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

Вступ

Оберненими задачами називають такі задачі, в яких необхідно відновити дані про деякий процес з використанням непрямих спостережень. Такі спостереження отримують за допомогою прямого процесу який, зазвичай, є необоротнім. Як наслідок, задачі такого типу можуть бути нерозв'язними або не мати єдиного розв'язку 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, таких як реконструкція знімків, видалення шуму, збільшення розмірності, заповнення втрачених даних та багато інших.

Класичні методи до розв'язання таких задач припускають 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 наявні спостереження, а також, є ймовірним враховуючи попередні знання.

Хоча такий підхід широко використовується, за останній час алгоритми глибокого навчання для розв'язування 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, спробуємо проаналізуємо їх ефективність та порівняти з класичними методами.

1 Постановка задачі

Оберненими задачами будемо вважати такі задачі, в яких невідомим є $n-$ піксельне зображення $\boldsymbol{x} \in \mathbb{R}^n$ яке було отримане з m вимірювань $\boldsymbol{y} \in \mathbb{R}^m$ відповідно до рівняння 1.1.

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

де $\mathcal{A}$ - це прямий оператор вимірювання та $\boldsymbol{\varepsilon}$ є певним вектором шуму. Метою задачі є відновлення x з y. Можна розглянути більш загальний випадок моделі неадитивного шуму, який має вигляд 1.2.

$$\boldsymbol{y} = \mathcal{N}(\mathcal{A}(\boldsymbol{x})) \tag{1.2}$$

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 x розв'язок задачі існує.*
2. *Розв'язок є єдиний для кожного x.*
3. *Розв'язок є стійкий до малих варіацій величини x, тобто достатньо малим зміненням величини x відповідають як завгодно малі зміни величини 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, який задовольняє спостереженням, є складною або неможливою за умови відсутності попередніх знань про дані.

Оцінку справжнього зображення $\boldsymbol{x}$ з $\boldsymbol{y}$ 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 $\boldsymbol{x} \in \mathbb{R}^n$ з вимірювань $\boldsymbol{y} \in \mathbb{R}^m$ у вигляді $\boldsymbol{y} = \mathcal{A}(\boldsymbol{x}) + \boldsymbol{\varepsilon}$.

Якщо розподіл шуму відомий, x можна відновити розв'язавши задачу оцінки максимальної ймовірності (maximum likelihood) 2.1.

$$\hat{\boldsymbol{x}}_{\mathrm{ML}} = \arg\max_{\boldsymbol{x}} p(\boldsymbol{y}|\boldsymbol{x}) = \arg\min_{\boldsymbol{x}} -\log p(\boldsymbol{y}|\boldsymbol{x}) \tag{2.1}$$

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

В залежності від умов задачі, можуть бути відомі попередні дані про те яким має бути x. Ці умови можна використати для формулювання задачі оцінки максимальної апостеріорної ймовірності (maximum a posteriori), що приводить до задачі 2.2.

$$\hat{\boldsymbol{x}}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{x}} p(\boldsymbol{x}|\boldsymbol{y}) = \arg - \max_{\boldsymbol{x}} p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x}) = \arg\min_{\boldsymbol{x}} -\ln p(\boldsymbol{y}|\boldsymbol{x}) - \ln p(\boldsymbol{x}) \tag{2.2}$$

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

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

де $|\mathcal{A}(\boldsymbol{x}) - \boldsymbol{y}\|_2^2$ відповідає за правдивість даних та позначає різницю між вихідним та шумним зображеннями, $\mathrm{R}(\boldsymbol{x})$- від'ємний логарифмічний пріор який позначає член регуляризації, а λ є параметром регуляризації. Для варіаційних методів видалення шуму, ключовим є пошук відповідного пріору зображення $\mathrm{R}(\boldsymbol{x})$. 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 з використанням параметра регуляризації $\mathrm{R_{TR}}$ за L_2 нормою, який можна подати у вигляді 2.4.

$$\mathrm{R_{TR}}(\boldsymbol{x}) = \|\nabla \boldsymbol{x}\|_2 = \sqrt{|\nabla_v x|^2 + |\nabla_h x|^2} \tag{2.4}$$

де $\nabla_h \boldsymbol{x}$ та $\nabla_v \boldsymbol{x}$ 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 Огляд 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 оберенне значення x. Цей підхід є доволі ефективним, однак є чутливим до змін в операторі вимірювання $\mathcal{A}$.

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 і ця робота націлена саме на методи контрольованого навчання, тому що очікується, що вони дадуть кращі результати в порівнянні з класичними методами.

3.2 Класифікація 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 $\mathcal{A}$ 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. До прикладів можна віднести задачу рентгенівських перетворень в комп'ютерній томографії.

- **Пряма модель $\mathcal{A}$ є відома тільки під час тестування**
  Такі алгоритми корисні для навчання моделей загального призначення. 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 $\mathcal{A}$ є відома тільки частково**
  Така ситуація можлива, наприклад, у випадку коли пряма модель залежить від деяких параметрів, для яких ми знаємо або розподіл, або достатню статистику.

- **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.

4 Автоенкодер для розв'язування обернених задач

Вперше автоенкодер був представлений в роботі [5] 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 Ian Goodfellow та ін. в [2]. Зокрема, автоенкодер є досить ефективною моделлю для розв'язування обернених задач.

4.1 Автоенкодер

Автоенкодером називають нейрону 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 нейрону 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.

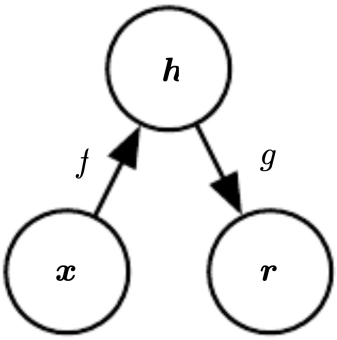

Рис. 4.1: Загальна структура автокодера, що відображає вхід x на вихід r (реконструкцію) через внутрішнє представлення h. Автокодер складається з двох компонентів: енкодера f (відображення x до h) та декодера g (відображення h до r) [2]

Якщо автоенкодеру вдається навчитися просто відтворювати $g(f(x)) = x$ для всіх прикладів, то це, зазвичай, не має особливої користі. Тому автоенкодери часто 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:

$$L(\boldsymbol{x}, g(f(\boldsymbol{x}))) \tag{4.1}$$

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

$$L(\boldsymbol{x}, g(f(\tilde{\boldsymbol{x}}))) \tag{4.2}$$

де $\tilde{\boldsymbol{x}}$ є копією $\boldsymbol{x}$ 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.

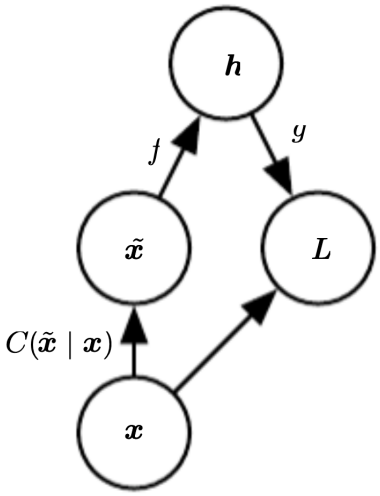

Рис. 4.2: Структура функції витрат для автоенкодера який навчається реконструювати чисті зображення x з пошкоджених. Тренування виконується на основі мінімізації функції втрат: $L = -\log p(\boldsymbol{x}|\boldsymbol{h} = f(\tilde{\boldsymbol{x}}))$, де $\tilde{\boldsymbol{x}}$ - пошкоджена версія прикладу $\boldsymbol{x}$, отримана в результаті деякого процесу руйнування $C(\tilde{\boldsymbol{x}}|\boldsymbol{x})$. [2]

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:

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

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

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

де

- x - матриця в якій кожен $i-$ий стовпець є розгорнутим у вектор чистим зображенням, $i = 1, \dots, m$
- y - матриця в якії кожен $i-$ий стовпець є розгорнутим у вектор пошкодженим зображенням, $i = 1, \dots, m$
- 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 має вигляд 5.1.

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

де

- l - номер шару нейронної мережі, при $l = 1 \dots L$
- $n^{[l]}$ - 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 $(n^{[l]} \times n^{[l-1]})$

- σ - 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 5.2 як середньоквадратичну похибку між активаціями останнього шару $a^{(L)}$ та справжніми зображеннями y.

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

де, функцією втрати для одного набору елементів визначимо половину евклідової відстані 5.3.

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

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 функції реконструкції:

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

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

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

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, які будемо застосовувати далі:

- **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)

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

Тепер можемо записати алгоритм зворотного поширення з використанням стохастичного градієнтного спуску та minibatch оптимізації 1.

Algorithm 1 Градієнтний спуск

Input: Тренувальні дані x, y, гіперпараметри: N, M, ε, α
Output: Оптимальні параметри моделі w, b
$i = 0$
while i $< N$ or ($\|d\omega\| < \varepsilon$ and $\|db\| < \varepsilon$) **do**
    $x^M \leftarrow$ випадкова група прикладів x, розміром M
    $y^M \leftarrow$ випадкова група прикладів y, розміром M
    for $l = 1$ to L **do**
        $w^{(l)} = w^{(l)} - \alpha \frac{\partial J(w, b, x^M, y^M)}{\partial w^{(l)}}$
        $b^{(l)} = b^{(l)} - \alpha \frac{\partial J(w, b, x^M, y^M)}{\partial b^{(l)}}$
        $i = i + 1$
    end for
end while
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 стандартного відхилення. Задамо загальну модель адитивного шуму як 6.1.

$$\mathbf{y} = \mathbf{x} + \varepsilon \tag{6.1}$$

де ε це шум який має нульове середнє значення та відповідає нормальному розподілу Гауса:

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

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 $y_1 \sim \mathcal{N}\left(\mu_1, \sigma_1^2\right)$ та $y_2 \sim \mathcal{N}\left(\mu_2, \sigma_2^2\right)$ також є розподілом Гауса $y_1 + y_2 \sim \mathcal{N}\left(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2\right)$. Таким чином, шумні вимірювання y_i будуть відповідати по-піксельному нормальному розподілу:

$$p\left(\mathbf{y}_i \mid \mathbf{x}_i, \sigma\right) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\left(\mathbf{y}_i - \mathbf{x}_i\right)^2}{2\sigma^2}} \tag{6.3}$$

де $i = 1 \ldots n$ 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 використовується у штрафній функції нейронної мережі.

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

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 [6]. Вона дає можливість краще оцінити схожість двох зображень x та y опираючись на

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

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

де

- μ_x середнє значення x
- μ_y середнє значення y
- σ_x^2 дисперсія x
- σ_y^2 дисперсія y
- σ_{xy} коваріація x та y
- $c_1 = (k_1 L)^2, c_2 = (k_2 L)^2$ 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 однакові.

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 28×28 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 та можуть вивчати складніші 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 випаду нашого датасету це 784. Кількість нейронів у вихідному шарі має відповідати кількості нейронів вхідного, тому що ми прагнемо відновити чисте зображення. Для прихованих шарів автоенкодера, зазвичай вибирають розмір між значеннями вхідного та вихідного шарів.

Ключовим гіперпараметром автоеркодера можна вважати 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.

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

Шар мережі (активаційна функція)	Розмірність
Енкодер	
Dense (Relu)	784×64
Dense (Relu)	64×32
Декодер	
Dense (Sigmoid)	32×784

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 заданий 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.

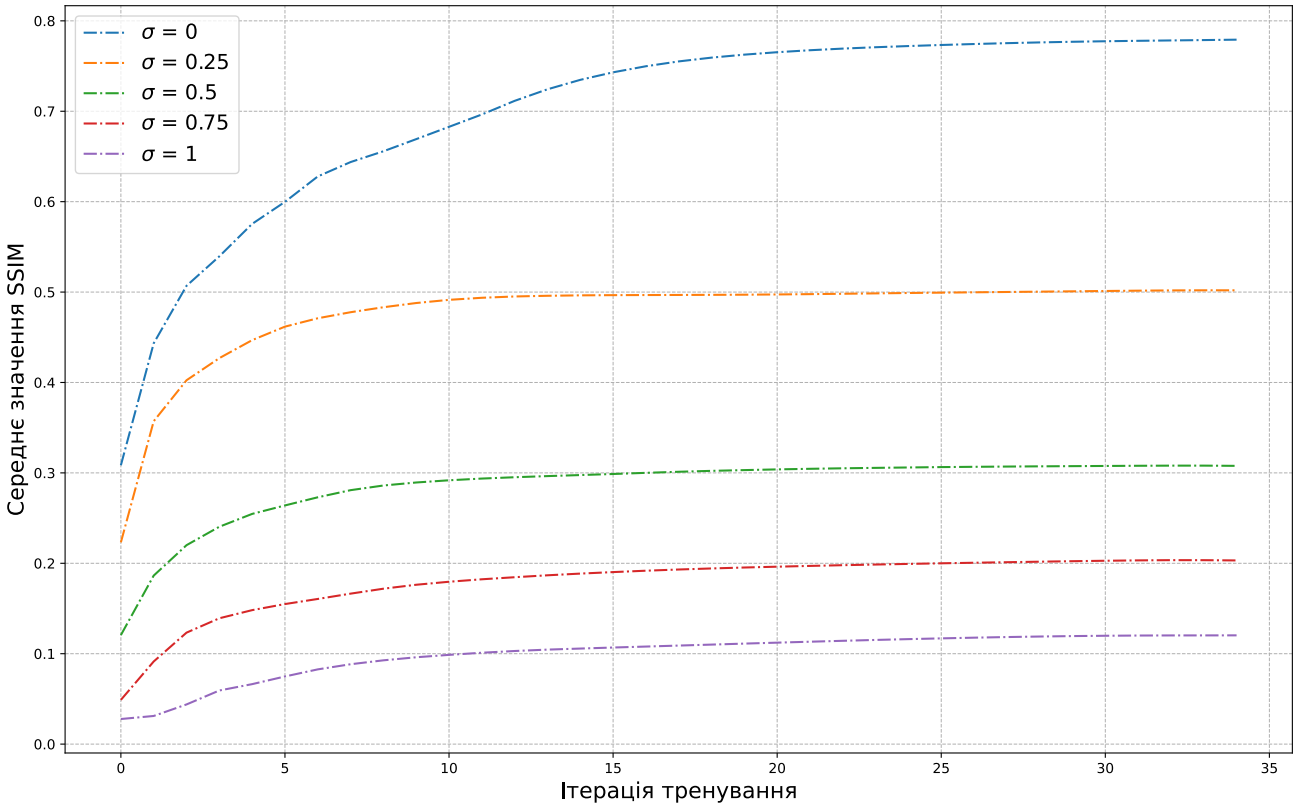

Рис. 7.1: Графік залежності усередненої SSIM оцінки для тестового датасету від кількості ітерацій тренування. σ відповідає середньоквадратичному відхиленню гаусівського шуму.

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 прямує до 0.8, коли для $\sigma = 1$ вона залишається в околі 0.1.

8 Аналіз результатів

Результати натренованих моделей автоенкодера для видалення шуму можна бачити на зображенні 8.1.

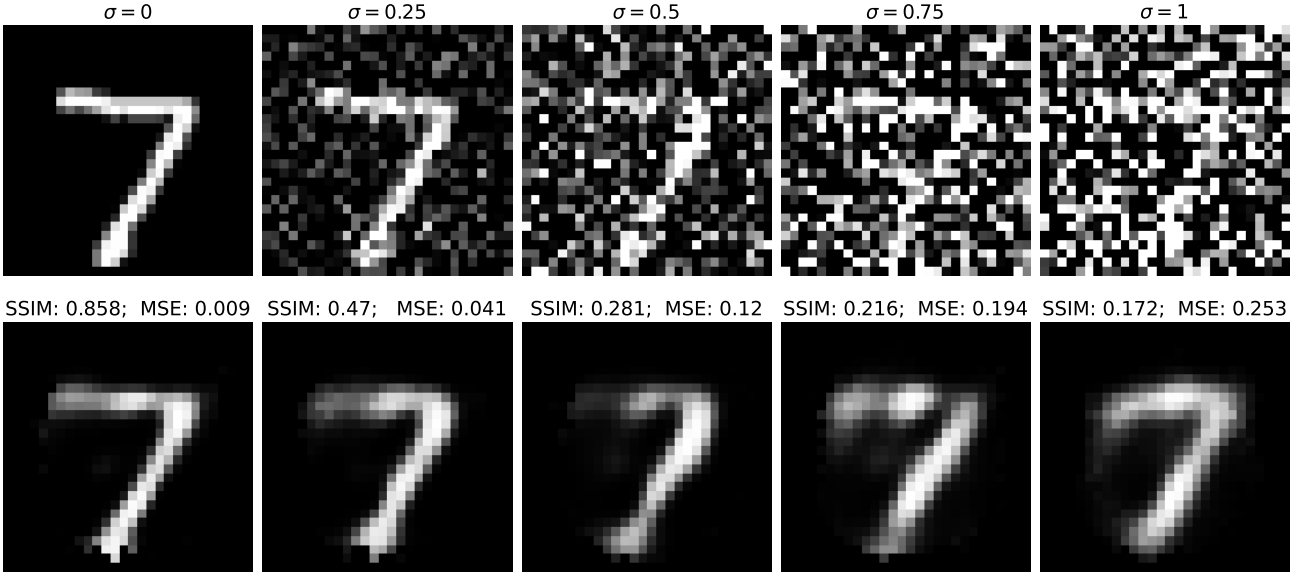

Рис. 8.1: Порівняння точності реконструкції зображень автоенкодером для різної величини стандартного відхилення σ білого шуму Гауса.

Очевидно, що при збільшенні кількості шуму, відтворенні зображення втрачають в чіткості та схожості з оригіналом. Залежність MSE похибки від σ є прямою, а для SSIM оцінки - навпаки оберненою.

Можна сказати, що навіть при дуже значній кількості шуму як $\sigma = 1$, відтворене зображення все одно залишається читабельним, на відмінно від пошкодженого варіанту. З цього можна зробити висновок, що автоенкодер досить ефективно справляється з задачею по видаленню шуму, при достатній кількості тренувальних прикладів.

Для порівняння результатів роботи автоенкодера з класичними методами, був також реалізований алгоритм регуляризації Тіхонова, де для регуляризації використовувався градієнтний пріор. На зображенні 8.2 продемонстрований результат роботи обох алгоритмів для видалення шуму заданого стандартним відхиленням $\sigma = 0.5$. Метод регуляризації суттєво покращив середньоквадратичну похибку, але загалом зображення залишилось сильно пошкодженим. З іншого боку, зображення відновлення за допомогою глибокої нейронної мережі демонструє непогані результати для обох наведених метрик.

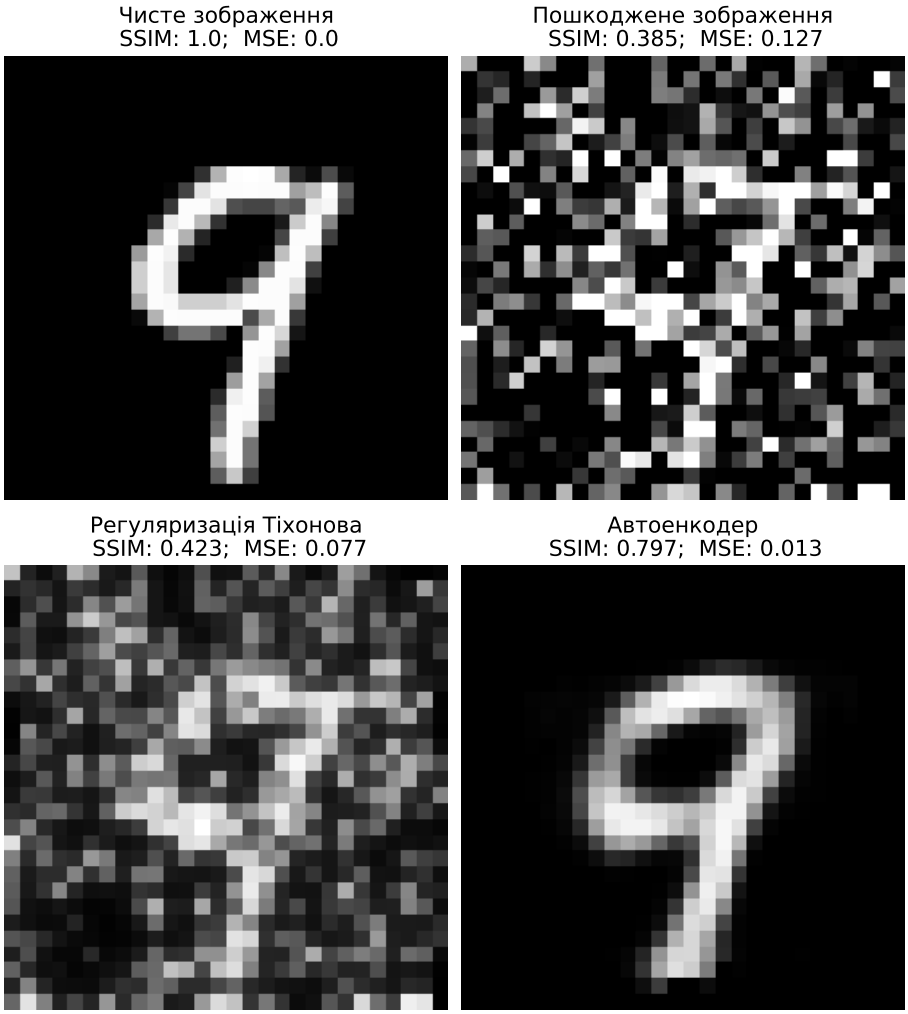

Рис. 8.2: Порівняння видалення шуму за допомогою автоенкодера з класичним методом основаним на регуляризації.

Висновок

В ході цієї роботи...

Додатки

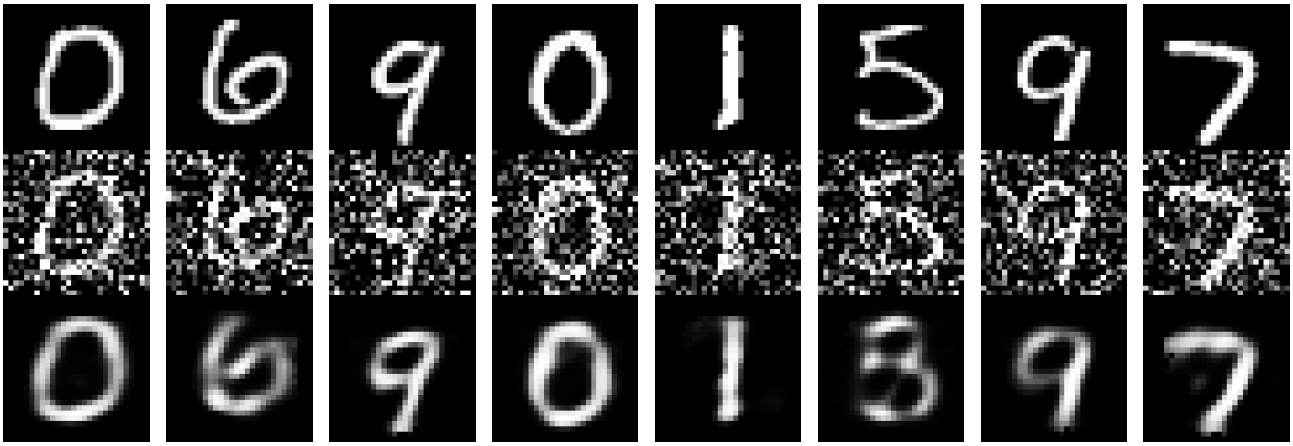

Рис. 8.3: Приклад видалення шуму за допомогою автоенкодера. Перший ряд це чисті зображення, другий - пошкодженні, третій - відновленні. $\sigma = 0.5$

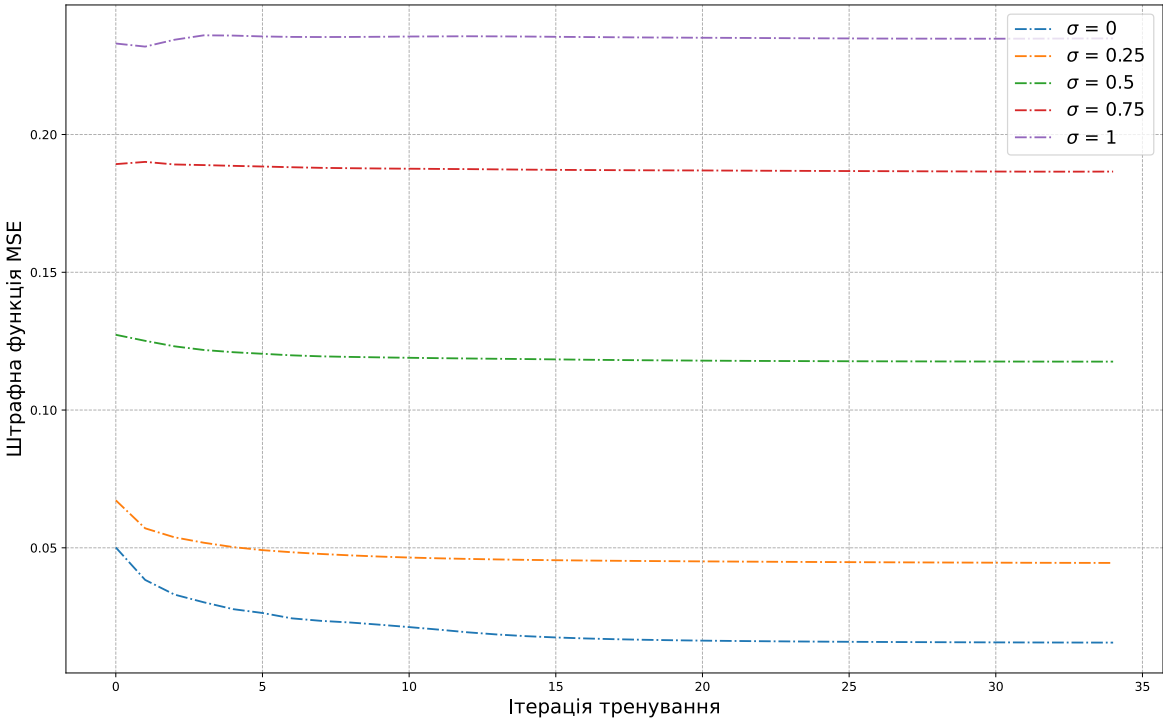

Рис. 8.4: Графік штрафної функції MSE для тренувального датасету від кількості ітерацій тренування. σ відповідає середньоквадратичному відхиленню гаусівського шуму.

Література

[1] Gregory Ongie та ін. *Deep Learning Techniques for Inverse Problems in Imaging*. 2020. arXiv: 2005.06001 [eess.IV].

[2] Ian Goodfellow, Yoshua Bengio та Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[3] 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.

[4] 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/6cdd60ea0045eb7a6ec44c54d29ed402-Paper.pdf.

[5] 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.

[6] 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.

[7] Ajay Kumar Boyat та Brijendra Kumar Joshi. *A Review Paper: Noise Models in Digital Image Processing*. 2015. arXiv: 1505.03489 [cs.CV].

[8] 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].

[9] Weisheng Dong та ін. "Denoising Prior Driven Deep Neural Network for Image Restoration". в: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.10 (жовт. 2019), с. 2305—2318. ISSN: 1939-3539. DOI: 10.1109/tpami.2018.2873610. URL: http://dx.doi.org/10.1109/TPAMI.2018.2873610.

[10] Kui Liu, Jieqing Tan та Benyue Su. "An Adaptive Image Denoising Model Based on Tikhonov and TV Regularizations". в: (серп. 2014), с. 2305—2318. DOI: 10.1155/2014/934834. URL: https://doi.org/10.1155/2014/934834.