або іншим критерієм, оптимізувати архітектуру (зменшення ваг, проріджування простору ознак)
- Зупиниться на варіанті мережі, який забезпечує найкращу здатність до узагальнення і оцінити якість роботи по тестовій множині

4. Використання і діагностика

- З'ясувати ступінь впливу різних чинників на прийняте рішення (евристичний підхід).

- Переконатися, що мережа дає необхідну точність класифікації (число неправильно розпізнаних прикладів мало)
- Практично використовувати мережу для вирішення завдання.

1.6 Нейромережі як інструмент розпізнавання об’єктів

Нейронна мережа для розпізнавання зображень - це, мабуть, найбільш популярний спосіб застосування ШНМ. При цьому незалежно від особливостей вирішуваних завдань, вона працює за наступними етапами:

- Збір та підготовка даних
- Вибір топології (архітектури)
- Підбір характеристик
- Підбір параметрів навчання
- Процес навчання
- Ревізія якості навчання
- Внесення коректувань
- Вербалізація

Як образи розпізнавання можуть виступати найрізноманітніші об'єкти, включаючи зображення, рукописний або друкований текст, звуки і багато іншого. При навчанні мережі їй пропонуються різні зразки з міткою того, до якого саме типу їх можна віднести. Як зразок застосовується вектор значень ознак, а сукупність ознак в цих умовах повинна дозволити однозначно визначити, з яким класом образів має справу ШНМ.

Важливо при навчанні навчити мережу визначати не тільки достатню кількість і значення ознак, щоб видавати хорошу точність на нових зображеннях, але і не перенавчити, тобто, надмірно не «підлаштуватися» під навчальну вибірку з зображень. Після завершення правильного навчання ШНМ повинна вміти визначати образи (тих же класів), з якими вона не мала справи в процесі навчання.

Важливо враховувати, що вихідні дані для нейромережі повинні бути однозначні і несуперечливі, щоб не виникали ситуації, коли ШНМ буде видавати високі ймовірності приналежності одного об'єкта до декількох класів.

2 РОЗРОБКА СТРУКТУРНОЇ СХЕМИ

Структурна схема нейромережі представляє собою двошарову ШНМ, вхідний шар має 784 нейрони, перший прихований шар має 18 нейронів, другий прихований шар має 18 нейронів, вихідний шар має 64 нейрони. Для більш явного представлення вхідний та вихідний шари представлено першими 10 нейронами та останніми 10 нейронами. Схематичне зображення представлено на рисунку 2.1.

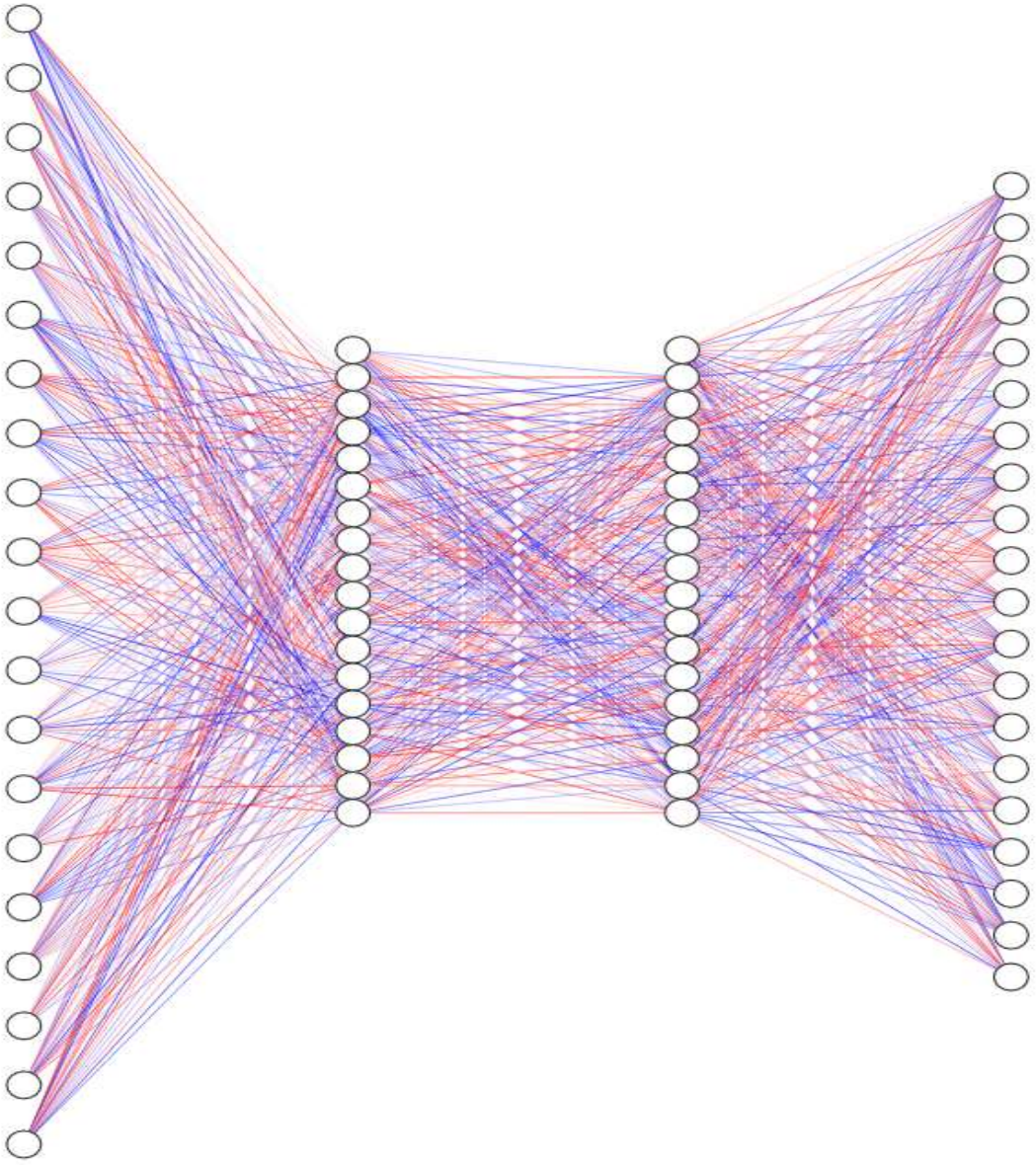

Рисунок 2.1 - Структурна схема нейромережі розпізнавання образів

При розробці даної нейромережі було вибрано архітектуру FFNN, тобто ШНМ у формі багатошарового персептрона. Обумовлюється це такими показниками як:

- простота
- ефективність використання ресурсів мобільного пристрою
- доцільність використання для задачі розпізнавання об’єктів

Головною властивістю, яке відрізняє багатошаровий персептрон від одношарового, є наявність декількох шарів обчислювальних нейронів між вхідним і вихідним шаром. У одношарового персептрона, як зрозуміло з назви, такий шар один.

Другим важливим властивістю багатошарового персептрона можна назвати односпрямованість (лінійність): сигнал від вхідних нейронів через шари обчислювальних чітко прямує до вихідних нейронів, зворотний рух неможливо. Крім цих двох головних властивостей, існує ще три додаткових, які безпосередньо пов'язані з подібним будовою персептрона.

У кожного нейрона є своя функція активації. Оскільки структура ускладнена в порівнянні з одношарової моделлю, то з'являється можливість відправляти сигнали з виходу кожного нейрона на один з декількох сусідніх з ним. З цієї причини функція активації, що здійснює вибір нейрона і відправку на нього сигналу, тепер є виділеної для кожного нейрона, в той час як в одношарової моделі така функція присутня лише у цілого шару.

Кілька шарів нейронів відкривають можливість для навчання мережі. Це одна з головних переваг даної моделі, яка і обумовлює використання такої архітектури для даної атестаційної роботи.

Велике значення зв'язків. Міжнейронні зв'язку (синапси) отримують все більше значення для багатошарових нейромереж, і чим більше в них шарів, тим вище значення таких зв'язків.

Для всіх нейронів функцією активації слугує сигмоїда:

$$y = \frac{1}{1 + e^{-CS}}$$

Така функція є однією з найбільш поширених функцій активації є нелінійна функція з насиченням - так звана логістична функція або сигмоїд. [2, с.34]

При зменшенні коефіцієнта C сигмоїд стає більш пологим, вироджуючись в межі при C = 0 в горизонтальну лінію на рівні 0.5. При збільшенні C сигмоїд наближається за зовнішнім виглядом до функції одиничного стрибка з порогом T в точці x = 0.

З виразу для сигмоїда можемо бачити, що вихідне значення нейрона лежить в діапазоні {0,1}. Популярність сигмоїдної функції визначають наступні її цінні властивості:

- здатність підсилювати слабкі сигнали краще, ніж великі, і чинити опір "насиченню" від потужних впливів;
- монотонність і диференційовність на всій осі абсцис;
- простий вираз для її похідної, що дає можливість використовувати широкий спектр оптимізаційних алгоритмів.

Розберемо значення кожного шару нейронів для ШНМ як системи.

Перший, вхідний шар складається з вхідних нейронів, які передають інформацію в прихований слів. Прихований шар в свою чергу передає інформацію в вихідний. Кожен нейрон має входи з вагами - синапсами, функцію активації, визначальну вихідну інформацію при заданої вхідної, і один вихід. Синапси - регульовані параметри, що конвертують нейронну мережу в параметризовану систему.

Стан нейрона визначається, як сума станів його входів виходячи з формули:

$$s = \sum_{i=1}^{n} x_i \cdot w_i$$

[1, с.42]

Як бачимо - значення на вході синапсу множиться на вагу даного синапсу, потім всі ці значення підсумовуються і отримуємо поточний стан нейрона. Виразимо цю формулу у вигляді програмного коду:

```
float sum = 0;
    for(int i = 0; i < N; i++) {
          sum += inputs[i]*weights[i];
    }
```

Після вхідного шару нейронів, для багатошарової моделі, характерна наявність прихованих шарів. Приховані шари значно розширюють можливості ШНМ та надають більше інструментів для навчання такої ШНМ.

Додавання прихованого шару між вхідним і вихідним шарами перетворює перцептрон в універсальний апроксиматор, що, по суті, означає, що він здатний захоплювати і відтворювати надзвичайно складні зв'язку вхід-вихід. [1, с.25]

Наявність прихованого шару робить навчання трохи складнішим, тому що вагові коефіцієнти між вхідним і прихованим шарами непрямим чином впливають на кінцеву помилку (цей термін я використовую для позначення різниці між вихідним значенням нейромережі і цільовим значенням, заданим навчальними даними).

Вибір числа нейронів для вихідного шару дуже схожий підходом для вхідного шару. Визначити кількість нейронів дуже просто. Воно повністю визначається розмірністю вихідного сигналу. Існує кілька правил, які дозволяють встановити кількість шарів і розмір нейронного шару для обох вхідних і вихідних шарів. [6, с.96]

Основне питання архітектури багатошарового персептрона стосується прихованих шарів. Скільки прихованих шарів повинно бути в нейронної мережі? Одна з проблем в цій темі, по котрій існує консенсус, тобто різниця продуктивності від додавання додаткових прихованих шарів - ситуація, в якій підвищується продуктивність з додаванням другого (третього, четвертого, і т.д.) прихованого шару дуже малоймовірна. Один прихований шар достатній для переважної більшості завдань. [3, с.53]

Підсумовуючи все вище сказане, для більшості завдань можна було б, ймовірно, отримати гідну продуктивність встановивши конфігурацію прихованого шару, використовуючи тільки два правила:

1) кількість прихованих шарі дорівнює одиниці;

2) кількість нейронів в цьому шарі є середнім між кількістю нейронів у вхідному і вихідному шарах і бути в кілька разів менше кількості навчальних прикладів, за умови наявності надмірності навчальних даних.

Для того, щоб нейронна мережа навчилася правильно на заданій вибірці даних з найбільшою точністю, необхідно підібрати це число нейронів найбільш оптимально. Їх повинно бути не багато, але і досить для гарної апроксимації

функції в просторі синоптичних вагів. У разі, якщо кількість нейронів в прихованому шарі недостатньо, то завдання навчання на навчальній множині не може бути вирішена. У разі, якщо нейронів занадто багато, мережа може навчитися великій кількості. При цьому фактично відбувається запам'ятовування різних шумів і похибок навчальних даних. При цьому процес навчання істотно сповільнюється через більшої кількості вагових коефіцієнтів мережі. Тому для настройки оптимальної архітектури мережі застосовуються різні методи. Такі способи діляться на дві групи: алгоритми скорочення і конструктивні алгоритмі.

ШНМ, представлена в даній атестаційній роботі, містить 784 нейрони для першого шару. Така кількість обумовлена тим, що в якості робочої області використовується квадрат пікселів розміром 28x28. Після обробки першої робочої області відбувається зсув на 28 пікселів вправо відносно верхнього лівого кута цільного зображення, отриманого з камери. Такий спосіб отримання вхідних значень для нейронів несе в собі наступні переваги:

- швидкість обробки зображення;
- точність передачі зваженої суми;
- потребує значно меншого ресурсу від мобільного пристрою.

Між першим (вхідним) та другим шарами розташовані синапси. Їх кількість дорівнює 14 112, у кожного синапса своя вага. Ваги зв'язків нейронів першого шару з шаром роздільників сигналу, а також значення порогів визначаються в процесі навчання. Таким чином, для налаштування в процесі навчання, ШНМ має більше 14 000 параметрів, що робить таку нейромережу дуже гнучкою.

Другий шар, він же перший прихований, складається з 18 нейронів та синаптичних з’єднань між попереднім та наступними шарами.

Третій шар, він же другий прихований, складається з 18 нейронів на синаптичних з’єднань між попереднім та наступними шарами.

В даному проекті передбачається використання камери мобільного пристрою в якості джерела отримання вхідної інформації.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Бібліотека OpenCV

OpenCV - бібліотека комп'ютерного зору і машинного навчання з відкритим вихідним кодом. У неї входять понад 2500 алгоритмів, в яких є як класичні, так і сучасні алгоритми для комп'ютерного зору і машинного навчання. Ця бібліотека має інтерфейси на різних мовах, серед яких є Python, Java, C ++ і Matlab. У моєму випадку, було використано інтерфейс для мови Java. Розпізнавання об'єктів відбувається за допомогою кольорової сегментації зображення. Для цього є дві функції:

cv2.findContours

cv2.drawContours

Щоб розпізнати образ, потрібно виділити його контури. Контури можна пояснити просто як криву, що з’єднує всі суцільні точки (разом із межею), маючи однаковий колір або інтенсивність. Контури є корисним інструментом для аналізу фігури та виявлення та розпізнавання об'єктів.

Застосування порогового значення для зображення у градаціях сірого робить його двійковим зображенням. Для цього встановлюється порогове значення, при якому всі значення нижче цього порогу стають чорними, а всі значення вище стають білими.

OpenCV використовує обгортки з пресетів OpenCPP із загальновживаних бібліотек дослідників у галузі комп'ютерного зору (FFmpeg, libdc1394, PGR FlyCapture, OpenKinect, librealsense, CL PS3 Eye Driver, videoInput, ARToolKit-Plus, flandmark, Leptonica та Tesseract) та на їх базі надає класи корисних програм, щоб полегшити їх функціонування на платформі Java, включаючи Android.[5, с.124]

OpenCV також постачається з апаратно прискореним повноекранним відображенням зображень (CanvasFrame та GLCanvasFrame), простими у використанні методами паралельного виконання коду на декількох ядрах (Parallel), зручним геометричним та кольоровим калібруванням камер та проекторів (GeometricCalibrator, ProCamGeometricCalibrator , ProCamColorCali-

brator), виявлення та узгодження точок об’єктів (ObjectFinder), набір класів, що реалізують пряме вирівнювання зображень систем проектора-камери (головним чином GNImageAligner, ProjectiveTransformer, ProjectiveColorTransformer, ProCamTransformer та ReflectanceInitializer), пакет аналізу крапок (Blobs) а також різну функціональність у класі JavaCV. Деякі з цих класів також мають аналог OpenCL та OpenGL, їх імена закінчуються на CL або починаються з GL, тобто: JavaCVCL, GLCanvasFrame тощо.

Основні модулі бібліотеки:

cxcore - ядро

Містить базові структури даних і алгоритми:

- базові операції над багатовимірними числовими масивами
- матрична алгебра, математичні ф-ції, генератори випадкових чисел
- Запис / відновлення структур даних в / з XML
- базові функції 2D графіки

CV - модуль обробки зображень і комп'ютерного зору

- базові операції над зображеннями (фільтрація, геометричні перетворення, перетворення колірних просторів і т. Д.)
- аналіз зображень (вибір відмінних ознак, морфологія, пошук контурів, гістограми)
- аналіз руху, спостереження за об'єктами
- виявлення об'єктів, зокрема осіб
- калібрування камер, елементи відновлення просторової структури

Highgui - модуль для введення / виведення зображень і відео, створення призначеного для користувача інтерфейсу

- захоплення відео з камер і з відео файлів, читання / запис статичних зображень.
- функції для організації простого UI (всі демо додатки використовують HighGUI)

Cvaux - експериментальні і застарілі функції

- просторів. зір: стерео калібрації, саме калібрації
- пошук стерео-відповідності, кліки в графах
- знаходження і опис рис обличчя

3.2 Процес навчання ШНМ по розпізнаванню образів

Під час навчання нейронної мережі, навчальні дані подаються на вхідний шар мережі. Цей етап називається прямим ходом алгоритму зворотного поширення. Під час прямого ходу, кожен вузол в прихованому шарі отримує від всіх інших вузлів вхідного шару значення, які перемножуються з відповідними ваговими коефіцієнтами і потім сумуються. Вихідні сигнали нейронів є нелінійними перетвореннями функції активації. Аналогічно, кожен вихідний вузол отримує від всіх нейронів в прихованому шарі отримані значення, які також перемножуються з відповідними ваговими коефіцієнтами і потім сумуються. Виходом кожного вихідного нейрона є функція нелінійного перетворення активаційної функції.

Вихідні значення для останнього шару порівнюються з ідеальними вихідними значеннями. Ідеальними вихідними значеннями вважаються вихідні навчальні дані. Помилка між ідеальним вихідним значенням і фактичним, отриманим після проходження сигналу через мережу розраховується і поширюється назад до прихованого шару. Такий напрям називається зворотним ходом алгоритму зворотного поширення помилки. Помилка використовується, щоб оновити вагові коефіцієнти між вузлами, наприклад, перераховуються ваги матриць між вхідним-прихованим і прихованим-вихідним шарами.

Локальні мінімуми функції помилки - це недолік, який властивий будь-якому спрямованому методу мінімізації, що використовується при навчанні нейронних мереж. На рисунку 3.1 показаний приклад локального мінімуму з більш високим рівнем помилки, ніж в інших місцях функції. У поверхні відгуку функції помилки існує так звана "долина", і якщо градієнтний спуск починається там, то алгоритм навчання зупиниться - відбувається так званий параліч навчання. [7, с.63]

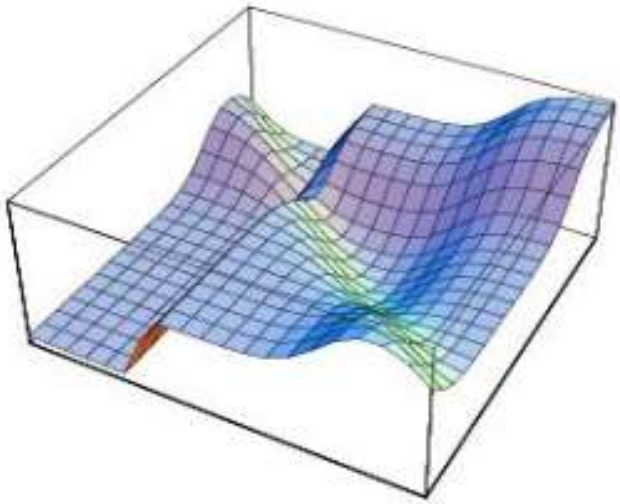

Рисунок 3.1 - Приклад локального мінімуму функції помилки

Для того щоб можна було почати навчання нашої мережі потрібно визначитися з тим, як вимірювати якість розпізнавання. У моєму випадку для цього буду використовувати найпоширенішу в теорії нейронних мереж функцію середньоквадратичної помилки (СКО, MSE):

$$E^p = \frac{1}{2}(D^p - O(I^p, W))^2$$

[9, с.117]

У цій формулі *Ep* - це помилка розпізнавання для *p*-ой навчальної пари, *Dp* - бажаний вихід мережі, *O* (*Ip*, *W*) - вихід мережі, що залежить від *p*-го входу і вагових коефіцієнтів *W*, куди входять ядра згортки, зміщення, вагові коефіцієнти *S*- і *F*- шарів. Завдання навчання так налаштувати ваги *W*, щоб вони для будь-якої навчальної пари (*Ip*, *Dp*) давали мінімальну помилку *Ep*. Щоб порахувати помилку для всієї навчальної вибірки просто береться середнє арифметичне по помилках для всіх навчальних пар. Таку усереднену помилку позначимо як *E*.

Для мінімізації функції помилки *Ep* найефективнішими є градієнтні методи. Розглянемо суть градієнтних методів на прикладі найпростішого одновимірного випадку (тобто коли у нас всього один вага). Якщо ми розкладемо в ряд Тейлора функцію помилки *Ep*, то отримаємо такий вираз:

$$E(W) = E(W_c) + (W - W_c)\frac{dE(W_c)}{dW} + \frac{1}{2}(W - W_c)^2\frac{d^2E(W_c)}{dW^2} + \dots$$

[9, с.119]

Тут *E* - все та ж функція помилки, *Wc* - деяке початкове значення ваги. Зі шкільної математики ми пам'ятаємо, що для знаходження екстремуму функції необхідно взяти її похідну і прирівняти нулю. Так і вчинимо, візьмемо похідну функції помилки по вагам, відкинувши члени вище 2го порядку:

$$\frac{dE(W)}{dW} = \frac{dE(W_c)}{dW} + (W - W_c)\frac{d^2E(W_c)}{dW^2}$$ [9, с.120]

з цього виразу випливає, що вага, при якому значення функції помилки буде мінімальним можна обчислити з наступного виразу:

$$W_{min} = W_c - \left(\frac{d^2E(W_c)}{dW^2}\right)^{-1} \frac{dE(W_c)}{dW}$$ [9, с.122]

Тобто, оптимальна вага обчислюється як поточна мінус похідна функції помилки по вазі, поділена на другу похідну функції помилки. Для багатошарового випадку (тобто для матриці ваг) все одно, тільки перша похідна перетворюється в градієнт (вектор приватних похідних), а друга похідна перетворюється в гессіан (матрицю других приватних похідних). І тут можливі два варіанти. Якщо ми опустимо другу похідну, то отримаємо алгоритм найшвидшого градієнтного спуску. Якщо все ж захочемо враховувати другу похідну, то для даної атестаційної роботи це неможливо, так як нам не вистачить обчислювальних ресурсів мобільного пристрою, щоб порахувати повний гессіан, а потім ще і звернути його. Тому зазвичай гессіан замінюють чимось простішим. Наприклад, один з найвідоміших і успішних методів - метод Левенберга-Марквардта (ЛМ) замінює гессіан, його апроксимацією за допомогою квадрадного Якобіана.

Але що нам важливо знати про ці дві методах, так це те, що алгоритм ЛМ вимагає обробки всієї навчальної вибірки, тоді як алгоритм градієнтного спуску може працювати з кожною окремо взятою навчальною вибіркою. В останньому випадку алгоритм називають стохастичним градієнтом. З огляду на, що наша база містить 60 000 навчальних зразків нам більше підходить стохастичний градієнт. Ще однією перевагою стохастичного градієнта є його менша схильність потрапляння в локальний мінімум в порівнянні з ЛМ.

Принцип роботи нейромережі полягає в тому, що активація в одному шарі визначає активацію в наступному. Збуджуючись, деяка група нейронів викликає збудження іншої групи. Якщо передати навченої нейронної мережі на

перший шар значення активації згідно яскравості кожного пікселя картинки, ланцюжок активацій від одного шару нейромережі до наступного призведе до переважної активації одного з нейронів останнього шару, відповідного розпізнаної цифрі - вибору нейронної мережі.

У процесі розпізнавання образів ми зводимо воєдино різні компоненти. Наприклад, чашка складається з циліндру, овалу зверху і напівовалу з однієї сторони . Вилку взагалі можна представити як набір прямих (а під певним кутом не зовсім прямих) коротких ліній. Клавіатура складається, по суті, з великої кількості квадратів, розташованих несиметрично. І так далі.

В ідеалізованому випадку можна очікувати, що кожен нейрон з другого шару співвідноситься з одним з таких примітивів (квадрат, овал, циліндр і т.д.). І, коли ви, наприклад, передаєте нейромережі зображення з овалом у верхній частині, існує певний нейрон, чия активація стане ближче до чашки. Таким чином, перехід від другого прихованого шару до вихідного відповідає знанням про те, який набір компонентів якому образу відповідає.

Задачу розпізнавання овалу так само можна розбити на підзадачі. Наприклад, розпізнавати різні маленькі межі, з яких він утворений. Аналогічно довгу вертикальну лінію можна уявити як шаблон з'єднання декількох менших шматочків. Таким чином, можна сподіватися, що кожен нейрон з першого прихованого шару нейромережі здійснює операцію розпізнавання цих малих граней.

Таким чином введене зображення призводить до активації певних нейронів першого прихованого шару, що визначають характерні малі шматочки, ці нейрони в свою чергу активують більші форми, в результаті активуючи нейрон вихідного шару, асоційованої певній цифрі.

3.3 Робота з камерою мобільного пристрою

В даній атестаційній роботі для реалізації повного функціоналу та досягнення мети нам знадобиться розробити мобільний додаток, в якому нам знадобилося використовувати камеру, щоб робити знімки та далі оброблювати їх.

Насамперед створимо базову активність, яка буде містити компонент ImageView для його відображення. Сюди ми будемо виводити зображення з камери. Також створимо неактивну область знизу екрану (бар), для виводу назви предмета.

Щоб наш додаток міг працювати з камерою мобільного пристрою, необхідно додати відповідний дозвіл в файл маніфесту додатка:

```
<uses-permission android:name="android.permission.CAMERA" />
```

Тепер нам потрібно зробити знімок і результат відобразити в віджеті ImageView. Для початку напишемо стандартний код активності і установки для користувача інтерфейсу. Такий код має наступний вигляд:

```
public class MainActivity extends AppCompatActivity {

    private ImageView imageView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        initUI();
    }

    private void initUI() {
```

```
        imageView = (ImageView) findViewById(R.id.imageView);
        FloatingActionButton fab = (FloatingActionButton)
findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //запускаем камеру
                //отображаем снимок в ImageView
            }
        });
    }
}
```

Нижче наведено код перевизначеного методу onActivityResult () з обробкою результату нашого запиту.

```
@Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode == MY_CAMERA_REQUEST) {
            if (resultCode == RESULT_OK) {
                if (data != null) {
                    Bitmap bitmap = (Bitmap) data.getExtras().get("data");
                    imageView.setImageBitmap(bitmap);
                }
            }
        }
    }
```

Метод зворотного виклику onActivityResult () отримує три параметри - код запиту, код результату (успіх або невдача) і об'єкт Intent, який зберігає дані відповіді. У цьому методі ми спочатку перевіряємо requestCode. Якщо він збігається з заданими нами кодом запиту, то далі виконується перевірка на успішність виконання нашого запиту. Якщо все ок, і дані не порожні, то ми отримуємо наше зображення і виводимо його в ImageView.

Перед початком роботи з версіями Android 6.0 і вище слід врахувати механізм так званих runtime permissions. Коротко - це механізм отримання прав на ті чи інші функції тоді, коли їх використання необхідно. Перед використанням камери нам необхідно запитати користувача: чи дозволяє він дати вашого додатком доступ до камери. Якщо доступу до камери у додатку немає - то показати йому відповідне повідомлення.

Додаємо метод, який буде запитувати у користувача дозвіл на використання камери:

```
private void requestPermissions() {
    ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.
    CAMERA}, MY_CAMERA_PERMISSION_REQUEST);
    }
```

Другим параметром ми передаємо масив потрібних нам дозволів, ми могли б запросити їх відразу кілька, а система послідовно запитала б користувача, які з функцій можна використовувати з додатком. Нам зараз потрібна тільки камера. Третій параметр - це такий же код запиту для даного набору дозволів, константу слід визначити заздалегідь.

Тепер нам потрібно перевизначити метод onRequestPermissionsResult ():

```
@Override
    public void onRequestPermissionsResult(int requestCode, String[] permissions,
                    int[] grantResults) {switch (requestCode) {
                case MY_CAMERA_PERMISSION_REQUEST:
                    if (grantResults.length > 0&& grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                        Intent intent = new
Intent(MediaStore.ACTION_IMAGE_CAPTURE);
                    startActivityForResult(intent, CAMERA_REQUEST_CODE);
                    } else {
                        Toast.makeText(this, "We need next permissions: Camera",
Toast.LENGTH_LONG).show();
                    }
                }
    }
```

Основний параметр, який нас тут цікавить - це масив grantResults. Він містить результати запитів відповідних дозволів. Кожен і-ий елемент з масиву grantResults відповідає і-ому елементу з масиву permissions. У нашому прикладі ми запитували всього один дозвіл і результат зберігається в самій першій клітинці масиву grantResults.

Якщо доступ дозволений, то ми точно також запускаємо камеру, а якщо - ні, то виводимо користувачеві повідомлення про помилку.

Тепер при натисканні на кнопку на екрані з'явиться спливаючий діалог з описом запитуваної permission'a.

В результаті маємо додаток для мобільних пристроїв, здатний звертатися до камери пристрою, отримувати зображення, та оброблювати його. Знімок головного екрану додатка на рисунку 3.2.

Рисунок 3.2 - Знімок головного екрану додатка по розпізнаванню образів

3.4 Режими роботи додатку

Для оптимізації та досягнення більш швидкої роботи додатка було прийнято рішення розділити образи розпізнавання на три групи:

Група “Дорожні знаки”:

Група “предмети побуту”:

Група “Двері та сходи”:

Так як робота додатку в декількох режимах одразу неможлива, значно зменшується навантаження на процесор мобільного пристрою, також зменшується споживання ОЗП. За рахунок цього вдалось прискорити роботу додатка, відповідно, і розпізнавання, до середнього значення в 2 сек для одного предмета.

ВИСНОВКИ

В даній атестаційній роботі було розроблено програмне забезпечення у вигляді Android-додатка для мобільних пристроїв. Додаток працює на основі ШНМ з архітектурою багатошарового персептрону, яка навчена розпізнавати образи предметів побуту.

Для даного програмного забезпечення було закладено функціонал самонавчання, що дозволило збільшити точність розпізнавання при використанні додатку.

Результати тестів та випробувань підтвердили, що представлене ПЗ розпізнає предмети побуту з точністю 91%, що є гарним показником.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Хайкин С.Нейронные сети: полный курс = Neural Networks: A Comprehensive Foundation. 2-е изд.—М.: Вильямс, 2006.–1104 с.
2. Ясницкий Л. Н.Введение в искусственный интеллект.—М.:Издат. центр «Академия», 2005.–176 с.
3. Каллан Р.Основные концепции нейронных сетей = The Essence of Neural Networks First Edition.—М.: Вильямс, 2001.–288 с.
4. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечёткие системы. //Д.Рутковская,пер.с польского И.Д.Рудинского. -М.: Горячаялиния–Телеком, 2007.–452с.
5. Тадеусевич Рышард, Боровик Барбара, Гончаж Томаш, Леппер Бартош. Элементарное введение в технологию нейронных сетей с примерами программ / Перевод И.Д.Рудинского.—М.:Горячая линия—Телеком, 2011.–408 с.
6. Круглов В. В.,Борисов В. В.Искусственные нейронные сети. Теория и практика.—М.: Горячая линия -Телеком, 2001.–382 с.
7. Головко В.А. Нейроинтеллект: теория и применения. Книга 1. Организация и обучение нейронных сетей с прямыми и обратными связями.1999.
8. Нейронныесети. Statistica Neural Networks: Пер. С англ.. М.: Горячая линия -Телеком, 2000. -182 с.
9. Bernard Widrow and Michael A. Lehr. Artificial neural networks of the perceptron and backpropagation family. Stanford University, CA 94305-4055.
10. G. Cybenko. Approximation by superpositions of sigmoidal functions. Mathematics of Control, Signals and Systems,1989.