ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ФРАНКА
Факультет прикладної математики та інформатики
Кафедра обчислювальної математики

Кусова 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 допомого певного прямого процесу який, зазвичай, є необоротнім, 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 сейсморозвідки на основі звукових сигналів або багато задач із зображеннями, такі як реконструкція рентгенівської або акустичної томографії, видалення шуму, збільшення розмірності, заповнення втрачених даних в зображеннях та інші.

Класичний підхід до розв'язання таких задач припускає наявність певної попередньої інформації про обернену задачу на основі якої будується прямий оператор та функціонал регуляризації з яких формується задача мінімізації.

Однак, за останій час алгоритми глибокого навчання набирають значну популярність в області розв'язуванні обернених задач через свою ефективність, та універсальність для багатьох різних типів задач [1].

Отже в межех цієх роботи будемо розглядати деякі методи по реконструкції зображень на основі глибокого навчання та проаналізуємо їх ефективність.

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

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

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

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

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, яка 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):

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

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 для формування задачі оцінки максимальної апостеріорної ймовірності (maximum a posteriori), що приводить до задачі 2.1.

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

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

$$\hat{x} = \arg\min_{x} \frac{1}{2}\|\mathcal{A}(\boldsymbol{x}) - \boldsymbol{y}\|_2^2 + \lambda \mathrm{R}(\boldsymbol{x}) \tag{2.2}$$

де $|\mathcal{A}(\boldsymbol{x}) - \boldsymbol{y}\|_2^2$ відповідає за правдивість даних та позначає різницю між вихідним та шумним зображеннями, $\mathrm{R}(\boldsymbol{x})$ є пропорційним до від'ємного логарифмічного TODO пріора та 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 параметра регуляризації R_{TR} за L_2 нормою, який можна подати у вигляді 2.3.

$$\mathrm{R}_{\mathrm{TR}}(\boldsymbol{x}) = \|\nabla \boldsymbol{x}\|_2 = \sqrt{|\nabla_v x|^2 + |\nabla_h x|^2} \tag{2.3}$$

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 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 оберенне 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

TODO Gregory Ongie та ін Відповідно до [1] існує багато варіантів застосування глибокого навчання для розв'язування обернених задач. Більшість з них можна поділита на декілька груп:

- **Пряма модель $\mathcal{A}$ є відома під час тренування та тестування**
  Для цьогу 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 $\mathcal{A}$ є відома тільки під час тестування**
  Для таких алгоритмів характерно, що якщо один раз навчити глибоку моделі, 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 нейроні мережі для різних прямих моделей є недоцільно.

- **Пряма модель $\mathcal{A}$ є відома тільки частково**
  Це може статися, наприклад, 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, не-

правильно визначеною або обчислювально неможливою для використання в навчанні та тестуванні. Тоді навчання може відбуватись лише з відповідними парами зображень та вимірювань.

3.3 Sample Complexity vs. Generality?

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 нейрону мережу яка навчається копіювати свої вхідні дані у вихідні. Така мережа має проміжний шар h, 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.

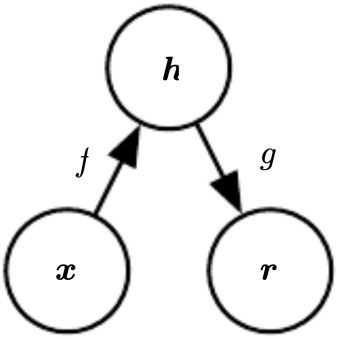

Рис. 4.1: Загальна структура автокодера, що відображає вхід x на вихід r (реконструкцію) через внутрішнє представлення h. Автокодер складається з двох компонентів: енкодера f (відображення x до h) та декодера g (відображення h до r) [3]

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 даним.

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 відображення 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 заданий на 4.2.

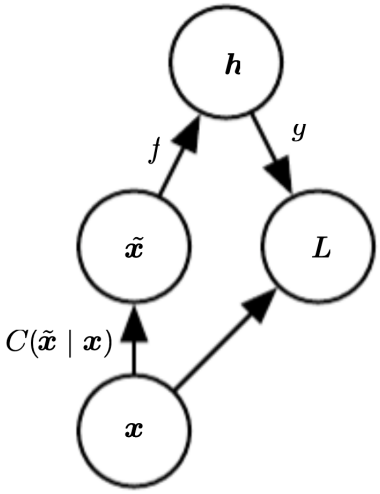

Рис. 4.2: Структура функції витрат для автоенкодера який навчається реконструювати чисті зображення x з пошкоджених. Тренування виконується на основі мінімізації функції втрат: $\tilde{x}$. $L = -\log p(\boldsymbol{x}|\boldsymbol{h} = f(\tilde{\boldsymbol{x}}))$, де $\tilde{\boldsymbol{x}}$ - пошкоджена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

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

TODO Автоенкодер доцільно використовувати у випадку коли 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 справжнього зображення, $i = 1, \dots, m$
- y - матриця в якії 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 нейроної мереді буде мати 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, 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 $(n^{[l]} \times n^{[l-1]})$

- 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 як середньо квадратичну 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 витрати для одного набору елементів визначимо половину евклідової відстані.

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

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 функці реконструкції:

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

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

$$da^{(L,i)} = \frac{\partial L(a^{(L,i)}, y^{(i)})}{\partial a^{(L,i)}} = \sum_{j=1}^{n}(a_j^{(L,i)} - y_j^{(i)})$$

$$dz^{(L,i)} = \frac{\partial L(a^{(L,i)}, y^{(i)})}{\partial z^{(L,i)}} = \frac{\partial L(a^{(L,i)}, y^{(i)})}{\partial a^{(L,i)}}\frac{\partial a^{(L,i)}}{\partial z^{(L,i)}} = \sum_{j=1}^{n}(a_j^{(L,i)} - y_j^{(i)})\sigma'(z^{(L,i)});$$

$$dw^{(L,i)} = \frac{\partial L(a^{(L,i)}, y^{(i)})}{\partial w^{(L,i)}} = dz^{(L,i)}\frac{\partial z^{(L,i)}}{\partial w^{(L,i)}} = dz^{(L,i)}a^{(L-1,i)};$$

$$db^{(L,i)} = \frac{\partial L(a^{(L,i)}, y^{(i)})}{\partial b^{(L,i)}} = dz^{(L,i)}\frac{\partial z^{(L,i)}}{\partial b^{(L,i)}} = dz^{(L,i)}$$

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

$$\frac{\partial J(w,b)}{\partial W^{(l)}} = \frac{1}{m}\sum_{i}^{m} \delta^{(l,i)} \left(a^{(l-1,i)}\right)^{\top}$$

$$\frac{\partial J(w,b)}{\partial b^{(l)}} = \frac{1}{m}\sum_{i}^{m} \delta^{(l,i)}$$

де

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

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

- **Sigmoid** (Logistic function)

$$\begin{aligned} \sigma(z) &= \frac{1}{1+e^{-z}} \\ \sigma'(z) &= a(1-a) \end{aligned} \tag{5.5}$$

- **ReLU** (Rectified Linear Units)

$$\begin{aligned} \sigma(z) &= \max(0,z) \\ \sigma'(z) &= \begin{cases} 0, z < 0 \\ 1, z \geq 0 \end{cases} \end{aligned} \tag{5.6}$$

- **Tanh** (Hyperbolic tangent)

$$\begin{aligned} \sigma(z) &= \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}} \\ \sigma'(z) &= 1-a^2 \end{aligned} \tag{5.7}$$

Далі запишемо алгоритм заворотнього 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.

5.3 Датасет

TODO

6 Реалізація та аналіз

6.1 Герерація шуму

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

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

де ε це шум який має нульове середє 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 використовувалась у штрафній функції нейронної мережі.

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

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

Однак, зазвичай, така метрика є не найкращим відображення людського сприйнятя зображень. Більш об'єктивною альтернативоє є SSIM (structural similarity index measure) метрика яка була предстадставлена в роботі [4]. Вона дає можливіть краще оцінити схожість двох зображень x та y операючись на

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

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

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 вападку коли зображення однакові.

6.3 Тренування

TODO

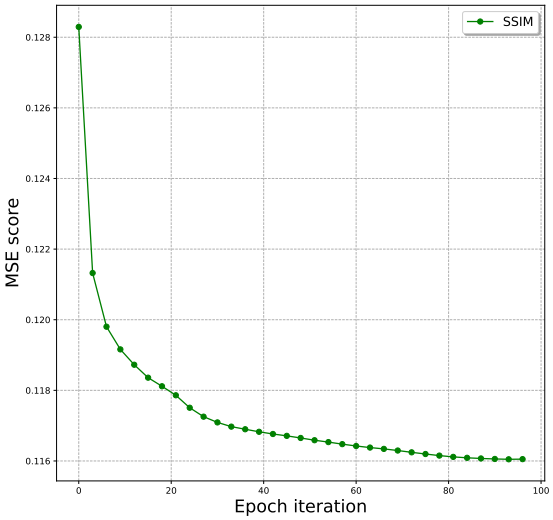

Рис. 6.1: Графік залежності штрафної функції MSE на тестовому датасеті від кількості епох. тренування.

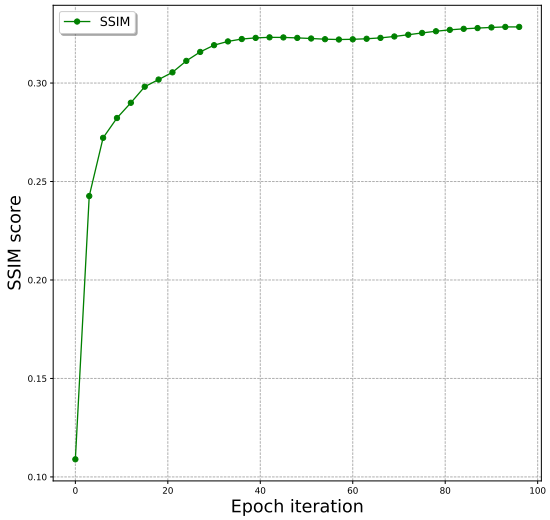

Рис. 6.2: Графік залежності оцінки SSIM на тестовому датасеті від кількості епох тренування.

6.4 Аналіз результатів

TODO

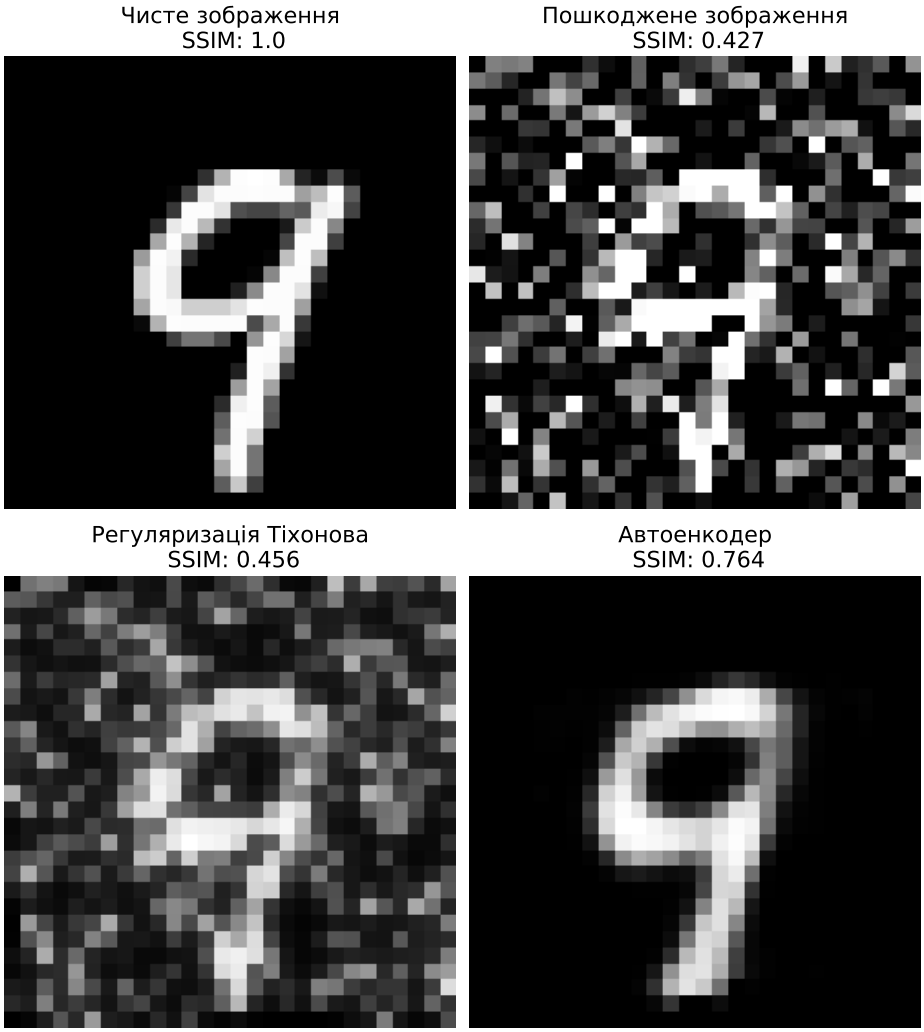
Чисте зображення
SSIM: 1.0
Пошкоджене зображення
SSIM: 0.427
Регуляризація Тіхонова
SSIM: 0.456
Автоенкодер
SSIM: 0.764

Рис. 6.3:

Висновок

TODO

Додатки

TODO

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].

https://arxiv.org/pdf/1312.6114.pdf https://arxiv.org/pdf/1505.03489.pdf ?