

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ  
ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики  
Кафедра обчислювальної математики

# Курсова робота

Використання глибокого навчання для обернених задач

Виконав студент IV курсу групи  
ПМп-41 напрямку підготовки  
(спеціальності)  
113 – “Прикладна математика”  
Середович В.В.

Керівник: Музичук Ю.А.

Львів - 2021

# Зміст

<b>Вступ</b>	<b>3</b>
<b>1 Постановка задачі</b>	<b>4</b>
<b>2 Структура обернених задач</b>	<b>5</b>
<b>3 Класифікація глибокого навчання для обернених задач</b>	<b>6</b>
3.1 Контрольоване і неконтрольоване навчання . . . . .	6
3.2 Огляд глибокого навчання для обернених задач . . . . .	6
3.3 ? . . . . .	7
<b>4 Автоенкодер для розв’язування обернених задач</b>	<b>8</b>
4.1 Автоенкодер . . . . .	8
4.2 Автоенкодер для видалення шуму . . . . .	9
<b>5 Модель глибокої нейронної мережі</b>	<b>10</b>
5.1 Пряме поширення . . . . .	10
5.2 Зворотнє поширення . . . . .	11
<b>6 Генерація та оцінка шуму в зображеннях</b>	<b>14</b>
6.1 Генерація шуму . . . . .	14
6.2 Оцінка зображень . . . . .	14
<b>7 Реалізація моделі для автоенкодера</b>	<b>16</b>
7.1 Датасет . . . . .	16
7.2 Архітектура автоенкодера . . . . .	16
7.3 Тренування . . . . .	17
<b>8 Аналіз результатів</b>	<b>19</b>
<b>Висновок</b>	<b>20</b>
<b>Додатки</b>	<b>21</b>
<b>Література</b>	<b>22</b>

# Вступ

Оберненими задачами називають такі задачі, в яких необхідно відновити дані про деякий процес з використанням непрямих спостережень. Такі спостереження отримують за допомогою певного прямого процесу який, зазвичай, є необоротним, а отже не має єдиного розв'язку. Як наслідок, задачі такого типу можуть бути нерозв'язними без додаткової інформації про об'єкт дослідження. До таких погано обумовлених задач можна віднести багато прикладів з реальних фізичних процесів, таких як задачі акустичної томографії або сейсморозвідки на основі звукових сигналів. Існує також широке поле обернених задач в області зображень та комп'ютерного бачення, такі як реконструкція рентгенівської, видалення шуму, збільшення розмірності, заповнення втрачених даних та багато інших.

Класичні методи до розв'язання таких задач припускає наявність певної попередньої інформації про обернену задачу, на основі якої будується прямий оператор та функція регуляризації. В результаті, формується задача мінімізації, яка зводиться до знаходження розв'язку який одночасно добре підходить під існуючі спостереження, а також, є ймовірним враховуючи попередні знання. Такий підхід є ефективним в багатьох задачах але може бути важким в реалізації.

Хоча такий підхід широко використовується, за останій час алгоритми глибокого навчання для розв'язуванні обернених задач набирають значну популярність через свою ефективність, та універсальність для багатьох різних типів задач що було продемонстровано в роботі Gregory Ongie та ін. [1].

В межах цієї роботи будемо розглядати деякі існуючі методи по реконструкції зображень на основі алгоритмів глибокого навчання та проаналізуємо їх ефективність.

# 1 Постановка задачі

Оберненими задачами будемо вважати такі задачі, в яких невідомим є  $n$ -піксельне зображення  $\mathbf{x} \in \mathbb{R}^n$  яке було отримане з  $m$  вимірювань  $\mathbf{y} \in \mathbb{R}^m$  відповідно до рівняння

$$\mathbf{y} = \mathcal{A}(\mathbf{x}) + \boldsymbol{\varepsilon} \quad (1.1)$$

де  $\mathcal{A}$  - це прямий оператор вимірювання та  $\boldsymbol{\varepsilon}$  є певним вектором шуму. Метою задачі є відновлення  $\mathbf{x}$  з  $\mathbf{y}$ . Можна розглянути більш загальний випадок моделі неадитивного шуму, який має вигляд

$$\mathbf{y} = \mathcal{N}(\mathcal{A}(\mathbf{x})) \quad (1.2)$$

де  $\mathcal{N}(\cdot)$  є прикладами вибірки з шумом.

**Означення 1.1** Відповідно до поняття, уведеного Жаком Адамаром, задачу 1.2 називають коректно поставленою, якщо вона задовольняє наступні умови:

1. Для кожного  $\mathbf{x}$  розв'язок задачі існує.
2. Розв'язок є єдиний для кожного  $\mathbf{x}$ .
3. Розв'язок є стійкий до малих варіацій величини  $\mathbf{x}$ , тобто достатньо малим зміненням величини  $\mathbf{x}$  відповідають як завгодно малі зміни величини  $\mathbf{y}$ .

**Означення 1.2** Задачу, яка не задовольняє хоча б одну з умов означення 1.1, називають некоректно поставленою.

Тому, очевидно, що розглянута обернена задача є некоректно (або погано обумовленою), оскільки в ній порушуються умови означення 1.1. Така задача знаходження єдиного розв'язку, яка задовольняє спостереженням є складною або неможливою, за умови відсутності попередніх знань про дані.

Оцінку справжнього зображення  $\mathbf{x}$  з  $\mathbf{y}$  вимірювання називають задачею реконструкції зображення. Класичні підходи до реконструкції зображень припускають наявність деякої попередньої інформації про зображення, яку називають пріором. В якості пріору можуть виступати параметри гладкості, щільності та інші геометричні властивості зображення.

Отже, метою даної роботи буде розв'язання таких обернених задач за допомогою алгоритмів глибокого навчання. Зокрема, будемо розглядати проблему видалення шуму у зображеннях.

## 2 Структура обернених задач

Відповідно до постановки задачі, ми прагнемо відновити векторизоване зображення  $\mathbf{x} \in \mathbb{R}^n$  з вимірювань  $\mathbf{y} \in \mathbb{R}^m$  у вигляді  $\mathbf{y} = \mathcal{A}(\mathbf{x}) + \boldsymbol{\varepsilon}$ .

Якщо розподіл шуму відомий,  $\mathbf{x}$  можна відновити розв'язавши задачу оцінки максимальної ймовірності (maximum likelihood):

$$\hat{\mathbf{x}}_{\text{ML}} = \arg \max_{\mathbf{x}} p(\mathbf{y}|\mathbf{x}) = \arg \min_{\mathbf{x}} -\log p(\mathbf{y}|\mathbf{x}) \quad (2.1)$$

де  $p(\mathbf{y} | \mathbf{x})$  це ймовірність спостереження  $\mathbf{y}$  за умови якщо  $\mathbf{x}$  є справжнім зображенням.

В залежності від умов задачі, можуть бути відомі попередні дані про те яким має бути  $\mathbf{x}$ . Ці умови можна для формування задачі оцінки максимальної апостеріорної ймовірності (maximum a posteriori), що приводить до задачі ??.

$$\hat{\mathbf{x}}_{\text{MAP}} = \arg \max_{\mathbf{x}} p(\mathbf{x}|\mathbf{y}) = \arg -\max_{\mathbf{x}} p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) = \arg \min_{\mathbf{x}} -\ln p(\mathbf{y}|\mathbf{x}) - \ln p(\mathbf{x}) \quad (2.2)$$

Для випадку білого гаусівського шуму, цільову функцію можна сформулювати як:

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x}} \frac{1}{2} \|\mathcal{A}(\mathbf{x}) - \mathbf{y}\|_2^2 + \lambda R(\mathbf{x}) \quad (2.3)$$

де  $\|\mathcal{A}(\mathbf{x}) - \mathbf{y}\|_2^2$  відповідає за правдивість даних та позначає різницю між вихідним та шумним зображеннями,  $R(\mathbf{x})$  - від'ємний логарифмічний пріор який позначає член регуляризації, а  $\lambda$  є параметром регуляризації. Для варіаційних методів видалення шуму, ключовим є пошук відповідного пріору зображення  $R(\mathbf{x})$ . Варіантами таких пріорів моделі можуть бути градієнтні або розріджені пріори.

Прикладом такого підходу до розв'язування некоректних задач є метод регуляризації Тіхонова. Він базується на мінімізації параметра регуляризації  $R_{\text{TR}}$  за  $L_2$  нормою, який можна подати у вигляді 2.4.

$$R_{\text{TR}}(\mathbf{x}) = \|\nabla \mathbf{x}\|_2 = \sqrt{|\nabla_v \mathbf{x}|^2 + |\nabla_h \mathbf{x}|^2} \quad (2.4)$$

де  $\nabla_h \mathbf{x}$  та  $\nabla_v \mathbf{x}$  є операторами градієнта по горизонталі та вертикалі зображення відповідно.

Задача максимальної апостеріорної оцінки може використовуватись для реконструкції зображень, однак такий підхід може бути не таким ефективним, якщо розподіл шуму або прямий оператор  $\mathcal{A}$  є невідомі. Алгоритми основані на використанні машинного навчання дають змогу побороти більшість з цих труднощів, що робить їх ефективною альтернативою класичному підходу.

## 3 Класифікація глибокого навчання для обернених задач

### 3.1 Контрольоване і неконтрольоване навчання

Перший і найпоширеніший тип розв'язування обернених задач з використанням глибокого навчання є контрольована інверсія. Ідея полягає у створенні співвідношення між датасетом справжніх зображень  $x$  та відповідними вимірюваннями  $y$ . Тобто ми можемо натренувати нейронну мережу приймати значення  $y$  та реконструювати обернене значення  $x$ . Цей підхід є дуже ефективним, однак є чутливим до змін в операторі вимірювання  $A$ .

Другим типом розв'язування обернених задач є неконтрольованого навчання. Він передбачає, що інформація про пари вхідної та вихідної інформації  $x$  та  $y$  невідомі під час тренування. До нього можна віднести ситуації коли відомі тільки справжні зображення  $x$  або тільки результати вимірювання  $y$ .

Ці два підходи мають фундаментальні відмінності та ця робота націлена саме на методи контрольованого навчання, тому що очікується, що вони дадуть кращі результати в порівнянні з класичними методами.

### 3.2 Огляд глибокого навчання для обернених задач

В роботі [1] наведена детальна класифікація основної кількості існуючих підходів до розв'язання обернених задач на основі глибокого навчання. Більшість з них можна поділити на декілька груп:

- **Пряма модель  $A$  є відома під час тренування та тестування**  
Для цього випадку найбільш доцільним підходом є використання контрольованих моделей машинного навчання, тому що маючи доступ до оригінальних зображень та прямого оператора можна легко згенерувати пари для тренування моделі.
- **Пряма модель  $A$  є відома тільки під час тестування**  
Для таких алгоритмів характерно, що якщо один раз навчити глибоку модель, цю ж саму модель можна буде використовувати для будь-якої іншої прямої моделі. Це вигідно в ситуаціях, коли є достатня кількість чистих зображень, а навчати глибокі нейронні мережі для різних прямих моделей є недоцільно.
- **Пряма модель  $A$  є відома тільки частково**  
Така ситуація можлива, наприклад, у випадку коли пряма модель є параметричною, і ми знаємо або розподіл, або достатню статистику про параметри.

- **Пряма модель  $\mathcal{A}$  є невідома**

У деяких випадках пряма-модель може бути абсолютно невідомою, неправильно визначеною або обчислювально неможливою для використання в навчанні та тестуванні. Тоді навчання може відбуватись лише з відповідними парами зображень та вимірювань.

### 3.3 ?

TODO

## 4 Автоенкодер для розв'язування обернених задач

Вперше автоенкодер був представлений в роботі [2] як нейронна мережа яка тренується відтворювати свої вхідні дані. З того часу він використовувався в багатьох задачах з області машинного навчання та був детально описаний автором в [3]. Зокрема, автоенкодер є досить ефективною моделлю для розв'язування обернених задач.

### 4.1 Автоенкодер

Автоенкодером називають нейронну мережу яка навчається копіювати свої вхідні дані у вихідні. Така мережа має проміжний шар  $h$ , до якого будемо звертатись як до репрезентаційного, який зберігає параметри необхідні для представлення вхідних даних. Таку нейронну мережу можна подати у складі двох частин: функції енкодера  $h = f(x)$  та декодера який відтворює  $r = g(h)$ . Ця архітектура подана на зображенні 4.1.

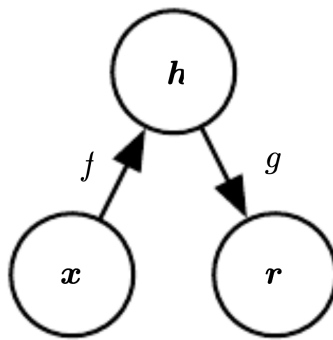


Рис. 4.1: Загальна структура автокодера, що відображає вхід  $x$  на вихід  $r$  (реконструкцію) через внутрішнє представлення  $h$ . Автокодер складається з двох компонентів: енкодера  $f$  (відображення  $x$  до  $h$ ) та декодера  $g$  (відображення  $h$  до  $r$ ) [3]

Якщо автоенкодеру вдається навчитися просто відтворювати  $g(f(x)) = x$  для всіх прикладів, то це не має особливої користі. Тому автоенкодери зазвичай обмежують таким чином, щоб вони не могли відтворювати ідеальну копію вхідних даних. Для цього можна, наприклад, регулювати розмірність репрезентаційного шару таким чином, щоб вона була суттєво меншою за розмірність вхідних даних.



## 4.2 Автоенкодер для видалення шуму

Класичні автоенкодери мінімізують деяку функцію:

$$L(\mathbf{x}, g(f(\mathbf{x}))) \quad (4.1)$$

де  $L$  це штрафна функція яка визначає відмінність функції  $g(f(\mathbf{x}))$  від  $\mathbf{x}$ , таку як, наприклад,  $L^2$  норма від їх різниці. В результаті це призводить до того, що композиція функцій  $g \circ f$  навчається бути тотожним відображення якщо для того є можливість. На відміну від цього, автоенкодер для видалення шуму мінімізує:

$$L(\mathbf{x}, g(f(\tilde{\mathbf{x}}))) \quad (4.2)$$

де  $\tilde{\mathbf{x}}$  є копією  $\mathbf{x}$  який був пошкоджений деяким шумом.

Отже, такий автоенкодер має не просто відтворити вхідні дані, а ще й відновити пошкодження. Процес тренування автоенкодера для видалення шуму заданий на 4.2.

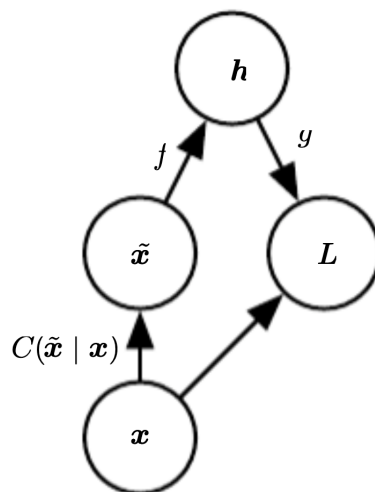


Рис. 4.2: Структура функції витрат для автоенкодера який навчається реконструювати чисті зображення  $\mathbf{x}$  з пошкоджених. Тренування виконується на основі мінімізації функції втрат:  $L = -\log p(\mathbf{x} | \mathbf{h} = f(\tilde{\mathbf{x}}))$ , де  $\tilde{\mathbf{x}}$  - пошкоджена версія прикладу  $\mathbf{x}$ , отримана в результаті деякого процесу руйнування  $C(\tilde{\mathbf{x}} | \mathbf{x})$ . [3]

Такий підхід до навчання видалення шуму змушує  $f$  та  $g$  явно вивчати структуру даних що дозволяє йому ефективно видаляти шум з пошкоджених зображень.

Автоенкодер для видалення шуму доцільно використовувати у випадку коли пряма модель  $\mathcal{A}$  та чисті зображення є відомі або коли є достатня кількість пар чистих та пошкоджених зображень для тренування нейронної мережі.

## 5 Модель глибокої нейронної мережі

Нехай маємо набір тренувальних даних:

$$(x^{(1)}, y^{(1)}), \quad (x^{(2)}, y^{(2)}), \quad \dots, \quad (x^{(m)}, y^{(m)})$$

$$x = \begin{bmatrix} \vdots & \vdots & \dots & \vdots \\ x^{(1)} & x^{(2)} & \dots & x^{(m)} \\ \vdots & \vdots & \dots & \vdots \end{bmatrix} \quad x^{(i)} \in \begin{bmatrix} x_1^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix}$$
$$y = \begin{bmatrix} \vdots & \vdots & \dots & \vdots \\ y^{(1)} & y^{(2)} & \dots & y^{(m)} \\ \vdots & \vdots & \dots & \vdots \end{bmatrix} \quad y^{(i)} \in \begin{bmatrix} y_1^{(i)} \\ \vdots \\ y_n^{(i)} \end{bmatrix}$$

де

- $x$  - матриця в якій кожен  $i$ -ий стовпець є розгорнутим у вектор справжнього зображення,  $i = 1, \dots, m$
- $y$  - матриця в якій кожен  $i$ -ий стовпець є розгорнутим у вектор пошкодженого зображення
- $m$  - кількість прикладів
- $n$  - кількість характеристик в кожному прикладі (довжина розгорнутого в вектор зображення)

### 5.1 Пряме поширення

Алгоритм прямого поширення для глибокої нейронної мережі буде мати вигляд 5.1.

$$z^{(l)} = w^{(l)} a^{(l-1)} + b^{(l)} \tag{5.1}$$
$$a^{(l)} = \sigma(z^{(l)})$$

де

- $l$  - номер шару нейронної мережі, де  $l = 1 \dots L$
- $n^{[l]}$  - кількість нейронів в  $l$  шарі
- $a^{(l)}$  - вектор стовпець активацій нейронів на для шару  $l$  ( $n^{[l]} \times 1$ )
- $b^{(l)}$  - вектор стовпець ваг зміщення ( $n^{[l]} \times 1$ )
- $w^{(l)}$  - матриця ваг поміж шарами  $l - 1$  та  $l$  ( $n^{[l]} \times n^{[l-1]}$ )

- $\sigma$  - це деяка активаційна (стискуюча) функція, яку ми можемо прийняти як логістичну (для діапазону від 0 до 1) або  $\tanh$  (для діапазону від -1 до 1), або будь-яку іншу диференційовану функцію.

Визначимо штрафну функцію як середньоквадратичну похибку між активаціями останнього шару  $a^{(L)}$  та справжніми зображеннями  $y$ .

$$J(\omega, b) = \frac{1}{m} \sum_{i=1}^m L(a^{(L,i)}, y^{(i)}) \quad (5.2)$$

де, функцією витрати для одного набору елементів визначимо половину евклідової відстані.

$$L(a^{(L,i)}, y^{(i)}) = \frac{1}{2} \|a^{(L,i)} - y^{(i)}\|_{L_2}^2 = \frac{1}{2} \sum_{j=1}^n (a_j^{(L,i)} - y_j^{(i)})^2 \quad (5.3)$$

де  $\|\cdot\|$  це  $L_2$  норма.

## 5.2 Зворотнє поширення

Задача полягає в тому, щоб знайти параметри  $w \in \mathbb{R}^m, b \in \mathbb{R}$  що мінімізують штрафну функцію реконструкції:

$$\arg \min_{w, b} J(w, b) \quad (5.4)$$

Мінімізацію будемо проводити алгоритмом градієнтного спуску. Для цього, спочатку обчислюємо похідну від функції середньо квадратичної похибки останнього шару нейронної мережі.

$$\begin{aligned} da^{(L,i)} &= \frac{\partial L(a^{(L,i)}, y^{(i)})}{\partial a^{(L,i)}} = \sum_{j=1}^n (a_j^{(L,i)} - y_j^{(i)}) \\ dz^{(L,i)} &= \frac{\partial L(a^{(L,i)}, y^{(i)})}{\partial z^{(L,i)}} = \frac{\partial L(a^{(L,i)}, y^{(i)})}{\partial a^{(L,i)}} \frac{\partial a^{(L,i)}}{\partial z^{(L,i)}} = \sum_{j=1}^n (a_j^{(L,i)} - y_j^{(i)}) \sigma'(z^{(L,i)}); \\ dw^{(L,i)} &= \frac{\partial L(a^{(L,i)}, y^{(i)})}{\partial w^{(L,i)}} = dz^{(L,i)} \frac{\partial z^{(L,i)}}{\partial w^{(L,i)}} = dz^{(L,i)} a^{(L-1,i)}; \\ db^{(L,i)} &= \frac{\partial L(a^{(L,i)}, y^{(i)})}{\partial b^{(L,i)}} = dz^{(L,i)} \frac{\partial z^{(L,i)}}{\partial b^{(L,i)}} = dz^{(L,i)} \end{aligned} \quad (5.5)$$

Перепишемо похідні в більш компактному вигляді, для всіх шарів нейронної мережі:

$$\begin{aligned}\frac{\partial J(w, b)}{\partial W^{(l)}} &= \frac{1}{m} \sum_i^m \delta^{(l,i)} \left( a^{(l-1,i)} \right)^\top \\ \frac{\partial J(w, b)}{\partial b^{(l)}} &= \frac{1}{m} \sum_i^m \delta^{(l,i)}\end{aligned}\tag{5.6}$$

де

$$\delta^{(l,i)} = \begin{cases} \sum_{j=1}^n \left( a_j^{(L,i)} - y_j^{(i)} \right) \sigma'(z^{(L,i)}), & \text{якщо } l = L \\ \left( (W^{(l)})^\top \delta^{(l+1,i)} \right) \sigma'(z^{(l,i)}), & \text{якщо } l < L \end{cases}\tag{5.7}$$

В залежності від активаційної функції визначеної на певному шарі нейронної мережі значення похідної буде відрізнятись. Розглянемо наступні варіанти активаційних функцій:

- **Sigmoid** (Logistic function)

$$\begin{aligned}\sigma(z) &= \frac{1}{1 + e^{-z}} \\ \sigma'(z) &= a(1 - a)\end{aligned}\tag{5.8}$$

- **ReLU** (Rectified Linear Units)

$$\begin{aligned}\sigma(z) &= \max(0, z) \\ \sigma'(z) &= \begin{cases} 0, & z < 0 \\ 1, & z \geq 0 \end{cases}\end{aligned}\tag{5.9}$$

- **Tanh** (Hyperbolic tangent)

$$\begin{aligned}\sigma(z) &= \frac{e^z - e^{-z}}{e^z + e^{-z}} \\ \sigma'(z) &= 1 - a^2\end{aligned}\tag{5.10}$$

Далі запишемо алгоритм заворотнього поширення з використанням стохастичного градієнтного спуску та minibatch оптимізації 1.

**Algorithm 1** Градієнтний спуск

---

```

1: Input: Тренувальні дані  $x, y$ , гіперпараметри:  $N, M, \varepsilon, \alpha$ 
2: Output: Оптимальні параметри моделі  $w, b$ 
3:  $i = 0$ 
4: while  $i < N$  or (  $\|dw\| < \varepsilon$  and  $\|db\| < \varepsilon$  ) do
5:    $x^M \leftarrow$  випадкова група прикладів  $x$ , розміром  $M$ 
6:    $y^M \leftarrow$  випадкова група прикладів  $y$ , розміром  $M$ 
7:   for  $l = 1$  to  $L$  do
8:      $w^{(l)} = w^{(l)} - \alpha \frac{\partial J(w, b, x^M, y^M)}{\partial w^{(l)}}$ 
9:      $b^{(l)} = b^{(l)} - \alpha \frac{\partial J(w, b, x^M, y^M)}{\partial b^{(l)}}$ 
10:     $i = i + 1$ 
11:   end for
12: end while
13: return  $w, b$ .

```

---

## 6 Генерація та оцінка шуму в зображеннях

### 6.1 Генерація шуму

Для того щоб застосувати описані методи видалення шуму, необхідно спочатку визначити яким чином цей шум, тобто прямий оператор  $\mathcal{A}$  вибрати. Для цього будемо використовувати адитивний білий гаусівський шум з різними параметрами середньокватдратичного відхилення. Задамо загальну модель адитивного шуму як 6.1.

$$\mathbf{y} = \mathbf{x} + \varepsilon \quad (6.1)$$

де  $\varepsilon$  це шум який має нульове середнє значення та відповідає нормальному розподілу Гауса:

$$\varepsilon_i \sim \mathcal{N}(0, \sigma^2). \quad (6.2)$$

Ми можемо змоделювати чисте від шумів зображення  $\mathbf{x} \in \mathbb{R}$  як розподіл Гауса з нульовою дисперсією, тобто  $\mathbf{x}_i \sim \mathcal{N}(\mathbf{x}, 0)$ , що дає нам змогу подати  $\mathbf{y}$  теж як розподіл Гауса. Сума двох розподілів Гауса  $y_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$  та  $y_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$  також є розподілом Гауса  $y_1 + y_2 \sim \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$ . Таким чином, шумні вимірювання  $y_i$  будуть відповідати по-піксельному нормальному розподілу:

$$p(\mathbf{y}_i | \mathbf{x}_i, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\mathbf{y}_i - \mathbf{x}_i)^2}{2\sigma^2}} \quad (6.3)$$

де  $i = 1 \dots n$  та  $n$  - кількість пікселів. Після додавання шуму до зображення необхідно також виконати по-піксельне обрізання так, щоб  $x_i \in [0, 1]$ , тобто значення не виходили за межі оригінального зображення.

### 6.2 Оцінка зображень

Для аналізу отриманих результатів необхідно мати метрики того, наскільки зображення оброблені алгоритмом видалення шуму, відрізняються від оригінальних зображень. Очевидним варіантом є використання середньо квадратичної похибки, яка також використовувалась у штрафній функції нейронної мережі.

$$MSE(x, y) = \frac{1}{n^2} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} [x(i, j) - y(i, j)]^2 \quad (6.4)$$

де  $x$  та  $y$  відповідає зображеннями розміром  $n \times n$ .

Однак, зазвичай, така метрика є не найкращим відображення людського сприйняття зображень. Більш об'єктивною альтернативою є SSIM (structural similarity index measure) метрика яка була представлена в роботі [4]. Вона дає можливість краще оцінити схожість двох зображень  $x$  та  $y$  оперуючись на

різницю в структурі всього зображення, а не окремих пікселів. SSIM метрика задається формулою 6.5.

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad (6.5)$$

де

- $\mu_x$  середнє значення  $x$
- $\mu_y$  середнє значення  $y$
- $\sigma_x^2$  дисперсія  $x$
- $\sigma_y^2$  дисперсія  $y$
- $\sigma_{xy}$  коваріація  $x$  та  $y$
- $c_1 = (k_1L)^2$ ,  $c_2 = (k_2L)^2$  змінні для стабілізації ділення
- $L$  динамічний діапазон пікселів
- $k_1 = 0.01$  та  $k_2 = 0.03$  - константи.

Значення цієї функції змінюється в діапазоні  $[-1, 1]$ , де 1 можна отримати у випадку коли зображення співпадають.

## 7 Реалізація моделі для автоенкодера

### 7.1 Датасет

Тренувати модель будемо використовуючи MNIST датасет який складається з 70 тис. тренувальних рукописних цифр. Всі зображення є чорно-білими та розмір кожного із них складає  $28 \times 28$  пікселів, а значення лежать в проміжку від  $[0, 255]$ . Для пришвидшення тренування дані були нормалізовані до діапазону  $[0, 1]$ . Весь датасет був поділений на частину тренувальних та тестових прикладів в пропорції 90/10.

### 7.2 Архітектура автоенкодера

Для роботи автоенкодера зазвичай достатньо мати по одному шару енкодера та декодера. Однак, глибокі автоенкодери мають додаткові переваги і можуть вивчати більш складні взаємозв'язки даних. Можна збільшувати як глибину енкодера для покращення представлення даних, так і декодера для покращення їх відтворення. Однак, збільшення кількості шарів нейронної мережі може суттєво вплинути на час її тренування і тому необхідно вибирати їх оптимальну кількість.

Важливим є також вибір розмірності кожного з шарів автоенкодера. Вхідний шар завжди має розмірність зображення, тобто для випадку нашого датасету це 784. Кількість нейронів у вихідному шарі має відповідати кількості нейронів вхідного адже ми прагнемо відновити чисте зображення. Для прихованих шарів автоенкодера, зазвичай вибирають розмір між значеннями вхідного та вихідного шару.

Ключовим гіперпараметром автоенкодера є розмір репрезентаційного шару. Від нього залежить наскільки енкодер буде компресувати вхідну інформацію, що прямо впливає на можливість її подальшого відтворення. Очевидно, що при малому розмірі цього шару модель декодер буде мати менше деталей для відтворення зображення, і як наслідок, буде давати гірші результати. Однак якщо зробити його розмірність занадто великою автоенкодер може просто навчитись копіювати свої вхідні дані у вихідні, не вивчаючи властивостей даних. Такий автоенкодер буде точно реконструювати навчальні дані, але через перетренованість він не зможе коректно працювати з новими прикладами.

Отже відповідно до наведених вище критеріїв була побудована наступна архітектура моделі автоенкодера, яка задана таблицею 1.



Шар мережі (активаційна функція)	Розмірність
Енкодер	
Dense (Relu)	$784 \times 64$
Dense (Relu)	$64 \times 32$
Декодер	
Dense (Sigmoid)	$32 \times 784$

Табл. 1: Архітектура щільної нейронної мережі для автоенкодера.

Як можна побачити для енкодера було вибрано два шари, а для декодера один. Така модель була вибрана емпірично і, як можна буде побачити далі, вона демонструє досить непогані результати по відтворенню зображень. Репрезентаційний шар складається з 32 нейронів що виявилось оптимальним значенням за критерієм точності відтворення даних та часу витраченого на тренування.

### 7.3 Тренування

Всього були натреновані п'ять моделей для різних значень стандартного відхилення білого гаусівського шуму. Для тренування кожної з них використовувався однаковий набір гіперпараметрів які задані в таблиці 7.3.

Кількість ітерацій	Розмір minibatch частини	Швидкість навчання
35	128	0.015

Окрім звичайної штрафної функції, під час тренування також використовувалась SSIM оцінка для аналізу ефективності відтворення зображень на тестових даних що продемонстровано на графіку 7.1.

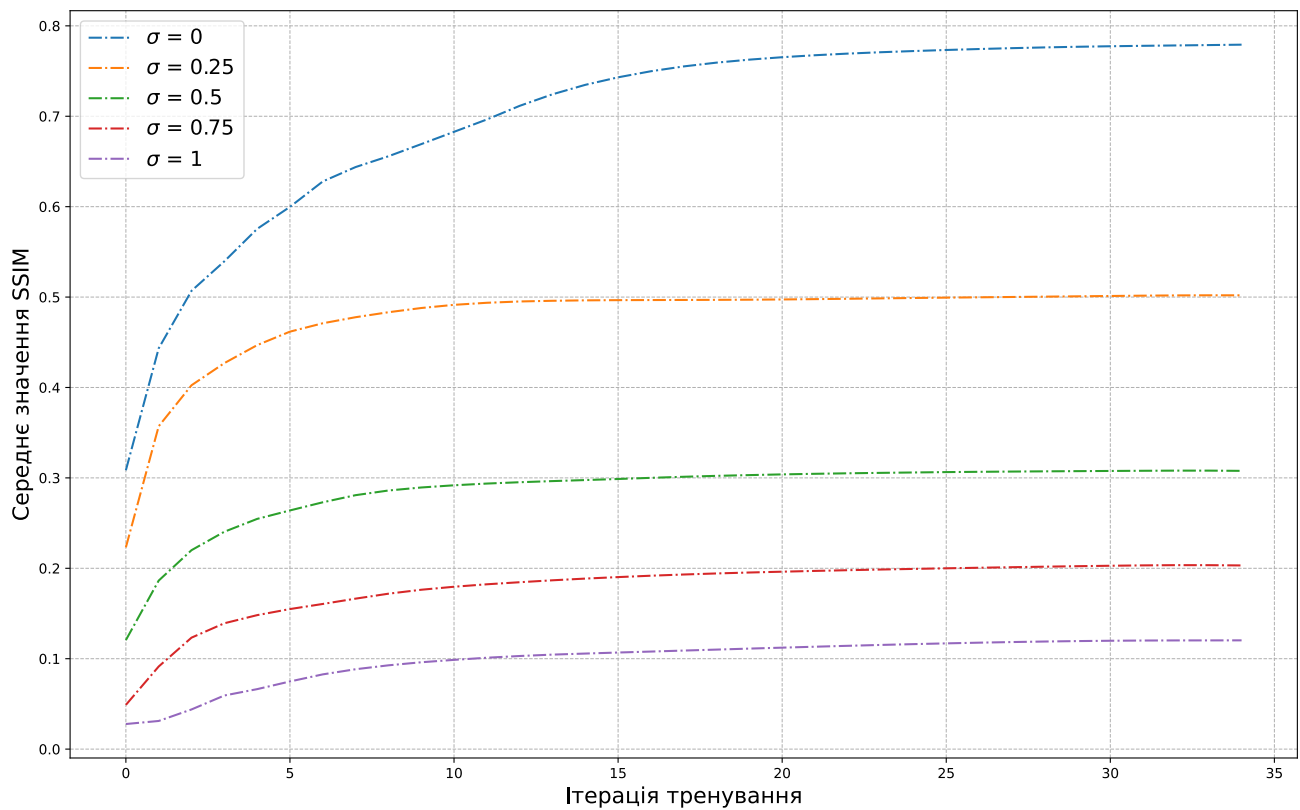


Рис. 7.1: Графік залежності усередненої SSIM оцінки для тестового датасету від кількості ітерацій тренування.  $\sigma$  відповідає середньоквадратичному відхиленню гаусівського шуму.

Для порівняння був також розглянутий варіант коли  $\sigma = 0$ . В такому випадку шум у зображення не додавався. Як можна бачити, існує чітка залежність між рівнем шуму доданим до чистих зображень та ефективністю його відтворення автоенкодером. Для  $\sigma = 0$ , середнє значення SSIM оцінки активно прямує до 0.8, коли для  $\sigma = 1$  вона залишається в околі 0.1.

## 8 Аналіз результатів

TODO Для порівняння результатів роботи автоенкодера з класичними методами був також реалізований алгоритм регуляризації Тіхонова з використанням градієнтним пріором в якості регуляризатора.

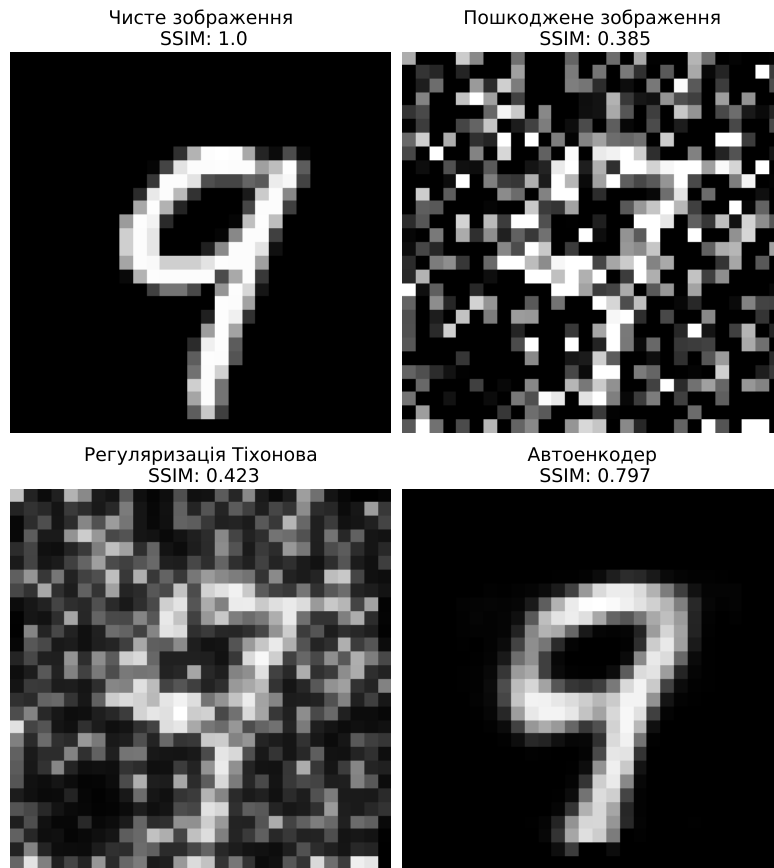


Рис. 8.1:

TODO Як можна бачити

# Висновок

TODO

## Додатки

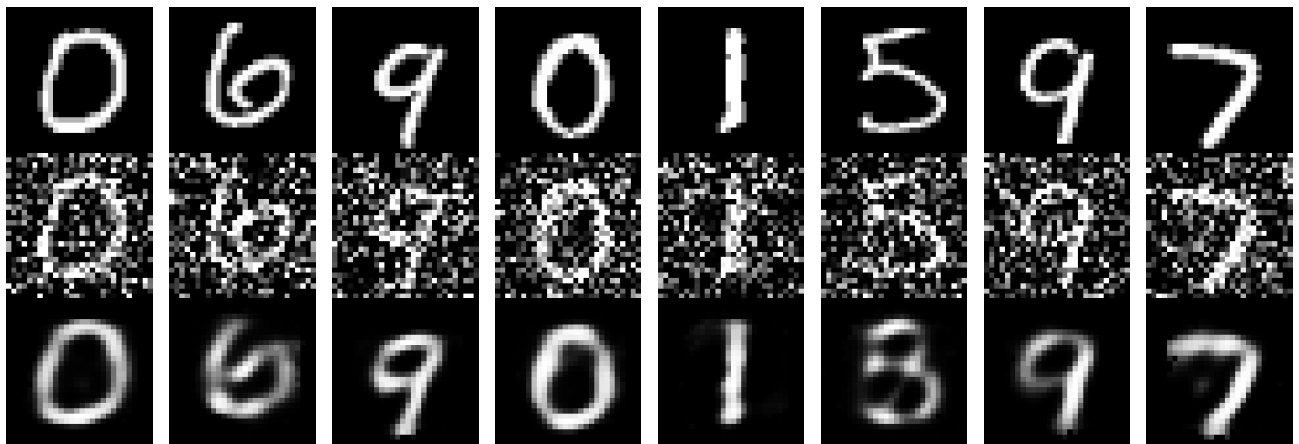


Рис. 8.2: Приклад видалення шуму за допомогою автоенкодера. Перший ряд це чисті зображення, другий - пошкодженні, третій - відновленні.

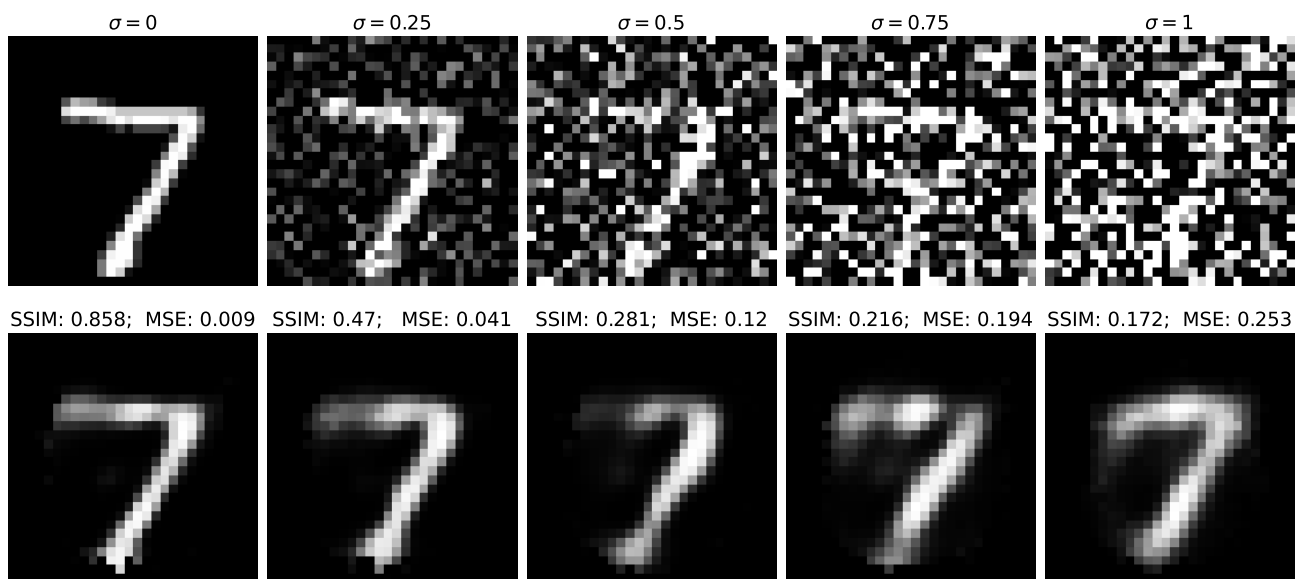


Рис. 8.3: Порівняння тоїності реконструкції зображень автоенкодером для різної величини стандартного відхилення  $\sigma$  білого шуму Гауса.

# Література

- [1] Gregory Ongie та ін. *Deep Learning Techniques for Inverse Problems in Imaging*. 2020. arXiv: 2005.06001 [eess.IV].
- [2] David E. Rumelhart, James L. McClelland та CORPORATE PDP Research Group, ред. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986. ISBN: 026268053X.
- [3] Ian Goodfellow, Yoshua Bengio та Aaron Courville. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016.
- [4] Zhou Wang та ін. “Image quality assessment: from error visibility to structural similarity”. в: *IEEE Transactions on Image Processing* 13.4 (2004), с. 600—612. DOI: 10.1109/TIP.2003.819861.
- [5] Jonas Adler та Ozan Öktem. “Solving ill-posed inverse problems using iterative deep neural networks”. в: *Inverse Problems* 33.12 (листоп. 2017), с. 124007. ISSN: 1361-6420. DOI: 10.1088/1361-6420/aa9581. URL: <http://dx.doi.org/10.1088/1361-6420/aa9581>.
- [6] Junyuan Xie, Linli Xu та Enhong Chen. “Image Denoising and Inpainting with Deep Neural Networks”. в: *Advances in Neural Information Processing Systems*. за ред. F. Pereira та ін. т. 25. Curran Associates, Inc., 2012. URL: <https://proceedings.neurips.cc/paper/2012/file/6cdd60ea0045eb7a6ec44c5Paper.pdf>.
- [7] Ankit Raj, Yuqi Li та Yoram Bresler. *GAN-based Projector for Faster Recovery with Convergence Guarantees in Linear Inverse Problems*. 2019. arXiv: 1902.09698 [cs.LG].
- [8] Hemant K. Aggarwal, Merry P. Mani та Mathews Jacob. “MoDL: Model-Based Deep Learning Architecture for Inverse Problems”. в: *IEEE Transactions on Medical Imaging* 38.2 (лют. 2019), с. 394—405. ISSN: 1558-254X. DOI: 10.1109/tmi.2018.2865356. URL: <http://dx.doi.org/10.1109/TMI.2018.2865356>.
- [9] Diederik P Kingma та Max Welling. *Auto-Encoding Variational Bayes*. 2014. arXiv: 1312.6114 [stat.ML].