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

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформаційних систем та технологій**

До захисту допущено:

Завідувач кафедри

_____Олександр РОЛІК

«__»_____2023р.

# Дипломний проєкт

**на здобуття ступеня бакалавра**
**за освітньо-професійною програмою**
**«Інформаційне забезпечення робототехнічних систем»»**
**спеціальності 126 «Інформаційні системи та технології»**
**на тему: «Згорткова нейронна мережа для пошуку аномалій на зображенні»**

Виконав:
студент IV курсу, групи ІК-93
Рябушко Максим Анатолійович _____

Керівник:
Доцент, к.т.н, доц.,
Батрак Євгеній Олександрович _____

Рецензент:
Доцент, к.т.н., доц.,
Волокита Артем Миколайович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2023 року

**Національний технічний університет України**
**«Київський політехнічний інститут імені Ігоря Сікорського»**
**Факультет інформатики та обчислювальної техніки**
**Кафедра інформаційних систем та технологій**

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 126 «Інформаційні системи та технології»
Освітньо-професійна програма «Інформаційне забезпечення робототехнічних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2023р.

**ЗАВДАННЯ**
**на дипломний проєкт студенту**
**Рябушко Максиму Анатолійовичу**

1. Тема проєкту «Згорткована нейронна мережа для пошуку аномалій на зображенні», керівник проєкту Батрак Євгеній Олександрович, доцент, к.т.н, затверджені наказом по університету від «31» травня 2023 р. № 2101-с

2. Термін подання студентом проєкту: 12 червня 2023 року

3. Вихідні дані до проєкту: нейромережа для пошуку аномалій на зображенні. Keras, Tesnorflow, Deep Learning model Autoencoder. It is made decoder and encoder, MNIST dataset, Loss function MSE.

4. Зміст пояснювальної записки Encoder and Decoder. Розуміння автокодерів, побудова кодера та декодера. Вивчення методів виявлення аномалій тестування трьох різних методів автокодування, пошук аномалій у зображеннях набору MNIST.

5. Перелік графічного матеріалу демонстрація зображень набору MNIST Візуальне представлення рукописних цифр з набору даних MNIST. Діаграма автокодера, що ілюструє структуру автокодера, включає кодер і декодер.

6. Дата видачі завдання 1 грудня 2022 року

Календарний план

| № з/п | Назва етапів виконання проєкту | Термін виконання етапів проєкту | Примітка |
|---|---|---|---|
| 1. | Дослідження актуальних рішень для знаходження аномалій | 01.05.2023 | |
| 2. | Імплентація гіпотез | 04.05.2023 | |
| 3. | Тестування гіпотез | 05.05.2023 | |
| 4. | Інтеграція кращої гіпотези | 07.05.2023 | |
| 5. | Розробка структури прототипу | 12.05.2023 | |
| 6. | Налагодження та перевірка програми | 20.05.2023 | |
| 7. | Оформлення пояснювальної записки | 28.05.2023 | |
| 8. | Захист | 19.06.2023 | |

Студент Максим РЯБУШКО

Керівник Євгеній БАТРАК

# АНОТАЦІЯ

Рябушко М.А. Згорткова нейронна мережа для пошуку аномалій на зображенні. Робота присвячена створення різноманітних автоенкодерів у keras.

Метою роботи є створення ефективної моделі, яка здатна автоматично виявляти аномальні об’єкти або відхилення від норми на зображеннях. Роботапочинається зі збору і попередньої обробки набору зображень, включаючи як нормальні, так і аномальні зразки. Після цього будується архітектурі знм, що складається з декількох шарів згортки та пулінгу, які допомагають виявити локальні особливості зображень. Далі використовуються повнозв’язні шари для класифікації зображень на аномальні та нормальні.

Основна увага приділяється процесу навчанням згорткової нейронної мережі з використанням алгоритмів оптимізації, таких як зворотне поширення помилки. Для досягнення кращої ефективності ефективності моделі розглядають різні архітектурні параметри. Кодер приймає вхідні дані та стискає їх у представлення латентного простору. Потім декодер намагається реконструювати вхідні дані з прихованого простору.

Крім цього розглядалось використання автокодерів для зашумлення зображень за допомогою keras, tensorflow і deep learning. Автокодери можна використовувати для для денойзингу, його можна назвати”шумозаглушенням”, тобто процесом видалення шуму з сигналу.

Ключові слова: автокодер, Keras, TensorFlow і Deep Learning, аномалії, кодер, декодер, шум.

Розмір пояснювальної записки 61 аркушів, 31 ілюстрацій та 4 додатки.

# ANNOTATION

Riabushko M. A. A convolution neural network for image anomaly detection.

The work is devoted to the creation of various Autoencoders in Keras. The aim of the work is to create an effective model capable of automatically dececting anomalous objects or deviations from the norm in images. The work begins with the collection and preprocessing of a set of images, including both normal and abnormal samples. After that, the architecture of convolution neural network is built, consisting of several layers, of convolution and pooling, which help to detect local features of images. Next, fully connected layers are used to classify the images into abnormal and normal.

The focus is on the process of training a convolutional neural network using optimization algorithms such as error backpropagation. To achieve better effictiency, the model considers various architectural parametres. An encoder takes input data and compresses it into a latent space representation. The decoder then tries to reconstruct the input data from the hidden space.

In addition, the use of autoencoders for image noise using Keras, TensorFlow, And Deep Learning was considered. Autoencoders can be used for denoising, it can be called “noise suppression”, that is, the process of removing oise from asignal.

Keywords: autoencoder, Keras, TensorFlow and Deep Learning, anomalies, encoder, decoder, noise.

The size of the explanatory note is 61 sheets, 31 illustrations and 4 appendic.

| Номер рядка | Формат | Позначення | Найменування | Кільк. аркушів | Номер екзем. | Примітка |
|---|---|---|---|---|---|---|
| 1 | | | Документація загальна | | | |
| 2 | | | | | | |
| 3 | | | Знову розроблена | | | |
| 4 | | | | | | |
| 5 | А4 | ІК93.220БАК.005 ПЗ | Пояснювальна записка | 61 | | |
| 6 | А3 | ІК93.220БАК.005Д1 | Згорткова нейронна мережа | 1 | | |
| 7 | | | для пошуку аномалій для | | | |
| 8 | | | на зображенні. | | | |
| 9 | | | Діаграма кодера | | | |
| 10 | А3 | ІК93.220БАК.005 Д2 | Згорткова нейронна мережа | 1 | | |
| 11 | | | для пошуку аномалій для | | | |
| 12 | | | на зображенні. | | | |
| 13 | | | Декодер діаграма | | | |
| 14 | А3 | ІК93.220БАК.005 Д3 | Згорткова нейронна мережа | 1 | | |
| 15 | | | для пошуку аномалій для | | | |
| 16 | | | на зображенні. | | | |
| 17 | | | Діаграма аугментації | | | |
| 18 | А3 | ІК93.220БАК.005 Д4 | Згорткова нейронна мережа | 1 | | |
| 19 | | | для пошуку аномалій для | | | |
| 20 | | | на зображенні. | | | |
| 21 | | | Діаграма звичайної згортки | | | |
| 22 | | | | | | |
| 23 | | | | | | |
| 24 | | | | | | |
| 25 | | | | | | |
| 26 | | | | | | |
| 27 | | | | | | |
| 28 | | | | | | |

ІК93.220БАК.005 ТП

| | | | | | |
|---|---|---|---|---|---|
| Розроб. | Рябушко М.А. | | | Літ. | Аркуш | Аркушів |
| Керівн. | Батрак Є.О.. | | | | 1 | 1 |
| | | | | КПІ ім. Ігоря Сікорського Група ІК-93 | | |
| | | | | | | |
| Затв. | Ролік О.І. | | | | | |

Згорткова нейронна мережа для пошуку аномалій на зображенні. Відомість проєкту

# Пояснювальна записка
## до дипломного проєкту
## на тему: «Згорткова нейронна мережа для пошуку аномалій на зображенні»

Київ – 2023 року

ЗМІСТ

# ВСТУП

Автокодер є одним з найпоширеніших архітектур глибокого навчання, який використовується для безлічі завдань, таких як компресія даних, зменшення розмірності, покращення якості зображень і відновлення пропущених або пошкоджених даних. Автокодери є формою нейронних мереж, які вчаться відтворювати вхідні дані на виході, пропускаючи їх через прихований шар з меншою кількістю нейронів.

Виявлення аномалій є важливим завданням в аналізі даних, і глибоке навчання може бути потужним інструментом для цього. Використання Keras, TensorFlow і глибокого навчання може сприяти ефективному виявленню аномалій в даних. Keras високорівнева бібліотека глибокого навчання, яка дозволяє швидкоі зручно побудувати та навчити різні нейронні мережі, включаючи автокодери. TensorFlow є відкритою бібліотекою машинного навчання, що використовується для побудови глибоких нейронних мереж. Він надає різноманітні інструменти для розробки і навчання моделей глибокого навчання, включаючи роботу з даними, визначення архітектури мережі та тренування моделей. Глибоке навчання, зокрема нейронні мережі, може бути використано для виявлення аномалій в даних. Зазвичай для цього використовуються автокодери, які навчаються відтворювати вхідні дані і мають здатність виявляти аномалії шляхом порівняння відтворених даних з оригіналами.У створенні моделі виявлення аномалій за допомогою Keras і TensorFlow можна скористатися різними типами шарів та архітектурою мережі. Зазвичай автокодери мають кодер, декодер та підсумковий шар. Кодер складається з послідовності шарів, які зменшують розмірність даних. Декодер складається з послідовності шарів, які відтворюють дані на виході. Після навчання автокодера можна використовувати його для виявлен. Усунення шуму є одним з застосувань автокодерів в глибокому навчанні. Основна мета автокодування полягає в тому, щоб код був ефективним представленням вхідних даних. зменшення шуму в даних, відбору найбільш інформативних ознак або збіжності даних. У Keras можна побудувати автокодери з використанням цих архітектур і налаштувати їх для відповідних завдань автокодування.

# 1 СТВОРЕННЯ АВТОКОДЕРІВ У KERAS

Автокодери є популярною архітектурою нейронних мереж для безпосереднього вивчення представлення даних та використання його для відтворення вхідних даних. Вони використовуються для зменшення розмірності даних, зображень, утворення нових зображень і використання їх для генерації нових даних. На рисунку 1.1 можна побачити як виглядає автокодувальник.

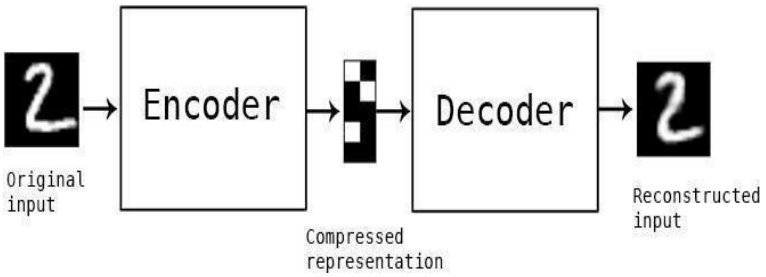


Рисунок.1.1 – Автокодувальник схема

Одним із різновидів штучних нейронних мереж, які використовуються для вивчення ефективних кодувань немаркованого введення, є автокодер. Відтворення вхідних даних із цього кодування дозволяє тестувати та вдосконалювати кодування. Автокодер навчає мережу ігнорувати сторонні дані, щоб навчитися подання для колекції даних, як правило, для зменшення розмірності.

Отримання цінних характеристик за допомогою навчених представлень може бути примусовим. Приклади включають варіаційні автокодери, які використовуються як генеративні моделі, і регуляризовані автокодери (Sparse, Denoising і Contractive відповідно). Регуляризовані автокодери корисні для навчання представлень для подальших завдань класифікації.

Вибір конкретного методу залежить від типу шуму, який потрібно усунути, та від даних, з якими працює автокодер. Експериментування з різними підходами та архітектурами може бути корисним для досягнення найкращих результатів у конкретній задачі. Основна мета автокодування полягає в тому, щоб код був ефективним представленням вхідних даних. Автокодери можуть бути побудовані за допомогою різних архітектур нейронних мереж, автокодери навчаються автоматично з прикладів даних, що є корисною властивістю.

## 1.1 Автокодування

Метод машинного навчання, який використовується для зменшення розмірності даних та вилучення їх характеристик. Воно ґрунтується на побудові та навчанні автокодерів (autoencoders). Автокодер - це нейронна мережа, яка навчається реконструювати вхідні дані якомога точніше. Вона складається з двох основних компонентів: енкодера (encoder) і декодера (decoder). Енкодер перетворює вхідні дані на меншорозмірний представлення, відоме як код (зазвичай вектор з меншою кількістю компонентів), а декодер намагається відтворити вхідні дані з цього коду. Процес навчання автокодера полягає в мінімізації різниці між вхідними даними та їх відтворенням, що забезпечує створення компактного подання даних. У процесі навчання автокодера можна використовувати різні оптимізаційні методи, такі як стохастичний градієнтний спуск (stochastic gradient descent), для знаходження оптимальних параметрів мережі. Автокодування має широкий спектр застосувань, включаючи компресію даних, вилучення ознак, покращення якості зображень, генерацію контенту тощо. Додатково, використовуючи набір даних, навчений автокодер може виконуватиреконструкцію або генерацію нових екземплярів, що розширює його застосування в області генеративного моделювання[1].

Метою автокодування є навчання моделі створювати ефективне представлення (код) вхідних даних, яке може бути використане для відновлення вихідних даних або виконання інших завдань, таких як зменшення шуму або виділення важливих ознак.

Вибір конкретного методу залежить від типу шуму, який потрібно усунути, та від даних, з якими працює автокодер. Експериментування з різними підходами та архітектурами може бути корисним для досягнення найкращих результатів у конкретній задачі. Автокодування - це процес зменшення розмірності даних шляхом створення компактного коду (представлення) для вхідних даних. Автокодування використовується для відбору найбільш інформативних ознак у вхідних даних і знаходження їх низькорозмірного представлення.

Автокодери є типом нейронних мереж, що використовуються для автокодування. Вони складаються з двох основних частин: енкодеру і декодеру. Енкодер перетворює вхідні дані на компактне представлення, яке може бути названо "кодом" або "простором ознак". Декодер відновлює дані з цього компактного представлення. Основна мета автокодування полягає в тому, щоб код був ефективним представленням вхідних даних. Значення коду повинно бути достатньою для відновлення вихідних даних, але при цьому воно має бути меншою розмірністю, щоб забезпечити зменшення розмірності даних. Це дозволяє використовувати автокодери для зменшення шуму в даних, відбору найбільш інформативних ознак або збіжності даних. Автокодери можуть бути побудовані за допомогою різних архітектур нейронних мереж, таких як звичайні перцептрони (feedforward neural networks), згорткові нейронні мережі (convolutional neural networks) або рекурентні нейронні мережі (recurrent neural networks). У Keras можна побудувати автокодери з використанням цих архітектур і налаштувати їх для відповідних завдань автокодування.

Автокодування з лінійною активацією (Linear Autoencoder): У цьому типі автокодування енкодер і декодер представлені лінійними шари. Така модель використовується для зменшення розмірності даних. Автокодування з використанням нейронних мереж (Neural Network Autoencoder). У цьому випадку енкодер і декодер представлені нейронними мережами.

## 1.2 Чи добре стискають дані автокодери

Автокодувальники залежать від даних, а це означає, що вони зможуть стискати лише ті дані, на яких їх навчали. Це відрізняється від, скажімо, алгоритму стиснення MPEG-2 Audio Layer III (MP3), який містить лише припущення щодо «звуку» загалом, але не щодо конкретних типів звуків. Автокодер, навчений на зображеннях облич, виконував би досить погану роботу зі стисненням зображень дерев, оскільки особливості, які він дізнавався, були б специфічними для обличчя. Автокодувальники мають втрату, що означає, що декомпресовані виходи будуть погіршені порівняно з вихідними вхідними (подібно до стиснення MP3 або JPEG).

Це відрізняється від арифметичного стиснення без втрат. Автокодери навчаються автоматично з прикладів даних, що є корисною властивістю означає, що легко навчити спеціалізовані екземпляри алгоритму, який добре працюватиме з певним типом введення. Для цього не потрібні нові інженерні розробки, лише відповідні навчальні дані[2].

## 1.3 Будова автокодер

Щоб побудувати автокодер, вам потрібні три речі: функція кодування, функція декодування та функція відстані між кількістю втрати інформації між стислим представленням ваших даних і розпакованим представленням (тобто функція «втрати»). Кодер і декодер буде обрано параметричними функціями (як правило, нейронними мережами) і диференційованими щодо функції відстані, тому параметри функцій кодування/декодування можна оптимізувати для мінімізації втрат при реконструкції за допомогою стохастичного градієнтного спуску. Навіть не потрібно розуміти жодне з цих слів, щоб почати використовувати автокодери на практиці.

Вони рідко використовуються на практиці. У 2012 році вони ненадовго знайшли застосування в жадібному пошаровому попередньому навчанні для глибоких згорткових нейронних мереж, але це швидко вийшло з моди, коли ми почали розуміти, що кращих схем ініціалізації випадкової ваги достатньо для навчання глибоких мереж з нуля.

У 2014 році пакетна нормалізація почала дозволяти ще більш глибокі мережі, а з кінця 2015 року ми могли навчати довільно глибокі мережі з нуля, використовуючи залишкове навчання. Найпростіший автокодувальник має лише два шари: шар кодування і шар декодування.

На сьогоднішній день два цікавих практичних застосування автокодерів це усунення шумів у даних (про що ми розповідаємо далі в цій публікації) і зменшення розмірності для візуалізації даних. З відповідними обмеженнями розмірності та розрідженості автокодери можуть вивчати проекції даних, які є більш цікавими, ніж PCA або інші базові методи.

Зокрема, для двовимірної візуалізації t-SNE (вимовляється як «ті-сні»), мабуть, є найкращим алгоритмом, але зазвичай для нього потрібні відносно малорозмірні дані. Отже, хороша стратегія для візуалізації зв'язків подібності у високовимірних даних полягає в тому, щоб почати з використання автокодувальника для стиснення ваших даних у низьковимірний простір (наприклад, 32-вимірний), а потім використовувати t-SNE для відображення стислих даних у 2D. літак. Зауважте, що гарна параметрична реалізація t-SNE у Keras була розроблена Кайлом Макдональдом і доступна на Github. В іншому випадку scikit-learn також має просту та практичну реалізацію.

Їхня головна претензія на популярність походить від того, що вони представлені на багатьох вступних курсах машинного навчання, доступних онлайн. Як наслідок, багато новачків у цій галузі дуже люблять автокодери й не можуть ними натішитися.

В іншому випадку одна з причин, чому вони привернули стільки досліджень і уваги, полягає в тому, що вони довгий час вважалися потенційним шляхом для вирішення проблеми неконтрольованого навчання, тобто вивчення корисних уявлень без потреби в ярликах[3].

Знову ж таки, автокодери не є справжньою технікою неконтрольованого навчання (що означало б зовсім інший процес навчання), вони є самоконтрольованою технікою, конкретним прикладом контрольованого навчання, де цілі генеруються з вхідних даних. Для того, щоб змусити самоконтрольовані моделі вивчати цікаві функції, ви повинні придумати цікаву синтетичну ціль і функцію втрат, і саме тут виникають проблеми просто навчитися реконструювати введені дані в найдрібніших деталях може бути не правильним вибором.

У самоконтрольованому навчанні, застосованому до зору, потенційно плідною альтернативою реконструкції вхідних даних у стилі автокодувальника є використання іграшкових завдань, таких як розгадування головоломки або узгодження деталей із контекстом (можливість зіставляти зображення з високою роздільною здатністю, але невеликі фрагменти зображень із версії зображень із

низькою роздільною здатністю, з яких вони витягнуті).

Наступна стаття досліджує розв’язування головоломок і є дуже цікавою для читання: Noroozi and Favaro (2016) Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles. Такі завдання забезпечують модель вбудованими припущеннями щодо вхідних даних, які відсутні в традиційних автокодерах, наприклад «візуальна макроструктура має більше значення, ніж деталі на рівні пікселів».

### 1.4 Найпростіший автокодувальник

Модель нейромережі, яка навчається стиснути вхідні дані до меншого представлення і потім реконструювати їх назад до початкового вигляду. Основна ідея полягає в тому, що модель навчається відтворювати свої вхідні дані, змушуючи прихований шар відображати інформацію про вхідні дані. Найпростіший автокодувальник має лише два шари, шар кодування і шар декодування. Вхідні дані подаються на шар кодування, де вони стискаються до меншого представлення, і потім розкодовуються на шарі декодування для отримання реконструкції початкових даних. Модель декодера може бути навчена за допомогою переважної навчання з використанням метрики, такої як перехресна ентропія (cross-entropy), яка вимірює відповідність між згенерованимі справжнім вихідним текстом.

### 1.5 Модель декодера

Ключовою складовою архітектури сучасних послідовних генеративних моделей,таких як Transformer. Її основна мета полягає в генерації тексту, що відповідає заданому вхідному контексту або послідовності. Модель декодера використовується після проходження вхідного тексту через енкодер (який може бути також заснований на Transformer) і отримання внутрішнього представлення цього тексту. Вона складається з декодувальних шарів, які взаємодіють між собою, щоб послідовно генерувати вихідний текст. Основним компонентом

моделі декодера є механізм виявлення уваги (attention mechanism). Цей механізм дозволяє моделі фокусуватися на різних частинах вхідного тексту при генерації вихідного. Він забезпечує здатність моделі враховувати важливість різних слів або фраз у вхідному контексті під час генерації тексту. Крім того, модельдекодера використовує самостійність (autoregression), що означає, що вона генерує вихідний текст по одному токену за раз. На кожному кроці декодуваннямодель використовує внутрішній стан і контекст інформації, щоб виробитиймовірності для наступного токену. Цей процес повторюється, доки не буде згенерований повний вихідний текст. Модель декодера може бути навчена за допомогою переважної навчання з використанням метрики, такої як перехреснаентропія (cross-entropy), яка вимірює відповідність між згенерованим і справжнім вихідним текстом.

Наведена діаграма кодера (кресленик ІК93.220БАК.005 Д1).

– Вхідне зображення;

– 2 Пробіжки, фільтри 32xconv, потім фільтри 64xconv(в якій згортка 2d 3x3,Функція повтиорної активації, нормалізація партії);

– Сплющування щільний шар;

– Щільний шар;

– Презентація латентного простору.

Ось що ми отримуємо. Верхній рядок оригінальні цифри, а нижній реконструйовані цифри. Завдяки цьому базовому підходу ми втрачаємо досить багато деталей. Можна побачити вигляд Найпростішого атокодувальника на рисунку 1.2.

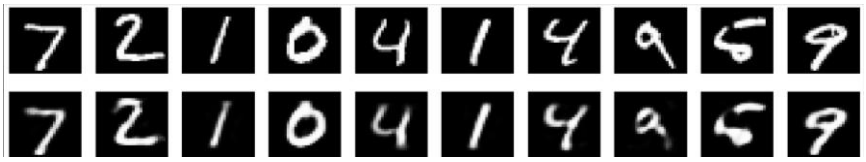

Рисунок 1.2 – Найпростіший можливий автокодувальник

Додавання обмеження розрідженості до закодованих представлень

У попередньому прикладі подання були обмежені лише розміром прихованого шару. У такій ситуації зазвичай відбувається те, що прихований рівень вивчає апроксимацію PCA (аналіз основних компонентів).

Давайте навчимо цю модель протягом 100 епох (з доданою регуляризацією модель має меншу ймовірність перевиконання, і її можна навчати довше). Моделі закінчуються втратою поїзда 0,11 і втратою тесту 0,10. Різниця між цими двома здебільшого пов’язана з регуляризаційним терміном, який додається до втрати під час навчання (вартість приблизно 0,01). Візуалізація наших нових результатів можна побачити на рисунку 1.3

Рисунок 1.3 – Глибокий атокодер

Глибокий автокодер є варіацією стандартного автокодера, в якому використовуються багатошарові нейронні мережі для енкодера і декодера. Глибокий автокодер має кілька прихованих шарів, що дозволяє йому вчитись більш складним та абстрактним представленням даних.

Згорткові шари забезпечують інваріантність до зміщень та масштабу, тобто вони можуть розпізнавати об'єкти, незалежно від їх положення в зображенні або їх розміру. Зменшення розмірності даних. Згорткові шари можуть виконувати зменшеннярозмірності даних шляхом застосування піддискретизації або згортки з великим кроком (strided convolution). Згорткові шари можуть виконувати зменшеннярозмірності даних шляхом застосування піддискретизації або згортки з великим кроком (strided convolution). Це дозволяє створювати більш компактне представлення даних, зберігаючи важливу інформацію. Після 100 епох глибокий автоенкодер досягає втрати під час перевірки ~0,08, що трохи краще, ніж наші попередні моделі. Наші реконструйовані цифри також виглядають трохи краще на рисунку 1.4.

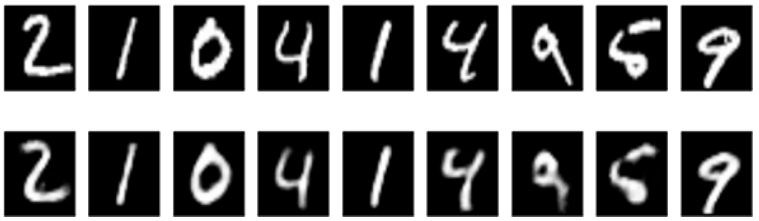

Рисунок 1.4 – Цифри глибокого автокодера

Автокодери можуть бути використані для реконструкції цифр або будь-яких інших зображень. Для цього модель автокодера навчається на вхідних даних, наприклад, наборі зображень цифр. Після тренування, автокодер може бути використаний для відтворення цих зображень на виході. Процес реконструкції полягає в тому, що вхідне зображення проходить через енкодер, який перетворює його в компактне представлення або "код". Потім цей код передається в декодер, який відтворює зображення з цього коду. Ідея полягає в тому, що декодер навчений генерувати реконструкції, якомога більш схожі на вхідні дані. Отримана реконструкція може бути порівняна з оригінальним зображенням для оцінки точності відтворення. Якщо автокодер правильно навчився, то реконструкція буде схожою на оригінал, і це може бути використано для перевірки якості моделі[4].

Оскільки нашими вхідними даними є зображення, має сенс використовувати згорточні нейронні мережі (convnets) як кодери та декодери. У практичних налаштуваннях автокодери, застосовані до зображень, завжди є згортковими автокодерами – вони просто працюють набагато краще.

Наведена діаграма декодера (кресленик ІК93.220БАК.005 Д2).

– Репрезентація латентного простору;

– щільний шар;

– 2 Пробіжки, фільтри 32xconv, потім фільтри 64xconv(в якій згортка 2d 3x3, Функція повтиорної активації, нормалізація партії);

– Транспортування згортки 2D(відновлення початкової глибини зображення);

– Активація сигмоіта;

– Вихід реконструйоване зображення.

Згортковий автокодер є варіацією автокодера, який використовує згорткові шари (convolutional layers) для енкодера та декодера. Згорткові шари дозволяють автоматично вилучати просторові ознаки зображень та здійснювати зменшення розмірності даних. Збереження просторової інформації. Згорткові шариврахову ють локальні залежності та просторову структуру даних, що особливо корисно для обробки зображень. Вони можуть вилучати різні рівні функцій, такі як контури, текстури та об'єкти. Інваріантність до зміщень та масштабу. Згорткові шари забезпечують інваріантність до зміщень та масштабу, тобто вони можуть розпізнавати об'єкти, незалежно від їх положення в зображенні або їх розміру. Зменшення розмірності даних. Згорткові шари можуть виконувати зменшення розмірності даних шляхом застосування піддискретизації або згортки з великим кроком (strided convolution). Це дозволяє створювати більш компактне представлення даних, зберігаючи важливу інформацію. Згорткові шари можуть автоматично вилучати важливі ознаки зображень підчас навчання автокодера. Це допомагає уникнути необхідності вручну задавати ознаки для вилучення. Згорткові шари допомагають зберігати локальні залежності в даних, що особливо корисно для відновлення зображень

Тоді давайте навчимо нашу модель. У списку зворотних викликів ми передаємо екземпляр зворотного виклику TensorBoard. Після кожної епохи цей зворотній виклик записуватиме журнали в /tmp/autoencoder, які може читати наш сервер TensorBoard. У контексті реконструкції зображень або фотографій, цифри можуть відтворюватися або реставруватися з використанням комп'ютерного зору або алгоритмів машинного навчання. Це може бути корисним для відновлення пошкоджених або стиснених зображень, відновлення деталей чи видалення шуму. У сфері культурної спадщинареконструкція цифр може стосуватися відтворення чисел або символів з археологічних знахідок, старовинних рукописів або історичних матеріалів. Використовуючи сучасні технології, дослідники можуть аналізувати, відновлювати та інтерпретувати цифри, які були втрачені або пошкоджені з часом. Розмивання шум, яке зменшує чіткість або деталі зображення.

Модель зворотних викликів екземпляр TensorBoard дозволяє контролювати гавчання у веб-інтерфейсі зображено на рисунку 1.5

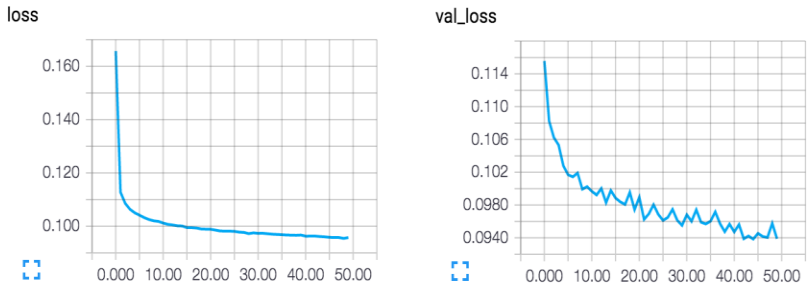


Рисунок 1.5 – Веб-інтерфейс TensorBoard

Реконструйовані цифри означають числа, що були відновлені або відтворені з певних даних, інформації чи спостережень. Це може відбуватися в різних контекстах і за допомогою різних методів. У наукових дослідженнях і аналітичних задачах реконструкція цифр може означати процес відновлення числових даних або параметрів на основі обробки вхідної інформації. Наприклад, у галузі обробки сигналів можна використовувати методи реконструкції сигналу для відновлення його форми чи властивостей із зазначених обмежених даних. У контексті реконструкції зображень або фотографій, цифри можуть відтворюватися або реставруватися з використанням комп'ютерного зору або алгоритмів машинного навчання. Це може бути корисним для відновлення пошкоджених або стиснених зображень, відновлення деталей чи видалення шуму. У сфері культурної спадщини реконструкція цифр може стосуватися відтворення чисел або символів з археологічних знахідок, старовинних рукописів або історичних матеріалів. Використовуючи сучасні технології, дослідники можуть аналізувати, відновлювати та інтерпретувати цифри, які були втрачені або пошкоджені з часом. Таким чином, реконструйовані цифри є результатом процесу відновлення числової інформації з вихідних даних або зображень за допомогою різних методів і технологій. На рисунку 1.6 зображено реконструйовані цифризгорткового автоенкодера. Деформації шум, який змінює форму або геометрію цифр. Наприклад, це можуть бути вигини, розтягнення або зсуви зображень.

Рисунок 1.6 – Реконструйовані цифри

Застосування автокодерів для реконструкції цифр має практичне значення, зокрема в задачах впізнавання рукописних цифр або виявлення аномалій у вхідних даних. Вони можуть також використовуватись для генерації нових цифр, подібних до навчального набору даних[4].

Ми також можемо поглянути на 128-вимірні закодовані представлення. Ці представлення мають розмір 8x4x4, тому ми змінюємо їх на 4x32, щоб мати можливість відображати їх як зображення у відтінках сірого рисунок 1.7

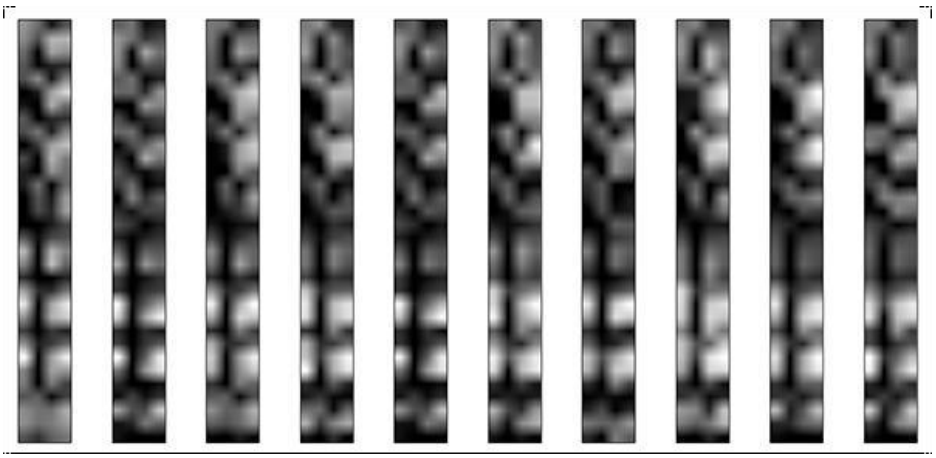

Рисунок 1.7 – 128 вимірні закодовані представлення

Давайте запустимо наш згортковий автокодер для вирішення проблеми зменшення шуму зображення.

128-вимірні закодовані представлення означають, що дані подаються у вигляді векторів розмірністю 128. Це можуть бути закодовані представлення об'єктів, зображень, тексту або будь-яких інших даних. Закодовані представлення зазвичай використовуються для виразного представлення даних у більш компактному та абстрактному вигляді, який містить важливу інформацію про дані. Це може бути досягнуто за допомогою різних моделей машинного навчання, таких як автокодери, глибокі нейронні мережі або моделі енкодер-декодер.

Якщо ви примружитесь, ви все одно зможете впізнати цифри а саме шумні цифри, але ледве на рисунку 1.8

Рисунок 1.8 – Зменшення шуму

Шумні цифри - це цифрові зображення цифр, до яких додано деякий рівень шуму або спотворень. Шум може бути розуміння як небажані артефакти або зміни, що впливають на якість або чіткість зображення. Додавання шуму до цифр може бути корисним для деяких завдань, таких як аналіз стійкості моделей до шуму або покращення навчання моделей розпізнавання цифр в умовах шуму або спотворень.

Адитивний шум, який додається безпосередньо до значень пікселів зображення. Це може бути, наприклад, шум Гауса або шум соль і перець.

Шум перекриття, при якому окремі пікселі зображення замінюються іншими значеннями або кольорами. Це може симулювати помилки при передачі даних або пошкодження зображення.

Розмивання шум, яке зменшує чіткість або деталі зображення. Це може бути спричинене рухом або іншими факторами, що призводять дорозмивання зображення. Деформації шум, який змінює форму або геометрію цифр. Наприклад, це можуть бути вигини, розтягнення або зсуви зображень. Додаванняшуму до цифр дозволяє використовувати моделі, які більш стійкі до змін та мають здатність розпізнавати та узагальнювати на шумні дані. Такі моделі можуть бути корисними в реальних умовах, де зображення можуть бути спотворені або містити шум [5]. Зверху цифри з шумом подаються в мережу, а знизу цифри реконструюються мережею рисунок 1.9

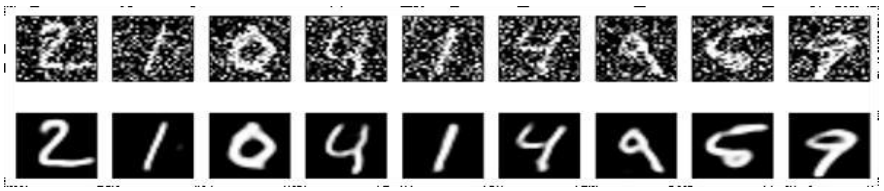

Рисунок 1.9 – Цифри з шумом та реконструйовані

Здається, це працює досить добре. Якщо ви масштабуєте цей процес до більшої мережі, ви можете розпочати створення моделей усунення шумів у документах або аудіо. У Kaggle є цікавий набір даних, який допоможе вам почати.

Автокодер послідовності в послідовність. Якщо вхідними даними є послідовності, а не вектори чи 2D-зображення, тоді ви можете використовувати як кодер і декодер тип моделі, яка може захоплювати часову структуру, наприклад LSTM. Щоб побудувати автокодер на основі LSTM, спочатку використовуйте кодер LSTM, щоб перетворити вхідні послідовності в єдиний вектор, який містить інформацію про всю послідовність, а потім повторіть цей вектор n разів (де n кількість кроків у вихідній послідовності), і запустіть декодер LSTM, щоб перетворити цю постійну послідовність у цільову послідовність.

## 1.6 Варіаційний автокодер (VAE)

Варіаційні автокодери трохи сучасніший і цікавіший погляд на автокодування. Це тип автокодувальника з додатковими обмеженнями на закодовані представлення, які вивчаються. Точніше, це автокодер, який вивчає модель прихованої змінної для своїх вхідних даних. Тож замість того, щоб дозволяти вашій нейронній мережі вивчати довільну функцію, ви вивчаєте параметри розподілу ймовірностей, що моделює ваші дані. Якщо ви вибираєте точки з цього розподілу, ви можете генерувати нові зразки вхідних даних: VAE є

«генеративною моделлю». Часто важко забезпечити ефективну обробку даних у реальному часі,особливо якщо вимагається висока точність. Достовірність даних Наявність неточностей, шуму і артефактів у даних може ускладнити виявлення аномалій. Втрата реконструкції вимірює різницю між оригінальними вхідними даними і реконструйованими даними.

У варіаційному кодувальнику використовуються дві головні складові: енкодер та декодер. Енкодер перетворює вхідні дані в закодоване представлення, яке зазвичай має низьку розмірність. Декодер використовує це закодоване представлення для генерації нових прикладів, наближених до оригінальних вхідних даних.

Однак, варіаційний кодувальник включає додатковий елемент - латентний простір, який використовується для згенерованих зразків. Замість простого закодованого представлення, варіаційний кодувальник вивчає розподіл у латентному просторі, який нормально розподілений. Це забезпечує гнучкість генерації нових зразків шляхом демонстрації різних значень у латентному просторі.

Процес навчання варіаційного кодувальника включає мінімізацію втрати, яка складається з двох частин: втрати від реконструкції (яка оцінює, наскільки добре модель може відтворити вхідні дані) і втрати KL-дивергенції [6].

## 1.7 Робота варіаційного автокодера

Параметри моделі навчаються за допомогою двох функцій втрат: втрати від реконструкції, що змушує декодовані зразки відповідати початковим вхідним даних (як і в наших попередніх автокодерах), і розбіжності KL між вивченим латентним розподілом і попереднім розподілом, діючи як термін регулярізації. Насправді ви можете повністю позбутися цього останнього терміна, хоча він допомагає вивчати добре сформовані латентні простори та зменшує переобладнання навчальних даних.

Робота варіаційного автокодувальника (Variational Autoencoder, VAE) включає два основних аспекти: кодування (енкодинг) і декодування. Вхідні дані (наприклад, зображення) подаються на вхід енкодера, який перетворює їх на закодоване представлення у латентному просторі. Достовірність даних наявність неточностей, шуму і артефактів у даних може ускладнити виявлення аномалій. Втрата реконструкції вимірює різницю між оригінальними вхідними даними і реконструйованими даними. Замість простого закодованого представлення, варіаційний кодувальник вивчає розподіл у латентному просторі, який нормально розподілений.

Отримана точка з латентного простору подається на вхід декодера. Декодер реконструює оригінальні дані (наприклад, зображення) з точки латентного простору. Отримана реконструкція порівнюється з оригінальними вхідними даними, і втрата реконструкції обчислюється. Функція втрат варіаційного автокодувальника складається з двох частин: втрати реконструкції і втрати KL-дивергенції. Втрата реконструкції вимірює різницю між оригінальними вхідними даними і реконструйованими даними. Втрата KL, дивергенції вимірює відстань між розподілом у латентному просторі та нормальним розподілом.

Висновок до розділу 1

Основні аспекти: Автокодери дозволяють здійснювати ефективне стиснення даних в невелику кількість прихованих параметрів, а потім відновлювати початкові дані з цих параметрів. Це дозволяє зберігати значну кількість інформації в компактному представленні. У створенні автокодерів важливо вибрати відповідну архітектуру, яка буде найкращим чином підходити до конкретної задачі. Існують різні типи автокодерів, такі як простий автокодер, варіаційний автокодер, згортковий автокодер та інші, кожен з яких має свої переваги та обмеження. Однією з цікавих властивостей автокодерів є їх здатність генерувати нові дані, подібні до навчальних прикладів. Це може бути корисною функцією для створення нових зображень, музики або тексту. Також Автокодери можуть використовуватись для моделювання користувачів та предметів у рекомендаційних системах. Вони допомагають здійснити стиснення та реконструкцію даних, що дозволяє розуміти схожість між користувачами та предметами та здійснювати персоналізовані рекомендації. Часто важко забезпечити ефективну обробку даних у реальному часі, особливо якщо вимагається висока точність. Достовірність даних наявність неточностей, шуму і артефактів у даних може ускладнити виявлення аномалій. Втрата реконструкції вимірює різницю між оригінальними вхідними даними і реконструйованими даними.

# 2 ВИЯВЛЕННЯ АНОМАЛІЙ ЗА ДОПОМОГОЮ KERAS, TENSORFLOW I DEEP LEARNING

Виявлення аномалій за допомогою Keras, TensorFlow і Deep Learning - це процес використання нейронних мереж для виявлення відхилень або аномалій в даних. Керас (Keras) і TensorFlow є двома популярними фреймворками глибокого навчання, які надають широкий набір інструментів для побудови та навчання нейронних мереж.Один з поширених підходів до виявлення аномалій за допомогою глибокого навчання - це використання автокодерів. Автокодери, про які було згадано раніше, можуть бути ефективними моделями для виявлення аномалій. Оскільки автокодери навчаються реконструювати вхідні дані, вони можуть виявити аномалії шляхом порівняння реконструйованих даних з оригінальними. Нижче наведений загальний процес виявлення аномалій за допомогою автокодерів та бібліотек Keras і TensorFlow[7].

## 2.1 Підготовка даних, побудова моделі автокодера

Збираємо набір даних, який складається з нормальних прикладів та аномалій. Розподілити дані на тренувальний набір і тестовий набір.

Побудова моделі автокодера, використовуючи Keras та TensorFlow, створити модель автокодера з енкодером та декодером. Енкодер зменшує розмірність вхідних даних, а декодер намагається відтворити вхідні дані з цього коду.Навчання моделі, використати тренувальний набір даних для навчання автокодера. Метою навчання є мінімізація різниці між вхідними даними та їх відтворенням. Виявлення аномалій за допомогою Keras, TensorFlow і Deep Learning

Виявлення аномалій є складним завданням, оскільки вимагає розуміння нормального функціонування системи та здатності виявити відхилення від цього нормального стану. Ось деякі причини, що роблять цей процес складним

Багатовимірність даних, більшість систем збирають велику кількість даних з різних джерел і датчиків. Ці дані можуть бути представлені у великій кількості

вимірів або характеристик, що робить аналіз складним. Зазвичай, аномалії відбуваються в підмножинах даних, які важко виявити в багатовимірному просторі.

Недостатня кількість марковської інформації Для виявлення аномалій потрібна деталізована інформація про нормальну поведінку системи. Часто важко визначити, які аспекти поведінки є нормальними, оскільки системи можуть бути складними і змінюватися з часом. Крім того, нормальна поведінка може мати сезонні або циклічні варіації, що додатково ускладнює процес виявлення аномалій.

Великий обсяг даних Виявлення аномалій великих масштабів може бути викликом. Обробка великого обсягу даних в реальному часі може вимагати значних обчислювальних ресурсів і швидких алгоритмів. Часто важко забезпечити ефективну обробку даних у реальному часі, особливо якщо вимагається висока точність. Достовірність даних Наявність неточностей, шуму і артефактів у даних може ускладнити виявлення аномалій.

Традиційні методи глибокого навчання, такі як згорткові нейронні мережі (Convolutional Neural Networks, CNN) або рекурентні нейронні мережі (Recurrent Neural Networks, RNN), зазвичай використовуються для вирішення завдань класифікації або регресії на основі навчальних даних. Вони навчаються розпізнавати шаблони і структури, які існують у навчальних даних, і використовують цю інформацію для здійснення передбачень на нових даних. Проте, коли мова йде про виявлення аномалій або викидів, ситуація стає складнішою.

Аномалії можуть представляти собою дані, які суттєво відрізняються від того, що було побачено в процесі навчання, і не мають чітких шаблонів або структур. Такі аномалії можуть бути непередбачуваними та випадковими, що робить їх важко виявити звичайними методами глибокого навчання. Окрім того, традиційні методи глибокого навчання зазвичай потребують значної кількості навчальних даних для досягнення задовільних результатів. Аномалії, зокрема рідкісні аномалії, можуть бути недостатньо представлені в навчальних даних, що ускладнює навчання моделі для їх виявлення.Для виявлення аномалій і викидів

зазвичай використовуються спеціалізовані методи, які можуть враховувати особливості цих даних.

Автокодувальні мережі (Autoencoders) можна використовувати для виявлення аномалій в наступний спосіб:Навчання автокодера: Спочатку автокодер навчається на звичайних даних, які є представниками нормальної поведінки. Автокодер має кодувальну (енкодер) та декодувальну (декодер) частини. В процесі навчання автокодер намагається реконструювати вхідні дані на виході, максимізуючи реконструкційну якість[8].

Після завершення навчання автокодера можна використовувати його для реконструкції нових даних, які не брали участі в навчанні. Отримана реконструкція порівнюється зі вхідними даними, і обчислюється розмір помилки або відхилення (реконструкційна помилка).

Задається порогове значення для реконструкційної помилки. Якщо реконструкційна помилка перевищує поріг, вважається, що даний вхід є аномалією або викидом.

Порогове значення можна налаштувати на основі обраної метрики або експертного знання. Важливо відстежувати баланс між виявленням аномалій і уникненням неправдоподібних сприйняттів нормальних даних як аномалій.

Якщо доступні мітки аномалій, можна застосувати підтримку навчанням з підкріпленням (reinforcement learning) для покращення виявлення аномалій.

Після цього ми реалізуємо архітектуру автокодувальника, яку можна використовувати для виявлення аномалій за допомогою Keras і TensorFlow. Потім ми навчимо нашу модель автокодувальника без нагляду.

Коли автокодер буде навчено, я покажу вам, як ви можете використовувати автокодер для виявлення викидів/аномалій як у вашому наборі для навчання/тестування, так і в нових зображеннях, які не є частиною ваших розділів набору даних. Комбінація декількох моделей або методів може покращити виявлення аномалій. Аномалії визначаються як події, які відхиляються від стандарту, трапляються рідкота не відповідають решті «шаблону».

## 2.2 Виявлення аномалій

Для виявлення аномалій можна використовувати різноманітні методи, включаючи традиційні статистичні підходи і методи машинного навчання. Статистичні методи, такі як контроль даних на основі відхилень від середнього значення або використання розподілів даних (наприклад, нормального розподілу), можуть бути використані для виявлення аномалій. Вони порівнюють спостережувані значення з очікуваними статистичними властивостями та виявляють значення, які суттєво відрізняються.

Алгоритми навчання без учителя, такі як кластеризація, автокодувальні мережі і генеративні моделі, можуть бути використані для виявлення аномалій. Вони шукають незвичайні зразки, які не вписуються в загальну структуру даних або мають низьку ймовірність відповідності моделі[9].

Якщо доступні мітки аномалій у навчальних даних, можна використовувати методи навчання з учителем, такі як класифікація або виявлення викидів. Навчальні дані використовуються для побудови моделі, яка може класифікувати нові дані як нормальні або аномальні на основі вивчених патернів. Ансамблеві методи: Комбінація декількох моделей або методів може покращити виявлення аномалій. Аномалії визначаються як події, які відхиляються від стандарту, трапляються рідко та не відповідають решті «шаблону». Несподівана зміна патернів активності, аномально високий обсяг повідомлень або підозріла поведінка. Щоб виявити аномалії, дослідники машинного навчання створили такі алгоритми, як Isolation Forests, One-class SVMs, Elliptic Envelopes і Local Outlier Factor, щоб допомогти виявити такі події; однак усі ці методи ґрунтуються на традиційному машинному навчанні. Аномалії можуть включати в себе затримки у доставці, втрату товару, незвичайні маршрутизації або несподівані зміни в логістичних потоках.

Несподівана зміна патернів активності, аномально високий обсяг повідомлень або підозріла поведінка користувачів можуть бути виявлені як аномалії.

## 2.3 Приклади аномалій

Незвичайна активність в комп'ютерних системах, така як намагання несанкціонованого доступу, атаки DDoS або вторгнення в мережу, можуть бути виявлені як аномалії.

Надмірно великі або дивергентні транзакції на банківських рахунках можуть вказувати на шахрайство або крадіжку. Такі випадки можуть бути виявлені як аномалії.

Аномалії можуть включати в себе виявлення рідкісних або незвичайних хвороб, виявлення аномальних змін у медичних зображеннях або незвичайний патологічний аналіз крові.

Аномалії можуть включати в себе виявлення дефектів на виробничій лінії, незвичайні або відхилені показники якості виробництва, або незвичайні зміни в параметрах процесу виробництва.

Аномалії можуть включати в себе затримки у доставці, втрату товару, незвичайні маршрутизації або несподівані зміни в логістичних потоках.

Несподівана зміна патернів активності, аномально високий обсяг повідомлень або підозріла поведінка користувачів можуть бути виявлені як аномалії.

Проблема лише ускладнюється тим фактом, що існує величезний дисбаланс у наших мітках класів.

За визначенням, аномалії трапляються рідко, тому більшість наших точок даних будуть дійсними подіями.

Щоб виявити аномалії, дослідники машинного навчання створили такі алгоритми, як Isolation Forests, One-class SVMs, Elliptic Envelopes і Local Outlier Factor, щоб допомогти виявити такі події; однак усі ці методи ґрунтуються на традиційному машинному навчанні.

Як я обговорював у своєму вступному посібнику з автокодувальника, автокодери тип неконтрольованої нейронної мережі, яка може прийняти вхідний набір даних, внутрішнє стиснення даних у представлення латентного простору, реконструювати вхідні дані з латентного представлення[10].

Для виконання цього завдання автокодер використовує два компоненти: кодер і декодер.

При наскрізному навчанні приховані рівні мережі вивчають фільтри, які є надійними та навіть здатними приглушувати вхідні дані.

Однак те, що робить автокодери такими особливими з точки зору виявлення аномалій, так це втрати при реконструкції. Коли ми навчаємо автокодер, ми зазвичай вимірюємо середньоквадратичну помилку (MSE) між реконструйоване зображення з автокодера.

Енкодер отримує вхідне зображення і перетворює його у вектор низькорозмірного представлення, яке називається кодом. Декодер отримує цей код і намагається відновити вихідне зображення з нього. Під час тренування автокодера, він навчається зменшувати розрізнення між вихідним і відновленим зображенням, що спонукає його вивчити корисні функції для реконструкції. Давайте розглянемо приклад реконструкції зображення з автокодера. Спочатку нам потрібно навчити автокодер на деякому наборі даних. Після навчання ми можемо використовувати автокодер для реконструкції зображень. Використовуйте підготовлений набір даних, наприклад, набір даних MNIST, для навчання автокодера. Навчання включає подачу зображень на вхід автокодера і навчання його відновлювати ці зображення на виході. Після навчання вхідного зображення подається на енкодер автокодера, який перетворює його у кодовий вектор. Кодовий вектор, отриманий з енкодера, подається на декодер, який намагається відновити вихідне зображення. Отримане від декодера відновлене зображення порівнюється з вихідним зображення. Чим менша втрата, тим кращу роботу виконує автокодер при реконструкції зображення. Під час тренування автокодера, він навчається зменшувати розрізнення між вихідним і відновленим зображенням, що спонукає його вивчити корисні функції для реконструкції.

Тепер припустімо, що ми навчили автокодер на всьому наборі даних MNIST можна побачити на рисунку 2.1

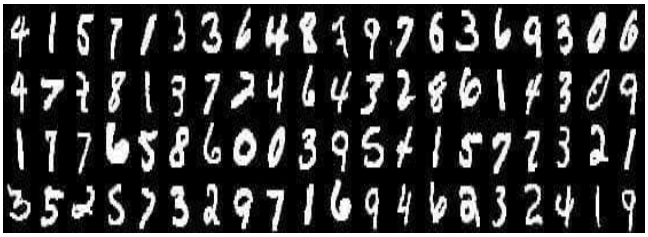

Рисунок 2.1 – Рукописні цифри MNIST

Зразки з набору даних порівняльного аналізу рукописних цифр MNIST. Ми використовуватимемо MNIST для розробки неконтрольованого автокодувальника з Keras, TensorFlow і глибоким навчанням.

Потім ми надаємо автокодеру цифру та кажемо їй її відновити можна побачити на рисунку 2.2


Рисунок 2.2 – Реконструкція цифри з MNIST

Реконструкція цифри з MNIST за допомогою автокодерів, Keras, TensorFlow і глибокого навчання.

Ми очікуємо, що автокодер справді добре справлятиметься з реконструкцією цифри, оскільки це саме те, що автокодер був навчений робити і якщо ми подивимось на MSE між вхідним зображенням і реконструйованим зображенням, ми виявимо, що це досить низько[11].

Давайте тепер припустимо, що ми надали нашому автокодеру фотографію слона і попросили його реконструювати її рисунок 2.3

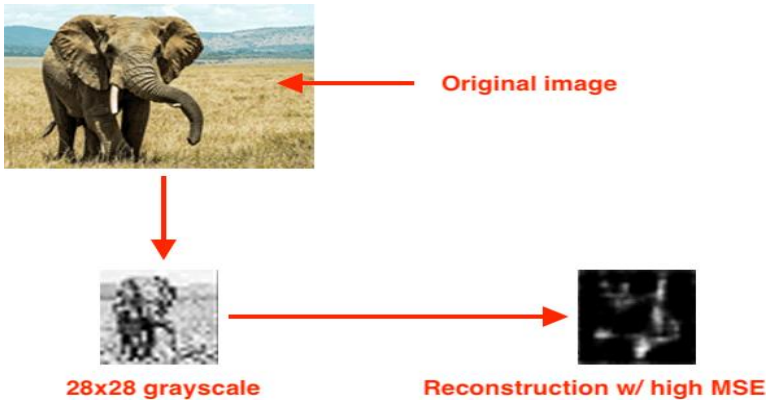


Рисунок 2.3 – Реконструкція невідомого зображення

Коли ми намагаємося реконструювати зображення за допомогою автокодувальника, але результат має високий MSE, ми маємо викид. Оскільки автокодер ніколи раніше не бачив слона, і, більш того, ніколи не навчався реконструювати слона, наш MSE буде дуже високим. Якщо MSE реконструкції високий, то ми, ймовірно, маємо викид. Зокрема, генеративні зіставлення прийняття (GAN) можуть бути використані для відновлення пошкоджених зображень. GAN складається з двох моделей, генератора і дискримінатора. Генератор намагається відновити пошкоджене зображення, тоді як дискримінатор намагається розрізнити відновлене зображення від оригіналу. Цей процес триває досягнення належної якості відновленого зображення. Існує також багато програм та онлайн-інструментів, які допоможуть відновити зображення. Наприклад, Adobe Photoshop має функції для виправлення пошкоджень, відновлення колірних тонів та інших змін у зображеннях. Інші популярні програми, які можуть бути використані для реконструкції зображень, включають GIMP та Photopea онлайн-редактор зображень. Реконструкція зображення може мати різні значення, в залежності від контексту. Під час тренування автокодера, він навчається зменшувати розрізнення між вихідним і відновленим зображенням, що спонукає його вивчити корисні функції для реконструкції. Давайте розглянемо приклад реконструкції зображення з автокодера[12]. Чим менша втрата, тим кращу роботу виконує автокодер при реконструкції зображення. Під час тренування автокодера, він навчається зменшувати розрізнення між вихідним і відновленим зображенням, що спонукає його вивчити корисні функції для реконструкції.

## 2.4 Впровадження автокодувальника

Першим кроком до виявлення аномалій за допомогою глибокого навчання є реалізація нашого сценарію автокодувальника.

Реалізація згорткового автокодувальника ідентична до тих, що описані в статтях про автокодери, а також у підручникках з усунення шумів автокодерів; однак ми розглянемо його тут для повноти якщо вам потрібні додаткові відомості про автокодери, обов’язково зверніться до цих публікацій.

## 2.5 Набір даних MNIST

Одним з найпопулярніших і широко використовується у галузі комп'ютерного зору та машинного навчання. Він складається з набору зображень рукописних цифр (0-9), що складається з тренувальної множини з 60 000 зображень і тестової множини з 10 000 зображень.

Кожне зображення в наборі даних MNIST має розмір 28x28 пікселів і представлене у відтінках сірого, де кожен піксель має значення від 0 до 255, де 0 означає чорний колір, а 255 - білий колір. Задача полягає у класифікації цих зображень на відповідні цифри.

Набір даних MNIST широко використовується для тренування та оцінки алгоритмів класифікації, зокрема для навчання нейронних мереж. Він став своєрідним стандартом для тестування нових методів машинного навчання.

Цей набір даних доступний для завантаження з офіційного сайту MNIST або з використанням бібліотек машинного навчання, таких як TensorFlow чи PyTorch, які мають вбудовану підтримку для MNIST.

## 2.6 Оптимізатор Adam

Популярний метод оптимізації для навчання нейронних мереж, включаючи автоенкодери. Щоб створити автоенкодер і використати оптимізатор Adam для його тренування, вам знадобиться збірка моделі та налаштування процесу навчання.

Фантастична робота з розробки сценарію навчання автоматичного кодувальника без нагляду рисунок 2.4

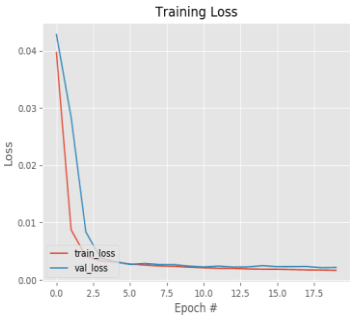


Рисунок 2.4 – Графік кривих втрат

Крім того, ми можемо переглянути наш вихідний файл візуалізації recon_vis.png, щоб побачити, що наш автокодер навчився правильно реконструювати 1 цифру з набору даних MNIST рисунок 2.5

Рисунок 2.5 – Реконструкція рукописної цифри

Перш ніж перейти до наступного розділу, вам слід переконатися, що обидва файли autoencoder.model і images.pickle були правильно збережені у вашому вихідному каталозі. Реалізація нашого сценарію для пошуку аномалій/викидів за допомогою автокодувальника.

## 2.7 Виявлення аномалій

Тепер ми готові виявляти аномалії в нашому наборі даних за допомогою глибокого навчання та нашої навченої моделі Keras/TensorFlow.

Почніть із того, що переконайтеся, що ви використали розділ «Завантаження» цього посібника, щоб завантажити вихідний код звідти ви можете виконати таку команду, щоб виявити аномалії в нашому наборі даних:

З пороговим значенням MSE ~0,0286, що відповідає квантилю 99,9%, наш автокодер зміг знайти сім викидів, п’ять із яких правильно позначені як такі Рисунок 2.6  показано аномалії, які були виявлені під час реконструкції даних

Рисунок 2.6 – Аномалій під час реконструкцій

Незважаючи на те, що автокодер був навчений лише на 1% з усіх 3 цифр у наборі даних MNIST, автокодер справляється напрочуд добре з їх реконструкцією, враховуючи обмежені дані але ми бачимо, що MSE для ці реконструкції були вищими за інші.

Крім того, цифри 1, які були неправильно позначені як викиди, також можуть вважатися підозрілими.

Зображення, які правильно позначені, але демонструють проблему для глибокої архітектури нейронної мережі, повинні вказувати на підклас зображень, які варто дослідити більше автокодери можуть допомогти вам помітити ці підкласи, що виходять за межі Виявлення аномалій за допомогою моделей глибокого навчання без нагляду рисунок 2.7 Точність виявлення аномалії автокодером недостатньо висока[13]. Виявлення аномалій глибоким навчанням - це процес використання методів глибокого навчання (deep learning) для виявлення відхилень або аномалій у вхідних даних. Глибоке навчання є підгалуззю машинного навчання, що базується на нейронних мережах з кількома шари, які можуть автоматично вивчати представлення даних. Для виявлення аномалій глибоким навчанням можна використовувати різні підходи. Робота з великими обсягами даних: Keras, Tensorflow і Deep Learning підтримують обробку великих обсягів даних та паралельне навчання моделей на розподілених системах.

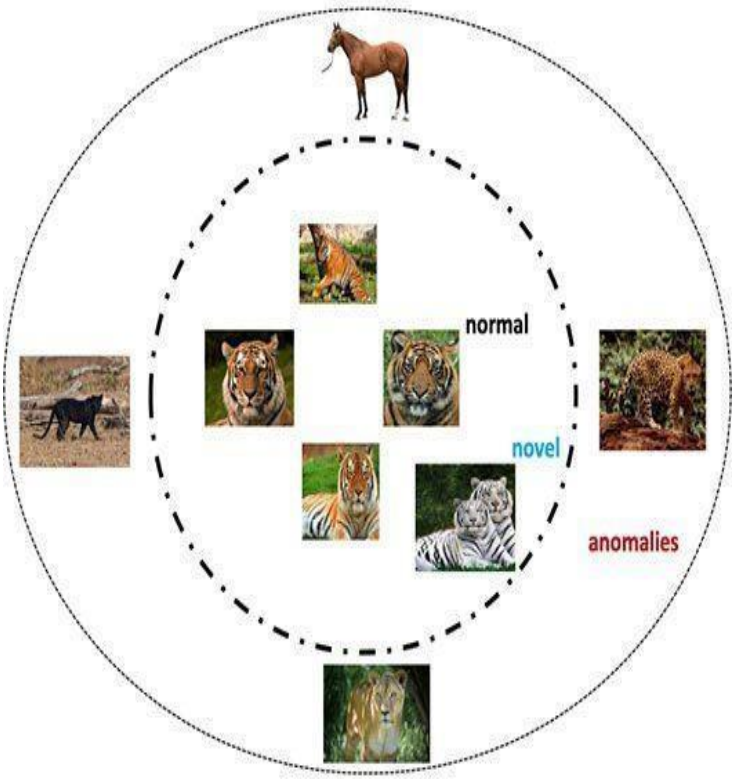


Рисунок 2.7 – Виявлення аномалій

Неконтрольоване навчання, зокрема виявлення аномалій/викидів, далеко не вирішена сфера машинного навчання, глибокого навчання та комп'ютерного зору не існує стандартного рішення для виявлення аномалій, яке було б на 100% правильним. Я б рекомендував вам прочитати оглядову статтю 2019 року «Глибоке навчання для виявлення аномалій: опитування», автори Чалапаті та Чавла, щоб дізнатися більше про поточний стан виявлення аномалій на основі глибокого навчання. Незважаючи на те, що це багатообіцяюче, майте на увазі, що ця галузь швидко розвивається, але знову ж таки, виявлення аномалій/викидів далеко не вирішені проблеми. Виявлення аномалій глибоким навчанням - це процес використання методів глибокого навчання (deep learning) для виявлення відхилень або аномалій у вхідних даних. Глибоке навчання є підгалуззю машинного навчання, що базується на нейронних мережах з кількома шари, які можуть автоматично вивчати представлення даних.Для виявлення аномалій глибоким навчанням можна використовувати різні підходи. Одним з найпоширеніших методів є використання автокодувальників (autoencoders). Автокодувальники

нейронні мережі, які навчаються відтворювати вхідні дані на виході. Якщо автокодувальник навчений на нормальних даних, він буде погано відтворювати аномальні дані, і це можна використовувати для їх виявлення. Ще одним підходом є використання генеративних моделей, таких як генеративні згорткові нейронні мережі (generative convolutional neural networks). Ці моделі навчаються генерувати дані з нормального розподілу, і якщо вхідні дані мають високу відстань до згенерованих даних, це може свідчити про наявність аномалій.Також можна використовувати рекурентні нейронні мережі (recurrent neural networks) або довільні комбінації різних типів нейронних мереж для виявлення аномалій у послідовних даних, наприклад, у часових рядах або текстових данних[14].

Висновок до розділу 2

З використанням Keras, Tensorflow і Deep Learning можна легко побудувати потужні моделі для виявлення аномалій. Можна використовувати різні архітектури нейронних мереж, такі як автокодери, глибокі згорткові мережі або рекурентні нейронні мережі, для виявлення незвичайних паттернів у вхідних даних. Keras і Tensorflow надають потужні інструменти для навчання моделей глибокого навчання. Після навчання моделей за допомогою Keras, Tensorflow і Deep Learning, їх можна використовувати для виявлення аномалій у нових даних. Це може бути корисно в різних сферах, наприклад, для виявлення шахрайства в фінансових операціях, виявлення несправностей в промислових процесах або виявлення аномальної поведінки в системах безпеки. Робота з великими обсягами даних: Keras, Tensorflow і Deep Learning підтримують обробку великих обсягів даних та паралельне навчання моделей на розподілених системах. Це дозволяє виявляти аномалії в реальному часі та працювати з великими потоками даних[15]. Виявлення аномалій глибоким навчанням процес використання методів глибокого навчання (deep learning) для виявлення відхилень або аномалій у вхідних даних. Глибоке навчання є підгалуззю машинного навчання, що базується на нейронних мережах з кількома шари, які можуть автоматично вивчати представлення даних. Для виявлення аномалій глибоким навчанням можна використовувати різні

підходи. Одним з найпоширеніших методів є використання автокодувальників (autoencoders). Автокодувальники нейронні мережі, які навчаються відтворювати вхідні дані на виході.

# 3 УСУНЕННЯ ШУМУ В АВТОКОДЕРАХ ЗА ДОПОМОГОЮ KERAS, TENSORFLOW I DEEP LEARNING

Усунення шуму в автокодерах є одним зі способів використання нейромереж для відновлення забруднених або шумних даних. Автокодери тип нейромереж, які навчаються відтворювати вхідні дані у вихідному шарі зі зменшеною кількістю нейронів. Вони використовуються для стиснення та відтворення інформації, яка включена у вхідні дані.

Для реалізації автокодерів з усуненням шуму можна використовувати бібліотеки Keras та TensorFlow, які надають потужні засоби для побудови та навчання нейромереж.Ось кілька кроків, які можна виконати для реалізації автокодера з усуненням шуму за допомогою Keras та TensorFlow.

Перш за все, необхідно підготувати дані для навчання автокодера. Зазвичай, усувають шум з вхідних зображень або інших типів даних. Можна використати набір даних, який містить забруднені і чисті зображення. Використовуючи Keras, необхідно побудувати модель автокодера. Вона складається з двох частин енкодера і декодера. Енкодер зменшує розмірність вхідних даних та отримує компактне подання (код) даних. Декодер відтворює зображення з цього коду. Зазвичай енкодер та декодер складаються зі шарів повністю з'єднаних нейронів[16].

Потім потрібно навчити модель на забруднених даних. Для цього можна використовувати оптимізатори та функції втрат, що надаються TensorFlow та Keras.

Автокодери для усунення шумів у зображеннях за допомогою Keras, TensorFlow і Deep Learning рисунок 3.1.

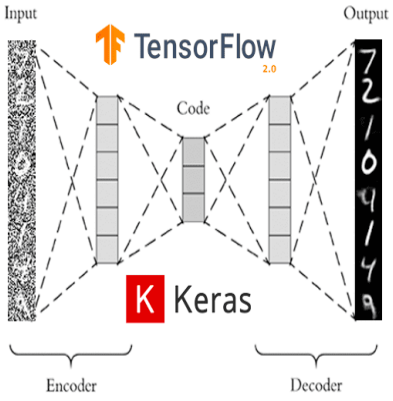


Рисунок 3.1 – Автокодери для усення шуму

Усунення шуму в автокодерах - це процес покращення якості відтворення вхідних даних автокодером, шляхом зменшення впливу шуму на результат відтворення. Автокодери є нейронними мережами, які навчаються відтворювати вхідні дані на виході шляхом перекодування їх у скорочений представлення і повернення вихідних даних з цього представлення[17].

### 3.1 Додавання шуму

Один з популярних підходів - це додавання шуму до вхідних даних під час тренування автокодера. Це може бути випадковий шум, який накладається на дані перед подачею їх на вхід автокодеру. Такий підхід допомагає навчити автокодер виявляти і усувати шум з вхідних даних.

Під час тренування автокодера можна використовувати реконструкційну помилку як функцію втрати. Це допомагає автокодеру зосередитись на точній реконструкції вхідних даних, враховуючи шум, який може бути присутній в даних.

Регуляризація є ефективним способом усунення шуму в автокодерах. Вона полягає в додаванні додаткових обмежень на ваги або шари автокодера, що допомагає уникнути перенавчання і поліпшити здатність автокодера відтворювати вхідні дані без шуму.

Дропаут є технікою, яка випадковим чином вимикає певні нейрони під час тренування.

Автоматично формує підбірку музики, яка вже є в бібліотеці користувача та доповнює її новими піснями, що можуть сподобатись. Також Spotify рекомендує елементи, які схожі на ті, які вже були додані до бібліотеки. Крім того, використовує ці системи для пропонування користувачам концертів та подій з улюбленими виконавцями, що є величезною перевагою перед іншими сервісами стрімінгової музики. Типовими явними даними для збору про користувачів є проставлення ними оцінок “подобається” чи “не подобається”. Ці дані використовуються для півищення ефективності рекомендаційних систем у додатку та забезпечення більш персоналізованого досвіду користувача [18].

Усі ці рекомендаційні системи в Spotify забезпечують користувачів більш індивідуальним досвідом прослуховування музики, дозволяють знайти нову музику та виконавців, які можуть сподобатись, та допомагають зберегти час, який користувачі витрачають на пошук нової музики самостійно.

Використання автокодерів для шумозаглушення, або зниження шуму, що є процесом видалення шуму із сигналу. Сьогодні ми збираємося глибше зануритися й дізнаємося, як можна використовувати автокодери для шумозаглушення, яке також називається «зниженням шуму», що є процесом видалення шуму із сигналу.

## 3.2 Термін шум

В контексті обробки сигналів і данних відноситься до небажаного впливу або спотворення, яке присутнє в сигналі або даних і може впливати на їхню якістьабо зрозумілість. У контексті сигналів, шум може включати в себе випадковіфлуктуації амплітуди, частоти або фази сигналу, а також інші випадкові зміни, якідодатково додаються до корисного сигналу. Шум може бути спричинений різнимифакторами, такими як електромагнітні перешкоди, електричні коливання, термічні шуми, квантові ефекти або помилки передачі даних. У контексті обробки даних, шум означає непотрібні або недоречні варіації або аномалії, які можуть бутиприсутні у вихідних даних[19].

Це може включати помилки вимірювання, артефакти від обладнання, недосконалості в процесі збору даних або інші випадкові спотворення, які виникають під час збору, передачі або обробки даних.

Усунення шуму є важливим завданням в багатьох областях, таких як обробка зображень, звуку, сигналів, медичне зображення, аналіз даних тощо. Методи та алгоритми для усунення шуму варіюються залежно від конкретного контексту та типу шуму, і можуть включати фільтри, статистичні методи, машинне навчання та інші техніки.

Шум квантування, також відомий як квантовий шум, є непередбачуваною зміною в кількості енергії або властивостей частинок на квантовому рівні. Він є однією з фундаментальних характеристик квантових систем і виникає внаслідок

дискретності та ймовірнісного характеру квантових процесів.Квантовий шум проявляється у випадкових коливаннях параметрів квантової системи.

Найбільш відомий приклад шум фотонів, який виникає при прийомі або передачі світла. Цей шум визначає мінімальну межу точності, з якою можна вимірювати світло, і обмежує швидкість передачі інформації в оптичних комунікаційних системах. Крім того, квантовий шум впливає на квантові обчислення, де він може викликати помилки при виконанні квантових операцій. Такі помилки є неуникним наслідком квантового шуму і потребують використання квантових алгоритмів корекції помилок. У зв'язку з тим, що квантовий шум є неуникним, розробка методів зменшення його впливу стала одним з важливих напрямків досліджень в квантовій інформатиці та квантовій технології. Ці методи включають в себе квантове виправлення помилок, кодування, ентанглемент і корекцію помилок. Квантовий шум є випадковими змінами, що виникають на квантовому рівні і впливають на точність вимірювань та виконання квантових операцій.

Стиснення JPEG (або Joint Photographic Experts Group) є одним з найпоширеніших методів стиснення зображень, який зменшує їхрозмір, зберігаючи деталі та забезпечуючи прийнятну якість зображення. Однак процес стиснення JPEG не є безвтратним, і може призводити до втрати якості зображення.

Під час стиснення JPEG використовується алгоритм, який видаляє незначні деталі та візуальні шуми, що менш важливі для людського сприйняття. Це досягається шляхом використання двох основних методів: дискретного косинусного перетворення (DCT) та квантування

Після застосування DCT кожний блок пікселів зображення перетворюється у набір коефіцієнтів, які представляють частотні складові. Деякі найважливіші коефіцієнти зберігаються з високою точністю, тоді як менш важливі коефіцієнти квантуються з меншою точністю.

У процесі квантування кожний коефіцієнт ділиться на певне число, називане квантовим кроком. Чим більше квантовий крок, тим більше коефіцієнтів буде

закруглено або відкинуто. Це призводить до втрати точності й деталей в зображенні.

В результаті стиснення JPEG можуть виникати артефакти, які проявляються у вигляді розмиття, розкрадання, блоків або видимих ступінчастих ефектів. Ці артефакти можуть бути помітними, особливо при високій стискальній якості або при повторному збереженні зображення у форматі JPEG.

З точки зору обробки зображень і комп'ютерного зору, ви повинні думати про шум як про будь-що, що може бути видалено дійсно хорошим фільтром попередньої обробки.

Наша мета полягає в тому, щоб навчити автокодер виконувати таку попередню обробку ми називаємо такі моделі автокодерами, що знімають шум.

Автоматичне кодування шумозаглушення та чому ми можемо їх використовувати[20].

## 3.3 Автоматичне кодування шумів

Автоматичний кодер із шумозаглушенням обробляє зображення з шумами, генеруючи чисте зображення на стороні виведення Рисунок 3.2

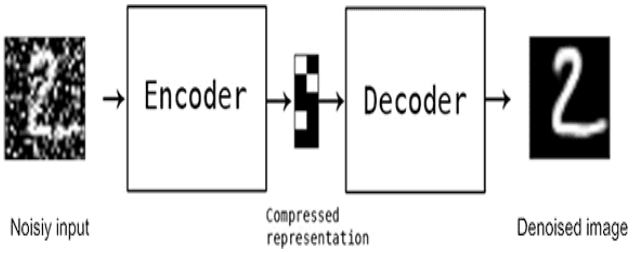


Рисунок 3.2 – Автоматичний кодер із шумозаглушенням

Чи можемо ми навчитися менш ніж за годину навчати автокодерів із зменшенням шуму за допомогою Keras, TensorFlow і Deep Learning? (джерело зображення)

Автокодери з усуненням шумів є розширенням простих автокодерів; однак варто зазначити, що автоматичне усунення шумів спочатку не було призначено для автоматичного усунення шумів у зображенні. Замість цього була

винайдена процедура автоматичного кодування шумів, щоб допомогти. Приховані шари автокодувальника вивчають більш надійні фільтри.

Зменшіть ризик переобладнання автокодера. Запобігайте автоматичному кодуванню вивчати просту функцію ідентифікації. Створимо фільтри нашого кодувальника. Keras високорівнева бібліотека для глибокого навчання, яка забезпечує інтерфейс для побудови та навчання різноманітних нейронних мереж. Вона дозволяє створювати різні типи моделей, включаючи автокодери.Автокодери - це нейронні мережі, які використовуються для навчання ефективного кодування та відкодування даних.

Вони складаються з двох основних компонентів: енкодера, який перетворює вхідні дані на більш компактне представлення, і декодер, який відновлює дані з цього компактного представлення.

Як метод був обраний автоенкодер завдяки його здатності краще справлятися з нелінійностями у вступних даних. На відміну від PCA, яке редукує матрицю фічів, залишаючи лише власні вектори з найвищими власними значеннями, за допомогою автоенкодера можна робити більш інформативну компресію даних.

Внаслідок цього метод автоенкодер краще підходить для завдання детекції аномалій на зображеннях цифр, можна спостерігати у вигляді рисунка 3.3

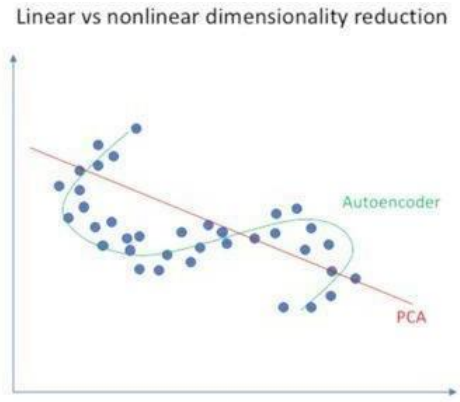


Рисунок 3.3 – Різниця автоенкодера з PCA

Наведена діаграма аугментації зображень (кресленик ІК93.220БАК.005 Д3).

– Оригінальне зображення;

– Додавання нормального шуму;

– Відсікання значень до [0,1];

– доповнене зображення.

Кількість вихідних зразків для візуалізації. Шлях до вихідного зображення візуалізації. Шлях до нашого вихідного графіка matplotlib.

Звуковий шум не може бути прямо застосований до цифр набору даних MNIST, оскільки MNIST набір зображень рукописних цифр. Звуковий шум застосовується до аудіоданих, а не до зображень.

Однак, якщо ви маєте на увазі додавання випадкового шуму до пікселів зображень MNIST, ви можете застосувати розмиття або спотворення до зображень, щоб створити ефект шуму.

Для додавання шуму до зображень набору даних MNIST використаємо гаусівський шум рисунок 3.4.

Рисунок 3.4 – Додавання шуму

Перед навчанням автокодувальника з усуненням шумів на MNIST за допомогою Keras, TensorFlow і Deep Learning ми беремо вхідні зображення (ліворуч) і навмисно додаємо до них шум (праворуч).

## 3.4 Стохастичний шум набору даних MNIST

Приклад результатів навчання автокодувальника глибокого навчання з усуненням шуму за допомогою Keras і Tensorflow на наборі даних порівняльного аналізу MNIST. У нашому сценарії навчання ми додали випадковий шум за допомогою NumPy до зображень MNIST. Випадковий шум за допомогою NumPy до зображень MNIST. Як показано на рисунок 3.5, наш навчальний процес був стабільним і не демонструє ознак переобладнання.

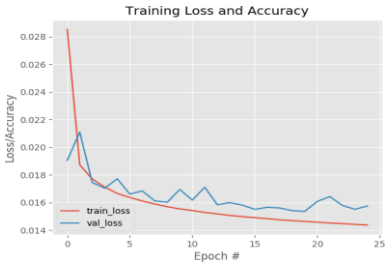


Рисунок 3.5 – Випадковий шум

Дешумуючий автокодер (denoising autoencoder) є одним з типів автокодерів, які використовуються для зменшення шуму в даних.Автокодери є нейронними мережами, що використовуються для навчання представлення даних без нагляду. Дещо особливе у дешумуючому автокодері полягає у використанні штучного шуму для навчання моделі видалення шуму з вхідних даних.Під час навчання дешумуючого автокодера, на вхід автокодера подаються зашумлені дані, тобто дані, які були модифіковані штучним шумом.

Наведена діаграма звичайно та згортки з транспозицією (кресленик ІК93.220БАК.005 Д4)

– Вхід 2х2 ;
– Операція згортки;
– Вихід 2х2д;
– Вхід 3х3 ;
– Операція згортки з трансозицією;
– Вихід 3х3д.

Модель навчається відновлювати оригінальні незашумлені дані з цих зашумлених вхідних даних. Процес навчання включає мінімізацію різниці між відтвореними даними і оригінальними незашумленими даними.Дешумуючі автокодери можуть бути корисними в таких задачах, як відновлення зображень, фільтрація сигналів або видалення шуму зі звукових записів.

Наш дешумуючий автокодер успішно навчено, але як він працював під час видалення шуму, який ми додали до набору даних MNIST.

Щоб відповісти на це запитання, подивіться на малюнок 3.6

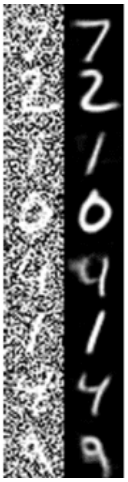

Рисунок 3.6. – Видалення шуму із зображень MNIST

Результати видалення шуму із зображень MNIST за допомогою автоматичного кодувальника з усуненням шумів, навченого Keras, TensorFlow і Deep Learning.

Ліворуч ми маємо вихідні цифри MNIST, до яких ми додали шум, а праворуч ми маємо вихідні дані автоматичного кодування шумопоглинання ми чітко бачимо, що автокодувальник зміг відновити вихідний сигнал (тобто цифру) із зображення при видаленні шуму.

Для автоматичної попередньої обробки зображень можна використовувати досконаліші автоматичні кодери декодування, щоб покращити точність OCR.

Keras високорівнева бібліотека для глибокого навчання, яка забезпечує інтерфейс для побудови та навчання різноманітних нейронних мереж. Вона дозволяє створювати різні типи моделей, включаючи автокодери.Автокодери - це нейронні мережі, які використовуються для навчання ефективного кодування та відкодування даних.

Вони складаються з двох основних компонентів: енкодера, який перетворює вхідні дані на більш компактне представлення, і декодер, який відновлює дані з цього компактного представлення. У цьому прикладі створюється автокодер з одним прихованим шаром розміром 128. Вхідні дані мають розмірність 784, що відповідає зображенням MNIST розміром 28x28 пікселів. Функція активації ReLU використовується для енкодера, а функція активації sigmoid для декодера. Після компіляції моделі автокодера використовується функція втрати

"binary_crossentropy".

Виявлення аномалій за допомогою Keras, TensorFlow і глибинного навчання є актуальною задачею в багатьох сферах, включаючи кібербезпеку, фінанси, медицину та інші.

Keras і TensorFlow популярні бібліотеки глибинного навчання, які надають потужні інструменти для побудови моделей нейронних мереж. Одним із підходів до виявлення аномалій є навчання без нагляду. Зазвичай він базується на автокодувальних моделях.

Автокодувальний підхід полягає в тому, щоб навчити модель відтворювати вхідні дані на виході, зменшуючи розмірність даних в процесі. При цьому, якщо модель натрапляє на аномалію, то відтворення буде мають велику помилку.

Усунення шуму в автокодерах є одним із важливих завдань обробки сигналів за допомогою глибокого навчання. Автокодери нейронні мережі, які використовуються для автоматичного вивчення репрезентації даних шляхом зменшення розмірності та повернення вихідного сигналу з найменшими втратами. Вони також можуть використовуватися для видалення шуму з даних. Одним з популярних фреймворків глибокого навчання є TensorFlow, а для високорівневої роботи з ним часто використовується Keras відкрита бібліотека для побудови нейронних мереж. Для усунення шуму в автокодерах зазвичай використовується так звана "зворотна" модель автокодера, де шумовий сигнал вводиться на вхід, а очищений сигнал відновлюється на виході. Основні кроки для реалізації цього процесу за допомогою Keras, TensorFlow і Deep Learning. Підготовка набору даних: Зібрати набір даних, який містить пари з шумовими та очищеними зображеннями або шумовими та очищеними векторами. Розділити дані на тренувальний і тестовий набори. Створити модель автокодера за допомогою Keras. Модель має складатися з енкодера і декодера. Енкодер зменшує розмірність вхідних даних, стискаючи їх у менший набір функцій або прихований шар. Декодер відновлює очищені дані зі стиснутого представлення. Можна використовувати різні типи шарів, такі як Dense (повнозв'язний), Convolutional (згортковий) або Recurrent.

Автокодери є ефективним способом вивчення представлення ваших даних,

що допомагає в таких завданнях, як зменшення розмірності або виділення ознак. Ви навіть можете навчити автокодер ідентифікувати та видаляти шум із ваших даних.У цій статті ми коротко ознайомимося з автокодувальниками (AE) і глибше заглибимося в певний тип, відомий як шумозаглушувальний автокодувальник (DAE).Якщо ви шукаєте приклади AE для зменшення розмірності та виділення ознак, ви можете звернутися до моєї попередньої статті, Неповні автоденкодери.

Усунення шумів автокодерів (DAE) у всесвіті машинного навчання. Структура DAE Як створити DAE на Python за допомогою Keras/Tensorflow. Усунення шумів автокодерів (DAE) у всесвіті машинного навчання Автокодери відрізняються від інших популярних типів нейронних мереж (прямих, рекурентних і згорткових), оскільки їм не потрібні позначені дані для їх навчання. Отже, ми можемо називати їх неконтрольованими або, якщо ми хочемо бути дуже точними, самоконтрольованими нейронними мережами. Враховуючи різноманітність нейронних мереж і їхній унікальний підхід до машинного навчання, я вважав, що вони заслуговують окремої категорії. Перевірте, чи зможете ви знайти дешумні автокодери (DAE), натиснувши на інтерактивну діаграму сонячних променів нижче та вивчивши її.

Якщо вам подобається наука про дані та машинне навчання, підпишіться, щоб отримувати листи з моїми новими статтями. Якщо ви не є учасником Medium, ви можете приєднатися тут. Структура DAEF. Спочатку давайте коротко нагадаємо про високорівневу структуру автокодерів. Вхідний рівень для передачі вхідних даних у мережу. Прихований рівень, що складається з кодувальника та декодера для обробки інформації шляхом застосування вагових коефіцієнтів, зміщень і функцій активації. Вихідний рівень зазвичай відповідає вхідним нейронам.

Ось ілюстрація наведеного вище підсумку рисунок 3.7

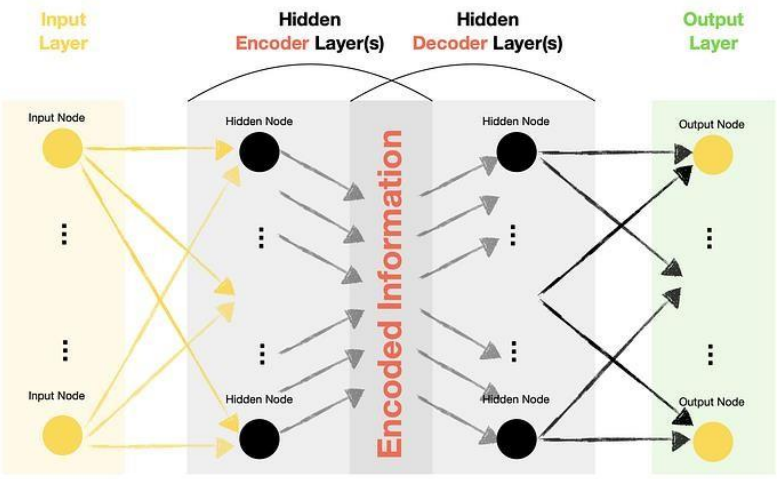


Рисунок 3.7 – Кодувальника та декодера

Найпоширенішим типом автокодувальника є Undercomplete Autocencoder, який стискає (кодує) дані в меншу кількість нейронів, одночасно видаляючи «неважливу» інформацію. Це досягається шляхом одночасного навчання кодера та декодера, щоб вихідні нейрони якомога точніше відповідали вхідним.

Ось приклад того, як виглядала б діаграма мережі для недоповненого автокодувальника рисунок 3.8

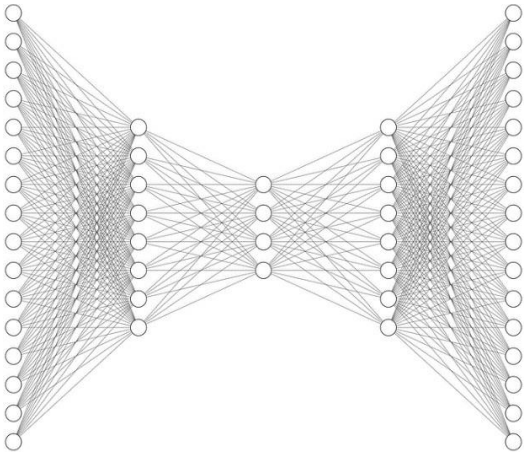

Рисунок 3.8 – Недоповнений автокодувальник

Нейронна мережа недоповненого автокодувальника. Зображення автора, створене за допомогою інструменту AlexNail NN-SVG.Дешумуючий автокодер (DAE)Метою DAE є видалення шуму. Ви також можете сприймати це як налаштований алгоритм усунення шумів, налаштований на ваші дані.Зверніть увагу на наголос на слові налаштований. Оскільки ми навчаємо DAE на певному наборі даних, його буде оптимізовано для видалення шуму з подібних даних.

Наприклад, якщо ми навчимо його видаляти шум із колекції зображень, він добре працюватиме на подібних зображеннях, але не підійде для очищення текстових даних. На відміну від недоповненого AE, ми можемо використовувати ту саму або більшу кількість нейронів у прихованому шарі, що робить DAE надповним.Друга відмінність полягає в тому, що вхідні та вихідні дані не використовуються. Натомість вихідні дані є оригінальними даними (наприклад, зображення), тоді як входи містять дані з деяким додаванням шуму рисунок 3.9

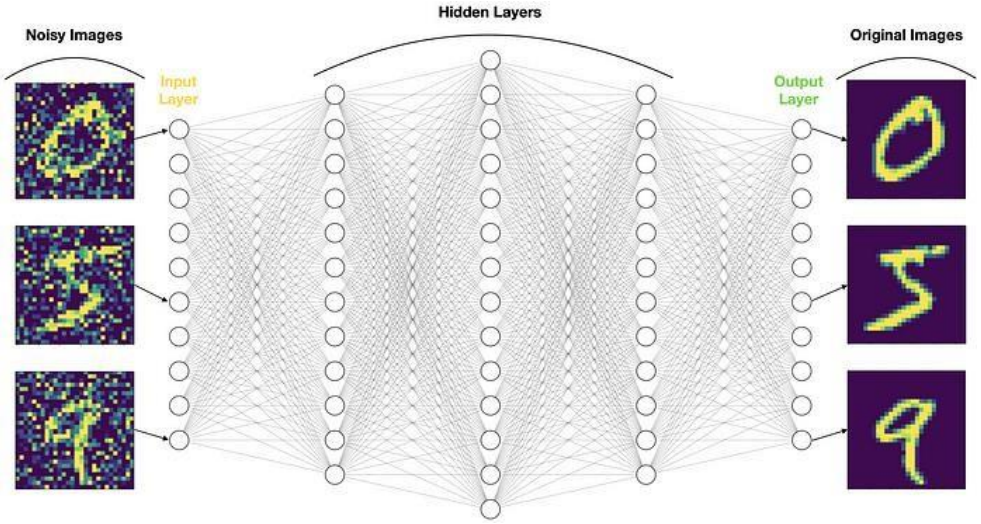


Рисунок 3.9 – Усунення шуму

Вище наведено приклад налаштування DAE для усунення шуму в рукописних цифрах MNIST. У наступному розділі я покажу вам, як налаштувати та навчити такий DAE.

Додавання шуму до зображень. Розв'язуючи постановку задачі, ми повинні пам'ятати про нашу мету створити модель, здатну видаляти шуми на зображеннях. Щоб це зробити, ми використаємо існуючі зображення та додамо їх до випадкового шуму. Тут ми подаємо оригінальні зображення як вхідні дані, а зображення з шумами отримуємо як вихідні дані, а наша модель (тобто автокодер) вивчатиме зв'язок між чистим зображенням і зображенням із шумами та навчиться очищати зображення з шумами. Отже, давайте створимо шумну версію нашого набору даних MNIST і передамо його як вхідні дані для мережі декодера. Ми починаємо з визначення коефіцієнта шуму, який є гіперпараметром. Коефіцієнт шуму множиться на випадкову матрицю, яка має середнє значення 0,0 і стандартне відхилення 1,0. Ця матриця бере зразки з нормального розподілу. Додаючи шум,

ми маємо пам’ятати, що форма випадкового нормального масиву буде подібна до форми даних, до яких ви будете додавати шум.

Побачивши наведений вище код, ви можете подумати, чому тут використовується ця функція кліпу. Щоб переконатися, що значення елементів масиву остаточних зображень знаходяться в діапазоні від 0 до 1, ми можемо використовувати метод np.clip. Кліп функція Numpy, яка відрізає значення за межами діапазону Min-Max і замінює їх визначеним мінімальним або максимальним значенням.

Зберіть набір даних, на якому ви хочете навчити автокодер. За необхідності виміріть та нормалізуйте дані. Модель повинна бути скомпільована з функцією втрати, яка буде оцінювати різницю між вхідними та відтвореними даними.

Згенерований набір даних для навчання моделі. Ви можете використовувати метод зворотного поширення помилки (backpropagation) для оновлення ваг моделі і зменшення функції втрати. Навчання може вимагати багатьох епох, залежно від складності задачі та розміру даних.Тестування та оцінка моделі: Після навчання використовуйте тестовий набір даних, щоб оцінити продуктивність моделі. Порівняйте вхідні дані з відтвореними даними, щоб побачити, наскільки добре автокодер усунув шум.

Автокодери часто використовуються для зменшення шуму та розміру зображення. Відтоді багато читачів зв’язалися зі мною і запитали, чи можу я обговорити усунення шумів зображення за допомогою автокодерів. Натхнення для цієї публікації походить з цього. У сфері нейронних мереж моделювання графічних даних вимагає унікальної стратегії. Згорточні нейронні мережі є найвідомішими нейронними мережами для моделювання даних зображень. Між пікселями зображення він може більш ефективно зберігати відповідну інформацію. CNN є кращим варіантом для обробки візуальних даних завдяки унікальній конструкції шарів.

Конструкцію CNN можна застосувати для усунення шумів або фарбування зображення, як показано на рисунках , або для ідентифікації та класифікації зображень, як показано на рисунку 4.1. На рисунку показано процес навчання

моделі CNN з використанням великої кількості зразки зображень як вхідні дані та мітки як вихідні дані. Потім, щоб визначити, чи є нове зображення «собакою», «кішкою» тощо, ми застосовуємо до нього навчену модель CNN. CNN можна використовувати як автокодер для фарбування зображення або усунення шумів.

CNN використовується в автокодувальнику, що означає, що він використовується як на етапах кодування, так і на етапі декодування автокодувальника, щоб зменшити шум або розфарбувати зображення. Автокодер CNN показано на рисунку. Кожен зразок зображення, який використовується як вхідний сигнал, є зашумленим зображенням, а кожен зразок, який використовується як вихідний сигнал, є безшумним зображенням. Ми можемо створити чисте зображення після застосування навченої моделі до зображення з шумом. Модель також можна навчити розфарбовувати картинки за допомогою цього методу. На рисунку 4.2 показано, як розфарбувати зображення за допомогою CNN Autoencoder.

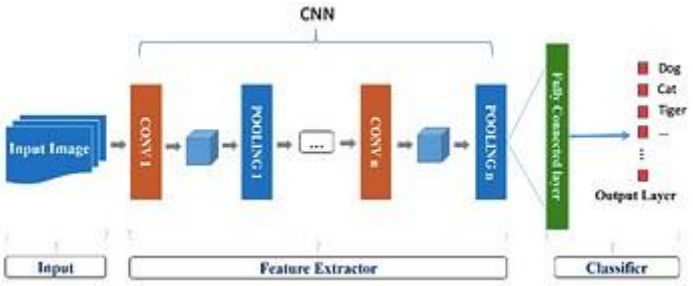


Рисунок 4.1 – Процес навчання моделі CNN

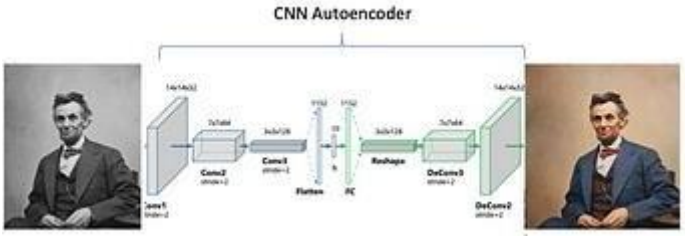


Рисунок 4.2 – Розфарбоване зображення

Оскільки не всі читачі знають про дані зображень, я почну цю частину з невеликого вступу до них (якщо ви вже знайомі, можете пропустити цей розділ). Я продовжую, описуючи базову нейронну мережу для візуальних даних. Це дозволить мені проілюструвати, чому краще використовувати згортковіавтокодери для обробки даних зображення.

Зокрема, я покажу, як згорткові автокодери зменшують шум зображення. У цій публікації я використовую модуль Keras і дані MNIST. За цим посиланням Github ви перейдете до блокнота. Keras на основі Python API нейронної мережі високого рівня, який можна використовувати з TensorFlow. Ця стаття є продовженням статті «Що таке розпізнавання зображень?», яку я наполегливо рекомендую вам прочитати.

Коли ми нарізаємо та складаємо дані, відбувається значна втрата інформації. Автоматичне кодування згортки м’яко витягує інформацію з так званого шару згортки, а не накопичує дані, зберігаючи просторову інформацію даних вхідного зображення. На рисунку (D) показано, як плоске двовимірне зображення перетворюється на товстий квадрат (Conv1), довгий куб (Conv2) і ще один довший куб (Conv3). Ця процедура передбачає збереження географічних зв’язків у даних. Ось як автокодер кодує дані. У центрі розташований повністю пов’язаний автокодер із прихованим шаром, що складається лише з 10 нейронів.

Далі йде етап декодування, який вирівнює кубики та створює двовимірне плоске зображення. На рисунку 4.3 (D) показано симетричний кодер і декодер. Більшість практиків просто дотримуються цих критеріїв, як зазначено в розділі «Просте виявлення аномалій за допомогою автокодерів», навіть якщо вони не обов’язково повинні бути симетричними.

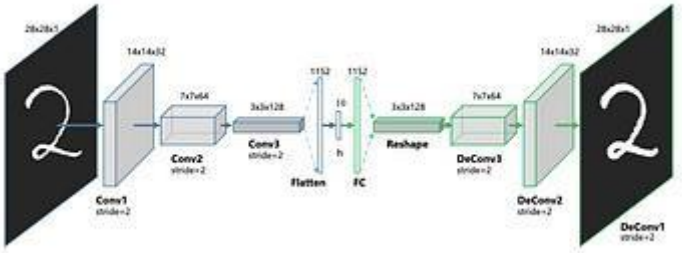


Рисунок 4.3 – Двовимірне плоске зображення

Наведене вище вилучення даних виглядає магічним. Він складається з наступних трьох рівнів, шару згортки, шару reLu та рівня об’єднання рисунок 4.4

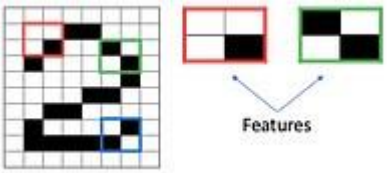


Рисунок 4.4 – Шар reLu

Процес згортки дає велику кількість крихітних компонентів, відомих як карти об’єктів або об’єктів, наприклад зелені, червоні або темно-сині квадрати на малюнку (E). Ці квадрати зберігають співвідношення між пікселями вхідного зображення. Як показано на малюнку (F), нехай кожна функція сканує вихідне зображення. Фільтрування – це назва процедури, яка використовується для створення балів, відфільтроване зображення рисунок 4.5

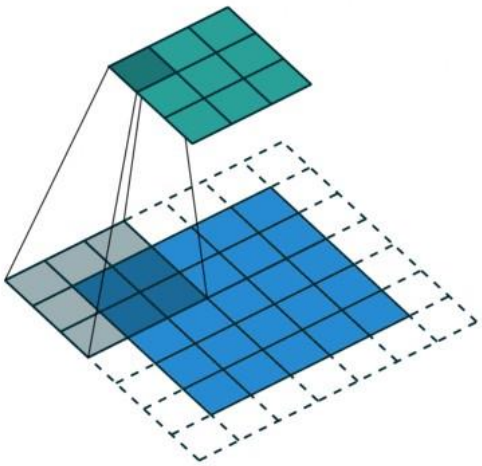

Рисунок 4.5 – Відфільтроване зображення

Кожна функція створює відфільтроване зображення з високими та низькими оцінками після сканування вихідного зображення, як показано на малюнку (G). Ідеальний збіг призводить до високого результату для цього квадрата. Оцінка низька або нульова, якщо збіг поганий або відсутній. Наприклад, червоний квадрат визначив чотири регіони на вихідному зображенні, які повністю збігаються з об’єктом; отже, ці чотири регіони отримали високі бали.

Багатошарові автокодери з усуненням шумів (SDAE) можна використовувати для вирішення низки проблем у цій галузі рисунок 4.6

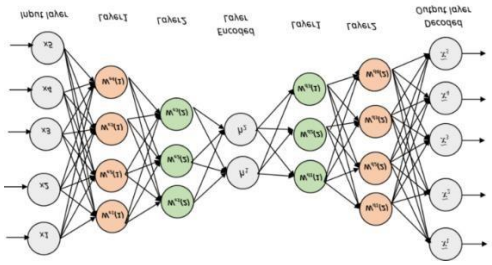

Рисунок 4.6 – Багатошаровий автокодер

Однак що саме таке самоконтрольоване навчання і як SDAE вписується в цю картину. Самоконтрольоване навчання – це різновид машинного навчання, коли модель може навчатися на основі вхідних даних без потреби в мітках чи анотаціях, які явно вказуються. Модель навчається на позначених даних, а потім перевіряється на свіжих, не позначених даних у контрольованому навчанні, що контрастує з неконтрольованим навчанням.

Самоконтрольоване навчання дає змогу моделі навчатися на величезних обсягах немаркованих даних і з часом покращувати свою продуктивність, що робить її потужним інструментом для таких програм, як класифікація зображень і обробка природної мови. Клас штучної нейронної мережі, відомий як SDAE, дуже ефективний у завданнях самоконтролю. Ці мережі складаються з багатьох рівнів автокодерів, які були навчені реконструювати вхідні дані з "зашумленої" версії даних. Мережа здатна вивчати корисні характеристики та шаблони, які можна використовувати для наступних завдань, навчаючи SDAE «очищати» дані та отримувати оригінальні вхідні дані. Багатошарові автокодери з шумозаглушенням нейронна мережа, метою якої є вивчення ефективного представлення вхідних даних, називається автокодером. Вони складаються з кодера та декодера.

Навчені відтворювати початкові вхідні дані з представлення нижчих розмірів, що називається латентним простором. У той час як декодер відповідає за повторне збирання оригінальних вхідних даних із прихованого простору, кодер відповідає за стиснення вхідних даних у прихований простір. Як правило, кодер і декодер навчаються в тандемі за допомогою цільової функції, яка порівнює реконструйований вхідний сигнал з вихідним вхідним.

Автокодери для шумозаглушення, підмножина автокодерів, відомих як автокодери з усуненням шуму, створена з явною метою вивчення надійного

представлення даних за наявності шуму. Їх навчають використовувати зашумлену версію вхідних даних для реконструкції вихідної форми даних без шумів. Спочатку ми вводимо шум у вхідні дані, додаючи гаусівський шум до даних або випадково спотворюючи частину вхідних значень, щоб навчити шумозаглушувальний автокодер. Потім, використовуючи зашумлену версію даних, автокодер навчається відтворювати вихідну безшумну форму даних. Під час навчання автокодер отримує можливість ідентифікувати та ігнорувати шуми вхідних даних під час реконструкції основного сигналу. Для таких видів діяльності, як обробка даних, автоматичне кодування шумів є надзвичайно корисним.

Багаторазове стекування автокодерів. Створення стосу автокодерів або поєднання кількох автокодерів може бути ефективним методом вивчення ієрархічного представлення даних. Вихідні дані кожного автокодувальника в стеку навчаються реконструювати з низьковимірного представлення наступним автокодером у стеку. Припустімо, наприклад, що у нас є стек із трьох автокодерів, перший з яких використовує вхідні дані як єдиний вхід, другий використовує вихід першого автокодера як єдиний вхід, а третій використовує вихід другого автокодера як його єдиний вхід. Спочатку ми навчимо перший автокодер перебудовувати вхідні дані з представлення меншої розмірності, щоб навчити стек автокодерів. Можемо вивчати дедалі складніші характеристики та закономірності в даних, коли ми просуваємося вгору по стеку.

Потім другий автокодер буде навчений відтворювати вихідні дані першого автокодера з представлення нижчої розмірності. Вихідні дані другого автокодувальника потім будуть реконструйовані з низьковимірного представлення шляхом навчання третього автокодувальника. Ми можемо вивчати дедалі складніші характеристики та закономірності в даних, коли ми просуваємося вгору по стеку, об’єднуючи численні автокодери. Після цього за допомогою цих функцій і шаблонів можна виконувати різноманітні наступні завдання, включаючи класифікацію та кластеризацію.

Переваги SDAE перед традиційними автокодерами, порівняно зі звичайними автокодерами, стекові автокодери з усуненням шумів (SDAE) пропонують ряд

переваг. Оскільки вони навчені реконструювати вхідні дані з шумної версії даних, SDAE мають перевагу в тому, що вивчають більш надійні та потужні представлення даних. Це дає їм змогу отримувати функції та шаблони, які можуть протистояти шуму та іншим перешкодам, що може бути особливо корисним для таких завдань, як вивчення функцій і усунення шумів у даних.

Оскільки вони навчені реконструювати вихідні дані одного автокодувальника з представлення нижчого розміру попереднього автокодувальника в стеку, SDAE також мають перевагу в тому, що можуть вивчати ієрархічні представлення даних. У міру того, як вони просуваються вгору в стеку, це дає їм змогу отримувати дедалі складніші функції та шаблони, що може бути корисним для таких завдань, як класифікація та кластеризація. Нарешті, оскільки вони можуть використовувати спільні функції та шаблони, виявлені багатьма автокодерами в стеку, SDAE можуть навчатися швидше, ніж типові автокодери. У результаті SDAE може навчатися ефективніше з меншою кількістю навчальних прикладів, що робить їх особливо вигідними для завдань з обмеженою кількістю навчальних даних.

Залежно від того, чим вони відрізняються один від одного. SDAE навчиться виявляти базові характеристики цифр, такі як лінії та криві, багаторазово виконуючи цей процес. Потім другий рівень SDAE буде навчено створювати вихід першого рівня з версії з шумом. Цей рівень намагатиметься ідентифікувати складніші шаблони, такі як форми окремих цифр.Для того, щоб перебудувати вихід другого рівня з версії з шумом, третій рівень буде потім навчений. Метою цього шару було б розпізнавання ще складніших візерунків, у тому числі самих окремих чисел.

Як працює SDAE у самоконтрольованому навчанні. Самоконтрольоване навчання як концепція, це різновид машинного навчання, коли модель може навчатися на основі вхідних даних без потреби в мітках чи анотаціях, які явно вказуються. Модель навчається на позначених даних, а потім перевіряється на свіжих, не позначених даних у контрольованому навчанні, що контрастує з неконтрольованим навчанням. Модель отримує набір вхідних даних і завдання, яке

потрібно виконати на основі цих даних під час самоконтролю. Завдання вибрано таким чином, щоб його можна було виконати, використовуючи лише вхідні дані, без додаткових міток чи анотацій. Передбачення пропущених слів у реченні на основі навколишнього контексту є прикладом навчального завдання для самостійного контролю.

Самоконтрольоване навчання дає змогу моделі навчатися на величезних обсягах немаркованих даних і з часом покращувати свою продуктивність, що робить її потужним інструментом для таких програм, як класифікація зображень і обробка природної мови. Надаючи моделі значну кількість даних для самоконтрольованого навчання перед тонким налаштуванням на меншому наборі даних з мітками, її також можна використовувати для попереднього навчання моделі для завдання навчання під контролем.

Самоконтрольоване навчання з використанням SDAE. Здатність стекових автокодерів із зменшенням шуму (SDAE) вивчати складніші функції та шаблони, коли вони просуваються в стек, робить їх ефективним інструментом для завдань із самоконтролем навчання. Після цього низка подальших дій, включаючи класифікацію зображень і обробку природної мови, може використовувати ці функції та шаблони.Категоризація зображень є одним із способів використання SDAE для самоконтролю. Скажімо, ми хочемо навчити модель ідентифікувати об’єкти, зображені на кожному зображенні, використовуючи колекцію зображень. Починаючи з шумної версії фотографій, ми могли б навчити SDAE реконструювати вихідні зображення.

Класифікатора можна навчити використовувати ці функції та шаблони, щоб ідентифікувати речі, які видно на фотографіях.Обробка природної мови є ще одним прикладом використання SDAE для самостійного навчання. Розглянемо сценарій, коли ми хочемо навчити модель класифікувати набір даних текстових документів. SDAE можна навчити спочатку відновлювати текст із зашумленої версії. SDAE підбирає складніші елементи та візерунки в тексті, коли просувається вверх по стеку. Класифікатор може бути навчений використовувати ці характеристики та шаблони для поділу текстів на різні групи.

Розпізнавання мовлення є додатковим додатком SDAE для самостійного навчання. SDAE також можна використовувати для таких завдань, як моделювання мови та розпізнавання мовлення. Мережа може поступово вивчати складніші особливості та шаблони мовлення, навчаючи SDAE реконструювати аудіо мовлення з шумової версії аудіо. Потім модель можна навчити розпізнавати вимовлені слова та речення або створювати текст, який звучить природно, використовуючи ці функції та шаблони.

Висновок до розділу 3

Усунення шуму в автокодерах є важливим завданням в глибокому навчанні,і Keras, TensorFlow і Deep Learning надають потужні інструменти для досягнення цієї мети. За допомогою Keras, TensorFlow і Deep Learning можна реалізувати різні підходи до усунення шуму в автокодерах. Один з підходів - це додавання шуму до вхідних даних перед подачею їх на вхід автокодеру.

Шум можна додавати, наприклад, шляхом зашумлення пікселів зображень або додавання випадкового шуму до числових даних.Денойзерні автокодери є іншим підходом, спеціально розробленим для усунення шуму. Вони навчаються з пошкодженими вхідними даними і використовують реконструкційну функцію для відновлення оригінальних даних з пошкодженого варіанту.Регуляризація також може бути використана для усунення шуму в автокодерах. Застосування регуляризаційних методів, таких як Dropout або L1/L2 регуляризація, може допомогти управляти перенавчанням та поліпшити здатність автокодера усувати шум.

В загальному, Keras, TensorFlow і Deep Learning надають широкі можливості для реалізації усунення шуму в автокодерах. Вибір конкретного підходу залежить від контексту та характеру даних. Експериментування з різними методами та налаштуваннями допоможе знайти найкращі рішення для конкретної задачі усунення шуму. Класифікатора можна навчити використовувати ці функції та шаблони, щоб ідентифікувати речі, які видно на фотографіях. Обробка природної

мови є ще одним прикладом використання SDAE для самостійного навчання. Розпізнавання мовлення є додатковим додатком SDAE для самостійного навчання. SDAE також можна використовувати для таких завдань, як моделювання мови та розпізнавання мовлення.

# ВИСНОВКИ

Згорткові нейронні мережі (Convolutional Neural Networks, CNN) є потужним інструментом для пошуку аномалій на зображеннях. Вони можуть виявляти незвичайні чи аномальні образи, виокремлюючи їх від нормальних зображень. Використовуючи згорткові шари для виявлення візуальних ознак і піксельних шаблонів, CNN можуть досить ефективно розпізнавати аномалії на зображеннях.

Під час тренування згорткової нейронної мережі для пошуку аномалій на зображенні, модель навчається розпізнавати нормальні зображення та будувати внутрішнє подання для них. Потім модель може використовуватись для класифікації нових зображень як нормальних або аномальних, залежно від різниці між вхідним зображенням і його внутрішнім поданням.

Згорткові нейронні мережі можуть ефективно виявляти аномалії, такі як викривлення, забруднення або рідкісні об'єкти на зображенні. Загалом, згорткові нейронні мережі є потужними інструментами для пошуку аномалій на зображеннях. Згорткові шари в CNN забезпечують автоматичне виявлення різноманітних ознак на різних рівнях абстракції, що дозволяє моделі впізнавати складні патерни в даних.

Згорткові шари дозволяють моделі автоматично виявляти прості ознаки, такі як краї та форми, а потім комбінувати їх для виявлення більш складних ознак, наприклад, облич, об'єктів або сцен.

Згорткові шари враховують просторову структуру даних і є інваріантними до зміщення об'єктів у вихідному зображенні. Це означає, що модель буде розпізнавати об'єкт незалежно від його положення на зображенні.

Згорткові шари використовують спільні ваги для різних областей зображення, що дозволяє ефективно використовувати параметри моделі і покращує здатність моделі узагальнювати на нові дані.

Згорткові шари зберігають локальну контекстну інформацію, що дає змогу моделі аналізувати даний об'єкт в контексті його оточення.

Основна перевага згорткових нейронних мереж полягає в їхній здатності до автоматичного вивчення ознак, що робить їх особливо корисними для задач обробки зображень. Вони використовують конволюційні шари, які сканують зображення з використанням фільтрів, щоб виділити різні ознаки, такі як краї, текстури або форми. Після цього виконуються додаткові операції, такі як пулінг і фільтрація, для зменшення розміру зображення і збереження найважливіших ознак.

У контексті пошуку аномалій, згорткові нейронні мережі можуть бути навчені розпізнавати типові зображення або шаблони вхідних даних. Потім, коли вони представляють нове зображення, можна порівняти його з вивченими шаблонами і виявити будь-які аномальні або непередбачувані образи.

Для досягнення цього результату, часто використовуються додаткові шари, такі як шари агрегації або шари відновлення. Шари агрегації допомагають скоротити розмірність векторів ознак, що дозволяє зменшити кількість параметрів і зменшити обчислювальні витрати. Шари відновлення використовуються для відновлення зображення з вектора ознак і порівняння відновленого зображення з оригіналом

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. How To Perform Data Compression Using Autoencoders. URL: https://medium.com/edureka/autoencoders-tutorial-cfdcebdefe37(дата звернення: 16.05.2023)

2. Denoising autoencoders with Keras, TensorFlow, and Deep Learning. URL: https://pyimagesearch.com/2020/02/24/denoising-autoencoders-with-keras-tensorflow-and-deep-learning/ (дата звернення: 16.05.2023)

3. Anomaly detection with Keras, TensorFlow, and Deep Learning. URL: https://pyimagesearch.com/2020/03/02/anomaly-detection-with-keras-tensorflow-and-deep-learning/ (дата звернення: 16.05.2023)

4. Stacked Autoencoder Networks Based Speaker Recognition. URL: https://ieeexplore.ieee.org/document/8526953 (дата звернення: 16.05.2023)

5. Building Autoencoders in Keras. URL: https://blog.keras.io/building-autoencoders-in-keras.html (дата звернення: 16.05.2023)

6. Convolutional neural networks. URL: https://conf.ztu.edu.ua/wp-content/uploads/2019/02/45-1.pdf (дата звернення: 16.05.2023)

7. Layers in CNN. URL: https://techukraine.net/згорткові-нейронні-мережі-cnn-вступ/ (дата звернення: 16.05.2023)

8. Application of convolutional neural network network for recognition of handwritten characters. URL: https://tech.vernadskyjournals.in.ua/eng/journals/2019/4_2019/part_1/15.pdf (дата звернення: 16.05.2023)

9. Simple explanation of CNN and its application. URL: https://evergreens.com.ua/ua/articles/cnn.htm (дата звернення: 16.05.2023)

10. Convolutional neural networks as the basis of machine learning. URL:

https://openarchive.nure.ua/server/api/core/bitstreams/a112beea-43f7-42d3-b63b-b7b40c46324e/content (дата звернення: 16.05.2023)

11. Hardware implementation of a neural network. URL: https://mc.today/uk/shho-take-nejronna-merezha/ (дата звернення: 16.05.2023)

12. The history of neural networks. URL: https://futurenow.com.ua/shho-take-nejronna-merezha-prostymy-slovamy/ (дата звернення: 16.05.2023)

13. Recurrent networks. URL: https://www.poznavayka.org/uk/nauka-i-tehnika-2/neyronni-merezhi-yih-zastosuvannya-robota/ (дата звернення: 16.05.2023)

14. Elements of the neural model.URL: https://hi-news.pp.ua/kompyuteri/print:page,1,16198-neyronn-merezh-priklad-viznachennya-znachennya-sfera-zastosuvannya.html (дата звернення: 16.05.2023)

15. Deep learning algoritms. URL: https://hi-news.pp.ua/kompyuteri/print:page,1,16198-neyronn-merezh-priklad-viznachennya-znachennya-sfera-zastosuvannya.html (дата звернення: 16.05.2023)

16. Deep learning algoritms.URL: https://hi-news.pp.ua/kompyuteri/print:page,1,16198-neyronn-merezh-priklad-viznachennya-znachennya-sfera-zastosuvannya.html ( дата звернення: 17.05.2023)

17. Hardware implementation of a neural network.URL: https://mc.today/uk/shho-take-nejronna-merezha/ ( дата звернення: 22.05.2023)

18. Classification of images and signals using neural networks.URL: https://evergreens.com.ua/ua/articles/cnn.html ( дата звернення: 28.05.2023)

19. The history of creation of neural networks.URL: https://livingfo.com/shcho-take-nejronni-merezhi-ta-iak-vony-pratsiuiut/( дата звернення: 30.05.2023)

20. Artificial neural network URL: https://termin.in.ua/neyromerezha/ ( дата звернення: 24.05.2023)