МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кваліфікаційна наукова праця
на правах рукопису

ЗІВАКІН ВАЛЕРІЙ ДМИТРОВИЧ

УДК 519.652:519.254

ДИСЕРТАЦІЯ

«ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ РОЗПІЗНАВАННЯ ДАНИХ АЄРОЗЙОМКИ З ВИКОРИСТАННЯМ БАГАТОВИМІРНИХ ПРЕДСТАВЛЕНЬ»

122 «Комп’ютерні науки»

12 «Інформаційні технології»

Подається на здобуття ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ В.Д. Зівакін

Науковий керівник:

Приставка Пилип Олександрович

доктор технічних наук, професор

Київ – 2024

АНОТАЦІЯ

Зівакін В. Д. Інформаційна технологія розпізнавання даних аерозйомки з використанням багатовимірних представлень – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії в галузі знань 12 «Інформаційні технології», за спеціальністю 122 «Комп'ютерні науки». — Національний авіаційний університет, Київ, 2024.

Дисертаційна робота присвячена дослідженню вирішення задач класифікації та розпізнавання даних повітряної зйомки, а також дослідженню методів передобробки таких даних, а саме методом класифікації даних за допомогою нейронних мереж та передобробки даних на основі запропонованої моделі оцінки функції розподілу даних аерозйомки.

Проведено аналіз існуючих підходів до отримання багатовимірних представлень набору даних, що складається з цифрових зображень шляхом використання методів зниження розмірності багатовимірних даних, що в подальшому стало основою для розробки інформаційної технології розпізнавання цифрових зображень на основі розробленої моделі оцінки розподілу представлень даних навчального набору повітряної зйомки в багатовимірному просторі та дослідженням використання запропонованих методів попередньої обробки отриманих даних.

- Розроблено алгоритми по-класового видалення дублікатів з навчального набору даних аерозйомки, який збільшує ентропію в середині кожного класу за рахунок пропорційного зменшення коефіцієнтів ексцесу в просторі головних компонент (~в 2 рази при пороговому рівні схожості 0.95).

Даний алгоритм видалення дублікатів може використовуватись в інших наборах даних. Окрім того метод може бути модифікований для видалення дублікатів в межах усього датасету. Перевагою методу є те, що із збільшенням кількості видалених елементів, коефіцієнт ексцесу продовжує падати пропорційно, що свідчить про стабільну роботу методу.

o Розроблено алгоритм по-класової генерації нових зображень для набору даних аєрозйомки, який збільшує швидкість навчання моделей мінімум в 1.5 рази та за рахунок перерозподілу точності розпізнавання, підвищує розпізнавання “проблемних” класів приблизно на 10 відсотків. Даний алгоритм генерації нових даних може використовуватись для інших навчальних наборівта для аугментації наборів даних таким чином, що деякі класи можусь на 75 відсотків складатись із згенерованих даних, і при цьому досягається попередньо заявлене збільшення точності.

Ключові слова: БПЛА, штучний інтелект, аерозйомка, повітряне спостереження, машинне навчання, глибоке навчання, дані, датасет, цифрові зображення, навчальний набір, стиснення даних, зниження розмірності, багатовимірність, представлення, класифікація, розпізнавання, нейронна мережа, згортка, класифікатор, аутоенкодер, алгоритм, метод головних компонент, оцінка розподілу, модель, CNN.

ABSTRACT

Zivakin, V. D. (2024). Information Technology for Recognition of Aerial Survey Data Using Multidimensional Representations – A qualification scientific work in the form of a manuscript. Dissertation for the degree of Doctor of Philosophy in the field of knowledge 12 "Information Technologies", specialty 122 "Computer Science". National Aviation University, Kyiv, 2024.

The dissertation is dedicated to solving the tasks of classification and recognition of aerial survey data, as well as exploring methods for preprocessing such data, specifically classification using neural networks and data preprocessing based on the data distribution function estimation model. The analysis of existing approaches for obtaining multidimensional representations of a dataset consisting of digital images using methods of dimensionality reduction of multidimensional data was conducted, which subsequently formed the basis for developing information technology for recognizing digital images based on the developed data distribution estimation model for the training set of aerial survey data in a multidimensional

space and investigating the use of the proposed preprocessing methods for the obtained data.

- Algorithms for class-wise duplicate removal from the aerial survey training dataset were developed, which increase entropy within each class by proportionally reducing kurtosis coefficients in the principal component space (by approximately 2 times at a similarity threshold of 0.95). This duplicate removal algorithm can be used in other datasets. Moreover, the method can be modified to remove duplicates within the entire dataset. The advantage of the method is that as the number of removed elements increases, the kurtosis coefficient continues to fall proportionally, indicating the stable operation of the method.

- An algorithm for class-wise generation of new images for the aerial survey dataset was developed, which increases the training speed of models by at least 1.5 times and, due to the redistribution of recognition accuracy, improves the recognition of "problematic" classes by approximately 10 percent. This data generation algorithm can be used for other training sets and for augmenting datasets so that some classes can consist of 75 percent generated data, achieving the previously stated increase in accuracy.

Keywords: UAV, artificial intelligence, aerial survey, aerial observation, machine learning, deep learning, data, dataset, digital images, training set, data compression, dimensionality reduction, multidimensionality, representation, classification, recognition, neural network, convolution, classifier, autoencoder, algorithm, principal component method, distribution estimation, model, CNN.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Праці, в яких опубліковані основні наукові результати дисертації:

Статті у наукових фахових виданнях:

1. Зівакін В. Д. Таврійський науковий вісник. Серія: Технічні науки: Імітація одновимірних вибірок на основі існуючих з використанням

поліноміальних сплайнів. Херсонський державний аграрно- економічний університет. Херсон : Видавничий дім «Гельветика», 2021. Вип. 6. С. 23-30

2. Зівакін В. Д., Приставка П. О. Дослідження імітації двовимірних вибірок з використанням поліноміальних сплайнів. Проблеми інформатизації та управління. Національний авіаційний університет. Київ, 2023. Том 2 Вип. 74. С. 38-43.

Особистий внесок автора: реалізація програмної частини, проведення експерименту та оформлення висновків.

3. Зівакін В. Д., Приставка П. О. Використання сплайн-моделі в просторі латентних представлень при вилученні дублікатів із набору спостережень. Проблеми інформатизації та управління. Національний авіаційний університет. Київ, 2024. Том1 Вип. 77. С. 36-43..

Особистий внесок автора: реалізація програмної частини, проведення експерименту та оформлення висновків.

Статті у виданнях, які включено до міжнародних наукометричних баз:

1. O. Cholyshkina. P. Prystavka. O. Kozachuk. V. Zivakin. Training set AERIAL SURVEY for data recognition systems from aerial surveillance cameras. IX INTERNATIONAL CONFERENCE Information Technology and Implementation. Paper 79. 1 December, 2022, Kyiv.

Особистий внесок автора: формування детального опису датасету та списку вже проведених та можливих подальших досліджень на його основі .

Наукові праці, які додатково відображають наукові результати дисертації:

1. Зівакін В. Д. «Застосування машинного навчання в обробці даних

аерозйомки» Політ. Сучасні проблеми науки”. Тези доповідей XXI науково практичної конференції здобувачів вищої освіти і молодих учених 5-9 квітня 2021 року. КІБЕРБЕЗПЕКА, КОМП'ЮТЕРНА ТА ПРОГРАМНА ІНЖЕНЕРІЯ–2021.–С. 8.

2. Зівакін В.Д. «Дослідження імітації одновимірних вибірок з використанням поліноміальних сплайнів» Тези доповідей XXII науково практичної конференції здобувачів вищої освіти і молодих учених 10 травня 2022 року. КІБЕРБЕЗПЕКА, КОМП'ЮТЕРНА ТА ПРОГРАМНА ІНЖЕНЕРІЯ–2022.–С. 5-6.

3. Зівакін В.Д., Козачук, О. О. «Навчальний датасет для нейромережевої обробки даних повітряного спостереження» Proceedings The Tenth World Congress " Aviation in the XXI-st Century " Safety in Aviation And Space Technologies (September 28-30), Kyiv, 2022, 2.5.27– 2.5.29 pp.

4. Зівакін В. Д. «Моделювання нових наборів даних на основі нейромережевих представлень» Конференція Proceedings The Tenth World Congress " Aviation in the XXI-st Century " Safety in Aviation And Space Technologies (September 28-30), Kyiv, 2022, 2.5.30– 2.5.32 pp.

5. Зівакін В. Д. «Імітація одновимірних вибірок методом зворотньої функції з використанням поліноміальних сплайнів». Конференція «Політ. Сучасні проблеми науки». (4-7 квітня) Секція «Сучасні інформаційні та комунікаційні технології в авіації. Прикладна математика» 2023. – С. 7-8

6. Зівакін В., Козачук О., Приставка П. Дослідження перехресного нейромережевого розпізнавання даних супутникової та повітряної зйомки. Матеріали XVI міжнародної науково-технічної конференції «АВІА-2023». – К.: НАУ, 2023, 15.78-15.81 сс.

7. Зівакін В.Д. «Вилучення дублікатів зображень з наборів даних із використанням багатовимірних представлень» Тези доповідей XXII науково практичної конференції здобувачів вищої освіти і молодих учених 2-5 квітня

2024 року. Сучасні інформаційні та комунікаційні технології в авіації–2024.–С. 6-7.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Скорочення, термін, позначення	Пояснення
ЦЗ	Цифрове зображення
ШНМ	Штучна нейронна мережа
ЗНМ/CNN	Згорткова нейронна мережа / convolutional neural network
ГЗМ/GAN	Генеративні змагальні мережі/Generative adversarial networks
МГК/PCA	Метод головних компонент/principal component analysis

ВСТУП

На сьогодні в Україні продовжується десятирічна війна, активну фазу якої називають також «війною дронів». Безпілотні літальні апарати показали себе як ефективний, дешевий, безпечний та універсальний технічний засіб для вирішення задач різних типів та рівнів (від тактичного до стратегічного).

Забезпечення кількісної та якісної переваги в таких засобах – один з факторів досягнення перемоги та укріплення подальшої обороноздатності країни.

Основні задачі, що вирішуються при застосуванні БПЛА: навігація по оптичному каналу (в умовах постійної дії засобів РЕБ), визначення та супровід цілей.

В контексті вирішення цих задач суттєвим є компонент розпізнавання об’єктів місцевості та цільових об’єктів, для чого використовується нейромережева обробка.

Результати роботи відповідають запиту наступних розпоряджень та постанов кабінету міністрів:

- Розпорядження:
  - №600 «Деякі питання розвитку критичних технологій у сфері виробництва озброєння та військової техніки»

1. Технології образної інтерпретації, селекції та класифікації цілей для систем самонаведення високоточної зброї
2. Технології машинного навчання, штучного інтелекту, нейронних мереж для проектування, виробництва та експлуатації військової та спеціальної техніки

  - №1556-р «Про схвалення Концепції розвитку штучного інтелекту в Україні».

- Постанови:
  - №256 – «Про реалізацію експериментального проекту щодо здійснення оборонних закупівель безпілотних систем та засобів радіоелектронної боротьби вітчизняного виробництва».

- №476 – «Про затвердження переліку пріоритетних тематичних напрямів наукових досліджень і науково-технічних розробок на період до 31 грудня року, наступного після припинення або скасування воєнного стану в Україні», в переліку якої є наступні пункти:

1. Інтелектуальні інформаційно-керуючі технології діагностики, експлуатації та ремонту військової та спеціальної техніки
2. Технології кодування, передачі та отримання (автоматичного розпізнавання, обробки, аналізу, генерації, візуалізації) інформації. Технології криптографічного захисту інформації
3. Методи та засоби інформаційно-аналітичного та нормативно-методичного забезпечення процесів прийняття рішень у сфері національної безпеки і оборони. Автоматизовані системи управління

Метою роботи є розробка інформаційної технології розпізнавання цифрових зображень на основі розробленої моделі оцінки розподілу представлень даних навчального набору повітряної зйомки в багатовимірному просторі та використанням запропонованих методів попередньої обробки.

Для цього сформульовано комплекс наступних науково-технічних задач:

1. Аналіз існуючих методів обробки даних аерозйомки.
2. Розробка та дослідження моделі розподілу представлень даних аерозйомки в багатовимірних просторах.
3. Розробка та дослідження методу видалення дублікатів з навчального набору даних аерозйомки для покращення результатів розпізнавання.
4. Розробка та дослідження генерації нових зображень для навчального набору даних аерозйомки для покращення результатів розпізнавання.
5. Створення інформаційної технології розпізнавання даних аерозйомки на основі запропонованої моделі розподілу представлень навчального набору даних аерозйомки та методів генерації нових даних та видалення дублікатів.
6. Тестування та оцінка ефективності розробленої технології розпізнавання цифрових зображень на реальних даних повітряного

спостереження.

Об'єкт дослідження – процес розпізнавання, класифікації та аналізу даних навчального набору аерозйомки з використанням багатовимірних представлень цифрових зображень.

Предмет дослідження – методи та технології обробки зображень аерозйомки, методи машинного та глибокого навчання, методи статистичного аналізу даних, методи не параметричної апроксимації, технології програмування.

Наукова новизна одержаних результатів полягає у наступному::

- Вперше запропоновано математичну модель розподілу представлення даних (цифрових зображень) повітряного спостереження в багатовимірному просторі, що будується на оцінці функції та щільності розподілу латентних представлень за допомогою багатовимірних локальних поліноміальних сплайнів близьких до інтерполяційних в середньому на основі B-сплайнів 2-го порядку. Самі представлення отримуються із цифрових зображень шляхом використання мереж-автокодувальників, та бажаним переходом до простору маргінальних розподілів за допомогою МГК. Модель дозволяє мати формальний опис неоднорідних даних для подальших перетворень.
- Вперше запропоновано метод видалення дублікатів з набору зображень даних повітряного спостереження на основі сплайн-моделі розподілу даних в просторі представлень, який регулується із введенням граничного рівня схожості зображень. Виявлено, що при використанні методу із граничним рівнем схожості 0.95 ентропія зображень в просторі класів збільшується за рахунок того, що коефіцієнт ексцесу в сереньому зменшується вдвічі вдвічі, що знижує ризик до перенавчання моделі.
- Вперше запропоновано метод імітаційного моделювання нових зображень на основі моделі розподілу навчального набору даних аерозйомки в просторі багатовимірних представлень, який дозволяє за рахунок використання змодельованих зображень підвищити швидкість навчання мінімум в 1.5 рази

- Розроблено інформаційну технологію розпізнавання та класифікації даних аерозйомки, яка в змозі тренувати різноманітні моделі мереж класифікаторі, які досягають точності розпізнавання не менше 0.9.

Практичне значення та використання результатів дисертаційного дослідження полягає у тому, що розроблена інформаційна технологія може використовуватись в різноманітних системах розпізнавання та системах автоматизації наповнення наборів даних, крім того:

- Розроблено алгоритми по-класового видалення дублікатів з набору даних аерозйомки, який збільшує ентропію в середені кожного класу за рахунок пропорційного зменшення коефіцієнтів ексцесу в просторі головних компонент (~в 2 рази при пороговому рівні схожості 0.95).
- Розроблений алгоритм видалення дублікатів може використовуватись в інших наборах даних.
- Розроблено алгоритм по-класової генерації нових зображень для набору даних аерозйомки, який збільшує швидкість навчання моделей мінімум в 1.5 рази та за рахунок перерозподілення точності розпізнавання підвищує розпізнавання “проблемних” класів приблизно на 10 відсотків.
- Розроблений алгоритм генерації нових даних може використовуватись для інших навчальних наборів.
- Розроблений алгоритм може використовуватись для аугментації наборів даних таким чином, що деякі класи можусь на 75 відсотків складатись із згенерованих даних, і при цьому досягається попередньо заявлене збільшення точності.
- Окремі модулі розробленої інформаційної технології можуть використовуватись в інших системах, для покращення результатів класифікації та розпізнавання.
- Розроблено програмне забезпечення для первинного аналізу та генерації різновимірних наборів даних на основі сплайн-моделі оцінки розподілу випадкових величин (.Net, C#).

- Розроблено програмне забезбечення (Python, Pytorch) для використання та навчання моделей нейронних мереж-класифікаторів та автокодувальників для різної кількості класів та отримання латентних представлень даних.
- Для програмного забезпечення (Python) розроблено модуль для по-класового видалення дублікатів на основі запропонованого методу та розробленого алгоритму видалення дублікатів.
- Для програмного забезпечення (Python) розроблено модуль по-класової генерації нових зображень на основі запропонованого методу імітаційного моделювання та розробленого алгоритму. Для використання в одному середовищі сплайн-оцінка буда дореалізована мовою Python.

Дисертація складається з анотації, вступу, чотирьох розділів, висновків, списку використаних джерел. Повний обсяг дисертації становить 114 сторінок, із них 96 – основного тексту. Робота містить 15 рисунків, 5 таблиць. Список використаних джерел налічує 66 найменувань.

РОЗДІЛ 1.
ОГЛЯД ІСНУЮЧИХ МЕТОДІВ ТА ТЕХНОЛОГІЙ

В роботі використовується навчальний набір даних повітряної зйомки. Нагадаємо, що дані аерозйомки – це зображення та відео, отримані за допомогою записуючих пристроїв, що розташовані на бортах літальних апаратів (тобто бортовими засобами). Якість отриманих даних обумовлюється технічними характеристиками засобів зйомки, літальних апаратів та зовнішніми умовами. Так як під час повітряної зйомки завжди є фактори, які негативно впливають на якість зображення (наприклад, вібрація літального апарату), то модель отриманого зображення представляє собою згортку функції відліку системи реєстрації із ідеальним зображенням[22]:

$$I(ra) = \int O(ia)H(ra - ia)d(ia),$$

де ra – зона (площина) реєстрації зображення, а ia – картина прощина.

Набір таких зображень можна використовувати для вирішення наступних задач:

1. Попередня обробка – набір процедур, які застосовуються до зображень перед подальшою обробкою, такі як фільтрація та покращення якості, масштабування та нормалізація, корекція спотворень.
2. Створення фотопланів – картографічних зображень, які точно відображають місцевість в певному маштабі.
3. Сегментація - поділ зображення на частини або сегменти, що містять об'єкти або області зі спільними характеристиками. У контексті аерозйомки, сегментація дозволяє виділити окремі об'єкти та аналізувати ландшафт.
4. Розпізнавання та класифікація – ідентифікація та категоризація об'єктів на зображенні.
5. Визначення координат місцевості – процес встановлення точних географічних координат для різних точок на зображенні.

В контексті створення інформаційної технології розпізнавання даних аерозйомки в роботі вирішувалася задача класифікації даних. На сьогодні, під вирішення цієї задачі створено або легко адаптується цілий ряд класів штучних нейронних мереж:

1. Конволюційні нейронні мережі (CNNs)[37,38] є стандартом для задач класифікації зображень і розпізнавання образів. Вони ефективно витягують ієрархічні візуальні особливості з зображень через послідовність конволюційних та пулінгових шарів, що дозволяє їм виявляти складні візуальні патерни на різних рівнях абстракції. Завдяки використанню локальних рецептивних полів, конволюційні шари здатні виявляти низькорівневі особливості, такі як краї, текстури і кути, які далі об'єднуються у вищі рівні абстракції, що репрезентують більш складні форми і об'єкти. Ця багаторівнева структура дає CNN перевагу у порівнянні з традиційними методами обробки зображень, оскільки дозволяє навчати моделі без необхідності ручного створення особливостей. Крім того, застосування пулінгових шарів зменшує розмірність представлення даних, зменшуючи кількість параметрів і покращуючи загальну продуктивність моделей. Сучасні досягнення у розвитку CNN, такі як архітектури ResNet [32]і Inception, продемонстрували високу точність у різних завданнях обробки зображень і значно вплинули на галузь комп'ютерного зору .
2. Мережі на основі автоенкодерів є важливим інструментом у галузі машинного навчання та обробки зображень. Автоенкодери [35]— це тип нейронних мереж, які навчаються стискати вхідні дані в компактніші внутрішні представлення і потім відтворювати вихід, який якомога ближчий до оригінального вхідного зображення. Ці мережі корисні для задач, де важливо виділити ключові особливості зображення для подальшої класифікації або аналізу. Зокрема,

автоенкодери успішно застосовуються для зниження розмірності даних, очищення зображень від шуму, а також для створення генеративних моделей, здатних генерувати нові зображення на основі вивчених патернів. Крім того, варіаційні автоенкодери [40](VAE) та автоенкодери з регуляризацією розширюють можливості стандартних автоенкодерів, дозволяючи отримувати латентні представлення, що підходять для моделювання складних розподілів даних і створення більш реалістичних синтетичних зображень. Ці інновації підвищують ефективність мереж на основі автоенкодерів і роблять їх незамінними інструментами у багатьох областях застосування .

3. Глибокі залишкові мережі (ResNets) є[35] однією з ключових архітектур у сучасному глибокому навчанні, відомою своїми високими результатами у складних задачах класифікації. Вони використовують архітектуру з "залишковими блоками", які дозволяють сигналам обходити один або кілька шарів через так звані "пропускні з'єднання". Це допомагає вирішити проблему зникнення градієнтів у дуже глибоких нейронних мережах, що часто виникає через експоненційне згасання сигналу в традиційних багатошарових мережах. Завдяки цим пропускним з'єднанням, ResNets можуть навчатися ефективніше і забезпечувати стабільний потік градієнтів навіть у дуже глибоких мережах, що дозволяє будувати моделі з сотнями або навіть тисячами шарів. Це значно покращує загальну точність класифікації в складних задачах, таких як розпізнавання об'єктів у зображеннях, сегментація і аналіз великих наборів даних. Впровадження ResNets стало важливим кроком у розвитку глибокого навчання, надавши дослідникам інструменти для створення більш потужних і точних моделей, що можуть вирішувати задачі, які раніше були за межами досяжності сучасних технологій.

4. Рекурентні нейронні мережі (RNNs) і Лонг-шорт термін меморі мережі (LSTMs) є потужними інструментами для обробки послідовних даних. Хоча ці типи мереж зазвичай асоціюються з обробкою тексту або аудіо, вони також можуть бути адаптовані для аналізу зображень, особливо в контексті відео або інших послідовних візуальних даних. RNNs і LSTMs здатні аналізувати зміни в часі, що робить їх корисними для задач, як-от розпізнавання дій або подій у відео. Завдяки їхній здатності зберігати і передавати інформацію з попередніх моментів часу, ці мережі можуть ефективно моделювати залежності у часових рядах даних. Це особливо важливо для відеоаналізу, де необхідно враховувати контекст попередніх кадрів для правильної інтерпретації поточних подій. LSTMs, завдяки своїй спеціальній структурі, можуть зберігати важливу інформацію на довгих часових проміжках і ігнорувати менш релевантну, що робить їх більш ефективними у порівнянні зі стандартними RNNs. Ці властивості роблять RNNs і LSTMs незамінними інструментами для різних завдань, пов'язаних з аналізом динамічних візуальних даних, таких як розпізнавання жестів, відстеження об'єктів та інші застосування в області комп'ютерного зору..
5. Складні архітектури детекторів, такі як YOLO (You Only Look Once) і SSD (Single Shot Multibox Detector), є провідними рішеннями для задач детекції об'єктів. Ці мережі призначені для завдань, де необхідно не тільки класифікувати візуальні об'єкти на зображенні, але й точно вказати їх розташування. Вони використовують спеціалізовані архітектури, які дозволяють виконувати детекцію в реальному часі з високою точністю. YOLO, наприклад, розділяє зображення на сітку і одночасно передбачає межі об'єктів і їх класифікацію, що дозволяє досягти високої швидкості обробки. Це робить YOLO особливо корисним для додатків, де потрібна миттєва

реакція, таких як системи безпеки та автономні транспортні засоби. SSD, з іншого боку, використовує багаторазові вікна з різними розмірами і співвідношеннями сторін, що дозволяє йому ефективно виявляти об'єкти різних масштабів і форм. Ця архітектура також забезпечує високу точність і швидкість, що робить її популярним вибором для мобільних і вбудованих систем. Використання таких передових технологій, як YOLO і SSD, значно розширює можливості комп'ютерного зору, дозволяючи створювати більш інтерактивні та ефективні рішення для широкого спектра застосувань.

1.1 Огляд методів класифікації цифрових зображень

На сьогоднішній день до вирішення задачі класифікації цифрових зображень повітряної зйомки існує дві основні умови:

- Можливість обробки зображень в режимі реального часу.
- Відсутність в потребі значної обчислювальної потужності.

Основним засобом класифікації в сфері комп’ютерного зору та машинного навчання стали штучні нейронні мережі, зокрема глибокі навчальні моделі, які демонструють значні результати в розпізнаванні образів, відділенні семантично значущих категорій та автоматичному анотуванні. Ці технології змінили підходи до традиційних задач обробки зображень та створили нові напрями досліджень та комерційних застосувань.

Прогрес у цій області був прискорений через появу потужних обчислювальних ресурсів та великих наборів даних, що дозволяють тренувати складні моделі на мільйонах зображень. Процес навчання ШНМ становить певну складність, але вже навчена модель не потребує значних ресурсів (наприклад для розгортання на мікрокомп’ютері) і при цьому здатна класифікувати ЦЗ в режимі реального часу. В процесі розвитку галузі з’явився цілий клас нейронних мереж, націлений на роботу із зображеннями – згорткові

(конволюційні) нейронні мережі (CNN), які замість обробки усього зображення працюють із визначеними в процесі обробки особливостями.

Така динаміка сприяла появі різних видів штучних згорткових нейронних мереж для вирішення задачі класифікації:

- У [1] описано AlexNet – згорткова нейронна мережа (CNN), яка мала ключову роль у відновленні інтересу до глибокого навчання в області комп'ютерного зору. Ця мережа була розроблена і представлена у 2012 році на конкурсі ImageNet Large Scale Visual Recognition Challenge (ILSVRC), де вона значно перевершила інші методи з рекордно низьким показником помилок у 15.3%, порівняно з 26.2% у найближчого конкурента. AlexNet складається з п'яти згорткових шарів, за якими йдуть три повнозв'язні шари, та використовує методи нормалізації, пулінгу та дроп-аут для підвищення продуктивності та запобігання перенавчанню. Впровадження цієї моделі стало важливим етапом у розвитку глибокого навчання і встановило нові стандарти для комп'ютерного зору, демонструючи потенціал глибоких нейронних мереж для обробки складних візуальних завдань. Успіх AlexNet стимулював подальші дослідження та розробки в галузі, приводячи до створення більш потужних архітектур, таких як VGG, ResNet та інші, які продовжують підвищувати точність і ефективність обробки зображень..
- В [2] подано VGG (Visual Geometry Group) – архітектуру з дуже глибокими згортковими вставками. VGG є однією з найбільш впливових архітектур у галузі глибокого навчання та комп’ютерного зору. Її простота і висока продуктивність зробили її популярною не тільки для класифікації зображень, але й як основу для більш складних задач обробки зображень, таких як детектування об’єктів та сегментація. Основною особливістю

VGG є використання невеликих (3×3) згорткових фільтрів, які застосовуються у великій кількості шарів для побудови глибоких моделей. Цей підхід дозволяє мережі вивчати більш детальні та складні візуальні представлення, що сприяє підвищенню точності класифікації. Архітектура VGG, зокрема моделі VGG-16 і VGG-19, стали основою для багатьох інших досліджень і розробок у галузі, демонструючи високу ефективність у різних застосуваннях комп’ютерного зору. Її впливовість підтверджується широким використанням у наукових дослідженнях та практичних додатках, що робить VGG важливим етапом у розвитку глибоких нейронних мереж

- GoogLeNet, також відома як Inception, представлена у [3], мала значний вплив на подальші дослідження та розробки в галузях глибокого навчання та комп'ютерного зору. Архітектура Inception стала основою для ряду подальших ітерацій та удосконалень, включаючи Inception v2, Inception v3, інтеграцію з residual connections (створення Inception-ResNet) та інші. Основною особливістю Inception є використання модулів, які виконують згортки з різними розмірами ядер одночасно, що дозволяє захоплювати різні рівні абстракції в межах одного шару. Цей підхід допомагає зменшити обчислювальну складність моделі, забезпечуючи високу продуктивність при відносно невеликій кількості параметрів. Вдосконалені версії архітектури, такі як Inception v3, включають додаткові техніки оптимізації, такі як факторизація згорток і більш ефективні методи нормалізації, що ще більше покращує точність і ефективність моделей. Інтеграція Inception з залишковими блоками (Inception-ResNet) об'єднує переваги обох архітектур, що дозволяє створювати ще глибші мережі без проблем зникнення градієнтів і зберігаючи високу

точність обробки зображень. Ці нововведення роблять архітектуру Inception важливим етапом у розвитку сучасних методів комп'ютерного зору та глибокого навчання.

- ResNet (Residual Network) [4] – вперше представлена у 2015 році. Здобула широке визнання після того, як виграла конкурс ILSVRC того ж року. Основною ідеєю ResNet є використання залишкових блоків, де вхід кожного блоку додається до його виходу через оператор прямого зв'язку (skip connection). Це дозволяє сигналам "перестрибувати" через один або кілька шарів без будь-яких перетворень, що знижує проблему зникнення або вибуху градієнтів при тренуванні дуже глибоких мереж. Завдяки цьому інноваційному підходу, ResNet успішно досягає великих глибин, які були недосяжними для попередніх архітектур, зберігаючи високу точність і стабільність під час навчання. Ця архітектура не тільки підвищила продуктивність у задачах класифікації зображень, але також стала базовою для багатьох інших застосувань, включаючи розпізнавання об'єктів, сегментацію зображень і інші завдання комп'ютерного зору. Успіх ResNet стимулював розвиток нових архітектур і підходів у глибокому навчанні, зокрема таких, як ResNeXt та DenseNet, які будують на ідеях залишкових з'єднань для досягнення ще кращих результатів.
- enseNet (Densely Connected Convolutional Network) [5] є ще однією інноваційною архітектурою у сфері глибокого навчання, яка була представлена у 2017 році. DenseNet, завдяки своїй унікальній структурі та високій ефективності, знайшла застосування у багатьох областях обробки зображень, включаючи медичну візуалізацію, аналіз аерофотознімків та інші завдання комп'ютерного зору. Основна ідея DenseNet полягає у щільних з'єднаннях між шарами, де кожен шар отримує як вхід всі

попередні шари і передає свої ознаки всім наступним шарам. Це дозволяє значно покращити передавання градієнтів через мережу, зменшуючи проблему зникнення градієнтів та сприяючи повторному використанню ознак, що веде до більш ефективного навчання. Завдяки цьому підходу DenseNet досягає високої точності з меншою кількістю параметрів у порівнянні з традиційними архітектурами. Використання цієї архітектури в різних галузях підтверджує її універсальність і ефективність у розв'язанні складних задач комп'ютерного зору.

Кожну з цих моделей або їхніх версій можна адаптувати під класифікацію даних повітряної зйомки. Основним компонентом для цієї адаптації є навчальний набір.

1.2 Навчальний набір «Аерозйомка»

Створення, наповнення та використання навчального набору є ключовим моментом не тільки для розпізнавання та класифікації даних з камер повітряного спостереження та адаптації відомих моделей під вирішення конкретної задачі, як сказано вище. Загалом, при використанні методів глибокого навчання, наявність представницького набору даних є обов’язковою для вирішення будь-якої задачі.

Тому, починаючи з 2018 року на кафедрі прикладної математики Національного авіаційного університету було розпочато створення та наповнення навчального набору «Аерозйомка» [6]. В першу чергу зазначений набір даних розглядається як стартовий майданчик для розгортання систем машинного навчання, їх дослідження та постійного вдосконалення. За час створення набору «Аерозйомка» вчені кафедри, студенти та аспіранти ставили та вирішували різноманітні задачі, вдосконалюючи, як сам навчальний набір, так і інформаційні технології на його основі. На момент написання [6] набір містив у собі 16 класів формату 64x64 пікселя. На момент написання цього

тексту в ньому вже 42 класи, в кожному з яких міститься від 6 до 8 тисяч зображень. Окрім того існують версії набору розмірності 128x128 та 256x256 пікселів.

В продовж проведення досліджень в рамках цієї роботи, використовувалися різні частини даного датасету. Наприклад, в роботі [6] використовувалося 10 класів загальною кількість майже 22 тисячі зображень. Найбільші експерименти проводилися із структурою, яка містила в собі 33 класи. Її опис міститься в таблиці 1.1.

Окремі частини або увесь набір даних використовується з моменту формування його першої версії у 2018 році і за цей час стали основою для досліджень десятків робіт як бакалаврського і магістерського рівня, так і окремих наукових праць. Приклади зображень окремих класів представлені у таблиці 1.2.

Таблиця 1.1 – опис навчального набору на базі датасету «Аерозйомка».

Позначення класу	**Клас**	**Опис**
AFV	Бойова броньована машина	Військові транспортні засоби, призначені для бойових дій, які можуть включати легкоброньовані машини або бойові машини підтримки.
APC	Бронетранспортер	Броньований транспортний засіб, призначений для перевезення військовослужбовців та їхнього обладнання на поле бою, не призначений для прямого бою.
Artillery	Артилерія	Великокаліберні ствольні системи, призначені для далекобійного обстрілу.
Autumn_Forest	Осінній ліс	Лісові масиви з характерним осіннім забарвленням листя.
Beach	Пляж	Прибережна зона з піщаним або гальковим покриттям.
Bridge	Міст	Конструкція, що перетинає водоймище або інші перешкоди для забезпечення проходу.
Buildings	Будівлі	Структури, використовувані для житла, роботи, навчання, адміністрації та інших потреб.

Продовження таблиці 1.1

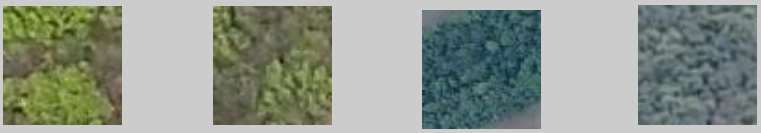
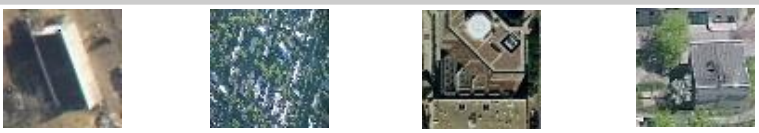
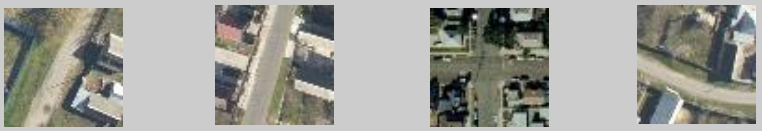
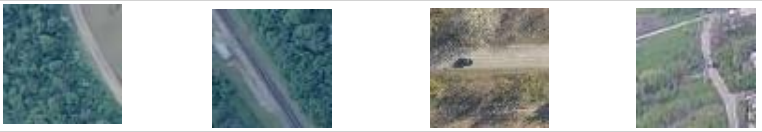
Buildings_Mid_Trees	Будівлі серед дерев	Будівлі, розташовані в лісистій місцевості, часто частково приховані деревами.
Civil_Vehicles	Цивільні транспортні засоби	Транспорт, використовуваний у повсякденному житті, що не є військовим.
Damaged_Buildings	Пошкоджені будівлі	Структури, які зазнали руйнувань чи пошкоджень.
Explosions	Вибухи	Виразні спалахи або розриви, часто від боєприпасів.
Hangars	Ангари	Великі будівлі для зберігання авіації або великогабаритної техніки.
Heavy_Armored	Важко броньовані	Бронетехніка з підвищеним рівнем захисту, здатна витримати серйозні пошкодження.
IFV	Бойова машина піхоти	Бронетехніка, призначена для транспортування піхоти, яка також здатна підтримувати піхоту вогнем.
Landing	Зона посадки	Визначена область для посадки повітряних чи водних транспортних засобів.
MLRS	Реактивна система залпового вогню	Артилерійська система, здатна запускати декілька реактивних снарядів одночасно.
Mountains	Гори	Великі природні підняття землі, вищі та крутіші за пагорби.
Narrow_Dirt_Roads	Вузькі ґрунтові дороги	Невеликі дороги, не призначені для інтенсивного трафіку, часто з природнім покриттям.
Non_ Vegetation_Field	Поле без рослинності	Відкриті простори без значної кількості рослин.
Pilars	Стовпи	Вертикальні конструктивні елементи, що підтримують структуру.
River	Річка	Природний водний потік, який тече по земній поверхні.
Roads	Дороги	Шляхи, призначені для руху транспорту.
Self_Propelled_Artillery	Самохідна артилерія	Артилерійські системи, встановлені на самохідних платформах.
Storage_Tanks	Зберігаючі резервуари	Великі контейнери для зберігання рідин або газів.
Summer_Forest	Літній ліс	Лісові масиви, які характеризуються літнім періодом року.
Tanks	Танки	Важкі броньовані бойові машини, озброєні гарматами.
Technics_Footprints	Технічні сліди	Сліди від технічних пристроїв або машин

Продовження таблиці 1.1

Traces	Сліди	Маркери або відмітки на поверхнях, що вказують на активність або рух.
Trenches	Окопи	Глибокі ями або канави, викопані для оборонних цілей або як частина будівельних робіт.
Vegetation_Field	Поле з рослинністю	Великі простори, покриті різноманітною рослинністю.
Water	Водойми	Поверхні водойм, такі як ставки, озера або моря.
Wide_Dirt_Roads	Широкі ґрунтові дороги	Більш широкі дороги, які не мають твердого покриття, здебільшого в сільських або малонаселених регіонах.
Winter_Forest	Зимовий ліс	Лісові масиви під час зимового періоду з покривом снігу або інших зимових особливостей.

Наявність навчального набору дає змогу навчати моделі штучних нейронних мереж, як ті що згадані вище, так і кастомні моделі, створені з нуля або на основі існуючих.

Таблиця 1.2 – приклади зображень класів навчального набору

Зображення	**Клас зображення**
	Ліси
	Будівлі
	Дорога серед будівель
	Дорога серед дерев
	Стовпи

Та на сьогодні навчання навіть складної моделі вже не є достатнім для покращення результатів у вирішені задач розпізнавання та класифікації, тому актуальним є пошук інших способів досягнення нових результатів.

1.3 Зниження розмірності даних

Ідея перетворення цифрових зображень (ЦЗ) для спрощення обробки або для вирішення конкретної задачі не є новою. Тривіальний приклад – перехід від звичайного триканального зображення до того ж самого, але в градаціях сірого, що спрощує обробку, при збереженні важливих особливостей зображення. Однак, зображення в градаціях сірого – це все ще слабоформалізовані сутності і для глибокого вивчення наборів таких даних, які до того ж мають поділ на класи, було б доречно застосовувати інструменти відображення ЦЗ у простір дійсних чисел, який уможливить застосування методів статистичного аналізу та машинного навчання, використання відомих моделей розподілів випадкових величин або ввід власних на основі проведення певних оцінок. Тобто, так як ЦЗ є реалізацією деякої випадкової величини, актуальним є спосіб їхнього відображення у простір дійсних чисел деякої розмірності(отримання багатовимірного представлення), який наддасть змогу провести оцінювання (наприклад, щільності розподілу) і на його основі ввести модель розподілу ЦЗ у просторі представлень. Але методи оцінки багатовимірних величин – це окреме питання, таке ж важливе, як і спосіб відображення даних у простори меншої розмірності.

На сьогодні, способи відображення ЦЗ у багатовимірний простір дійсних чисел можна умовно поділити на дві групи: методи, в яких використовуються обробка нейронними мережами (найчастіше - аутоенкодерами) та ті, в яких використовуються класичні методи машинного навчання та аналізу даних.

1.3.1 Методи зниження розмірностей

Методи, які не використовують нейромережеву обробку, мають різне математичне підґрунтя та цілі: якісна візуалізація, перехід до статистично чи кореляційно незалежних представлень, тощо.

Наприклад, у [7] детально розглядається модифікація методу стохастичного вбудовування сусідів (SNE) з t-розподілом (t-SNE), який автори позиціонують як потужний інструмент для візуалізації багатовимірних даних. t-SNE є ітеративним методом, який перетворює багатовимірні дані в дво- або тривимірний простір, мінімізуючи дивергенцію Кульбака-Лейблера між початковими даними та їхніми відображеннями. Цей підхід надає можливість максимально зберігати взаємозв'язки між точками при зменшенні розмірності простору.

Метод t-SNE стає особливо корисним для візуалізації високорозмірних даних, так як він здатен ефективно відокремлювати схожі дані та зберігати їхню геометричну структуру в низькорозмірному просторі. Це дає можливість дослідникам і аналітикам глибше розуміти структуру даних і виявляти складні взаємозв'язки між їхніми компонентами.

У [8] представлений алгоритм Uniform Manifold Approximation and Projection (UMAP) – ще один інноваційний метод зменшення розмірності та візуалізації багатовимірних даних, заснований на теорії топологічних даних. У відмінність від t-SNE, який використовує попарний розподіл імовірностей для оцінки подібності між спостереженнями, UMAP будує зважений граф, де ребра з’єднують лише найближчі сусіди (це один з гіперпараметрів методу). Множина ребер графу утворює нечітку множину, і функція належності визначає ймовірність існування ребра між вершинами. У маломірному просторі UMAP створює новий граф, де ребра наближаються до ребер початкового графу за допомогою тієї ж дивергенції Кульбака-Лейблера. Таким чином, UMAP подібний до t-SNE, але має іншу математичну основу, що

дозволяє йому бути більш ефективним для обробки великих наборів даних і краще зберігати глобальні зв'язки між даними, наприклад, між різними класами.

UMAP відкриває нові можливості для аналізу високорозмірних даних, зокрема застосовується в медичній діагностиці, аналізі аерофотознімків та інших сферах комп'ютерного зору, де точність та глибина аналізу грають критичну роль.

У [9] подано опис алгоритму локального лінійного вкладення (LLE) - метод зниження розмірності, що використовується в машинному навчанні та статистиці для проекції даних високої розмірності у простір низької розмірності, зберігаючи при цьому важливі геометричні структури. LLE відмінно підходить для обробки нелінійних структур даних і широко застосовується в задачах, де потрібно виявити внутрішню структуру даних, таких як розпізнавання образів, аналіз часових рядів та біоінформатика.

Ключова ідея LLE полягає в тому, що кожна точка даних може бути апроксимована лінійною комбінацією її найближчих сусідів, і цю локальну геометричну структуру можна зберегти у просторі з нижчою розмірністю.

Алгоритм можна розділити на три основні кроки:

- Вибір сусідів: для кожної точки даних у вихідному просторі визначається набір найближчих сусідів. Зазвичай використовується евклідова відстань, і кількість сусідів задається користувачем як параметр алгоритму.
- Обчислення ваг: для кожної точки обчислюється, як може бути представлена як лінійна комбінація її сусідів. Для цього мінімізується квадратична помилка між точкою та лінійною комбінацією її сусідів, при цьому сума ваги для кожної точки повинна дорівнювати одиниці. Це створює матрицю ваг, яка відбиває внесок кожного сусіда в апроксимацію цієї точки.

- Вкладення в низькорозмірний простір: з використанням отриманої матриці a_{ar} будується вкладення в новий простір меншої розмірності таким чином, щоб мінімізувати розбіжність між кожною точкою та лінійною комбінацією її сусідів у новому просторі. Це досягається шляхом мінімізації вартісної функції, яка оцінює, наскільки добре локальні властивості кожної точки зберігаються у просторі зниженої розмірності.

Мінусом методу є відсутність прямої можливості для повернення в простір початкової розмірності.

[10] – є всебічним посібником з використання методу головних компонент (PCA), класичного методу зменшення розмірності у машинному навчанні, який базується на обчисленні власних векторів та власних значень. На відміну від t-SNE та UMAP, головною перевагою PCA є ортогональність вихідних компонент між собою, тобто їхня незалежність, що дозволяє в подальшій обробці працювати окремо з кожною компонентною. Окрім того, PCA має реалізацію прямого та зворотного ходу – можливість повернення даних до вихідної розмірності. Саме ці властивості були ключовими у вирішенні використовувати метод головних компонент як один з інструментів у інформаційній технології, що пропонується.

Метод незалежних компонент (ICA) [11] – спосіб розділення багатовимірного сигналу на адитивні підкомпоненти. Метод ґрунтується на припущенні, що компоненти є незалежними і мають не-гаусовий розподіл. Таким чином, якщо результатом PCA є компоненти, які пояснюють частку дисперсії даних, яка послідовно зменшується, є ортогональні і можуть бути інтерпретовані як напрямки, у яких дані мають найбільшу варіативність, то результат ICA – компоненти, що є найбільш статистично незалежними одна від одної, що може не обов'язково відповідати напрямкам найбільшої дисперсії. Статистична незалежність компонент є корисною в розрізі

спрощення обробки і роботи з окремими компонентами, але без впевненості у відсутності гаусових розподілів використання методу ICA стає недоречним. Тому, як сказано вище, перевага була надана методу PCA.

1.3.2 Нейромережева обробка навчального набору

Латентне представлення — це концепція в машинному навчанні і штучному інтелекті, що стосується внутрішнього представлення даних у скомпресованому або абстрактному вигляді, яке нейронна мережа або інша модель обчислює під час навчання. Латентні представлення використовуються для вилучення суттєвої інформації з великих і складних датасетів, зменшення розмірності даних та для покращення ефективності обчислень.

Виділяються наступні аспекти латентних представлень:

- Внутрішній простір: латентне представлення створює "внутрішній простір" або "прихований простір" (latent space), де дані представлені у вигляді векторів. Ці вектори зазвичай мають набагато меншу розмірність порівняно з вхідними даними.
- Згортання інформації: латентне представлення згортає або конденсує вхідні дані так, що суттєва інформація зберігається, а неважлива або шумова інформація відкидається. Це дозволяє моделі зосередитися на релевантних особливостях даних.
- Застосування: латентні представлення широко використовуються в задачах зорового перцепції (наприклад, розпізнавання образів, класифікація зображень), обробці мови (наприклад, моделі вкладення слів, переклад), рекомендаційних системах та інших сферах, де потрібно ефективно обробляти великі обсяги даних.
- Властивості: латентні представлення можуть мати різні властивості залежно від типу моделі та її архітектури. Наприклад, у варіаційних автоенкодерах (VAE) латентний простір моделюється як розподіл імовірностей, що дозволяє генерувати нові дані, схожі на навчальний набір.

Таким чином, у визначеннях машинного та глибокого навчання, латентне представлення – це результат відображення даних з навчального набору у простір дійсних чисел, який представляє собою вектори відповідної розмірності, тобто є різновимірним представленням даних. Такі представлення можуть використовуватись як в подальшій обробці моделями штучних нейронних мереж, так і іншими інструментами, наприклад, методами, що згадані вище.

Основними типами моделей для отримання латентних представлень є аутокодувальники (аутоенкодери) [12] - моделі штучних нейронних мереж, які використовуються для зменшення розмірності даних та отримання латентних представлень. Вони працюють за принципом навчання без вчителя (unsupervised learning) і складаються з двох основних компонентів: енкодера і декодера.

Енкодер відповідає за перетворення вхідних даних у латентне представлення. Він зменшує розмірність даних, захоплюючи лише найважливішу інформацію у формі вектора у латентному просторі. Цей вектор часто має значно меншу розмірність порівняно з оригінальними даними, і він слугує стислим представленням вхідних даних.

Декодер виконує зворотне завдання. Він використовує латентне представлення, створене енкодером, і намагається відновити оригінальні дані на основі цього представлення. Мета декодера — відтворити вхідні дані якомога точніше з врахуванням інформації, закодованої у латентному просторі.

Процес навчання автоенкодера включає мінімізацію втрати між оригінальними даними та їх відновленою версією, що виводиться декодером. Функція втрат (різниця між входом і виходом) зазвичай є середньоквадратичною помилкою (MSE) або бінарною ентропією перехрестя, залежно від типу даних. Оптимізація цієї функції дозволяє енкодеру навчитися виводити стиснене та інформативне представлення початкових даних.

Автокодувальники мають декілька варіацій, які створені для вирішення додаткових задач паралельно із стандартним стисканням/відновленням. Наприклад, такі архітектури як Sparse Autoencoders та Denoising Autoencoders (для розрідження представлень та видалення шуму, відповідно), але в рамках даної роботи необхідно згадати про варіаційні автокодувальники (VAE) [13], тип генеративних моделей. VAE поєднує ідеї глибокого навчання та баєсових методів, щоб створювати потужні моделі для генерації даних.

Додаткове завдання енкодера VAE полягає у тому, щоб перетворити вхідні дані у параметри латентного (прихованого) розподілу, конкретно у середнє значення (μ) і дисперсію (σ) латентного простору. Ці параметри описують розподіл, з якого може бути згенероване латентне представлення даних для генерування, який за замовченням вважається гаусовим. Енкодер використовує нейронну мережу для обчислення цих параметрів. Вхідні дані пропускаються через цю мережу, яка окрім латентних представлень, видає два вектори: один для μ і один для σ. Замість прямого використання таких векторів VAE використовує процес званий "reparameterization trick". Випадкова змінна ε генерується зі стандартного нормального розподілу, і латентна змінна z обчислюється як $z = \mu + \varepsilon \times \sigma$. Цей крок дозволяє моделі генерувати змінні, які можуть ефективно пропагувати градієнти під час зворотного поширення, забезпечуючи стійкість і ефективність навчання.

Завдання декодера полягає у відновленні вхідних даних з латентних змінних z. Декодер латентні змінні як вхід і використовує їх для генерації вихідних даних, які намагаються максимально точно відтворити оригінальні вхідні дані. Вихід декодера зазвичай пройде через функцію активації, щоб забезпечити вихід даних у певному діапазоні.

Функція втрати VAE складається з двох основних компонентів: перша – реконструкційна (між оригінальними даними і відтвореними), як і в звичайному автокодувальнику, а друга – KL розбіжність (дивергенція), яка вимірює відстань між апріорним розподілом латентних змінних (зазвичай

нормальним розподілом) і розподілом, що генерує енкодер. Цей термін додає штраф за відхилення від апріорного розподілу, сприяючи регуляризації моделі.

Другим потужним видом генеративних мереж є змагальні (GAN) [14] – це моделі, які складаються з двох основних компонентів: генератора та дискримінатора, що змагаються один з одним, звідки і назва.

Генератор у GAN відповідає за створення даних, що максимально схожі на реальні. Він починає з випадкового шуму або латентного вектора, отриманого з латентного простору, і перетворює цей шум у дані, які імітують розподіл реальних даних (знову ж таки, нормальний). Латентний простір тут — це не результат, а деяке джерело, яке генератор використовує для вивчення та генерації нових екземплярів.

Дискримінатор у GAN відповідає за розрізнення справжніх даних від синтезованих генератором. Його завдання — визначити, чи є вхідні дані реальними чи штучно створеними. Це змагання між генератором та дискримінатором сприяє покращенню якості генерованих даних.

Процес навчання GAN включає наступні кроки:

- Генерація даних: генератор отримує випадковий шум і перетворює його в синтетичні дані.
- Оцінка дискримінатором: дискримінатор оцінює як реальні дані, так і дані, сгенеровані генератором, і визначає їх автентичність.
- Оптимізація: обидва компоненти тренуються одночасно: дискримінатор намагається максимізувати свою здатність правильно класифікувати реальні та синтетичні дані, тоді як генератор намагається обдурити дискримінатор, роблячи його вироблення нерозрізненними від реальних даних.

Генерація нових даних – один зі способів наповнення навчальних наборів та їх балансування (аналог звичайної аугментації), який в змозі покращити ефективність вирішення задачі класифікації на рівні набору даних.

Адже, як було згадано вище, будь-який процес пов'язаний із глибоким навчанням починається з навчального датасету.

1.4 Технологія попередньої обробки даних. Постановка задачі дослідження

Отже, на сьогодні існує потужний інструмент для вирішення задач розпізнавання та класифікації – нейромережеві моделі різноманітних архітектур, але для їхнього повноцінного функціонування необхідно пройти якісний етап навчання, який базується на наявності навчального набору.

Закономірним є те, що покращення навчального набору як якісно так і кількісно має позитивно вплинути на процес навчання (скоротити необхідний час, збільшити точність, тощо). Тобто, актуальною є технологія попередньої обробки даних для існуючого навчального набору повітряного спостереження, яка б дала змогу не тільки генерувати нові дані в разі потреби, а також видаляти ті, що не підходять (аномалії, дублікати, тощо). Такі процедури простіше провести для випадкової величини, для якої існує змога оцінки функції розподілу та щільності, тобто існує модель розподілу набору даних, як випадкової величини.

Процес вводу такої моделі для набору зображень спрощується у випадку, коли є змога відобразити цифрові зображення у простір дійсних чисел деякої не критичної розмірності, так щоб замість оцінки розподілу навчального набору зображень аерозйомки, можна було провести оцінку розподілу для набору багатовимірних представлень даних навчального набору.

Маючі таку модель, за допомогою відомих методів або їхньої адаптації можна ввести методи попередньої обробки даних набору.

Враховуючі усе сказане вище, поставимо наступні задачі дослідження:

- Введення моделі розподілу даних навчального набору повітряного спостереження на основ багатовимірних представлень елементів набору.

- Розробка методів попередньої обробки набору (для генерації нових та видалення зайвих елементів набору) на основі моделі
- Розробка і тестування інформаційної технології для вирішення задачі класифікації на основі введених моделі та методів.

РОЗДІЛ 2.
МОДЕЛЬ РОЗПОДІЛУ НАВЧАЛЬНОГО НАБОРУ У ПРОСТОРІ ПРЕДСТАВЛЕНЬ

Так як дослідження проводиться для покращення вирішення задачі класифікації, нагадаємо її постановку.

Нехай маємо n – вимірний набір спостережень, кожне з яких належить до деякого класу, приналежність до котрого визначається міткою:

$$(x_l, y_l),$$

де $x_l -$ деяке цифрове зображення, $l = 1, N$, а $y_l = \{1...k\}$, де $k \in Z$ -кількість класів, а $x_l \in \mathbb{R}^n$, де $n -$ розмірність простору спостережень. Стверджуємо, що $x_l \in S_i$ тільки тоді, коли $y_l = i$, де $S_i = \{x_{l_i}, l_i = \overline{1, N_i}\}$ - деяка підмножина, така що $N = \sum_{i=1}^{k} N_i$ і $S_i \cap S_j = \varnothing, i \neq j$.

Як вже було сказано, основна ідея – за допомогою деякої функції перетворення відобразити набір зображень у простір дійсних чисел, отримавши таким чином набір багатовимірних представлень і оцінити розподіл для цього набору. Так як і в самій класифікації, і в отриманні представлень фігуруватимуть нейромережеві моделі, спочатку буде надано опис принципу роботи таких моделей. Після буде надано опис способу оцінки розподілу набору даних у просторі представлень та опис методів маніпуляції даними на основі представлень.

2.1 Принцип роботи нейронних мереж

Протягом проведення досліджень використовувались різні варіації нейромережевих класифікаторів та автокодувальників, перші – для проведення класифікації даних з навчального набору повітряних спостережень, другі – для виділення латентних векторів (тобто багатовимірних представень) з того самого набору, відповідно.

Тому в подальшому необхідно надати принцип роботи таких моделей, починаючі з простішої складової – нейрона.

2.1.1Нейрони

Сучасна концепція штучних нейронних мереж була закладена Уорреном С.Мак-Каллоком і Вальтером Питтсом. Головний принцип їхньої теорії полягав в тому, що довільні явища, які стосуються вищої нервової діяльності, можуть бути проаналізовані і зрозумілі, як деяка активність в мережі, що складається з логічних елементів, які приймають тільки два стани ("все або нічого") [15]. При цьому для будь-якого логічного виразу, який відповідає наведеним авторами умовам, може бути знайдено мережу логічних елементів, що має описану цим виразом поведінку.

В подальшому "логічний елемен" отримав назву логічного (формального) нейрону, схема роботи якого відображено на схемі 2.1.

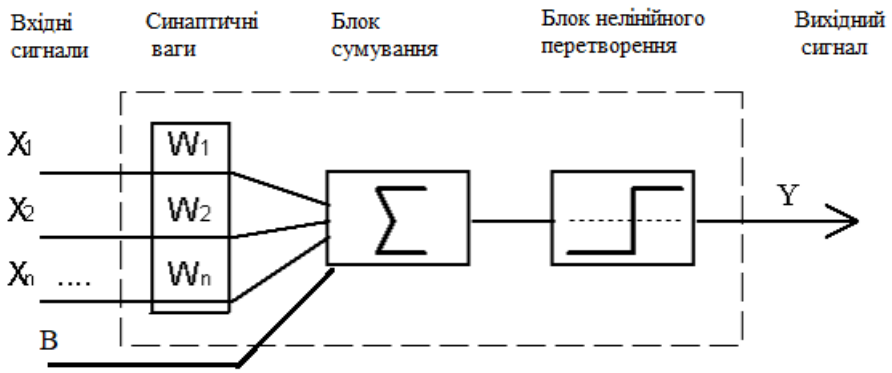


Рис. 2.1 функціональна схема формального нейрону

На цій схемі X – окреме n-вимірне спостереження (сигнал)

$$X = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{pmatrix};$$

W – вектор ваг нейрону

$$W = \begin{pmatrix} W_1 \\ W_2 \\ \vdots \\ W_n \end{pmatrix};$$

B – деякий зсув.

З сучасної точки зору, формальний нейрон являє собою математичну модель простого процесора, що має кілька входів і один вихід. Вектор вхідних сигналів перетворюється нейроном у вихідний сигнал з використанням трьох функціональних блоків: локальної пам'яті, блоку підсумовування і блоку нелінійного перетворення.Вектор локальної пам'яті містить інформацію про вагові множники, з якими вхідні сигнали будуть інтерпретуватися нейроном. Ці змінні ваги є аналогом чутливості пластичних синаптичних контактів. Вибором ваг досягається та чи інша інтегральна функція нейрона.У блоці підсумовування відбувається накопичення загального вхідного сигналу (*net*), який дорівнює сумі зсуву та зваженої суми складових вхідного сигналу:

$$net = B + \sum_{i=1}^{n} W_i X_i.$$

У блоці нелінійного перетворення відбувається так звана "активація" загального вхідного сигналу за допомогою функції нелінійного перетворення (активаційної), результатом якої і є вихідний сигнал:

$$Y = f_a\left(net\right),$$

де f_a - активаційна функція, Y - вихідний сигнал нейрону.

Подальшим розвитком теорії формального нейрона є перехід до аналогових (безперервних) сигналів, а також до різних типів нелінійних перехідних функцій. Така робота формального нейрону знайшла своє подальше застосування у перших видах штучних нейронних мереж – перцептронах, а самі вони отримали назву штучного нейрону. Узагальнене перетворення вхідного сигналу у вихідний за допомогою штучного нейрону можна описати наступним чином:

$$Y = f_a\left(B + \sum_{i=1}^{n} W_i X_i\right) = f_a\left(B + W^T X\right) \tag{2.1}$$

Нижче наведено деякі види активаційних функцій. Зазначимо, що значення Q, яке присутнє у перших варіантах цих функції, - це порогове значення, яке часто приймається за нуль, що відображено у других варіантах.

Порогова функція (розглянута Маккалоком і Питтсом):

$$Y = f_a(net) = \begin{cases} 1, net > Q, \\ 0, net \le Q. \end{cases}$$

або (2.2)

$$Y = f_a(net) = [\text{net} > 0].$$

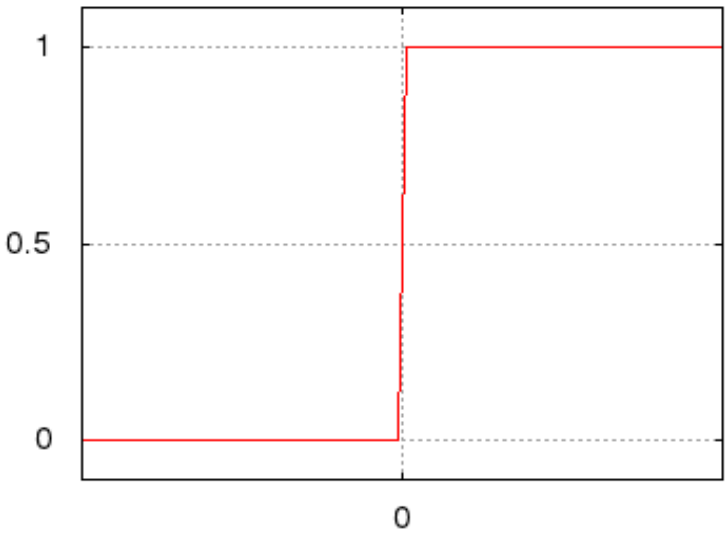


Рис. 2.2 Вид порогової функції

Лінійна функція, а також її варіант - лінійна функція з погашенням негативних сигналів (ReLu):

$$Y = f_a(net) = \begin{cases} net, net > Q, \\ 0, net \le Q. \end{cases}$$

або (2.3)

$$Y = f_a(net) = \max(0, net).$$

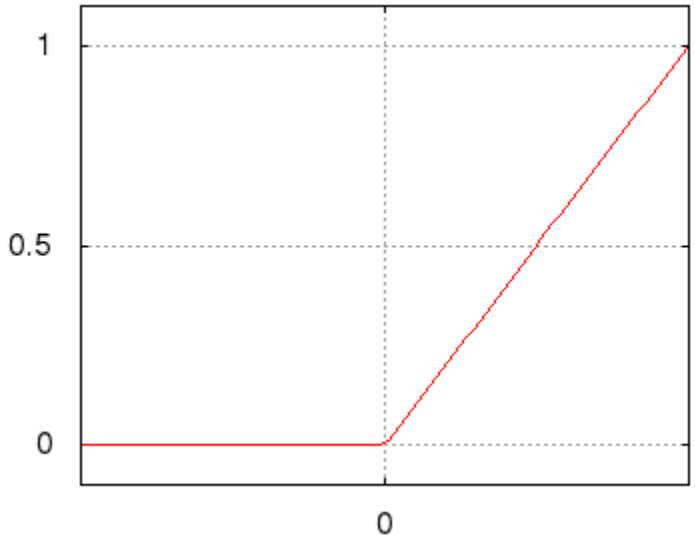


Рис. 2.3 Вид ReLu функції

Сигмоїдальна функція:

$$Y = f_a(net) = \frac{1}{1+\exp(-(net-Q))},$$

або (2.4)

$$Y = tanh(net) = \frac{\exp(net)-\exp(-net)}{\exp(net)+\exp(-net)}.$$

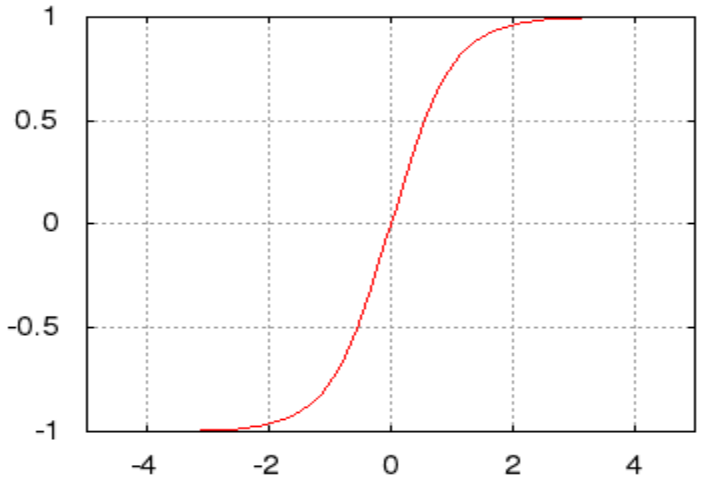


Рис. 2.4 Вид сигмоїдальної функції

2.1.2 Багатошарові перцептрони

Вище згадувалося, що подальшим кроком у використані штучних нейронів стали перцептрони та їхні нащадки – багатошарові перцепторони, які

і є першими штучними нейронними мережами (ШНМ). В [16] пояснюється, що перцептрон – це детермінований варіант логістичної регресії. Дійсно, для бінарної логістичної регресії модель представлена наступним чином:

$$p(y \mid x, \theta) = Ber\left(f_{sig}\left(y \mid w^T x + b \right) \right), \tag{2.5}$$

де w- вагові коефіцієнти, b- деякий зсув, а f_{sig}- сигмоїдальна функція. Позначення $\theta = (w, b)$ об'єднує в собі всі параметри. Легко побачити схожість між формулою (2.1) та моделлю (2.5), тобто по суті така модель є прикладом нейрону, що реалізує лінійне розбиття області класифікації. Для реалізації моделі нелінійної розбиття між таким нейроном і вхідними даними пропонується підставити цілий шар штучних нейронів із своєю функцією активації, таким чином застосувавши деяке перетворення до окремих вхідних ознак і отримавши нову модель:

$$p\left(1 \mid X; \vec{\Theta}\right) = f_{sig}\left(\vec{\Theta}^T F_a\left(X\right)\right) =$$

$$= f_{sig}\left(\theta_0 + \theta_1 f_{a1}\left(X\right) + \theta_2 f_{a2}\left(X\right) + \ldots + \theta_K f_{aK}\left(X\right) = \theta_0 + \sum_{i=1}^{K} \theta_i f_{ai}\left(X\right)\right)$$

де

$$F_a\left(X\right) = \begin{pmatrix} 1 \\ f_{a1} \\ f_{a2} \\ \vdots \\ f_{an} \end{pmatrix}; \qquad \vec{\Theta} = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{pmatrix}.$$

Яка еквівалентна одношаровому перцептрону, а отже і найпростішій ШНМ. Якщо ж розбиття має бути не бінарним, то замість одного вихідного нейрону потрібно використати декілька.

Також у [16] згадується, що перцептрони не здатні були вирішити задачу XOR, поки з них не сформували так званий стек. Ідея полягає в наступному: якщо мережі потрібно "підлаштуватися" під складнішу модель, тобто збільшити нелінійність, а збільшення кількості нейронів в одному шарі було б

недоцільним, тоді необхідно додати до моделі ШНМ нові шари нейронів. Такі моделі стали називатися багатошаровими перцептронами, загальна структура яких відображена на рисунку 2.5.

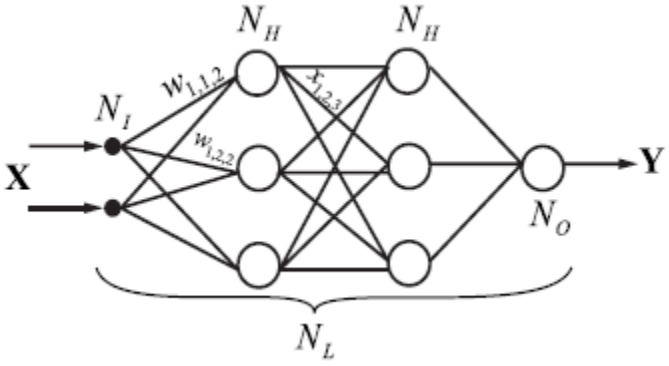


Рис. 2.5 Структура багатошарового перцептрону.

Де N_I - кількість вхідних ознак, N_O - кількість нейронів в вихідному шарі, N_L - кількість шарів ШНМ,

В загальному випадку, такі нейронні мережі – це складні функції, що складаються з великої кількості композицій інших (можливо також складних) функцій, а саме – штучних нейронів із їхнім виходом у вигляді функції активації зваженого вхідного сигналу. То ж загальне перетворення в багатошаровій ШНМ можна описати складною функцією $g(X)$, в якій кожен шар мережі – своя вкладена функція:

$$g(X) = g_L\left(g_{L-1}\left(\ldots\left(g_1(X)\right)\ldots\right)\right), \tag{2.6}$$

де $g_l(\cdot)$, $l = \overline{1,L}$ – функція *l*-го шару.

Тому, якщо припустити, що до мережі із *L* шарами надходить *n*-вимірний вектор *X*, а *l*-тий шар має в собі m_l нейронів то функція $g_1(\cdot): \mathbb{R}_n \to \mathbb{R}_{m_1}$ відображає *n*-вимірний простір в m_1-вимірний. Аналогічно $g_l(\cdot): \mathbb{R}_{m_{l-1}} \to \mathbb{R}_{m_l}$, $l = \overline{2, L-1}$ та $g_L(\cdot): \mathbb{R}_{m_{L-1}} \to \mathbb{R}_m$. При цьому згадані вище вагові вектори *l*-го шару утворюють вагову матицю розмірності $m_{l-1} \times m_l$:

$$W_l = \begin{pmatrix} w_{l,1,1} & w_{l,1,2} & \cdots & w_{l,1,m_l} \\ w_{l,2,1} & w_{l,2,2} & \cdots & w_{l,2,m_l} \\ \vdots & \vdots & \ddots & \vdots \\ w_{l,m_{l-1},1} & w_{l,m_{l-1},2} & \cdots & w_{l,m_{l-1},m_l} \end{pmatrix}.$$

Зсуви формують вектор зсувів:

$$B_l = \begin{pmatrix} b_{l.1} \\ b_{l,2} \\ \vdots \\ b_{l.m_l} \end{pmatrix}.$$

Якщо $g_{l-1}(\cdot)$ – вихідний вектор з попереднього шару

$$g_{l-1}(\cdot) = \begin{pmatrix} g_{l-1,1}(\cdot) \\ g_{l-1,2}(\cdot) \\ \vdots \\ g_{l-1,m_{l-1}}(\cdot) \end{pmatrix},$$

то в матричному вигляді визначення $g_l(\cdot)$ можна записати так:

$$g_l(\cdot) = f_{al}\left(B_l + W_l^T g_{l-1}(\cdot)\right), \tag{2.7}$$

де f_{al} – будь-яка функція активація на *l*-му шарі мережі, при чому на кожному шарі це можуть бути різні функції, хоча зазвичай відрізняється лише функція вихідного шару.

2.1.3 Навчання ШНМ

Мережа (2.6) називається багатошаровим перцептроном або глибокою нейронною мережею, а навчання такої мережі – глибоким навчанням.

Складність навчання нейронних мереж полягає в тому, що через нелінійність нейронної мережі більшість функцій втрат, що підлягають мінімізації, виявляються невипуклими. Тому нейронні мережі зазвичай

навчаються за допомогою ітеративних градієнтних оптимізаторів, які знаходять мале значення функції втрат, а не за допомогою методів вирішення лінійних рівнянь, котрі застосовуються при навчанні моделей регресії, або алгоритмів опуклої оптимізації, що гарантовано сходяться до глобального оптимуму і які використовуються при навчанні логістичної регресії або моделі опорних векторів. Алгоритми опуклої оптимізації теоретично стійкі та сходяться за будь-яких початкових параметрів, хоча й у такому випадку можуть бути обчислювальні проблеми, а метод стохастичного градієнтного спуску, що застосовується до неопуклих функцій втрат, не дає подібних гарантій збіжності та чутливий до початкових значень параметрів. Тому, для нейронних мереж прямого поширення важливо ініціалізувати всі ваги невеликими випадковими значеннями, а зміщення - нулями або невеликими позитивними значеннями.

Звісно, можна навчати методом градієнтного спуску і такі моделі, як лінійна регресія, і так справді роблять, коли навчальний набір дуже великий. З цієї точки зору навчання нейронної мережі не сильно відрізняється від навчання будь-якої іншої моделі. Для нейронної мережі обчислення градієнта дещо складніше, але його можна виконати ефективно і точно.

Нагадаємо основні поняття, які стосуються аналізу градієнта функції багатьох змінних. Щоб поняття мінімуму мало сенс при мінімізації функції декількох змінних, результатом функції має бути скаляр. Якщо функція, що розглядається, має вигляд $f(X), X = \left(x_k; k = \overline{1,n}\right)$, то її часткова похідна $\frac{\partial f(X)}{\partial x_k}$ показує, як змінюється функція при зміні аргументу *X* лише в одному (*k*-му) напрямку. Градієнт узагальнює поняття похідної на вектор: градієнтом функції f називається вектор всіх її часткових похідних, *i*-м елементом якого є часткова похідна f по x_i. Тобто, якщо функція диференційована в точці

$P_0 = \left(x_{0,k}; k = \overline{1,n} \right)$, то вектор координат $\frac{\partial f\left(P_0\right)}{\partial x_k}, k = \overline{1,n}$ називається її градієнтом та позначається ∇f :

$$\nabla f = \begin{pmatrix} \frac{\partial f\left(P_0\right)}{\partial x_1} \\ \frac{\partial f\left(P_0\right)}{\partial x_2} \\ \vdots \\ \frac{\partial f\left(P_0\right)}{\partial x_n} \end{pmatrix}.$$

Критичною багатовимірною точкою називається та, в якій всі елементи градієнту дорівнюють нулю. Якщо o_i - координатні орти, то:

$$\nabla f = \sum_{k=1}^{n} \frac{\partial f\left(P_0\right)}{\partial x_k} o_k$$

Також відомо, що функція має похідну в довільному напрямку, якщо є диференційованою в точці і визначається ця похідна так:

$$\frac{\partial f\left(P_0\right)}{\partial s} = \sum_{k=1}^{n} \frac{\partial f\left(P_0\right)}{\partial x_k} \cos\gamma_k ,$$

де $s-$ приріст в деякому напрямку.

Іншими словами, похідною за напрямом у напрямку одиничного вектору u називається кутовий коефіцієнт функції f в напрямку u. Тобто, похідна за напрямом – це похідна функції $f\left(X + \alpha u\right)$ по α, що обрахована при $\alpha = 0$. За правилом обчислення складної похідної

$$\frac{\partial f\left(X + \alpha u\right)}{\partial \alpha} = u^T \nabla f\left(X\right),$$

при $\alpha = 0$. Отже, похідна за напрямом визначається як скалярний добуток градієнта та вектора u.

Якщо треба мінімізувати функцію, то потрібно знайти напрямок в якому f спадає найшвидше всього. Для цього можна скористатись умовою мінімізації похідної за напрямом:

$$\min_{u,u^T,u=1} u^T \nabla f(X) = \min_{u,u^T,u=1} \|u\|_{L_2} \|\nabla f(X)\|_{L_2} \cos\varphi,$$

де φ – кут між u та градієнтом.

Якщо в останній вираз підставити $\|u\|_{L_2} = 1$ та ігнорувати множники, що не залежать від u, то залишається $\min_u \cos\varphi$. Ця величина досягає мінімуму, коли u направлено протилежно градієнту.

Тож, напрям градієнту вказує напрям найшвидшого зростання функції, а його величина дорівнює похідній в цьому напрямку. Тому для мінімізації f потрібно рухатись в напрямку від'ємного градієнту. Реалізація такого підходу до мінімізації функції має назву метод градієнтного спуску.

За методом градієнтного спуску пропонується обирати нову точку наступним чином:

$$X' = X - \mu \nabla f(X),$$

де μ – швидкість навчання, додатній скаляр, що визначає довжину кроку наближення до мінімуму, найчастіше – це деяка невелика константа.

Нехай, як і вище у викладенні, на вхід мережі з L шарами надходить n-вимірний вхідний вектор X, а на виході мережі отримано деяку скалярну або векторну функцію, яку можна порівняти з навчальним еталоном $y(X)$.

Тобто, для кожного окремого прикладу, після проходу через мережу, можна отримати деяку функцію втрат $Loss(X,\Theta)$, яка відображає відхилення за деякою метрикою отриманого результату від еталону:

$$Loss(X,\Theta) = delta(g(X), y(X)),$$

де $delta(g(X), y(X))$ – метрика відстані між результатом нейромережі та еталоном; Θ – вектор параметрів нейромережі, що підлягає оцінці під час навчання.

Мінімізація функції втрати відбувається методом градієнтного спуску. Тобто, першим важливим питання є обрахунок для кожного окремого навчального прикладу вектору

$$\nabla Loss = \begin{pmatrix} \frac{\partial Loss}{\partial \theta_A} \\ \frac{\partial Loss}{\partial \theta_{A-1}} \\ \vdots \\ \frac{\partial Loss}{\partial \theta_1} \end{pmatrix}, \tag{2.8}$$

де θ_i, $i = \overline{1, A}$ – всі ваги та зсуви нейромережі; A – їх кількість.

Алгоритм зворотного поширення – дозволяє передавати інформацію про значення втрати назад по мережі для обчислення градієнта на всіх попередніх шарах мережі.

Під терміном «зворотне поширення» часто розуміють алгоритм навчання багатошарових нейронних мереж, в той час як воно відноситься лише до обчислення градієнта, тоді як для навчання за допомогою такого градієнту застосовують інші алгоритми. Крім того, іноді помилково вважають, що зворотне поширення стосується лише багатошарових нейронних мереж, хоча, фактично, мова йде про обчислення похідних будь-якої складної функції, за умови, що такі існують.

Якщо розглядати приклад мережі, що має L шарів та містить на кожному із них лише один блок, одне значення вході та на виході, а функція активації на кожному шарі буде однаковою, можна визначити її так:

$$g_1 = f_a(w_1 x + b_1),$$

$$g_2 = f_a(w_2 g_1 + b_2),$$

$$\dots$$

$$g_L = f_a(w_L g_{L-1} + b_L).$$

Нехай функція втрати – квадратична:

$$Loss(x, w_1, b_1, \ldots, w_L, b_L) = \frac{1}{2}(y - g_L)^2.$$

До початку навчання всі ваги w_l та зсуви b_l, $l = \overline{1, L}$ визначені деякими випадковими значеннями і метою є змінити їх так, щоб забезпечити мінімізацію функції *Loss*.

У процесі використання розглянутої нейромережі, після надходження конкретного прикладу *x*, вираховуються g_l та безпосередньо саме значення функції *Loss* для конкретного вхідного прикладу.

Не складно отримати, що для останнього шару даної мережі

$$\frac{\partial Loss}{\partial w_L} = -(y - g_L) \cdot f_a'(w_L g_{L-1} + b_L) \cdot g_{L-1},$$

$$\frac{\partial Loss}{\partial b_L} = -(y - g_L) \cdot f_a'(w_L g_{L-1} + b_L),$$

де $f_a'(w_L g_{L-1} + b_L)$ – обчислене значення похідної функції активації.

Не складно отримати інші часткові похідні для ваг на *l*-му шарі:

$$\frac{\partial Loss}{\partial w_l} = -(y - g_L) \cdot f_a'(w_L g_{L-1} + b_L) \cdot f_a'(w_{L-1} g_{L-2} + b_{L-1}) \cdot \ldots f_a'(w_l g_{l-1} + b_l) g_{l-1} =$$

$$= -(y - g_L) \cdot \left(\prod_{j=l}^{L} f_a'(w_j g_{j-1} + b_j) \right) \cdot g_{l-1},$$

$$\frac{\partial Loss}{\partial b_l} = -(y - g_L) \cdot \left(\prod_{j=l}^{L} f_a'(w_j g_{j-1} + b_j) \right).$$

Нарешті, часткові похідні по w_1 та b_1 такі:

$$\frac{\partial Loss}{\partial w_1} = -(y - g_L) \cdot \left(\prod_{j=2}^{L} f_a'(w_j g_{j-1} + b_j) \right) \cdot (w_1 x + b_1) \cdot x,$$

$$\frac{\partial Loss}{\partial b_1} = -(y - g_L) \cdot \left(\prod_{j=2}^{L} f_a'(w_j g_{j-1} + b_j) \right) \cdot (w_1 x + b_1).$$

Нескладно показати, що в даному прикладі для обчислення часткових похідних на l-му шарі не обов'язково обраховувати добуток похідних функції активації на попередніх. Досить використати вже обчислене значення часткової похідної на $(l+1)$-му шарі:

$$\frac{\partial Loss}{\partial w_l} = \frac{\partial Loss}{\partial w_{l+1}} f_a'\left(w_l g_{l-1} + b_l\right)\frac{g_{l-1}}{g_l},$$

$$\frac{\partial Loss}{\partial b_l} = \frac{\partial Loss}{\partial b_{l+1}} f_a'\left(w_l g_{l-1} + b_l\right), \quad l = \overline{2, L-1},$$

$$\frac{\partial Loss}{\partial w_1} = \frac{\partial Loss}{\partial w_2} f_a'\left(w_1 x + b_1\right)\frac{x}{g_2},$$

$$\frac{\partial Loss}{\partial b_1} = \frac{\partial Loss}{\partial b_2} f_a'\left(w_1 x + b_1\right).$$

Таким чином, відносно нескладним обчислювальним способом отримують значення градієнту для кожного з прикладів, що використовується для навчання розглянутої мережі:

$$\nabla Loss = \begin{pmatrix} \frac{\partial Loss}{\partial w_L} \\ \frac{\partial Loss}{\partial b_L} \\ \vdots \\ \frac{\partial Loss}{\partial w_1} \\ \frac{\partial Loss}{\partial b_1} \end{pmatrix}.$$

Повернемось до розгляду нейромереж прямого поширення з довільною кількістю блоків активації на кожному шарі: нехай функція активації на кожному шарі мережі – однакова та застосоовується без врахування зсуву. Тоді L-й шар можна представити так:

$$g_L(\cdot)=\begin{pmatrix} g_{L,1} \\ \vdots \\ g_{L,m} \end{pmatrix}=\begin{pmatrix} f_a\left(\sum_{j=1}^{m_{L-1}} w_{L,j,1}g_{L-1,j}\right) \\ \vdots \\ f_a\left(\sum_{j=1}^{m_{L-1}} w_{L,j,m_{L-1}}g_{L-1,j}\right) \end{pmatrix}=f_a\left(W_L^T g_{L-1}(\cdot)\right).$$

Нехай вектор похибки E_L на виході мережі визначається так:

$$E_L=y-g_L=\begin{pmatrix} y_1-g_{L,1} \\ \vdots \\ y_m-g_{L,m} \end{pmatrix}.$$

Подамо *m*-вимірну функцію втрати у вигляді:

$$Loss(X,\Theta)=\frac{1}{2}\|y-g_L\|^2{}_2=\frac{1}{2}\sum_{j=1}^{m}\left(y_j-g_{L,j}\right)^2=\frac{1}{2}\sum_{j=1}^{m}\left(y_j-f_a\left(\sum_{k=1}^{m_{L-1}} w_{L,k,j}g_{L-1,k}\right)\right)^2=$$

$$=\frac{1}{2}\sum_{j=1}^{m}\left(y_j-f_a\left(z_{L,j}\right)\right)^2=\frac{1}{2}\left\|y-f_a\left(W_L^T g_{L-1}\right)\right\|^2{}_2, \qquad (2.9)$$

де

$$z_{L,j}=\sum_{k=1}^{m_{L-1}} w_{L,k,j}g_{L-1,k}, \qquad j=\overline{1,m}.$$

Тоді, застосовуючи правило диференціювання складної функції, отримаємо:

$$\frac{\partial Loss}{\partial W_L}=C\cdot \underset{m\times m_{L-1}}{J_{g_L}}, \qquad (2.10)$$

де $\underset{m\times m_{L-1}}{J_{g_L}}$ – матриця Якобі

$$J_{\underset{m\times m_{L-1}}{g_L}} = \begin{pmatrix} \frac{\partial g_{L,1}}{\partial w_{L,1,1}} & \cdots & \frac{\partial g_{L,1}}{\partial w_{L,1,m_{L-1}}} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_{L,m}}{\partial w_{L,1}} & \cdots & \frac{\partial g_{L,m}}{\partial w_{L,m,m_{L-1}}} \end{pmatrix} = \begin{pmatrix} f_a'\left(z_{L,1}\right) \\ \vdots \\ f_a'\left(z_{L,m}\right) \end{pmatrix} \cdot \begin{pmatrix} g_{L-1,1} \\ \vdots \\ g_{L-1,m_{L-1}} \end{pmatrix}^T \in \mathbb{R}_{m\times m_{L-1}};$$

C – скаляр

$$C = -\sum_{i=1}^{m}\left(y_i - g_{L,i}\right).$$

Варто нагадати, що на l-му шарі мережі маємо m_l-вимірну функцію $g_l(\cdot)$, тож матриця Яккобі переходу від n-вимірного вхідного вектора X до вихідної m-вимірної функції g можна визначити як добуток матриць Якобі переходів на кожному з шарів мережі:

$$J_{\underset{m\times n}{g}} = J_{\underset{m\times m_{L-1}}{g_L}} \cdot J_{\underset{m_{L-2}\times m_{L-3}}{g_{L-1}}} \cdot \ldots \cdot J_{\underset{m_l\times m_{l-1}}{g_l}} \cdot \ldots \cdot J_{\underset{m_2\times m_1}{g_2}} \cdot J_{\underset{m_1\times n}{g_1}}, \tag{2.11}$$

де

$$J_{\underset{m_l\times m_{l-1}}{g_l}} = \begin{pmatrix} \frac{\partial g_{l,1}}{\partial w_{l,1,1}} & \cdots & \frac{\partial g_{l,1}}{\partial w_{l,1,m_{l-1}}} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_{l,m_l}}{\partial w_{l,m_l,1}} & \cdots & \frac{\partial g_{l,m_l}}{\partial w_{l,m_l,m_{l-1}}} \end{pmatrix} = \begin{pmatrix} \nabla g_{l,1}{}^T \\ \vdots \\ \nabla g_{l,m_l}{}^T \end{pmatrix} = \begin{pmatrix} \frac{\partial g_l}{\partial w_{l,1}} & \cdots & \frac{\partial g_l}{\partial w_{l,m_l-1}} \end{pmatrix} \in \mathbb{R}_{m_l\times m_{l-1}}$$

.

Таким чином, для довільного l-го шару нейромережі можна визначити матрицю часткових похідних функції втрати по вагам так:

$$\frac{\partial Loss}{\partial W_l} = C\cdot E_l^T \cdot J_{\underset{m_l\times m_{l-1}}{g_l}} = C\cdot\left(W_{l+1}\cdot E_{l+1}\right)^T \cdot J_{\underset{m_l\times m_{l+1}}{g_l}}, \qquad l = \overline{2, L-2}. \tag{2.12}$$

Відмітимо, що для першого шару мережі матриція Яккобі визначається так:

$$J_{\underset{m_1\times n}{g_1}} = \begin{pmatrix} f_a'\left(z_{1,1}\right) \\ \vdots \\ f_a'\left(z_{1,m_1}\right) \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}^T .$$

Вираз (2.12) демонструє ідею методу зворотного поширення похибки, а саме: обрахунок значень градієнту похибки для l-го шару здійснюється після обрахунку відповідних градієнтів похибки на шарах, що розташовані ближче до виходу мережі. Тим самим забезпечується відносна обчислювальна простотота, адже фактично на l-му шарі обчислюється лише обмежена кількість координат вектору градієнту похибки на основі вже раніше обчислених:

$$\frac{\partial Loss}{\partial W_l} = \underbrace{\frac{\partial Loss}{\partial g_L}\frac{\partial g_L}{\partial g_{L-1}}\cdots\frac{\partial g_{l+2}}{\partial g_{l+1}}}_{\frac{\partial Loss}{\partial g_{l+1}}}\frac{\partial g_{l+1}}{\partial g_l}\frac{\partial f_a}{\partial z_l}\frac{\partial z_l}{\partial W_l} = \frac{\partial Loss}{\partial g_{l+1}}\boxed{\frac{\partial g_{l+1}}{\partial g_l}\frac{\partial f_a}{\partial z_l}\frac{\partial z_l}{\partial W_l}}. \qquad (2.13)$$

2.1.4 Принцип роботи Автокодувальників

Як було сказано в 1.3.2, автокодувальник — це тип нейронної мережі, який навчається кодувати вхідні дані в компактніші представлення і відновлювати їх назад до оригінального формату. Там же описувалося, що така мережа ділиться на дві частини: кодувальник (кодер), який перетворює вхідні дані на латентний вектори, та декодувальник (декодер), який відновлює оригінальний вид даних з цих векторів. Схема такого поділу представлена на рисунку 2.6, де представлений простий автокодувальник із чотирма шарами: вхідний, вихідний, та два прихованих

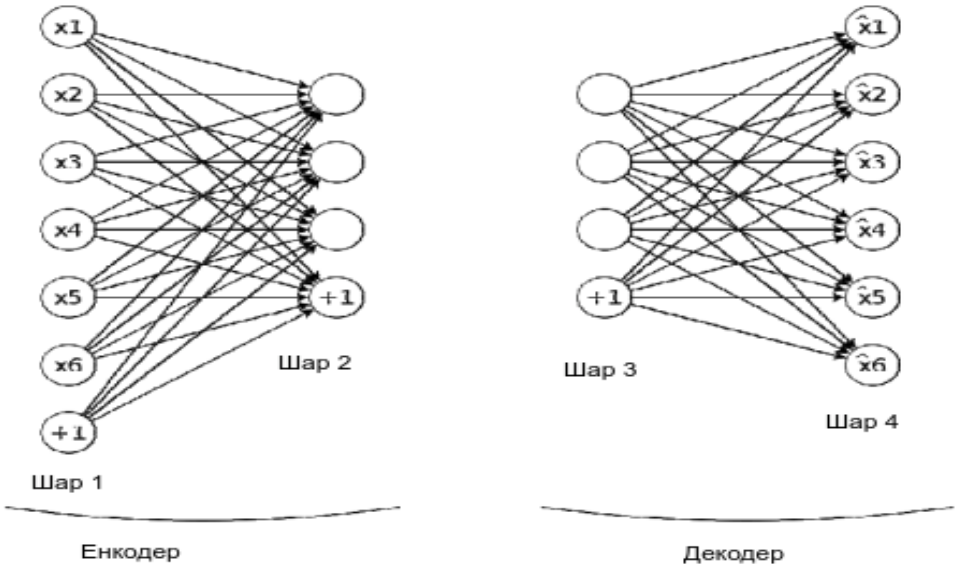


Рис. 2.6 Автокодувальник із чотирма шарами

Для цифрового зображення і автоенкодера із одним прихованим шаром процес відображення можна описати наступним чином:

$$t_l = W^1 x_l, \tag{2.14}$$

де t_l – результат відображення, а W^1 – матриця вагів прихованого шару. Для того щоб збільшити нелінійність кінцевого перетворення достатньо додати додаткові приховані шари.

Декодер, в свою чергу намагається відновити вихідні дані з цього стисненого представлення:

$$x_l = W^2 t_l = W^2 W^1 x_l = W x_l$$

Навчатися автокодувальник може за тим самим принципом, що описано вище, але, що також згадувалось у 1.3.2, його функцією втрат найчастіше обирається середньоквадратична помилка (MSE), на відміну від перехресної ентропії при навчанні класифікаторів.

Шари нейронних мереж, які були розглянуті в цьому розділі, використовуються не тільки у перцептронах, а і в інших, більш складних, моделях і називаються повнозв'язними або повністю з'єднаними (fully connected) шарами. Але моделі, які цілком складаються тільки з таких шарів

(fully connected models) не ефективно обробляють зображення, через що прийнято використовувати згорткові мережі.

2.1.5 Згорткові нейронні мережі

При роботі із ЦЗ у повнозв'язних моделей виникають наступні недоліки:

- Повністю з'єднані мережі потребують величезної кількості параметрів, коли їх застосовують до зображень. Наприклад, зображення розміром 256x256 пікселів з трьома каналами кольору має 196,608 вхідних одиниць. Якщо перший прихований шар також містить велику кількість нейронів, це може призвести до мільйонів ваг, що робить мережу великою і обчислювально важкою для тренування.
- Повністю з'єднані мережі не зберігають просторову структуру зображення. Всі пікселі обробляються однаково без урахування їхньої просторової близькості. Це означає, що такі мережі не можуть ефективно вивчати зображення, в яких просторовий контекст (тобто взаємне розташування пікселів) є важливим.
- Через велику кількість параметрів, повністю з'єднані мережі вимагають значних обчислювальних ресурсів і великих наборів даних для тренування, щоб уникнути перенавчання. Це обмежує їхню практичність, особливо для мобільних або вбудованих систем.
- Інша проблема полягає у тому, що патерн, що зустрівся в одному місці, може не розпізнаватись в іншому – тобто модель не буде інваріантною щодо зсуву, шаблон лишатиметься «зафіксованим» відносно певної позиції в зображенні.

Для вирішення зазначених проблем, що виникають при обробці цз, використовують спеціальні моделі – згорткові нейронні мережі (ЗНМ, CNN), в яких множення великих матриць вагових коефіцієнтів замінено локальною операцією згортки.

Основна полягає в тому, щоб розбити вхідне зображення на двомірні патчі (невеликі ділянки), що перетинаються, і порівняти кожен патч з безліччю невеликих матриць ваг, або фільтрів, які представляють частини об'єкта. Це можна розглядати як різновид зіставлення з шаблоном. Оскільки шаблони невеликі (часто всього 3x3 або 5x5), число параметрів значно зменшується. А оскільки для порівняння з шаблоном використовується згортка, а не множення матриць, модель виявляється інваріантною щодо зсуву. Це корисно в таких задачах, як класифікація зображень, де мета – зрозуміти, є об'єкт чи ні, незалежно від його розташування .

Згорткою $f * g$ двох функцій $f = f(t)$ та $g = g(t)$ є функція, що залежить від t і визначається наступним чином:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(a)g(t-a)da. \qquad (2.15)$$

Функцію $f * g$ можна сприймати як згладження функції g, де f є функцією згладження: $g(t)$ замінюються на зважені сусідні значення $g(t – a)$, де вагою для кожного значення a є $f(t)$. Проста заміна змінних ілюструє, що згладження функції f з використанням g дасть аналогічний результат (тобто згортка є комутативною):

$$(f * g)(t) = (g * f)(t).$$

Ця операція називається згорткою. Операцію звичайної згортки позначено зірочкою:

$$s(t) = (x * w)(t)$$

У прикладі w має бути дійсною функцією щільності імовірності, або вихід не є середньозваженим. Крім того, w має бути 0 для всіх негативних аргументів, або в майбутньому він буде виглядати таким, що, перевершує можливості. Ці обмеження є характерними для цього прикладу. Загалом, згортка визначається для будь-яких функцій, для яких визначений вище інтеграл існує, і може використовуватися для інших цілей, крім розрахунку середніх значень.

У термінології згорткової мережі перший аргумент (в цьому прикладі, функція x) до згортки називають входом, а другим аргументом (у цьому прикладі функцією w) – ядром (kernel). Вихід називають картою властивостей (feature map).

Часто використовують згортки понад однієї вісі розмірності за раз. Наприклад, якщо використовується двовимірне зображення I як вхід, то необхідно використовувати двомірне ядро K:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n) K(i-m, j-n). \quad (2.15)$$

Згортка є комутативною, отже ми можемо еквівалентно написати:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n) K(m,n). \quad (2.16)$$

Зазвичай остання формула є більш простою для реалізації в бібліотеці машинного навчання, оскільки в діапазоні дійсних значень m та n менше варіацій.

Натомість, часто реалізують пов’язану функцію, яка називається крос-кореляцією, яка еквівалентна згортці, якщо вектор ваг симетричний:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n). \quad (2.17)$$

У літературі з глибокого навчання під «згорткою» зазвичай розуміють саме перехресну кореляцію, тож в контексті розгляду нейронних мереж будемо під згорткою розуміти останній вираз, на рисунку 2.7 продемонстровано приклад дії такої згортки.

Використовуючи локальні фільтри з різними властивостями, можна для вхідного зображення отримати набір карт ознак, кожна з яких буде орієнтована на детектування певної особливості. У загальному випадку достатньо визначити лише кількість таких карт, а конкретний тип ядра згортки може бути отримано в результаті навчання мережі.

Нехай на деякій карті знак детектовано деяку особливість. Це означатиме, що в конкретному місці карти отримано досить «великий» відгук

на фільтр відповідної згортки. Проте, в багатьох випадках достатньо встановити лише факт наявності особливості, без прив’язки до місця. Такий підхід надає універсальності, наприклад, при навчанні для прикладів, які містять цільовий об’єкт в різних ракурсах або масштабах. Тож, для забезпечення інваріантності детектованої особ-ливості відносно положення, в згорткових нейронних мережах після згоркового шару реалізують пулінг (рис. 2.8).

Типовий шар згорткової мережі складається з трьох стадій. На першому етапі шар виконує кілька згорток паралельно для створення набору лінійних активацій (карт ознак). На другому етапі кожна лінійна активація проходить через нелінійну функцію активації. Цей етап іноді називають етапом детектора. На третьому етапі використовують функцію пулінгу, щоб змінити вигляд шару далі.

У всіх випадках пулінг допомагає зробити подання приблизно інваріантним для малих відображень входу. Інваріантність відображень означає, що, якщо ми відображаємо вхід невеликою сумою, значення більшості об’єднаних виходів не змінюється.

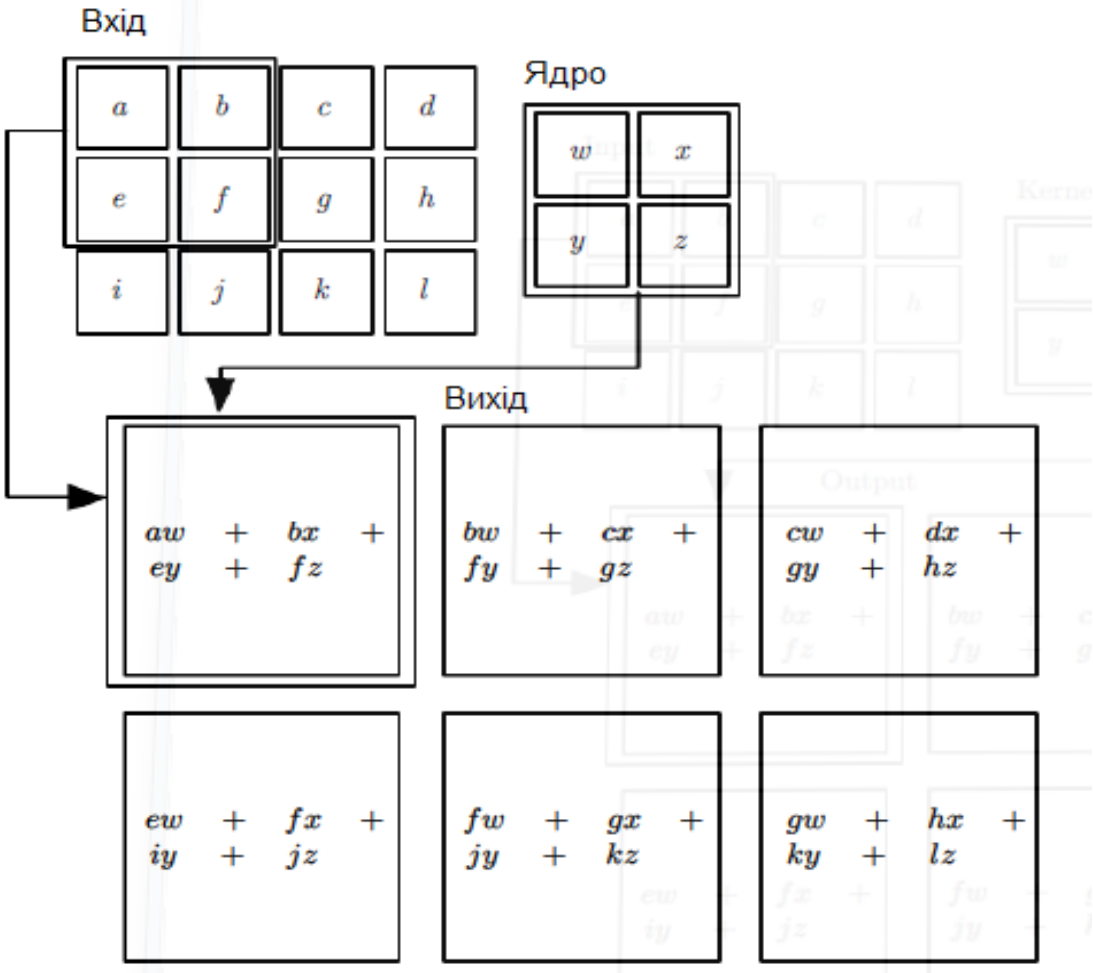


Рис. 2.7 Приклад прямої згортки

Інваріант до місцевого відображення може бути дуже корисною властивістю, якщо більше цікава наявність певної особливості, аніж її точне розташування. Наприклад, при визначенні того, чи містить зображення обличчя, не потрібно знати місцезнаходження точок з точністю до пікселів, а потрібно знати, що є око на лівому боці обличчя та око праворуч. В інших контекстах важливіше зберегти місце розташування об’єкта.

Стандартний згортковий класифікатор використовує шари згортки, пулінгу та повнозв’язні шари. Таким чином реалізується ідея про класифікацію не набору пікселів, а виділених ознак. Через переваги згортки в аутокодувальниках також використовуються відповідні шари, такі архітектури називають згортковими аутокодувальниками. В їхніх кодерах шари повного зв’язку також замінені на послідовність згорток і пулігнів.

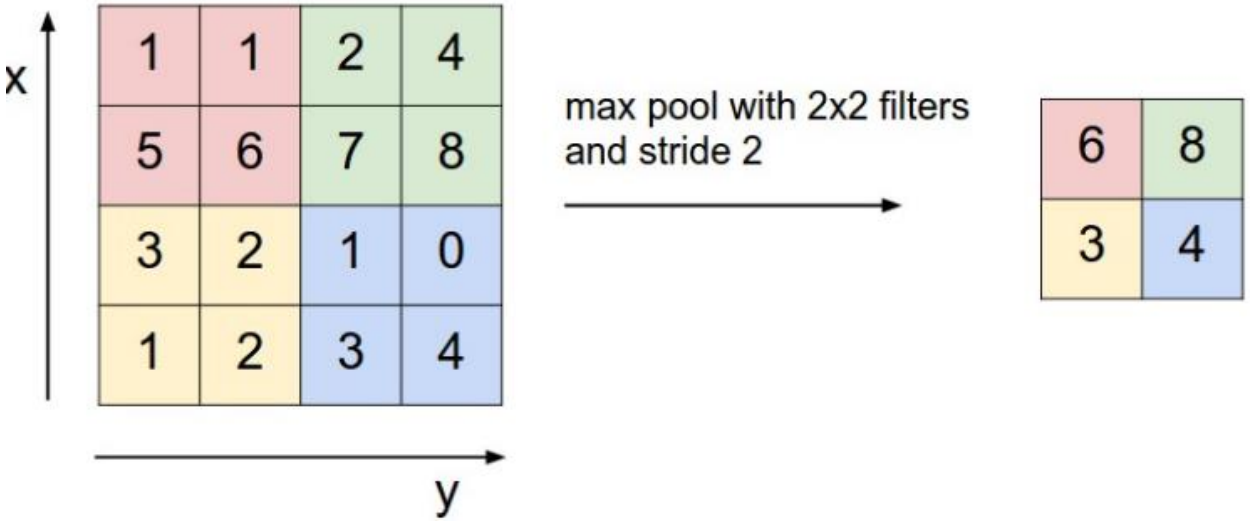


Рис. 2.8 Операція max pooling

Декодувальна частина таких мереж використовує транспоновані згорткові шари, які розширюють латентні представлення назад до розмірів оригінального зображення. Так як основний інтерес маємо саме до латентних представлень, застосування усіх шарів кодувальника позначимо наступним чином:

$$t_l = Code(x_l), \tag{2.18}$$

де *Code* – набір перетворень.

2.2 Оцінка розподілу латентних представлень

Тепер, коли зрозуміло, що вираз (2.18) і буде функцією переходу у простір меншої розмірності для кожного окремого зображення повітряної зйомки, постає питання про спосіб оцінки функції розподілу або функції щільності для набору таких зображень.

2.2.1 Оцінка на основі суміші нормальних розподілів

Як зазначалося в першому розділі, варіаційні автокодувальники під час навчання формують два вектори параметрів тієї ж розмірності, що й латентні вектори: один для середніх значень, другий для дисперсій. Ці параметри використовуються для генерації нових даних [17]. Схожим чином, в змагальних мережах (GAN), латентні вектори виступають як шум на вході

генератора, який перетворює їх у дані, імітуючи реальний розподіл даних, що часто моделюється як нормальний [18].

Концепції роботи обох архітектур підтверджують іншу закономірність: при оцінці розподілу даних, більше ніж у двовимірному просторі, параметричні методи оцінювання функцій розподілу та щільності зводяться до оцінки на основі суміші нормальних розподілів. В такому випадку щільність розподілу для набору X обраховується так:

$$f(X) = \sum_{k=1}^{K} \alpha_k Norm(X \mid \mu_k, DC_k),$$

де $\sum \alpha_k = 1, k = 1..K$ - ваги кожного окремого розподілу в суміші, а $Norm(X \mid \mu_k, DC_k)$ - щільність нормального розподілу із відповідним середнім та дисперсійно-коваріаційною матрицею. Покладемо $Norm(X \mid \mu_k, DC_k) = f_k(X)$, яка в свою чергу дорівнює:

$$f_k(X) = \frac{1}{2\pi^{n/2} \mid DC_k \mid^{1/2}} e^{-\frac{1}{2}(X-\mu_k)^T DC_k^{-1}(X-\mu_k)}$$

Такий підхід, як демонструє [19], широко застосовується для моделювання розподілів, але має певні обмеження, особливо у випадках, коли даних недостатньо або їх розподіл значно відхиляється від нормального. Суміші нормальних розподілів ефективно застосовуються у багатьох задачах, включно з кластеризацією та виявленням аномалій, проте вони можуть бути обмежені у випадках, коли справжні розподіли даних мають складні або мультимодальні форми, які не адекватно описуються простими гаусовими компонентами. Отже, умовах високої розмірності і обмеженого набору даних, вектори параметрів часто визначаються неточно. Також, при необхідності працювати з даними окремого класу, існує ризик, що дані цього класу не будуть добре апроксимовані нормальними розподілами. Тому було вирішено розглянути непараметричні методи оцінки, які могли б надати більш гнучкі можливості для моделювання розподілів.

2.2.2 Сплайн – оцінка функції щільності

Найшвидшим непараметричним способом оцінки багатовимірної функції щільності є гістограмна оцінка. Вона використовує дискретизацію даних за допомогою розділення області значень змінних на інтервали, які формують "біни" або класи. Кількість спостережень у кожному біні дає нам оцінку щільності в цій області. Правильний вибір ширини бінів є критичним: занадто велика ширина може призвести до втрати важливих деталей розподілу, а занадто мала — до надмірної шумності у оцінці [20].

Сплайн-обробка може значно покращити результати гістограмної оцінки. Застосування сплайнів дозволяє згладити гістограму, вирівнюючи переходи між класами і забезпечуючи більш плавне і неперервне представлення щільності. Це може бути особливо корисним у випадках, коли досліджуються складні мультимодальні розподіли, де необхідно точно відобразити наявність кількох піків чи "горбів" у розподілі [21].

Для покращення гістограмних оцінок пропонується використовувати сплайн-апроксимацію поліноміальними сплайнами на основі B-сплайнів, близькими до інтерполяційних у середньому, які представлені у [22], де обґрунтовано, що, як в багатовимірному, так і в одновимірному випадку цей апарат не поступається іншим в оцінці функції щільності і функції розподілу, але й має свої переваги (особливо при їхній неоднорідності) за рахунок того, що локальна апроксимація є більш гнучкою з точки зору врахування локальних особливостей функцій.

Отже, нехай після використання процедури (2.18) до кожного елементу навчального набору було отримано набір латентних представлень із збереженими мітками класів (t_l, y_l). Збереження міток класів є важливим елементом, адже для більшою гнучкості у перетвореннях може знадобитися оцінка розподілу окремого класу, що буде продемонстровано далі, але на разі вважаємо, що нас цікавить оцінка розподілу всього набору. Так як цифрові зображення – це випадкова величина, а t – є результатом їхнього відображення

в простір дійсних чисел, то очевидним є існування функції розподілу такої випадкової величини: $F\left(t_1,\ldots,t_n\right)=P\left\{\omega:-\infty<\xi_1\left(\omega\right)<t_1,\ldots,-\infty<\xi_n\left(\omega\right)<t_n\right\}$, де n – розмірність простору відображення. Якщо вона неперервна, то існує функція щільності:

$$f\left(t_1,\ldots,t_n\right)=\frac{\partial^n F\left(t_1,\ldots,t_n\right)}{\partial t_1\ldots\partial t_n}$$

Саме через те, що про вигляд такого розподілу важко навіть робити припущення, пропонують використовувати непараметричні методи, а саме – покращену за допомогою сплайн-оцінки гістогрмну оцінку.

Розглянемо забезпечення проведення ймовірнісної оцінки за масивом $\left\{t_l;l=\overline{1,N}\right\}$ реалізацій n-вимірної випадкової величини $\vec{\xi}=\left(\xi_1\left(\omega\right),\ldots,\xi_n\left(\omega\right)\right)$. При умові незалежності одновимірних випадкових величин $\xi_k\left(\omega\right)$, $k=\overline{1,n}$ та зважаючи на можливість проведення ймовірнісної оцінки відповідних масивів реалізацій $\left\{t_{k,l};l=\overline{1,N}\right\}$, уведемо низки рівномірних розбиттів за вісями спостережень:

$$t_k:\Delta_{h_{t_k}},\ h_{t_k}>0,\ k=\overline{1,n},$$

можемо розглядати й можливість ймовірнісної обробки n-вимірного варіаційного ряду, розбитого на класи, згідно рівномірного розбиття $\Delta_{h_{t_1},\ldots,h_{t_n}}$:

$$\left\{\left(t_{1,i_1},\ldots,t_{n,i_n}\right),n_{i_1\ldots i_n},p_{i_1\ldots i_n};i_k=\overline{0,m_k-1},k=\overline{1,n}\right\},$$

де $\left(t_{1,i_1},\ldots,t_{n,i_n}\right)$ – варіанта, яка визначає центральну (або мінімальну) точку $\left(i_1,\ldots,i_n\right)$-го елементу розбиття $\Delta_{h_{t_1},\ldots,h_{t_n}}$; m_k – кількість елементів (класів) розбиття $\Delta_{h_{t_1},\ldots,h_{t_n}}$ за напрямками t_k, $k=\overline{1,n}$; $n_{i_1,\ldots,i_n}$ – частота (кількість потраплянь точок з масиву $\Omega_{n,N}$ в $\left(i_1,\ldots,i_n\right)$-й елемент розбиття $\Delta_{h_{t_1},\ldots,h_{t_n}}$); $p_{i_1,\ldots,i_n}$ – випадковість варіанти n-вимірного варіаційного ряду:

$$p_{i_1,\ldots,i_n} = \frac{n_{i_1,\ldots,i_n}}{N},\ \sum_{i_1=0}^{m_1-1}\ldots\sum_{i_n=0}^{m_n-1} p_{i_1,\ldots,i_n} = 1,$$

причому

$$p_{i_1,\ldots,i_n} = P\{\omega : t_{1,i_1} - 0{,}5h_{t_1} \le \xi_1(\omega) < t_{1,i_1} + 0{,}5h_{t_1},\ldots,$$

$$\ldots t_{n,i_n} - 0{,}5h_{t_n} \le \xi_n(\omega) < t_{n,i_n} + 0{,}5h_{t_n}\} = f_{i_1,\ldots,i_n} h_{t_1} \cdot \ldots \cdot h_{t_n},$$

де $f_{i_1,\ldots,i_n}$ – усереднене значення щільності розподілу ймовірностей $f(t_1,\ldots,t_n)$ величини $\vec{\xi}$ на $(i_1,\ldots,i_n)$-му елементі розбиття $\Delta_{h_{t_1},\ldots,h_{t_n}}$:

$$f_{i_1,\ldots,i_n} =$$

$$= \frac{1}{h_{t_1} \cdot \ldots \cdot h_{t_n}} \int_{t_{1i_1}-0{,}5h_{t_1}}^{t_{1i_1}+0{,}5h_{t_1}} \ldots \int_{t_{1i_n}-0{,}5h_{t_1}}^{t_{1i_n}+0{,}5h_{t_1}} f(t_1,\ldots,t_n)\,dt_1\ldots dt_n,$$

окрім того виконується рівність

$$p_{i_1,\ldots,i_n} = p_{i_1} \cdot \ldots \cdot p_{i_n}.$$

Для непараметричної оцінки функції щільності розподілу (з точністю до константи $h_{t_1} \cdot \ldots \cdot h_{t_n}$) в просторі представлення автокодувальника пропонується така сплайн-модель на основі *B*-сплайнів другого порядку:

$$\hat{f}(t_1,\ldots,t_n) = S_2(p,t_1,\ldots,t_n) = \tag{2.19}$$

$$= \sum_{i\in Z}\ldots\sum_{j\in z} B_{2,h_{t_1}}(t_1 - ih_{t_1})\ldots B_{2,h_{t_n}}(t_n - jh_{t_n})\underbrace{p_{i,\ldots,j}}_{n},$$

де з точністю до аргумента:

$$B_{2,h}(t) = \begin{cases} 0, & t \notin [-3h/2;3h/2], \\ (3+2t/h)^2/8, & t \in [-3h/2;-h/2], \\ 3/4-(2t/h)^2/4, & t \in [-h/2;h/2], \\ (3-2t/h)^2/8, & t \in [h/2;3h/2]. \end{cases}$$

(2.19) – це модель оцінки багатовимірної функції щільності набору (t_l, y_l). Тобто, модель (2.19) дозволяє оцінити функцію щільності набору (x_l, y_l) у просторі латентних представлень автокодувальника.

Змога працювати із ЗЦ як із вектором дійсних чисел через перехід (2.18) відкриває цілий спектр можливостей: проведення різноманітних перетворень, видалення аномальних значень або дублікатів, тощо. Але змога оцінки розподілу через (2.19) – більш значущий крок. Єдина проблема, яку можна легко помітити – залежність ступеня поліному від кількості вимірів латентного простору. Так в двовимірному просторі оцінка функції щільності проводитиметься поліномом другого ступеня від двох змінних, в тривимірному – третього ступеня від трьох і так далі. І це лише в тому випадку, якщо за основу сплайн-моделі брати B-сплайни другого порядку. Перша ідея – переходити до такої розмірності, щоб оцінюючий поліном був невеликого ступеня. Але в такому випадку виникає ризик занадто сильно скомпресувати дані. З рештою, якщо ЦЗ розмірністю 64x64 пікселі привести до вектора розміром 1024 елемента, це вже зменшує розмірність вхідного елемента в чотири рази і при цьому залишає латентний вектор інформативним. Але будувати поліном 1024 ступеня на таку ж кількість змінних – не виправдано складно.

Для вирішення цієї проблеми вирішено скористуватися властивістю моделі (2.19), яка полягає в тому що, якщо в просторі представлення існує деяке афінне перетворення, наприклад за реалізації методу головних компонент [27,28,29], що забезпечує перехід до незалежної системи координат, то така модель може бути представлена як добуток оцінок щільності одновимірних маргінальних розподілів:

$$S_2\left(p, t_1, \ldots, t_n\right) = \hat{S}_2\left(t_1\right) \cdot \ldots \cdot \hat{S}_2\left(t_n\right),$$

де

$$\hat{S}_2\left(t_k\right) = \sum_{i \in Z} B_{2,h_{t_k}}\left(t_k - ih_{t_k}\right) p_i \,. \qquad (2.20)$$

Оцінки маргінальних розподілів $\hat{S}_2(t_k)$, $k=\overline{1,n}$ можуть мати використання для більш «гнучкого» дослідження особливостей набору зображень в латентному просторі, але основна перевага використання методу головних компонент полягає в ортогональності (тобто незалежності) результуючих між собою, а це означає, що оцінювати кожну компоненту фактично одновимірною сплайн-оцінкою, що відкриває додаткові можливості в роботі в просторі представлень, в тому числі і генерації нових даних.

2.3 Методи попередньої обробки даних начального набору

З урахуванням інформації, яка викладена в попередніх пунктах, для дослідження процесу вирішення задачі класифікації на основі модифікованого навчального набору отримується не складний алгоритм дій:

- Отримати відображення набору у просторі латентних представлень за (2.18).
- Скористатися моделлю оцінки (2.19) або перейти в простір головних компонент і працювати з моделлю у маргінальних розподілах(2.20)
- Провести необхідні маніпуляції над даними або оцінки.
- При необхідності відновити дані декодером (попередньо пройти зворотній МГК в разі роботи із (2.20))
- Дослідити модифікований набір у класифікаторі

Єдина складність полягає в тому, що на момент початку роботи із набором в нас має буди вже навчена модель автокодувальника. З іншого боку, така умовність розкриває перевагу таких моделей. Так як основна ціль таких мереж – компресія та відновлення вхідних даних, то є можливість використання вже натренованої моделі для отримання латентних представлень будь-яких даних. Головне, щоб навчальний набір був якісним. Після тренування, будь-яке зображення може бути представлене у формі латентного вектора, який отримується при роботі частини-кодувальника. Під

час проведення досліджень частина навчального набору аерозйомки була спрямована у аутокодуальник, який навчався на датасеті супутникового спостереження і повітряні дані були відновлені з тією ж якістю, що й супутникові, на яких навчалась модель.

Під час проведення даного дослідження тренувалися декілька видів автокодувальників, з яких була обрана одна, представлена у таблиці 2.1. В подальшому викладенні буде згадуватись ця модель, яка тренувалась на навчальному наборі з пункту 1.2.

Таблиця 2.1 Архітектура аутокодувальника

Шар	**Вихідна форма даних**	**Кількість параметрів**
Conv2d-1	32, 32, 32	896
ReLU-2	32, 32, 32	0
Conv2d-3	64, 16, 16	18,496
ReLU-4	64, 16, 16	0
Conv2d-5	128, 8, 8	73,856
ReLU-6	128, 8, 8	0
Conv2d-7	256, 4, 4	295,168
ReLU-8	256, 4, 4	0
Conv2d-9	256, 2, 2	590,08
ReLU-10	256, 2, 2	0
ConvTranspose2d-11	256, 4, 4	590,08
ReLU-12	256, 4, 4	0
ConvTranspose2d-13	128, 8, 8	295,04
ReLU-14	128, 8, 8	0
ConvTranspose2d-15	64, 16, 16	73,792
ReLU-16	64, 16, 16	0
ConvTranspose2d-17	32, 32, 32	18,464
ReLU-18	32, 32, 32	0
ConvTranspose2d-19	3, 64, 64	867
Tanh-20	3, 64, 64	0

2.3.1 Метод видалення дублікатів з навчального набору

Необхідно зазначити, що задача по пошуку схожих зображень поставала задовго до популяризації нейромережевого підходу, і вже має деякий масив

рішень. Популярними є методи, що базуються на порівнянні локальних та глобальних дескрипторів зображення або на побудові порівняльних хеш-функцій. Обидва види методів та їх окремі реалізації охарактеризовані в роботі [23]

В нашому випадку, коли за допомогою (2.18) замість ЦЗ отримується вектор дійсних чисел, виник певний інтерес перевірити який результат дасть застосування відомих метрик, таких як косинусна схожість, до таких векторів.

Косинусна схожість — це метрика, яка вимірює косинус кута між двома векторами у багатовимірному просторі. Ця метрика використовується для визначення ступеня схожості між двома векторами на основі кута між ними, незалежно від їхньої довжини. Косинусна схожість визначається наступним рівнянням:

$$Sim(A,B) = \frac{A \bullet B}{\|A\| \bullet \|B\|}, \tag{2.21}$$

де A, B – вектори, а $\|A\|, \|B\|$ – їхні норми.

Косинусна схожість має декілька ключових переваг, зокрема:

- Ігнорування розмірності векторів: ця метрика вимірює лише напрямок векторів, ігноруючи їх масштаб. Це робить її ідеальною для порівняння векторів, де абсолютні значення компонентів не мають значення, як у випадку з текстовими даними або зображеннями.
- Ефективність в обчисленнях: косинусна схожість зазвичай потребує менше обчислювальних ресурсів, особливо коли працює зі стислими або розрідженими даними.
- Робота з нормалізованими даними: ця метрика особливо корисна, коли дані були нормалізовані або коли важливо зменшити вплив різних масштабів характеристик.

Наукові дослідження, такі як робота [24] про векторні просторові моделі в інформаційному пошуку, підкреслюють значення косинусної схожості в

аналізі текстових даних, демонструючи її здатність ефективно виявляти семантичні відносини між документами, навіть у великих текстових корпусах.

Косинусна схожість може застосовуватись в широкому спектрі задач, включаючи машинне навчання, аналіз соціальних мереж, організацію груп документів і класифікацію текстів, де вона допомагає забезпечити точність у групуванні та пошуку схожості.

Тепер, маючі (2.18) та (2.20) сформуємо послідовність дій для видалення дублікатів в кожному класі навчального набору повітряних спостережень:

- Обрахунок векторів: Спочатку потрібно отримати латентні вектори для кожного об'єкта у датасеті, використовуючи (2.18). При цьому зберігаються мікти класів y_l.
- Для кожного класу виділити набір латентних векторів кількості N_c, такої що $N = \sum_{c=1}^{k} N_c$
- Розрахунок: для кожної пари векторів в класі обраховується косинусна схожість згідно вищезазначеної формули (2.21).
- В результаті будується матриця схожості *M* для кожного класу, де кожен елемент матриці відображає схожість між парою зображень. Великі значення в матриці позначають високу схожість, тоді як низькі значення вказують на низьку(0 .. 1).
- Задається граничний рівень T, в порівнянні з яким приймається рішення про видалення: якщо $Sim(x_i, x_j) > T$, то x_i, x_j - дублікати
- Проходимо над або під основною діагоналлю матриці *M*, і визначаємо дублікати

Метод орієнтовано на оцінку і видалення дублікатів у межах кожного класу окремо через питання оптимізації та доречності. Будувати матрицю схожості на весь датасе, тобто розміром *N*x*N*, а потім проходити по ній може викликати великі затрати часу. Також, більш доречним є розгляд маніпуляцій саме на рівні класу набору, адже вони можуть знадобитися лише одному або

декільком класам. Однак, це не заважає у разі необхідності розширити метод на увесь набір навчальних даних.

2.3.2 Метод генерації нових даних для класів навчального набору

Головною ціллю генерації нових даних на основі існуючих є можливість проведення аугментації набору за рахунок згенерованих. Це корисно, якщо в навчальному наборі відсутня збалансованість даних і в деяких класах зображень значно менше за інші. Але в перспективі при маніпуляції з даними на рівні моделі оцінки розподілу може з'явитись можливість генерувати зображення, які виходять за межі класу, на основі якого проводилася оцінка.

По аналогії із методом видалення дублікатів вибір методу генерації полягає в тому, що при наявності моделі оцінки функції щільності та розподілу, можна використати методи, які спираються на ці функції. Пам'ятаємо, що завдяки (2.20) нам не обов'язково оцінювати функції тієї ж розмірності, що і розмір векторів латентного простору (2.18). Тобто, при переході у простір головних компонент, кожну з них ми можемо оцінити окремо і для кожної окремо згенерувати нові значення, які потім комбінуються разом і підлягають зворотного ходу МГК, формуючи таким чином вже згенеровані латентні вектори.

В процесі дослідження було випробувано три методи генерації:

- Одновимірний метод режекції (виключень)[25]
- Одновимірний метод зворотньої функції[26]
- Двовимірний метод виключень[27]

Останнім варіантом був саме двовимірний метод виключень, але принцип його роботи (рис. 2.9) простіше пояснити з одновимірного випадку.

Нехай g — двовимірна область, яка з одного боку обмежена деяким інтервалом $[a, b]$ осі абсцис, а з іншого — графіком функції $f_\eta(t) \geq f$, яка є щільністю розподілу випадкової величини η, що моделюється рис.

Помістимо область g всередину іншої області G(на рис. 1 $G -$ прямокутна область, як це зазвичай буває на практиці; $t -$ абсциси точок, $f -$

ординати точок), що також обмежена на осі абсциси інтервалом $[a, b]$і нехай точка (t, f) – реалізація випадкового двовимірного вектора (η, ξ), рівномірно розподіленого в області G.

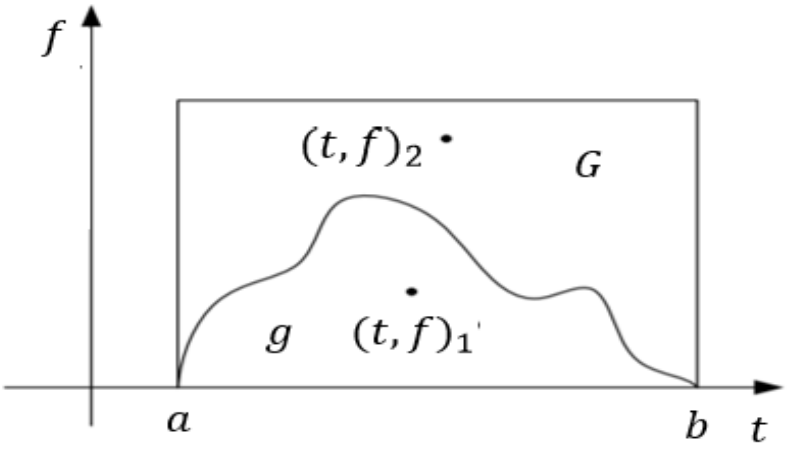


Рис.2.9 Ілюстрація методу виключень

Тоді процедура полягає в тому, що при виконанні нерівності $f_\eta(t) \geq f$ значення y приймається як реалізація випадкової величини із заданим законом розподілу.

Відповідно в двовимірному випадку функція буде залежити від двох змінних, а інтервал стане площиною і алгоритм для такого моделювання полягає у наступному:

- Обирається один з тренувальних класів зображень, який необхідно змоделювати.
- Кожне зображення даного класу проходить через (2.18), після чого отримується матриця розмірності $N_c \times 1024$, де N_c – кількість зображень обраного класу.
- Дані ортогоналізуються за допомогою методу головних компонент. Кількість перших компонент – 1000.
- Методом двовимірних виключень компонента розглядаються попарно як окремі двовимірні вибірки з N_c значень. На основі кожної з 500 вибірок моделюються нові значення (одна вибірка –

дві компоненти), які в сукупності дадуть нове зображення в області представлень МГК. Моделюється N_{nc} таких зображень.

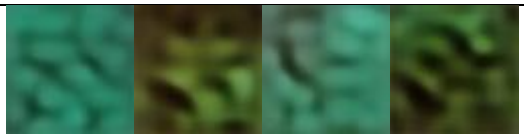
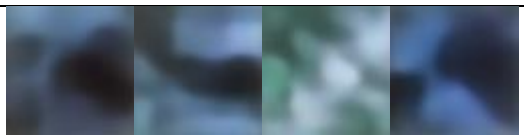
- Змодельовані дані проходять зворотне МГК – перетворення, набуваючи вигляду $N_{nc} \times 1024$.
- Новоутворений набір даних проходить через декодер, вертаючи нові зображення.

Необхідно зауважити, що така кількість перших компонент була взята вже після першої варіації, де їх було лише 20. Це здавалось логічним, адже перші 20 компонент несли в собі більше 95 відсотків варіабельності даних, але результат моделювання (таблиця 2.2) і подальшої класифікації був незадовільним

Для того, щоб впевнетись в тому, що перших 20 компонент замало, через процедуру були пропущенні оригінальні дані, із пропуском в кроці моделювання методом виключень. Після проведення прямого і зворотного ходу МГК, і наступного відновлення через декодер оригінали вийшли занадто згладженими і після цього експериментально була підібрана кількість в 1000 головних компонент, після якої оригінальні зображення практично не зазнали ушкоджень.

Також зазначимо, що на відміну від попередньо введеного методу, метод генерації має застосовуватись саме для по-класової генерації зображень, тому що його застосування до всього датасету призведе до того, що ознаки різних класів можуть потрапити на одне зображення, що на цьому етапі не бажано.

Таблиця 2.2 Результат моделювання при взятті перших 20 компонент

Клас зображень	Нові зображення
Будова	
Ліс	
Водойми	

2.4 Висновки до другого розділу

В даному розділі подано принцип роботи нейромережевих класифікаторів та аутоенкодерів, а також їхніх згорткових аналогів. Введена процедура отримання латентних представлень для навчального набору та описана модель оцінки розподілу цього набору у просторі латентних (багатовимірних) представлень, та надано опис методів видалення дублікатів та генерації нових даних для класів навчального набору.

РОЗДІЛ 3.
ОПИС СКЛАДОВИХ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

Дана інформаційна технологія є результатом розробки, яка базується на матеріалах, що описані в другому розділі. Вона спрямована на ефективне вирішення задач класифікації через комплексний підхід до аналізу і обробки даних.

Дана технологія включає п’ять ключових модулів, кожен з яких відіграє важливу роль у процесі підготовки, аналізу та використання даних:

- Основні сутності — визначають структуру та типи даних, з якими працює система, закладаючи основу для усієї архітектури рішення.
- Модуль тренування автоенкодера — центральний елемент для вивчення глибинних прихованих зв’язків у даних, що дозволяє стисло представляти інформацію та покращувати якість класифікації.
- Модуль тренування та тестування класифікаторів — відповідає за формування та оцінку моделей класифікації на основі навчальних та тестових наборів даних.
- Модуль генерації нових даних — забезпечує збільшення обсягу тренувальних даних за рахунок синтезу нових прикладів, що сприяє збалансуванню датасетів і підвищенню узагальнюючої спроможності моделей.
- Модуль видалення дублікатів — покликаний очищати набори даних від повторень, що покращує якість та здатність моделей точно класифікувати унікальні об'єкти.

Кожен з цих модулів має стратегічне значення і сприяє підвищенню ефективності всієї системи класифікації, дозволяючи вирішити поставлені задачі.

3.1 Основні сутності

Класи нейронних мереж:

- Autoencoder - це базовий клас для конволюційного автоенкодера. Він має дві основні частини: енкодер і декодер. Енкодер послідовно застосовує згортки, нормалізацію, ReLU активацію та пулінг для стискання вхідного зображення до меншого латентного представлення. Декодер виконує обернене перетворення, використовуючи зворотні згортки для розширення латентного представлення назад до розмірів оригінального зображення.
- ConvAutoencoderNoPool - модифікована версія автоенкодера, яка використовує згортки з кроком у два для зменшення розміру замість використання пулінгу. Це може допомогти у збереженні більше деталей при кодуванні.
- ConvAutoencoder128 і ConvAutoencoder256 - це спеціалізовані версії автоенкодерів, призначені для роботи з зображеннями різних розмірів. Вони використовують різні конфігурації та глибини згорток для адаптації до різних вхідних розмірів.
- ModifiedClassifier - класифікатор, який використовує кілька повнозв'язних шарів для класифікації векторів особливостей, отриманих з автоенкодера. Класифікатор проходить через кілька шарів з ReLU активацією та закінчується шаром softmax для визначення класу.
- AutoencoderClassifier - клас, що об'єднує автоенкодер і класифікатор у єдину модель, де вхідні дані спочатку кодуються автоенкодером, а потім результат передається класифікатору.

Їхні функції:

- train_coder і train_coder_with_cross - функції для тренування автоенкодера з можливістю використання крос-валідації для підвищення надійності оцінки моделі.
- train_AutoClass і train2 - функції для тренування комбінованої моделі автоенкодера та класифікатора. Ці функції використовують різні стратегії для оптимізації, включаючи окреме та спільне тренування автоенкодера та класифікатора.

На даний момент, як говорилося вище, в основному дослідженні використовується клас ConvAutoencoderNoPool – аутокодувальник без шарів пулінгу, який тренується з використанням функції train_coder_with_cross(). Але всі ці класи в тій чи іншій мірі зіграли свою роль упродовж проведення досліджень і в перспективі ще знадобляться.

Наступний набір сутностей пов’язаний із вибірками та сплайн-оцінками.

Клас _2DimentionalSpline, який використовується для роботи з сплайнами однієї змінної. Цей клас надає функціональність для обрахунку значень нових точок за допомогою різних типів сплайнів. Клас реалізує методи:

- new_point_value(points_values, x, spline_order='second', refill_degree='zero'). Параметри методу:
    - points_values: Список значень точок, який використовується для розрахунку нової точки.
    - x: Значення аргументу, для якого потрібно знайти значення сплайна.
    - spline_order: Порядок сплайна (може бути 'second' або 'third').
    - refill_degree: Ступінь інтерполяції (може бути 'zero' або 'first').
  - Функціональність: вибирає відповідний метод розрахунку на основі порядку і рівня уточнення сплайна і повертає обраховане значення.
- spline20(p_values, x). Параметри методу:

        - p_values: Список значень точок для обрахунку.
        - x: Значення аргументу.
    - Функціональність: Реалізує сплайн другого порядку з нульовим рівнем уточнення. Використовує три точки для обрахунку.
- spline21(p_values, x). Параметри:
        - p_values: Список значень точок для обрахунку.
        - x: Значення аргументу.
    - Функціональність: Реалізує сплайн другого порядку з першим рівнем уточнення. Використовує п'ять точок для обрахунку.
- spline30(p_values, x). Параметри:
        - p_values: Список значень точок для обрахунку.
        - x: Значення аргументу.
    - Функціональність: Реалізує сплайн третього порядку з нульовим рівнем уточнення. Використовує чотири точки для обрахунку.

В дослідженнях фігурувала функція spline20(), інші були реалізовані для подальшого порівняння роботи методів при усталених інших параметрах.

Клас _3DimentionalSpline, який використовується для роботи з сплайнами двох змінних – основних в цій роботі. Цей клас надає функціональність для обрахунку значень нових точок за допомогою сплайнів в тривимірному просторі.Методи класу:

- new_point_value(points_values, x, y, spline_order=2, refill_degree=0). Параметри:
        - points_values: Список значень точок.
        - x: Значення аргументу x.
        - y: Значення аргументу y.
        - spline_order: Порядок сплайна (2 або 3).
        - refill_degree: Ступінь заповнення (0 або 1).
    - Функціональність: Вибирає відповідний метод розрахунку на основі порядку і ступеня сплайна і повертає обраховане значення.

- spline20(p_values, x, y):
    - Функціональність: Реалізує сплайн другого порядку з нульовим рівнем уточнення. Використовується для обрахунку значення на основі трьох точок.
- spline21(p_values, x, y):
    - Функціональність: Реалізує сплайн другого порядку з першим рівнем уточнення. Використовує більшу кількість точок для точнішого обрахунку.

В цьому класі також реалізовані методи на перспективу подальших досліджень, а саме обрахування сплайну із більшим рівнем уточнення.

Наступні сутності пов’язані із реалізаціями вибірки та багатовимірної вибірки.

Клас Sample, який реалізує статистичний аналіз одновимірного набору даних. Цей клас може використовуватись для різних цілей, включаючи аналіз розподілу даних, оцінку їх параметрів та формування емпіричних функцій розподілу. Конструктор класу має наступний вигляд:

- __init__(self, massive_of_data, classes_count=0): Ініціалізує клас з масивом даних. Під час ініціалізації відбувається калькуляція різних статистичних параметрів та формування класів для гістограми.
    - massive_of_data: Вхідний масив даних.
    - classes_count: Кількість класів для гістограми. Якщо не задано, то клас сам визначає оптимальну кількість.

Основні атрибути класу:

- data_list: Масив вхідних даних.
- midle_value, dispercion_value, sigma_value, asimetric_coef, exces: Різні статистичні характеристики набору даних.
- classes: Словник, що містить дані, розбиті на класи за значеннями.

Методи класу

- midle_value_calculate(): Обраховує середнє значення набору даних.
- dispercion_calculate(): Обраховує дисперсію.
- sigma_calculate(): Обраховує стандартне відхилення.
- asimetric_calculate(): Обраховує коефіцієнт асиметрії.
- ecxes_calculate(): Обраховує ексцес.
- ecxes_z_calculate(): Обраховує скоригований ексцес.
- define_classes_count(): Визначає кількість класів для розбиття даних.

Основна ідея цього класу полягає в тому, що він дозволяє здійснювати початковий статистичний аналіз одновимірного набору даних, включаючи формування емпіричної функції розподілу та визначення різних параметрів розподілу. Тому саме він використовувався при оцінці одновимірної функції щільності.

Клас MultiSample, який є контейнером для обробки та аналізу багатовимірних даних. Цей клас використовується для створення множини одновимірних зразків і надає можливості для розрахунку багатовимірних частот та кореляційних матриць між ознаками. Конструктор має наступний вигляд:

- __init__(self, lines=None): Ініціалізує клас із можливістю додавання зразків даних.
    - lines: Масив даних, з якого формуються зразки.

Атрибути класу

- samples: Список об'єктів класу Sample, кожен з яких містить зразок даних.
- dimensions: Кількість зразків (вимірів) у контейнері.
- rel_frequency: Масив, що відображає взаємні частоти різних комбінацій класів зразків.
- correl_matrix: Матриця кореляції між різними зразками.

Методи

- add_samples(self, lines): Додає декілька зразків до контейнера.
- add_sample(self, sample): Додає один зразок до контейнера.
- add_range_samples(self, samples_list): Додає список зразків до контейнера.
- calculate_multidimensional_frequencies(self): Розраховує багатовимірні частоти для всіх комбінацій класів ознак.
- new_correlation_matrix(self): Створює нову матрицю кореляції на основі існуючих зразків.

Цей є корисним для статистичних аналізів, де потрібно одночасно розглядати взаємозв'язки між кількома зразками, як при попарній генерації нових головних компонент.

Останні сутності несуть важливу роль зв’язку з іншими модулями, тому що саме вони звертаються до класів сплайнів під час процесу моделювання.

Клас Modulator. Цей клас використовується для генерації випадкових чисел з рівномірним розподілом.

- even_distr_element(self, low, high): Генерує випадкове число між low та high.

Функції для взаємодії зі сплайнами

1. find_value_in_1d_spline(sample, X):
   Використовує 1D сплайн для визначення значення за допомогою індексу, заснованого на вхідному значенні X.
2. find_value_in_2d_spline_old(ms, X, Y):
   Отримує значення з 2D сплайна використовуючи старий метод, який залежить від дискретних індексів X та Y.
3. find_value_in_2d_spline(ms, X, Y):
   Оновлений метод для отримання значення з 2D сплайна, використовуючи numpy для пошуку індексів.

Функції для моделювання за допомогою сплайнів

1. modul_2d_with_spline(multi_s, n):

   Генерує нові точки за допомогою 2D сплайна до тих пір, поки не буде досягнута необхідна кількість точок n.
2. modeling_through_2d_splines_threaded(multi_sample, need_count, last_file_name):

   Використовує багатопоточність для паралельного моделювання за допомогою 2D сплайнів. Записує результати у файл.
3. modeling_through_2d_splines(multi_sample, need_count, last_file_name):

   Моделює дані використовуючи 2D сплайни без використання багатопоточності.

Ці функції та класи забезпечують важливий інструментарій для аналізу та моделювання даних, зокрема для створення інтерполяцій та оцінювання щільності за допомогою сплайнів у багатовимірному просторі.

На цьому опис основних сутностей завершено, перейдемо то опису більш функціональних модулів.

3.2 Модуль тренування автокодувальників

Цей модуль є стартовим для початку досліджень, адже як вже було сказано раніше, тільки після тренування автокодувальника для переходу у простір латентних векторів можна рухатись далі. Він включає в себе основний скрипт і два тестових, які запускалися для виявлення оптимальної кількості ГК. Вони згадувалися в минулому розділі. Тому опишимо тільки основний скрипт:

1. Імпорти:
   - torch, torch.nn, torch.optim: Основні компоненти PyTorch для моделювання нейронних мереж та оптимізації.
   - torchvision: Завантаження стандартних датасетів та трансформацій.

- matplotlib.pyplot: Бібліотека для візуалізації результатів тренування.

2. Налаштування пристрою:
   - Використання GPU за наявності (CUDA), інакше CPU.
3. Трансформація даних:
   - Перетворення зображень до заданого розміру, нормалізація за допомогою заданих середніх значень та стандартних відхилень.
4. Завантаження даних:
   - Створення тренувального набору з вказаної папки.
   - Ініціалізація DataLoader для ефективного завантаження даних.
5. Створення та тренування моделі:
   - Вибір моделі на основі розміру вхідних зображень (64x64, 128x128 або 256x256).
   - Задання критерію втрат (MSE) та оптимізатора (Adam).
   - Тренування моделі з використанням функції train_coder_with_cross, що забезпечує тренування та валідацію.
6. Збереження моделі:
   - Модель зберігається у файл після тренування для подальшого використання.

Цей скрипт також можна назвати типовим прикладом використання PyTorch для тренування нейронних мереж на задачах обробки зображень. Він демонструє загальний підхід до тренування з використанням конволюційних автоенкодерів для зменшення вимірів даних та можливості генерації нових даних за допомогою навченої моделі.

3.3 Модуль тренування та тестування класифікатор

Цей модуль складається із двох великих скриптів, один з яких відповідає за тренування, а другий – за тестування, відповідно.

Перший скрипт використовується для тренування класифікатора з використанням автоенкодера як частини попереднього тренування. Це дозволяє використовувати вивчені автоенкодером ознаки для підвищення ефективності класифікації. Файл включає наступні основні компоненти:

Імпорти

- ConvAutoencoder: Модуль, який містить класи для автоенкодера.
- torch, torch.nn, torch.optim, torchvision: Основні бібліотеки для роботи з нейронними мережами.
- matplotlib.pyplot: Використовується для візуалізації процесу тренування.

Конфігурація тренування

- Підготовка даних: Використання трансформацій для зображень, які включають зміну розміру, конвертацію в тензори та нормалізацію.
- Ініціалізація DataLoader для ефективного завантаження даних.

Модель

- autoencoder: Ініціалізація і завантаження автоенкодера без пулінгу.
- classifier: Класифікатор, який приймає ознаки з автоенкодера.

Функції втрати та оптимізатори

- Два види втрат: CrossEntropyLoss для класифікації та MSELoss для автоенкодера.
- Оптимізатори: SGD для класифікатора та Adam для автоенкодера.

Тренування моделі

- Виконує тренування моделі, використовуючи спеціалізовану функцію train_AutoClass, яка оптимізує класифікатор та автоенкодер.
- Збереження тренованої моделі для подальшого використання.

Візуалізація

- Відображення зміни функції втрати під час тренування за допомогою графіка.

Скрипт демонструє інтеграцію попереднього навчання автоенкодера та класифікаційних завдань в єдиний тренувальний процес, що забезпечує більш ефективне використання ознак, видобутих з автоенкодера.Як сказано вище, під час тренування скрипт вимірює значення точності навчання і функції втрат. В кінці ця інформація відображається на спеціальних графіках (рис. 3.1 та 3.2)

Така модель дозволяє заощадити час, адже згорткова частина мережі – класифікатора по суті заміняється частиною-енкодером попередньо навченого аутокодувальника, вона може як донавчатися, так і залишити свої ваги незмінними.

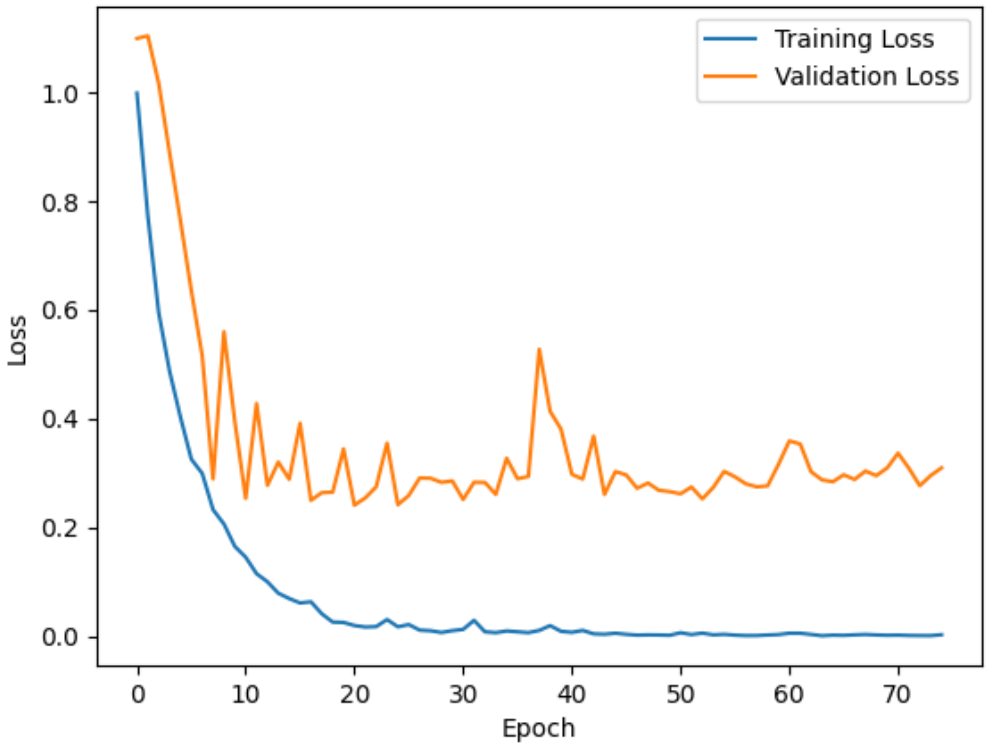


Рис.3.1 Приклад графіку зміни функції втрати під час навчання класифікатора

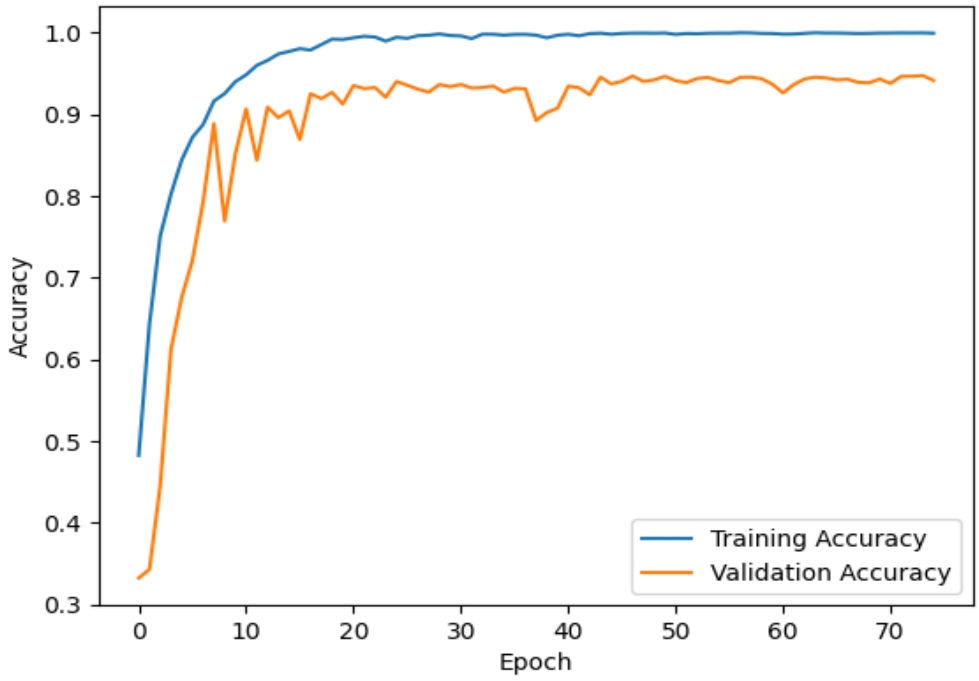


Рис. 3.2 Приклад зміни точності мережі під час навчання класифікатора

Другий скрипт містить код для оцінки класифікатора, зокрема через використання матриці помилок (confusion matrix). Скрипт завантажує попередньо навчену модель та використовує її для класифікації зображень з тестового датасету. Основні компоненти цього скрипту включають:

Імпорти

- Імпорт бібліотек та модулів, необхідних для роботи з нейронними мережами, обробкою зображень і візуалізацією.

Функції

- CalculateConfusionMatrix: функція для розрахунку та візуалізації нормалізованої матриці помилок. Виводить матрицю у форматі зображення, що дозволяє оцінити якість класифікації.

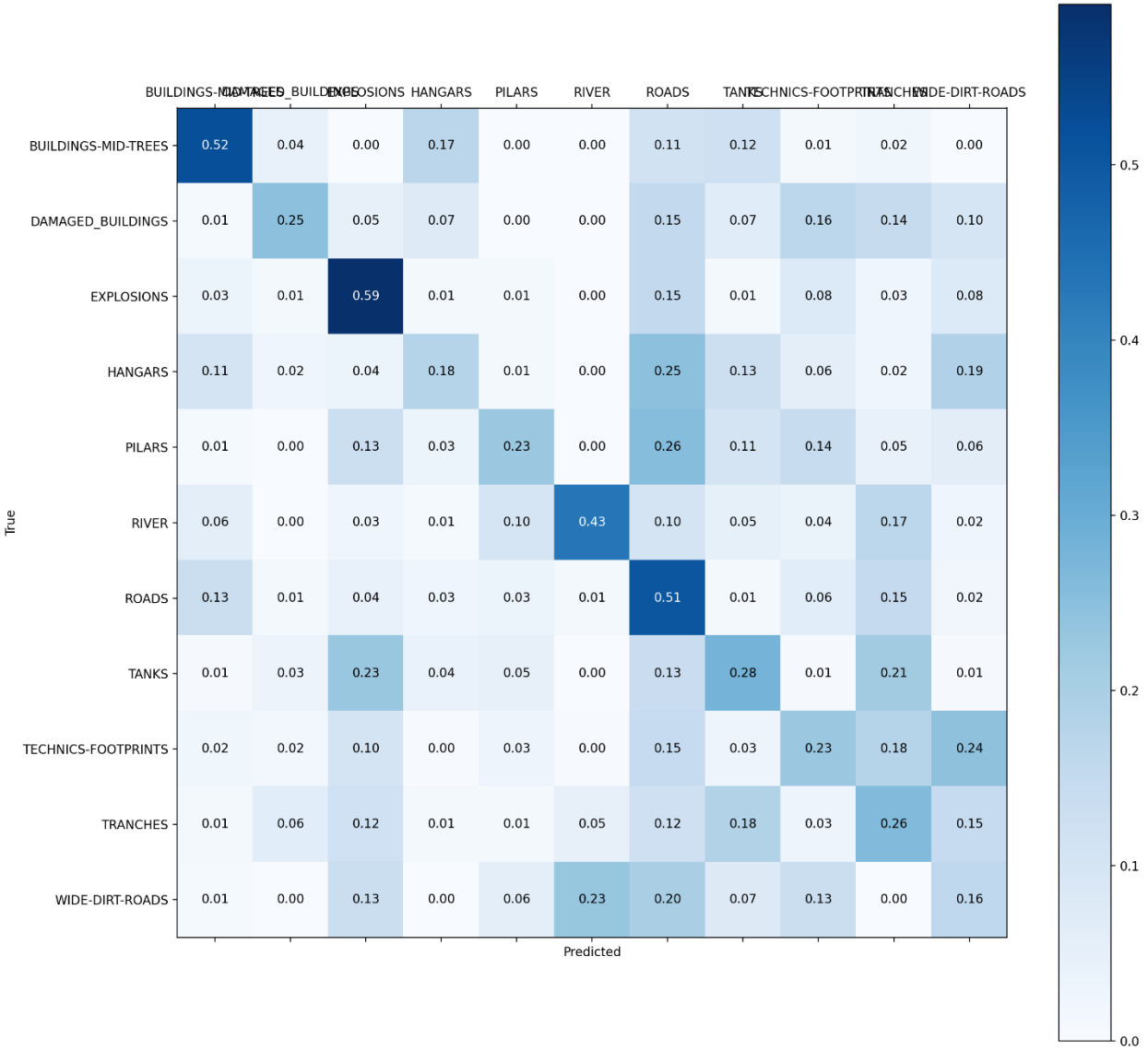
Основний блок

- Ініціалізація моделі: завантаження збереженої моделі з вказаного шляху та її підготовка до використання.
- Підготовка датасету: завантаження тестових зображень та їх передобработка з використанням заданих трансформацій.
- Процес тестування: проходження через тестовий набір даних, збір передбачень моделі та відповідних міток.
- Аналіз результатів: виклик функції CalculateConfusionMatrix для аналізу результатів класифікації та збереження матриці помилок.

Цей скрипт є важливим інструментом для оцінювання ефективності тренованих класифікаційних моделей, зокрема для аналізу точності розпізнавання кожного класу.

Саме на основі цих двох скриптів будується аналіз всього дослідження, адже графік зміни функції втрати та графік точності несуть суттєву інформацію про поведінку мережі під час тренування, навіть на прикладі рисунку 3.1 можна сказати що мережа, яка тренувалася в той момент увійшла у перенавчання, про що свідчить сильно асцильований графік валідаційної функції втрат, яка до того ж після сорокової епохи почала збільшуватись, при

тому тренувальна функція втрат. Матриця помилок (рис. 3.3), в свою чергу показує результативність мережі після навчання та допомагає визначити



проблемні класи.

Рис. 3.3 Приклад матриці точності (помилок)

3.4 Модуль генерації нових зображень

Модуль генерації складається з трьох скриптів, перші два працюють у звя’зці – перший генерує файли із головними компонентами класів, а другий приймає до себе файли головних компонент і пропускає їх через декодер. Така організація була присутня до реалізації сутностей пов’язаними із сплайнами у python – середовище. Якийсь час моделюванням головних компонент на

основі існуючих займалася окрема програма, що була реалізована на .Net. Але при цьому перші два скрипти залишаються актуальними, адже проміжна робота в просторі ГК часто може допомогти виявити недоліки або закономірності. Третій скрипт вже використовує нові сутності і процес оцінки функції щільності латентних векторів у просторі головних компонент виконується вже в середені нього. Для нього і була реалізована функція з підтримкою багатопоточності, показана вище.

Отже, файловий генератор використовується для генерації нових файлів ГК на основі латентних представлень, отриманих з автоенкодера, і їх подальшої обробки за допомогою методу головних компонент (PCA

Імпорти

- Використовуються стандартні бібліотеки Python та фреймворки для роботи з нейронними мережами (torch, torchvision) і обробкою даних (numpy).

Функції

- load_model: Завантажує навчену модель автоенкодера для подальшого використання.
- get_data_loader: Створює DataLoader для зчитування даних з заданого шляху з використанням попередньо визначених трансформацій.
- get_generals: Виконує PCA трансформацію на латентних векторах і зберігає результати у файл.
- save_latents: Зберігає латентні представлення та PCA трансформовані дані для кожного класу.

Процес

- Модель автоенкодера використовується для кодування зображень у латентні простори.
- Завантажені дані трансформуються та подаються до моделі.

- За допомогою PCA аналізується структура латентних просторів, що дозволяє зменшити розмірність даних і виділити основні складові.
- Отримані латентні та PCA трансформовані дані зберігаються у відповідних файлах, розподілених за класами.

Скрипт – генератор зображень зчитує файли вигляду ГК і пропускає їх через декодер.

Імпорти:

- Використання модулів для роботи з файловою системою, обчисленнями, машинним навчанням та обробкою зображень.
- torch та torchvision для роботи з нейронними мережами та даними.
- sklearn.decomposition.PCA для виконання PCA трансформацій.

Функція load_model:

- Завантажує модель автоенкодера і переміщує її на доступний обчислювальний пристрій (CPU або GPU).
- Встановлює модель у режим оцінювання.

Функція save_images_from_PCA:

- Виконує інверсію PCA для генерації даних, які були редуковані до менших вимірів.
- Перетворює ці дані назад у зображення за допомогою декодера моделі автоенкодера.
- Зберігає реконструйовані зображення в певну директорію.
- Використовує нормалізацію та інші трансформації для коректного відображення зображень.

Головний скрипт (if __name__ == "__main__"):

- Ініціалізація пристрою для виконання обчислень.
- Завантаження та налаштування моделі.
- Перебір кожного класу з датасету для генерації нових зображень за допомогою PCA.

Як згадувалось раніше, цей модуль є частиною більшої системи для автоматизації процесу створення нових зображень із заданих латентних представлень.

Останній скрипт проводить аналіз без генерації файлів з головними компонентами і генерує нові зображення із використанням багатопоточності.

Імпорти

- Використовуються модулі для обробки даних (numpy), машинного навчання (torch, torchvision, PCA з sklearn), і файлової системи (os).
- Імпорт спеціалізованих функцій і класів з інших модулів, що входять до складу проекту, таких як ConvAutoencoder, MultiSample тощо.

Основний процес

1. Завантаження Моделі:
    - Модель автоенкодера завантажується і встановлюється в режим оцінки.
2. Завантаження Даних:
    - Дані завантажуються за допомогою функції get_data_loader, яка підготовляє DataLoader для обробки зображень.
3. Генерація Латентних Векторів:
    - Для кожного зображення з датасету генерується латентний вектор за допомогою енкодера моделі.
4. PCA Трансформація:
    - Латентні вектори кожного класу обробляються за допомогою PCA для зменшення розмірності та формування нового простору ознак.
5. Генерація Зображень:
    - Використовуються сплайни для моделювання нових даних в просторі, створеному PCA. Модельовані дані перетворюються назад до вихідного простору латентних векторів.
    - Декодер автоенкодера використовується для реконструкції зображень з отриманих латентних векторів.

6. Збереження Зображень:
    - Реконструйовані зображення зберігаються у відповідні директорії на диску.

На цьому завершено опис модулю генерації нових зображень.

3.5 Модуль видалення дублікатів

Для видалення дублікатів створено один основний скрипт і два допоміжних, як і з модулем тренування аутокодувальників. Він використовує глибокі нейронні мережі та косинусну схожість для визначення схожих зображень.

Імпорти

- Модулі для роботи з файлами, виконання математичних обчислень та обробки даних.
- Функції та класи з інших модулів проекту для обробки зображень і маніпуляцій з даними.

Основні класи і функції

1. Клас MyImageFolder:
    - Спеціалізований клас, що наслідує ImageFolder для керування завантаженням зображень. Він дозволяє ігнорувати піддиректорії без файлів.
2. Функція get_data_loader:
    - Створює DataLoader для обробки зображень з директорії, використовуючи трансформації для нормалізації та зміни розміру зображень.
3. Функція sort_images:
    - Основна функція для сортування зображень за класами і видалення дублікатів на основі обчисленої косинусної схожості між латентними векторами, отриманими від моделі автоенкодера.
4. Функція define_2d_pca:

- Використовує PCA для зниження розмірності латентних векторів зображень до двох вимірів, що полегшує обробку та аналіз.

5. Головний скрипт:

- Визначає параметри запуску, такі як пристрій для обчислень, параметри нормалізації зображень та параметри сортування (розмір зображення та поріг для визначення дублікатів).

Цей скрипт використовується для автоматизації процесу видалення дублікатів в наборах даних, забезпечуючи ефективніше використання обчислювальних ресурсів та покращення якості навчальних даних для машинного навчання.

3.6 Висновки до третього розділу

Даний розділ присвячено опису складових реалізованої інформаційної технології, описано п’ять модулів, кожен з яких відповідає за свою функції. Така організація технології забезпечує її модульність і змогу використання окремих частих без прив’язки до іншого коду. Після висвітлення структури технології можна перейди до аналізу отриманих результатів.

РОЗДІЛ 4.
ТЕСТУВАННЯ РОБОТИ РЕАЛІЗОВАНОЇ ТЕХНОЛОГІЇ

Після реалізації інформаційної технології був проведений ряд тестів для оцінки впливу запропонованих методів передобробки даних навчального набору повітряної зйомки. Ціль такого тестування – пересвідчитися у позитивному сприянні методів маніпуляції даними аерозйомки на вирішення задачі класифікаці та розпізнавання.

Було проведено 4 контрольні екперименти.

4.1 Тестування обману класифікатора

Для проведення цього тесту була взята уся структура, що описується в таблиці 1.1. Метою екмперименту було пересвідчитись, що класифікатор, який навчався на навчальному наборі аерозйомки буде обманутим і розпізнає згенеровані дані згідно міткою їхнього класу

Для генерації використовувався попередньо навчений автоенкодер, структура якого описана в таблиці 2.1 та модуль генерації нових зображень, описаний у розділі 3. Для кожного з тридцяти трьох класів методом, що описаний у пункті 2.3.2 було згенеровано по дві тисячі зображень. Тобтто було отримано цілком згенерований датасет із шестидесяти шести тисяч зображень.

Для цього експерименту на початковому датасеті був навчений з нуля згортковий класифікатор, який не приймав ніякої участі у формуванні згенерованого набору даних, ці зображення вперше потрапили в цей класифікатор при проведені екперименту.

Результати проведеного екперименту відображає рисунок 4.1, який є матрицею помилок – на основній її діагоналі знаходяться частки класів, що були правильно класифіковані.

Аналіз цієї матриці показав, що в середньому згенеровані зображення були розпізнанні вірно із точністтю 0.71. Що є задовільним результатом, враховуючі велику незбалансованість початкового набору, в якому різниця кількості між зображеннями класів могла досягати більше тисячі зображень.

Також, можна звернути увагу на те, що класи, які були класифіковані найгірше. Наприклад клас APC, який мав найнижчу точність 0.36, найбіль помилково був класифікований як AFV, що говорить про те що в цих класів багато спільних деталей

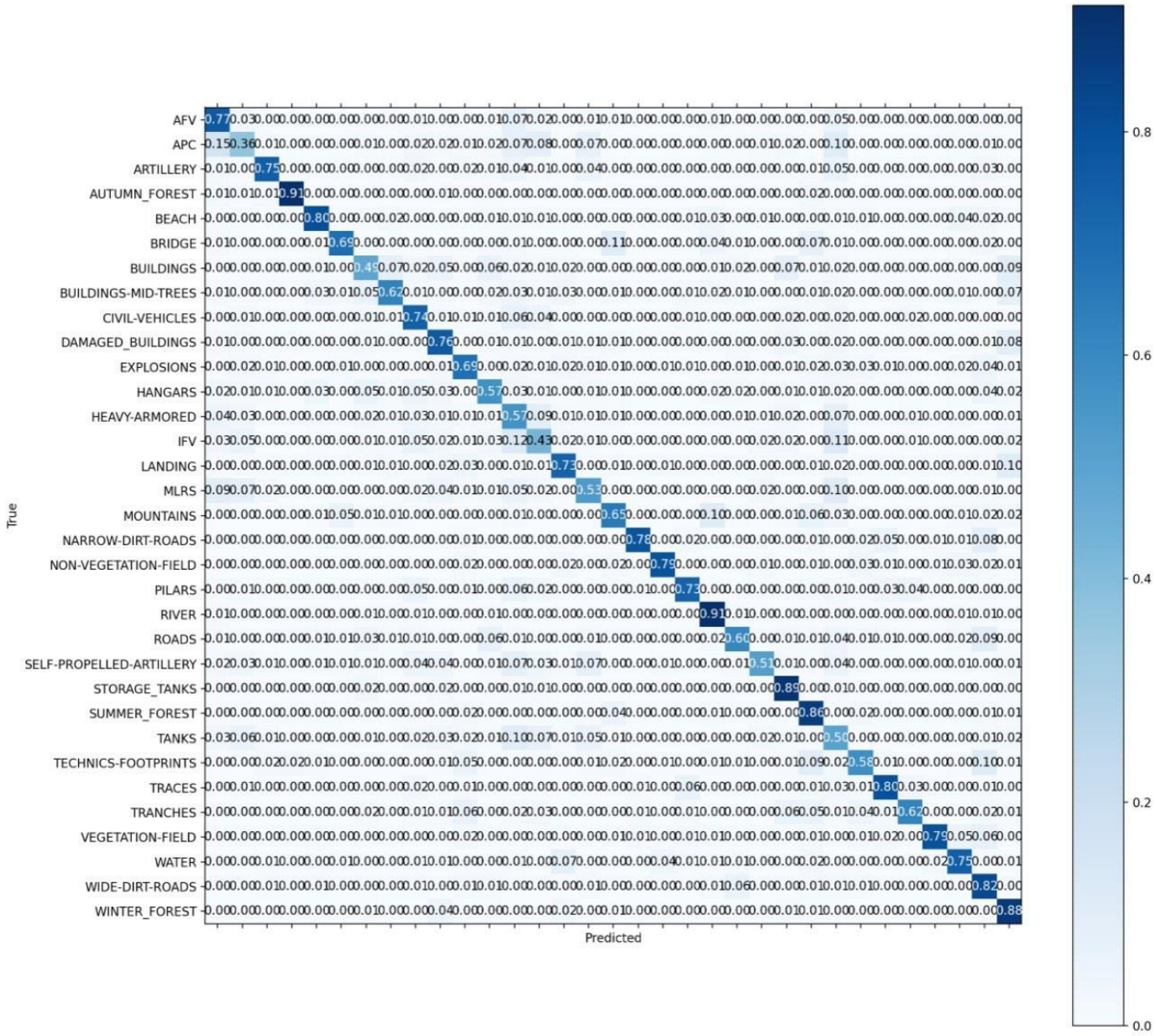


Рис. 4.1 Результат розпізнавання згенерованих зображень

4.2 Обернений експеримент з обманом класифікатора

Коли попередній експеримент показав задовільний результат, постало питання про те, чи зможе вибірка, повністю сформована із згенерованих зображень достатньо якісно натренувати класифікатор. Ті самі шістдесят

шість тисяч зображень виступили в ролі навчальної вибірки для нового класифікатора тієї самої архітектури, що і в попередньому тесті. На тестування в цей раз віддавався оригінальний набір зображень з 33 класів. Результат можна побачити на рисунку 4.2:

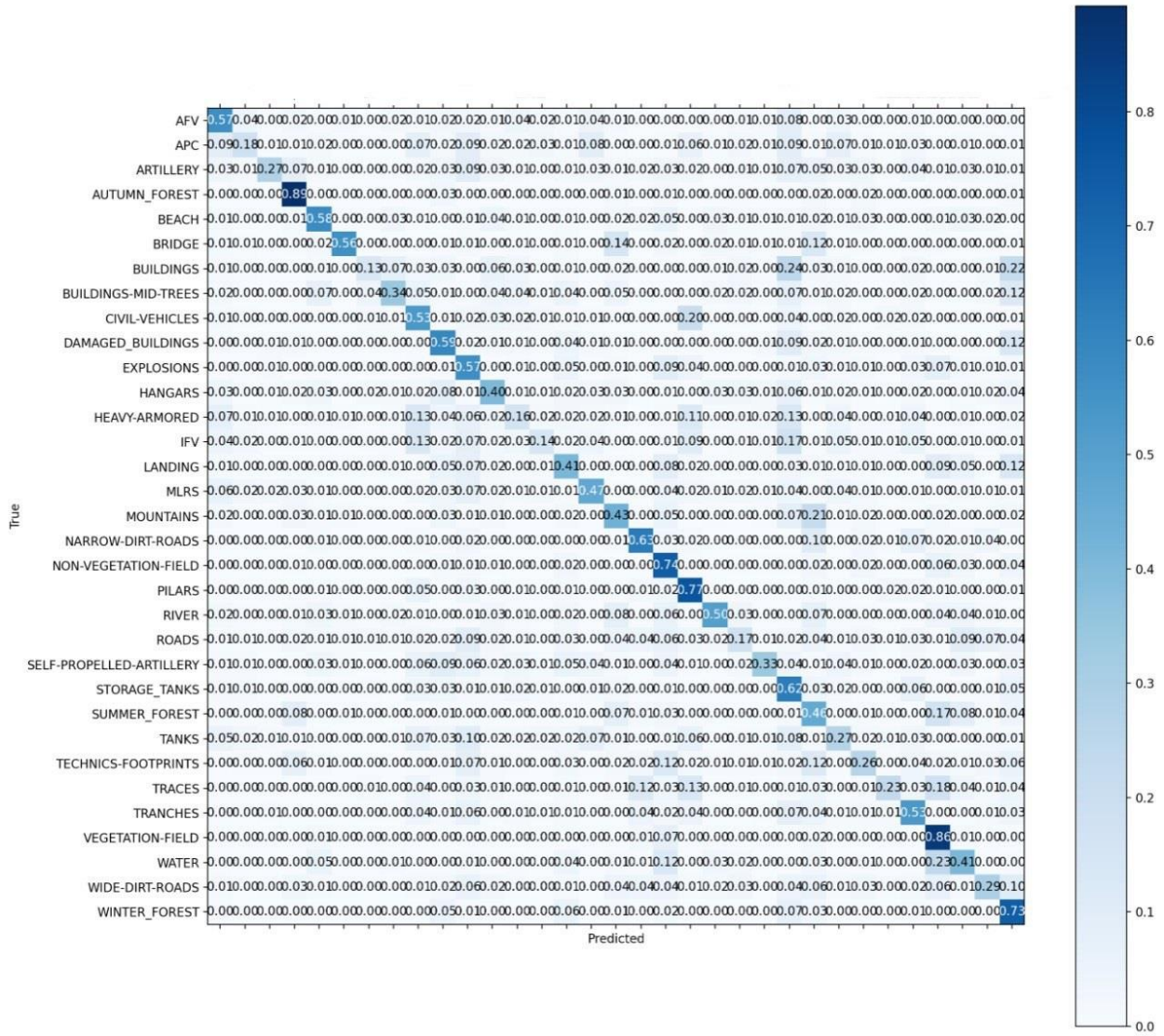


Рис. 4.2 Результат обернено експерименту обману класифікатору

Як видно з матриці розпізнавання, на цей раз отриманий результат був менш задовільний, хоча він також має пояснення. При тренуванні на згенерованих зображеннях найкраще було розпізнано класи, які мають дуже яскраві характеристики, такі як осінні та зимні ліси, водойми, або стовпи. І хоча результат виявився гіршим, ніж в першому експерименті, його також можна вважати успішним, бо практично усі неточності мали своє пояснення.

4.3 Експеримент із видаленням дублікатів

Після реалізації методу вилучення дублікатів постало питання про, наскільки якісно проходить видалення. Просто видаляти зображення з набору, зменшуючі його при цьому і не отримувати при цьому покращень – контрпродуктивно.

Тут стала в нагоді модель оцінки розподілу набору даних в просторі латентних представлень. Ідея полягає в тому, щоб довести що при видаленні дублікатів зменшується гостровершинність розподілу класу, тобто зменшується його коефіцієнт ексцесу (рисунок 4.3).

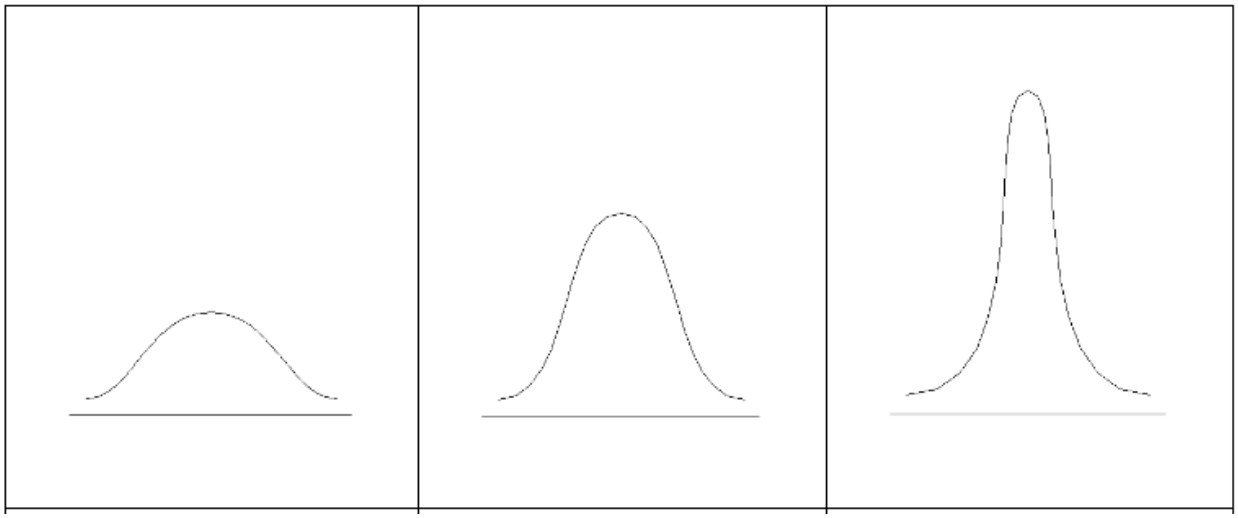


Рис. 4.3 Залежність виду щільності розподілу від коефіцієту ексцесу

Зменшення гостровершинності впливає на ентропію розподілу окремого класу, підвищуючи її. І хоча ентропія – це міра хаосу, її збільшення в межах окремого класу, може зменшити імовірність того, що мереже перенавчиться.

Диференціальна ентропія обраховується на основі функції щільності розподілу $f(x)$:

$$h(x) = -\int_x f(u) \ln f(u) du$$

Так як оцінити функцію щільності при розмірності латентного простору у 1024 виміри складно, знов скористаймось властивістю (2.20) і оцінимо коефіцієнти ексцесів у перших трьох компонент кожного з 33 класів навчального набору.

Отже, суть експерименту наступна:

- отримати латентні представлення для всіх класів з навчального набору
- перевести їх у простір головних компонент
- в перших трьох компонентах кожного класу оцінити коефіцієнти ексцесу
- проведення процедури видалення дублікатів, що описана у Розділі 2
- Повторення визначення оцінок ексцесів уже на отриманих, нових даних.

Таким чином, ми отримуємо по 33 коефіцієнти ексцесу для кожної з першої трьох головних компоненти до та після видалення. Після цього у розрізі кожної компоненти, можна побудувати лінійну регресію, де на осі абсцис лежать коефіцієнти до видалення дублікатів, а на осі ординат- після видалення дублікатів. Якщо коефіцієнт лінійної регресії при X менше 1, це означає, що коефіцієнт ексцесу після видалення зростає повільніше ніж коефіцієнт ексцесу до видалення.

Це означає, що гостровершинність розподілу перших трьох компонент після видалення дублікатів спадає. Так як перші три головні компоненти, зазвичай несуть в собі більше 70% варіабельності всього розподілу, то можна стверджувати, що коефіцієнт ексцесу загального розподілу також знижується. Експеримент проводився для двох порогових значень (t = 0,95 і t = 0,97), відповідні лінії регресії для перших головних компонент можна побачити на рисунках 4.4 та 4.5 відповідно.

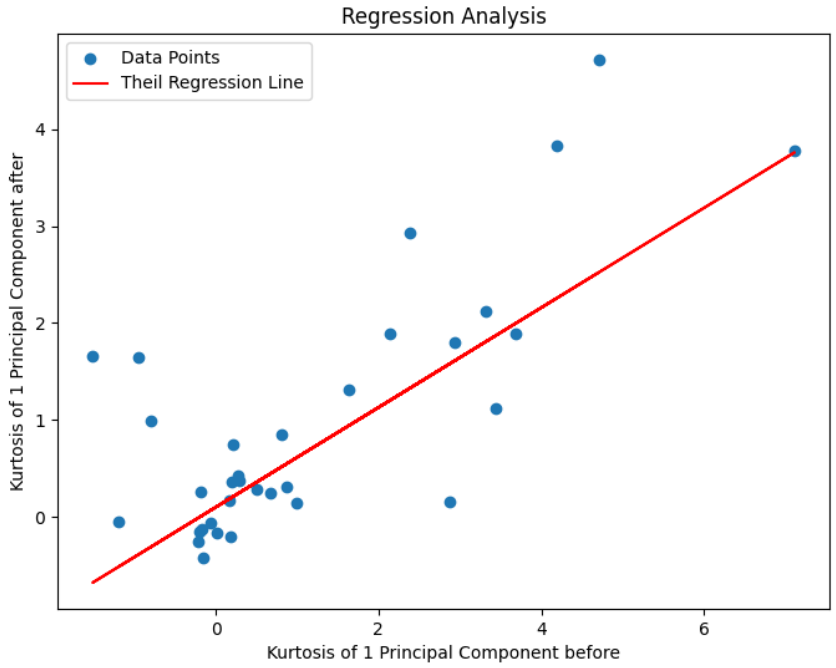


Рисунок 4.4. Регресія для коефіцієнтів ексцесів першої головної компоненти при t = 0,95.

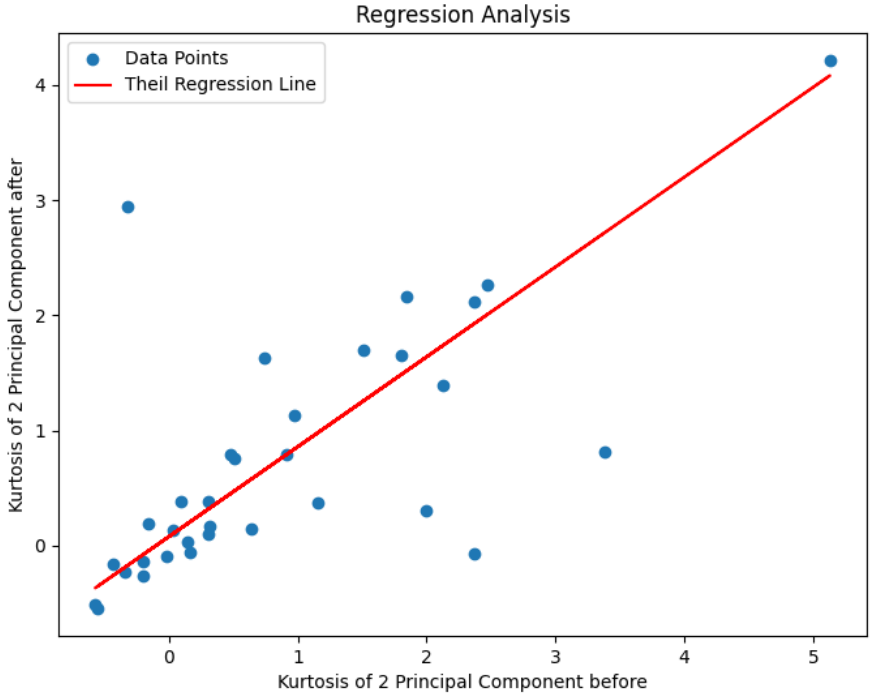


Рисунок 4.5. Регресія для коефіцієнтів ексцесів першої головної компоненти при t = 0,97.

Візуально може бути незрозуміло як саме поводить себе регресія, тому на таблиці 4.1 представлено усі рівняння залежності для всіх трьох компонент.

Таблиця 4.1 Рівняння залежності коефіцієнтів ексцесу для різних порогових рівнів.

T (граничний рівень)	**Номер компоненти**	**Рівняння**
0.97	1	y = 0.82x + 0.05
0.97	2	y = 0.78x + 0.08
0.97	3	y = 0.74x + 0.22
0.95	1	y = 0.51x + 0.10
0.95	2	y = 0.52x + 0.18
0.95	3	y = 0.61x + 0.43
0.93	1	y = 0.30x + 0.14
0.93	2	y = 0.30x + 0.26
0.93	3	y = 0.42x + 0.80

Як видно з таблиці 4.1, при зменшенні порогового рівня, коефіцієнт при X зменшується пропорційно. Якщо при t = 0,95, коефіцієнт ексцесу після видалення дублікатів зростав вдвічі менше, практично на всіх трьох компонентах, то при t = 0,93 коефіцієнт при X в середньому дорівнює 0,35. Тому ми можемо стверджувати, що процедура видалення дублікатів зменшує гостровершинність функції щільності, що в свою чергу збільшує ентропію розподілу. А це в свою чергу, зменшує ймовірність, що модуль, яка

навчатиметься на даних, отриманих після видалення дублікатів, потрапить в перенавчання.

4.4. Аугментація незбалансованої вибірки.

Як заявлялося на початку даного дослідження, можливість використання згенерованих даних як аугментація початкового датасету розглядалась в першочергово. Можливість якісно збалансувати датасет, в якому існує значний розрив у кількості даних між класами, на даний момент є основною ціллю для згенерованих даних.

Для перевірки такої можливості був сформований трьохкласовий тестовий набір в якому різниця між найменшою і найбільшою кількістю даних в класах, становила 2000 зображень. Ці дані не були присутні у початковому навчальному наборі, тому цілком підходять для тестування.

Спочатку, нейромережевий класифікатор навчався на цій незбалансованій вибірці і намагався розпізнавати дані з нашого навчального набору. Після цього, маленький набір був дозбалансований за рахунок згенерованих зображень. Знову проводилась процедура розпізнавання отриманих даних.

Результати цього експерименту відображаються на рисунках 4.6 та 4.7.

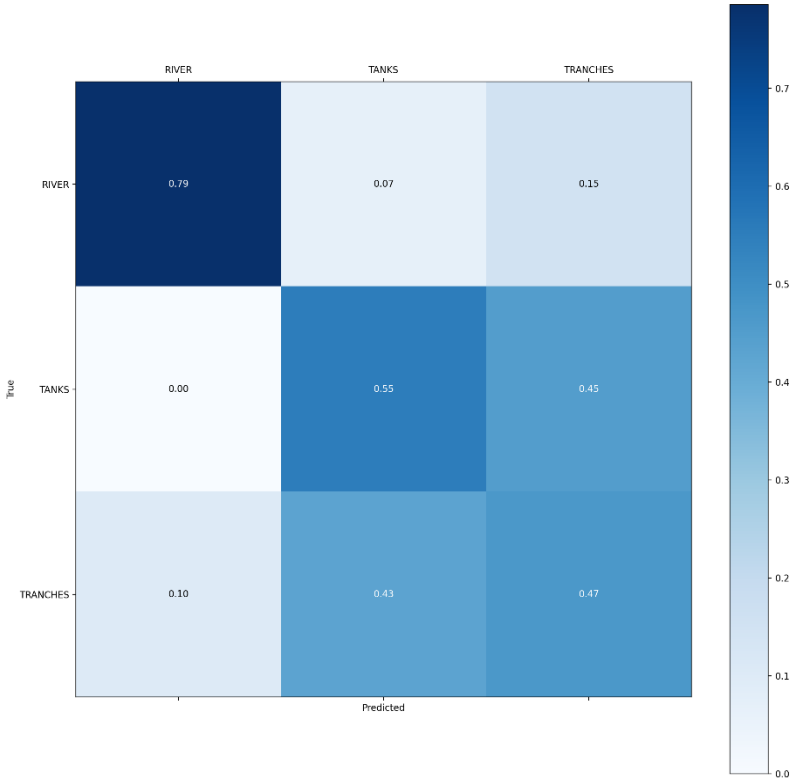


Рис. 4.6. Результат класифікації після навчання на не збалансованій вибірці.

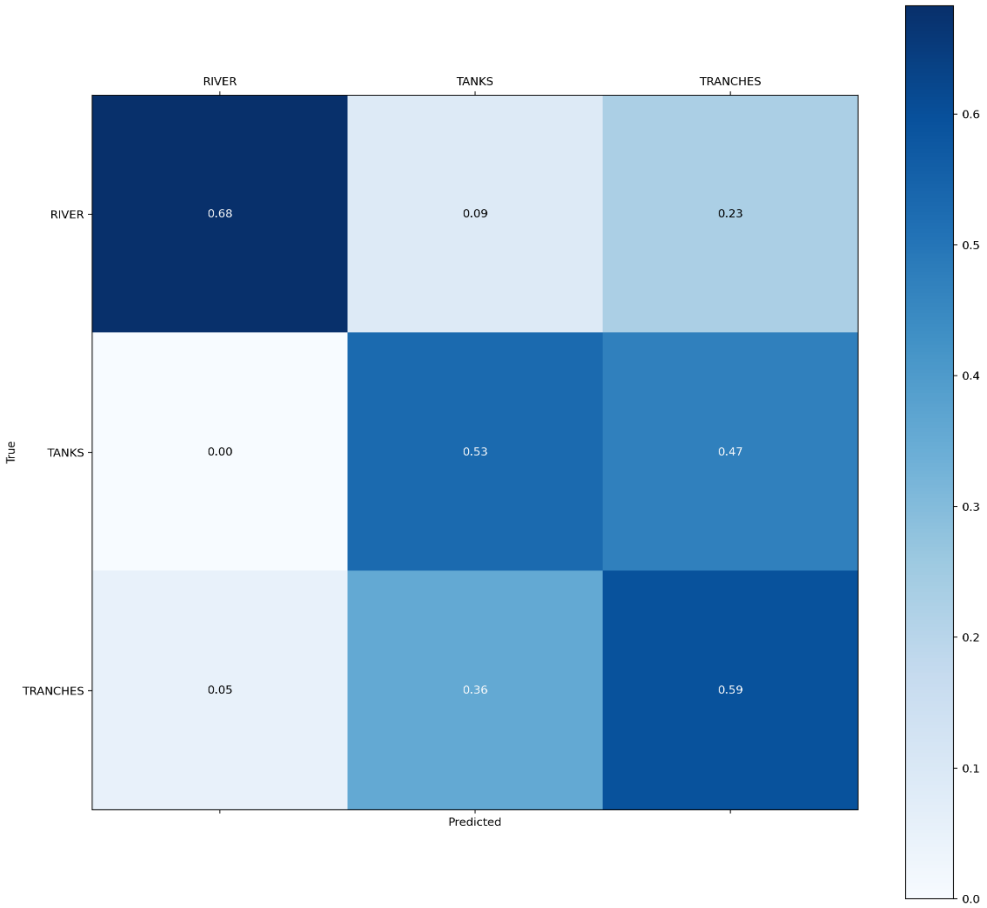


Рис. 4.7. Результат класифікації після навчання на доповненій вибірці.

З наведених матриць видно, що середня точність розпізнавання практично не змінилась, але точність розпізнавання найбільш проблемного класу (tranches) збільшилась на 12%. Отже, прослідковується позитивний ефект від збалансування.

Але більший інтерес представляє сам процес навчання. З рисунків 4.8 та 4.9 видно, що після збалансування, мережа досягла граничної точності у півтора рази швидше, ніж до нього.

Це означає, що не дивлячись на досить слабкий ефект впливу на точність розпізнавання, швидкість навчання моделей за допомогою згенерованих даних можна значно підвищити.

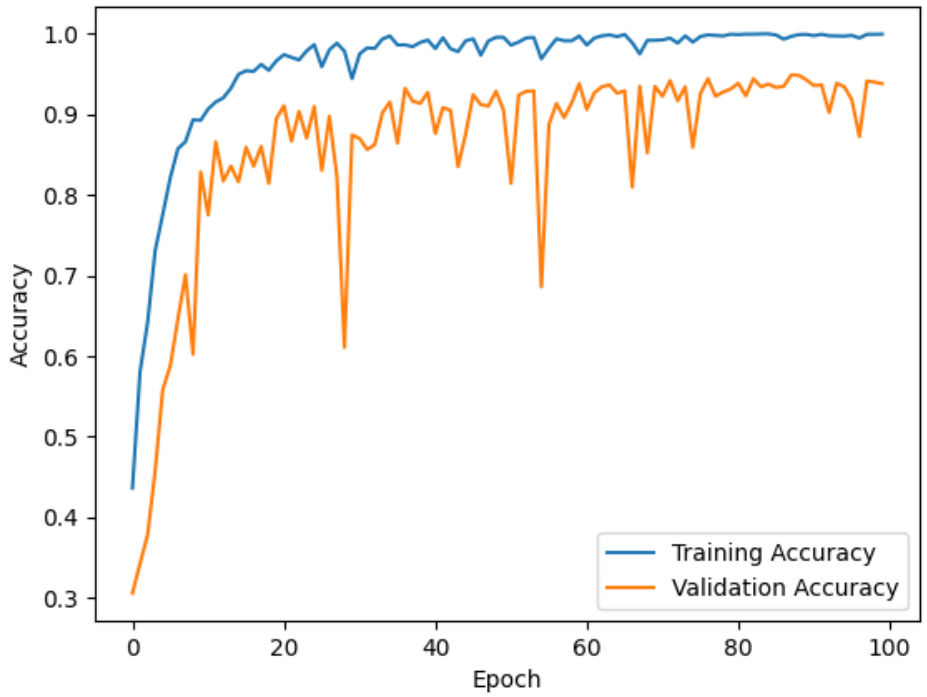


Рис. 4.8. Процес навчання на незбалансованій вибірці.

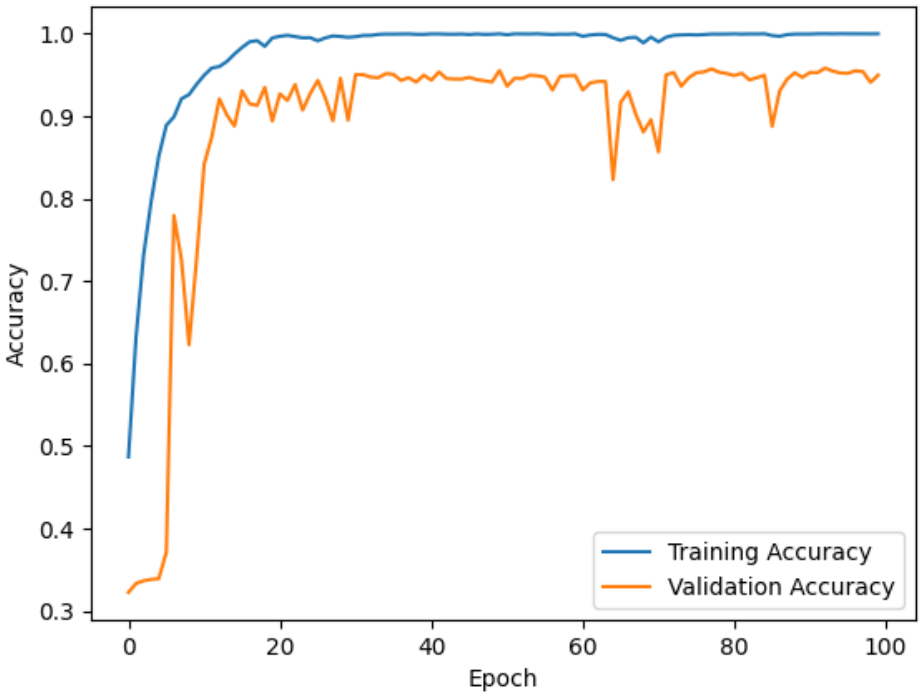


Рис. 4.9. Процес навчання на доповненій вибірці.

4.5 Висновки до Розділу 4.

В ході проведених експерементів було показано:

1. Згенеровані дані здатні «обманути» класифікатор, в якості тестових.
2. В якості навчальної вибірки ще рано подавати повністю згенерований датасет, але перспектива для цього є.
3. Метод видалення дублікатів довів свою ефективність, що підтверджено оцінками на основі розробленої моделі оцінок розподілу даних навчального набору.
4. Згенеровані дані продемонтрували себе достатньо ефективно для того, щоб користуватись нами в якості аугементації вибірок.
5. Отже, розроблена інформаційна технологія може покращити процес вирішення задачі класифікації та розпізнавання.

ЗАГАЛЬНІ ВИСНОВКИ

Дисертаційна робота є завершеною науково-дослідною роботою, в якій обґрунтовано та вирішено актуальна науково-технічна проблематика, а саме, проведено аналіз результатів вирішення задачі розпізнавання і класифікації розробленої інформаційної технології, в якій основою для покращення слугував процес передобробки даних, що виконується за допомогою розроблених методів: методу генерації даних на основі існуючих та методом видалення дублікатів, що в свою чергу є похідними від запропонованої моделі оцінки розподілу набору даних в просторі латентних представлень.

Також розроблена інформаційна технологія має функцію навчання кастомних моделей нейромереж, що показують високу точність.

Основні наукові та практичні результати дисертаційного дослідження, отримані під час роботи:

1. Проведено огляд існуючих моделей нейромережевих класифікаторів.
2. Проведено аналіз існуючих генеративних моделей, таких як VAE і GAN.
3. Проаналізовано існуючі методи зменшення розмірності багатовимірних даних. З них в подальшу роботу був взятий метод головних компонент, через взаємну ортогональність головних компонент та можливість зворотнього перетворення. Для реалізації цієї функції було обрано мережу автокодування.
4. Запропоновано модель оцінки розподілу навчального набору даних повітряної зйомки.
5. На основі запропонованої моделі оцінки та методу переходу до латентних просторів були запропоновані два методи передобробки даних: метод генерації нових зображень на основі існуючих та метод видалення дублікатів з набору даних.

На базі всього вище описаного функціоналу реалізовано інформаційну технологію розпізнавання даних аерозйомки, яка покращує або поліпшує процес вирішення задачі класифікації і розпізнавання.

На основі розробленої технології існує:

1. Можливість зменшити ймовірність перенавчання за рахунок зменшеного мінімум вдвічі коефіцієнту ексцесу розподілу навчального набору.
2. Збільшити швидкість навчання моделей мінімум в півтора рази.

І все це при можливості тренувати моделі, які показують мінімум 0,9 точності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*. Retrieved from [https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf]

2. ImageNet Large Scale Visual Recognition Challenge. (2024). Retrieved from [https://www.semanticscholar.org/reader/eb42cf88027de515750f230b23b1a057dc782108]

3. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going Deeper with Convolutions. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Retrieved from [https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Szegedy_Going_Deeper_With_2015_CVPR_paper.pdf]

4. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Retrieved from [https://openaccess.thecvf.com/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf

5. Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely Connected Convolutional Networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Retrieved from [https://openaccess.thecvf.com/content_cvpr_2017/papers/Huang_Densely_Connected_Convolutional_CVPR_2017_paper.pdf]

6. O. Cholyshkina. P. Prystavka. O. Kozachuk. V. Zivakin. Training set AERIAL SURVEY for data recognition systems from aerial surveillance cameras. IX INTERNATIONAL CONFERENCE Information Technology and Implementation. Paper 79. 1 December, 2022, Kyiv

7. Van Der Maaten, L., & Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9, 2579-2605. Retrieved from [https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf?fbcl]

8. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Retrieved from https://arxiv.org/pdf/1802.03426

9. Shalizi, C. R. (2013). Advanced Data Analysis from an Elementary Point of View. Lecture Notes. Retrieved from [https://www.stat.cmu.edu/~cshalizi/350/lectures/14/lecture-14.pdf]

10. Jolliffe, I. (2002). *Principal Component Analysis* (2nd ed.). Springer. Retrieved from [http://cda.psych.uiuc.edu/statistical_learning_course/Jolliffe%20I.%20Principal%20Component%20Analysis%20(2ed.,%20Springer,%202002)(518s)_MVsa_.pdf]

11. MRI Questions. (2013). Mathematical Foundations of Independent Component Analysis. Retrieved from [https://www.mriquestions.com/uploads/3/4/5/7/34572113/ica_math.pdf]

12. Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786), 504-507. Retrieved from [https://dbirman.github.io/learn/hierarchy/pdfs/Hinton2006.pdf]

13. Kingma, D. P., & Welling, M. (2013). Auto-Encoding Variational Bayes. Retrieved from [https://arxiv.org/pdf/1312.6114]

14. Goodfellow, I., Poulin, J., & Vincent, P. (2014). Generative Adversarial Nets. *Advances in Neural Information Processing Systems*. Retrieved from [https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf]

15. 1. McCullh W.S. and W.Pitts ,“A logical calculus of the ideas immanent in nervous activity”, Bulletin of Mathematical Biopysics, 1943, vol. 5, p. 115 – 133

16. Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.

17. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative adversarial nets. Advances in neural information processing systems, 27, 2672-2680.

18. McLachlan, G., & Peel, D. (2000). *Finite mixture models*. Wiley.

19. Wand, M. P. (1997). Data-based choice of histogram bin width. The American Statistician, 51(1), 59-64.

20. Eilers, P. H., & Marx, B. D. (1996). Flexible smoothing with B-splines and penalties. Statistical Science, 11(2), 89-121.

21. Приставка П. О. Поліноміальні сплайни при обробці даних : монографія. Д.: Вид-во Дніпропетровського університету, 2004. С. 155–164

22. Шабан Н. В. Система пошуку нечітких дублікатів зображення. Дип…магістр/КПІ Київ. 2018. 117 с

23. Singhal, A. (2001). Modern information retrieval: A brief overview. IEEE Data Engineering Bulletin, 24(4), 35-43.

24. Зівакін В. Д. Таврійський науковий вісник. Серія: Технічні науки: Імітація одновимірних вибірок на основі існуючих з використанням поліноміальних сплайнів. Херсонський державний аграрно- економічний університет. Херсон : Видавничий дім «Гельветика», 2021. Вип. 6. С. 23-30

25. Зівакін В. Д. «Імітація одновимірних вибірок методом зворотньої функції з використанням поліноміальних сплайнів». Конференція «Політ. Сучасні проблеми науки». (4-7 квітня) Секція «Сучасні інформаційні та комунікаційні технології в авіації. Прикладна математика» 2023. – С. 7-8

26. Зівакін В. Д., Приставка П. О. Дослідження імітації двовимірних вибірок з використанням поліноміальних сплайнів. Проблеми інформатизації та управління. Національний авіаційний університет. Київ, 2023. Том 2 Вип. 74. С. 38-43.

27. Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: A review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 374*(2065), 20150202. https://doi.org/10.1098/rsta.2015.0202

28. Shlens, J. (2014). A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*. https://arxiv.org/abs/1404.1100

29. Cherapanamjeri, Y., Krishnamurthy, A., Jain, P., & Netrapalli, P. (2021). Robust principal component analysis: A median of means approach. *arXiv preprint arXiv:2102.03403*. https://arxiv.org/abs/2102.03403

30. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436-444. https://doi.org/10.1038/nature14539

31. Rawat, W., & Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation, 29*(9), 2352-2449. https://doi.org/10.1162/neco_a_00990

32. Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., ... & Chen, T. (2018). Recent advances in convolutional neural networks. *Pattern Recognition, 77*, 354-377. https://doi.org/10.1016/j.patcog.2017.10.013

33. Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science, 313*(5786), 504-507. https://doi.org/10.1126/science.1127647

34. Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning* (pp. 1096-1103). https://doi.org/10.1145/1390156.1390294

35. Masci, J., Meier, U., Cireşan, D., & Schmidhuber, J. (2011). Stacked convolutional auto-encoders for hierarchical feature extraction. In *International Conference on Artificial Neural Networks* (pp. 52-59). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-21735-7_7

36. Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 39*(12), 2481-2495. https://doi.org/10.1109/TPAMI.2016.2644615

37. Mao, X., Shen, C., & Yang, Y.-B. (2016). Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In *Advances in Neural Information Processing Systems* (pp. 2802-2810). https://arxiv.org/abs/1603.09056

38. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems* (pp. 1097-1105). https://doi.org/10.1145/3065386

39. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. https://arxiv.org/abs/1409.1556

40. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9). https://doi.org/10.1109/CVPR.2015.7298594

41. Hyvärinen, A., & Oja, E. (2000). Independent component analysis: Algorithms and applications. *Neural Networks, 13*(4-5), 411-430. https://doi.org/10.1016/S0893-6080(00)00026-5

42. Comon, P. (1994). Independent component analysis, a new concept? *Signal Processing, 36*(3), 287-314. https://doi.org/10.1016/0165-1684(94)90029-9

43. Bell, A. J., & Sejnowski, T. J. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural Computation, 7*(6), 1129-1159. https://doi.org/10.1162/neco.1995.7.6.1129

44. Faisal, R., Kitasuka, T., & Aritsugi, M. (2012). Semantic cosine similarity. *The 7th International Student Conference on Advanced Science and Technology ICAST*, 4(1), 1-10.

45. Maher, K., & Joshi, M. S. (2016). Effectiveness of different similarity measures for text classification and clustering. *International Journal of Computer Science and Information Technology*, 7(4), 1715-1720.

46. Zhang, Y., Jin, R., & Zhou, Z. H. (2010). Understanding bag-of-words model: A statistical framework. *International Journal of Machine Learning and Cybernetics*, 1, 43-52.

47. Wang, P., et al. (2016). Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification. *Neurocomputing*, 174, 806-814.

48. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

49. Silverman, B. W. (1986). Density estimation for statistics and data analysis. *Chapman and Hall*.

50. Scott, D. W. (2015). Multivariate density estimation: Theory, practice, and visualization. *John Wiley & Sons*.

51. Wand, M. P., & Jones, M. C. (1994). Kernel smoothing. *Chapman and Hall/CRC*.

52. Sheather, S. J. (2004). Density estimation. *Statistical Science*, 19(4), 588-597.

53. Terrell, G. R., & Scott, D. W. (1992). Variable kernel density estimation. *Annals of Statistics*, 1236-1265.

54. Goodfellow, I., et al. (2014). Generative adversarial nets. *Advances in Neural Information Processing Systems*, 2672-2680.

55. Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

56. Karras, T., Laine, S., & Aila, T. (2019). A style-based generator architecture for generative adversarial networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4401-4410.

57. Brock, A., Donahue, J., & Simonyan, K. (2018). Large scale GAN training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*.

62. Jain, A. K., Duin, R. P., & Mao, J. (2000). Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1), 4-37.

63. Bay, H., Tuytelaars, T., & Van Gool, L. (2006). SURF: Speeded up robust features. *European Conference on Computer Vision*, 404-417.

64. Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2), 91-110.

65. Radenović, F., Tolias, G., & Chum, O. (2018). Fine-tuning CNN image retrieval with no human annotation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(7), 1655-1668.

66. Zheng, L., et al. (2015). Scalable person re-identification: A benchmark. *Proceedings of the IEEE International Conference on Computer Vision*, 1116-1124.