Міністерство освіти і науки України
Кам’янець-Подільський національний університет імені Івана Огієнка
Фізико-математичний факультет
Кафедра комп’ютерних наук

Кваліфікаційна робота
магістра
з теми «МЕТОДИ ТА ЗАСОБИ ОРГАНІЗАЦІЇ НЕЙРОМЕРЕЖ ДЛЯ ГЕНЕРАЦІЇ ЗОБРАЖЕНЬ»

Виконав: здобувач вищої освіти
2 курсу, групи KN1-M23
спеціальності 122 Комп’ютерні науки
Кондрушенко Микола Володимирович

Керівник: Пилипюк Тетяна Михайлівна,
кандидат фізико-математичних наук,
доцент, доцент кафедри комп’ютерних наук

Рецензент: Газдюк Катерина Петрівна,
доктор філософії з інженерії програмного забезпечення, доцент, в.о. зав. кафедри програмного забезпечення комп'ютерних систем Чернівецького національного університету імені Юрія Федьковича

м. Кам’янець-Подільський, 2024 р.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CNN (Convolutional Neural Network) – згорткова нейронна мережа

GAN (Generative Adversarial Network) – генеративно-змагальна мережа

VAE (Variational Autoencoder) – варіаційний автоенкодер

ViT (Vision Transformer) – трансформер для обробки зображень

SGD (Stochastic Gradient Descent) – стохастичний градієнтний спуск

MLP (Multi-Layer Perceptron) – багатошаровий персептрон

SGD (Stochastic Gradient Descent) – стохастичний градієнтний спуск

ШНМ – штучні нейронні мережі

ВСТУП

У сучасному світі нейронні мережі відіграють важливу роль у багатьох сферах життя. Однією з найбільш інноваційних та перспективних технологій є генеративні нейронні мережі, здатні створювати реалістичні зображення та інші форми контенту. З одного боку це може сприяти розв'язанню численних наукових завдань, а з іншого – розширювати творчі та креативні можливості художників, дизайнерів та розробників. Генеративні моделі можуть використовуватися для створення нових візуальних елементів, архітектурних прототипів, анімацій тощо.

Актуальність. Генеративні нейромережі використовуються, у тому числі, для створення високоякісних зображень у таких сферах, як реклама, кіно, ігри, віртуальна та доповнена реальність. Компанії застосовують їх для залучення додаткових ресурсів у сфері дизайну, це дозволяє їм значно скоротити витрати на створення маркетингових матеріалів та різного роду візуальних продуктів. Це робить дані нейромережі особливо актуальними для малого бізнесу. Також застосування даного типу нейромереж дозволяє значно скоротити час роботи із візуальною складовою маркетингу. Не менш значною перевагою є ширші можливості створення персоналізованого контенту, що значно підвищує привабливість товарів і послуг, що пливає на розвиток компаній.

Ще один важливий напрямок, де потенційно можуть використовуватися генеративні нейромережі, – медична галузь. Тут за їх допомогою можна створювати нові зображення медичних знімків для навчання моделей діагностики. Це особливо важливо для тренування нейромереж в умовах обмежених даних, адже часто важко чи неможливо отримати велику кількість реальних медичних знімків для обчислень. Завдяки такій генерації можна підвищити точність діагностики та досягти вищої ефективності медичних процедур.

Швидкий розвиток сфери машинного навчання неодмінно призводить до значного зростання потреб в обчислювальних потужностях. У свою чергу це призводить до появи нових апаратних рішень, таких як GPU більшої потужності та тензорні процесори TPU, які були спеціально розроблені для машинного навчання нейронних мереж компанією Google. Як ніколи стали актуальними дослідження в напрямку розробки оптимізованих моделей нейромереж, які б використовували мінімум ресурсів, забезпечуючи при цьому високу продуктивність роботи.

Загалом, актуальність обумовлена інноваційністю самої технології, її широкими можливостями для застосування в науковій та комерційній сферах, а також необхідністю вирішення важливих методологічних питань, пов’язаних із підвищенням ефективності, точності та якості генерації зображень.

Метою дослідження є аналіз та порівняння методів та засобів організації нейронних мереж, що використовуються для генерації зображень, а також розробка рекомендацій щодо підвищення ефективності їхнього використання.

Завдання дослідження:

- здійснити аналіз досліджень та публікацій;
- ознайомитися та здійснити вибір відповідних бібліотек;
- здійснити дослідження ШНМ для генерації зображень.

Дослідження теоретичних засад організації нейронних мереж включає дослідження математичних основ функціонування нейромереж, алгоритмів навчання, функцій активації, функцій втрат.

Аналіз сучасних підходів до організації архітектури нейромереж, призначених для роботи із зображеннями, зосереджено на згорткових нейронних мережах (CNN), глибоких нейромережах із залишковими блоками, трансформерах, які також застосовують елементи залишкових нейронних мереж, що найкраще зарекомендували себе у задачах генерації зображень.

Об’єктом дослідження є нейронні мережі, що використовуються для генерації зображень.

Предметом дослідження є методи та засоби організації нейронних мереж, їх архітектури, а також підходи до покращення якості згенерованих зображень та ефективності генерації.

Методи дослідження. Застосовані теоретичні та емпіричні методи дослідження, які включають аналіз наукової літератури, систематизацію та узагальнення інформації для визначення стану досліджуваної проблематики, проведення експериментальних досліджень, аналіз результатів.

Апробація результатів дослідження. Матеріали досліджень були оприлюднені на звітній науковій конференції за підсумками науково-дослідної роботи здобувачів вищої освіти фізико-математичного факультету Кам’янець-Подільського національного університету імені Івана Огієнка у 2023-2024 н.р., 9-10 квітня 2024 року [3] та на ІІ Міжнародній науково-практичній інтернет-конференції «Актуальні аспекти розвитку STEAM-освіти в умовах євроінтеграції» (м. Кропивницький, 26 квітня 2024 року) [5]. Статтю «Порівняльний аналіз нейромереж-трансформерів в задачах генерації зображень» опубліковано у Віснику Кам’янець-Подільського національного університету імені Івана Огієнка. Фізико-математичні науки. Випуск 17 [4].

Структура роботи. Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел, додатків.

1. ТЕОРЕТИЧНІ ЗАСАДИ ОРГАНІЗАЦІЇ НЕЙРОМЕРЕЖ

1.1. Математичні основи генеративних моделей

Генеративні моделі базуються на статистичних і математичних принципах, які дозволяють нейронним мережам генерувати дані, що нагадують реальні. Головними математичними основами є ймовірнісне моделювання, теорія оптимізації, принципи латентного простору та глибинне навчання.

Ймовірнісне моделювання є основою генеративних моделей, що дозволяє моделювати розподіл даних P(X) або спільний розподіл між спостережуваними та латентними змінами P(X, Z). Це дозволяє виявляти структуру даних, вираховувати невідомі параметри, і використовувати ці дані для створення нових зразків. Ймовірнісне моделювання намагається визначити розподіл даних P(X) через використання латентних змінних Z [6]:

$$P(X) = \int P(X, Z) dZ \tag{1.1}$$

де, Z – визначає приховані фактори, що впливають на X.

Оптимізація є ключовим поняттям в процесі навчання нейромереж. Теорія оптимізації досліджує як мінімізувати функцію втрат, що забезпечує ефективне навчання нейромереж. Стохастичний градієнтний спуск (SGD) є основним методом оптимізації для великих наборів даних. Функція втрат мінімізується шляхом поступового оновлення ваг [7]. Формула градієнтного спуску має вигляд:

$$\omega^T = W^{t-1} - \eta \cdot \nabla t(\omega) \tag{1.2}$$

де, W^{t-1} – градієнт вирахуваний на попередньому кроці, η – крок навчання, $\nabla t(W)$ – градієнт даної точки.

Генеративні моделі використовують латентний простір для компактного представлення даних. Цей простір визначається як багатовимірний простір, у якому кожна точка відповідає прихованому представленню даних. Для моделювання латентного простору часто використовують методи варіаційного виведення. Для прикладу, у варіаційних автоенкодерах (VAE) мінімізується

дивергенція Кульбака-Лейблера між апроксимацією та істинним апостеріорним розподілом [19].

1.2. Особливості навчання нейромереж

Робота мережі розділяється на навчання та адаптацію. Під навчанням розуміється процес адаптації мережі до пропонованих еталонних зразків шляхом модифікації (відповідно до тих чи інших алгоритмів) вагових коефіцієнтів зв'язків між нейронами. Для процесу навчання необхідно мати модель зовнішнього середовища, у якій функціонує нейронна мережа та потрібну для вирішення задачі інформацію. По-друге, необхідно визначити, як модифікувати вагові параметри мережі. Існують три види навчання нейромереж: «із учителем», «без учителя», змішаний (частина ваг визначається за допомогою навчання «з учителем», а частина – за допомогою навчання «без учителя») [1].

Градієнтний спуск – один з основоположних алгоритмів навчання нейромереж. Він покликаний мінімізувати функцію втрат, тобто підібрати такі вагові коефіцієнти для нейромережі, за яких помилка у роботі моделі буде мінімальною. Суть методу градієнтного спуску – знайти мінімум функції. В точках екстремуму похідна функції рівна нулю. Щоб до функції застосовувались градієнтні методи вона має бути неперервною і гладкою, без зубчиків та частих перепад висот графіка. Інакше алгоритм постійно застрягатиме у точках локального мінімуму, але якщо функція відповідатиме вимогам це не означає, що алгоритм не застрягне в локальних мінімумах. Це основна проблема цього алгоритму і більшість покращень та вдосконалень якраз спрямовані на боротьбу із цією особливістю [9].

Що таке градієнт? Градієнт – це вектор, який визначає крутість схилу і вказує його напрямок щодо будь-якої із точок на поверхні. Щоб знайти градієнт потрібно взяти похідну від графіка за цією точкою. Рухаючись у напрямку цього градієнта, ми будемо плавно скочуватися в низину (рис. 1.1) [2].

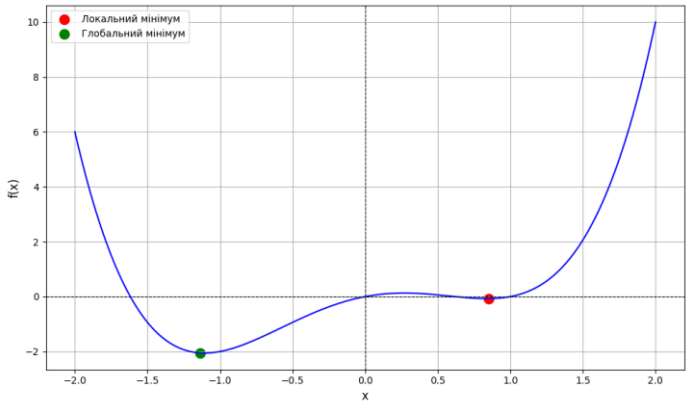


Рис. 1.1. Глобальний, локальний мінімум

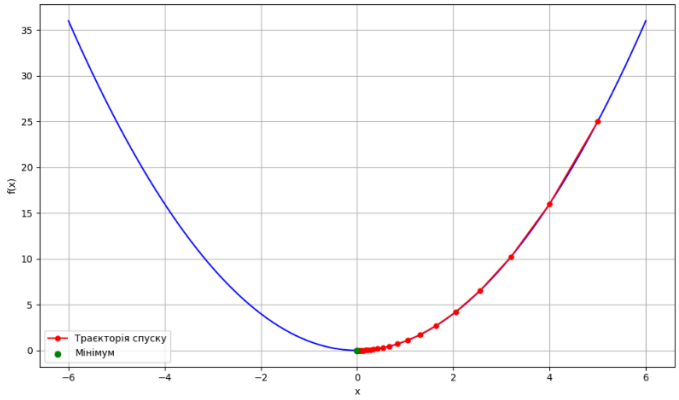


Рис. 1.2. Сходження в мінімум

Із рис. 1.2 можна побачити як працює градієнтний спуск. Ми будемо поступово спускатися вниз по графіку функції, оскільки похідна (градієнт) буде поступово зменшуватися, так, з часом, ми потрапимо у точку мінімуму, де похідна буде рівна нулю.

Якщо говорити про крок навчання λ, то його підбирають експерементально, наприклад 0,1; 0,01; 0,005; … . При надто великому кроці спуск не працюватиме або працюватиме набагато гірше, точка не наблизиться до екстремуму. Якщо крок буде надто малий, то навчання нейромережі відбуватиметься дуже довго та з надмірними витратами обчислювальних ресурсів. Зі значенням λ можна експерементувати, щоб робити його непостійним, тоді можуть покращитися результати [9].

Ще одна проблема градієнтних методів – повільне сходження вниз на плоских ділянках (рис. 1.3). Для вирішення потрібно оптимізовувати навчання одним із наступних способів:

– оптимізація на основі моментів;
– прискорені градієнти Нестерова;
– метод Adagrad;
– методи RMSProp і Adadelta;
– методи Adam і NAdam.

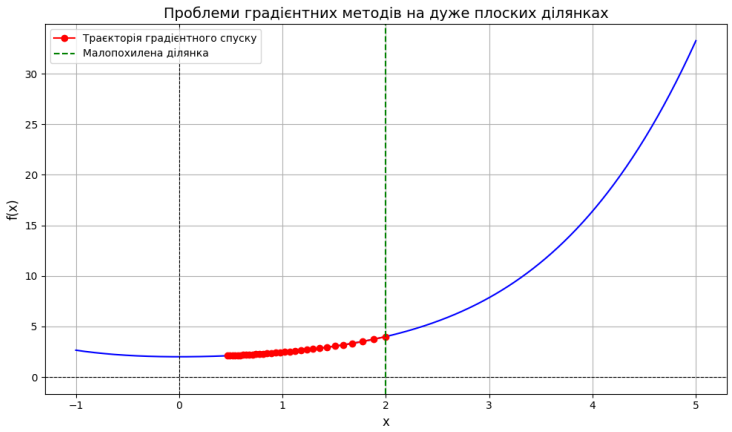


Рис. 1.3. Проблемна ділянка градієнтних методів

Один із найпоширеніших алгоритмів навчання – алгоритм зворотного поширення помилки (Back Propagation). Він базується на алгоритмі градієнтного спуску, а тому основна його проблема це «застрягання» в локальних мінімумах функції. Під час навчання неможливо дізнатися чи було досягнуто глобального мінімуму, тому процес навчання припиняється тоді, коли досягається бажана якість роботи нейромережі. Тому навчання завжди слід запускати кілька разів із різними початковими значеннями коефіцієнтів і відбирати найкращі варіанти. Початкові значення генеруються випадковим чином в області нуля, окрім зміщень (bias).

Нормування вхідних даних виконується тому, що серед них може трапитися якесь велике числове значення, через що ми потрапимо в область насичення, де значення похідної активаційної функції близьке до нуля, що призводить до повільних змін вагових коефіцієнтів. Це спричиняє збільшення

часу навчання нейромережі. Найшвидше навчається нейромережа на крутій ділянці функції.

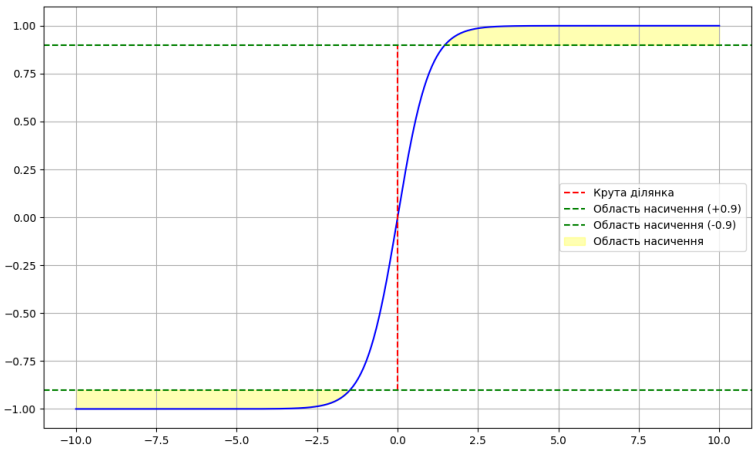


Рис. 1.4. Крута ділянка та область насичення

З огляду на вищевикладене вхідні дані необхідно нормувати, щоб подавати на вхід мережі значення в діапазоні 0-1. При такому підході функцію нормування слід застосовувати й після навчання у роботі нейромережі. Інакше вихідні значення ніяк не зможуть бути вірними.

Щодо навчальної вибірки, то у ній мають бути найрізноманітніші дані, які охоплюють найбільшу область ситуацій і ці ситуації повинні з’являтися із однаковою частотою. Вся множина навчальних даних називається епохою, вона розділяється на частини, які називаються mini-batch. Епоху доцільно розбивати на mini-batch, коли в ній міститься від кількох тисяч елементів. Це робиться для оптимізації процесу корекції ваг, що дозволяє в рази прискорити навчання. Mini-batch «пропускається» через мережу, локальні градієнти на кожному нейроні сумуються, а після проходження всього mini-batch корегуються ваги по результуючій сумі. У такому процесі всі незначні +/- значення будуть взаємно компенсовані і залишиться лише корисне зміщення, яке, власне, і буде застосоване. Це дозволить значно економити обчислювальні ресурси та час на навчання нейромереж. Після того, як вся епоха була «пропущена» через нейромережу, вираховується критерій якості її роботи Q. Якщо цей критерій незадовільний, то вибірка перетасовується, знову ділиться на mini-batch і знову «проганяється» через нейромережу. З часом показник Q

має зменшуватися, що означає покращення якості роботи. Також, окрім навчальної вибірки робиться тестова вибірка, на якій робиться фінальна перевірка якості роботи нейронної мережі після проходження всіх епох навчання.

Перенавчання – явище, коли нейромережа «завчає» навчальні дані й успішно їх використовує з підвищенням критерію якості роботи, але в реальній роботі, на реальних даних, робить помилки. Причиною перенавчання, в тому числі, може бути завелика кількість нейронів, тому існує рекомендація, що у всіх моделях слід використовувати мінімально необхідну кількість нейронів. Існує кілька способів запобігання перенавчанню. Розглянемо їх.

Перший – це використання валідаційної вибірки. Щоб контролювати явище перенавчання навчальна вибірка розбивається на навчальну та валідаційну вибірки, за певним співвідношенням. При навчанні на вхід подається навчальна вибірка і на кожній епосі змінюються вагові коефіцієнти та підраховується критерій якості Q. Так само, після навчальної вибірки, на вхід подається валідаційна вибірка і по ній також на кожній епосі підраховується критерій якості Q. Але при проходженні валідаційної вибірки вагові коефіцієнти не змінюються. В точці, де починається розходження графіків критеріїв якості цих вибірок й буде починатися перенавчання, тому значення вагових коефіцієнтів у цій точці є найоптимальнішим у конкретно цьому процесі навчання.

Другий – це застосування алгоритму Dopout. Ціль цього алгоритму – зменшити спеціалізацію кожного нейрона і зробити із них спеціалістів більш широкого профілю. Це потрібно, щоб не зменшувати число нейронів, адже зі зменшенням кількості нейронів може сильно погіршитися якість роботи нейромережі. Якщо Dopout не допомагає, його слід відкинути. Також не слід використовувати цей алгоритм одразу, а лише в разі якщо станеться перенавчання. Детальніше алгоритм буде описано у наступному підрозділі.

Загалом перенавчання, – це неприємне явище з яким буває складно боротися. Окрім вже перерахованих методів його запобігання також потрібно експериментувати із кроком навчання (він же крок градієнтного спуску), розмірами mini-batch та іншими гіперпараметрами, оскільки всі вони так чи інакше впливають на процес навчання, а отже можуть бути причетні й до перенавчання.

Існують також критерії зупинки процесу навчання, за яких подальше навчання є неефективним. Тоді краще перервати процес навчання, змінити певні гіперпараметри та знову запустити навчання. Ось основні критерії:

- розходження графіків на навчальній та варіаційній вибірках;
- показник критерію якості Q практично не змінюється;
- незначна зміна ваг, що може означати, що було досягнуто локального мінімуму або знаходження у зоні малих градієнтів;
- було досягнуто максимального числа ітерацій по епохам.

Функції активації – складова частина кожного нейрона, яка формує його вихід. Приховані шари, як правило, мають одну й ту саму функцію активації, а для вихідного шару вона обирається інша. Розглянемо функції, які використовуються переважно для прихованих шарів.

Одна з найпростіших функцій активації – порогова функція, яка часто використовується на початкових етапах вивчення нейромереж або для нескладних задач бінарної класифікації. Може підходити для задач розробки логічних функцій XOR, AND тощо. Ця функція не є гнучкою та має багато проблем під час навчання, однією із них є нулеві градієнти [8]. Її формула має вигляд:

$$f = \{\, 1, \text{при } x \geq n;\ 0, \text{при } x < n \qquad (1.3)$$

Сигмоїдна функція, це – функція, яка має S-подібну форму, діапазон її значень 0-1, що запобігає використанню великих значень при навчанні. Вихід такої функції є нелінійним. Завдяки своїй формі вона є придатною для застосуванню градієнтного спуску. Можна побачити, що значення Y дуже швидко зростають коли X знаходиться в діапазоні від -2 до 2. Це робить

функцію схильною до спуску значення Y до обох кінців кривої, що робить дану функцію ще більш придатною для неглибоких нейромереж. Ще однією проблемою є те, що її центр перебуває не у значенні 0, що даватиме постійний позитивний результат [8]. Формула сигмоїдної функції має наступний вигляд:

$$f(x) = \frac{1}{1+e^{-x}} \tag{1.4}$$

Гіперболічний тангенс (tanh), – функція, що може слугувати альтернативною сигмоїдній функції. В певному сенсі можна сказати, що tanh є однією з варіацій сигмоїди. Але tanh є менш плавною, ніж сигмоїдна, що може означати кращу роботу градієнтного спуску. Вихідні значення коливаються в діапазоні від -1 до 1, а це означає, що негативні входи будуть співставлятися із негативними виходами, позитивні із позитивними і тд. Тому мережа, яка використовує таку функцію навчатиметься ефективніше [8]. Формула tanh має вигляд:

$$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{1.5}$$

Гіперболічний тангенс як і сигмоїдна функції застосовуються в неглибоких нейромережах. Похідні таких функцій менші за одиницю і залежно від кількості шарів значення похідних зменшується по експоненті. Це означає, що похідна прямує до нуля, а тому навчання відбуватиметься лише на кількох останніх шарах. Загалом обидві ці функції покликані боротися із проблемою затухаючих градієнтів [8].

Щоб вирішити проблему коли похідна активаційної функції прямує до нуля потрібно підібрати таку функції, похідна якої дорівнювала б одиниці. В такому випадку локальний градієнт не буде ні збільшуватись, ні зменшуватись, як і має бути в ідеалі. Така функція носить назву ReLU, лише потрібно додати обмеження, щоб вона була більша за нуль. Дана функція є незамінною для глибоких нейромереж. ReLU або ж її варіації широко застосовують у більшості нейромереж для роботи із зображеннями, в тому числі у генеративних моделях. ReLU має наступний вигляд:

$f(x) = x$ – початковий варіант функції ReLU (1.6)

$f(x) = \max(x, 0)$ – варіант функції ReLU із обмеженням

ReLU має серйозну проблему, яка називається «проблема вмирання ReLU». Ця проблема виникає, коли функція повертає 0 для певних нейронів і ці нейрони припиняють оновлювати свої ваги. Для вирішення цієї проблеми була запропонована модифікація – Leaky ReLU. Модифікація полягає у застосуванні малого множника для вхідних значень, які ≤ 0, а це дозволяє уникнути перетворення всіх від'ємних значень на 0, як у звичайній ReLU. Це має наступний вигляд [8]:

$$f = \{ x, \text{при } x > 0;\ \alpha x, \text{при } x \leq 0 \tag{1.7}$$

де, α – малий множник, який зазвичай рівний 0,01.

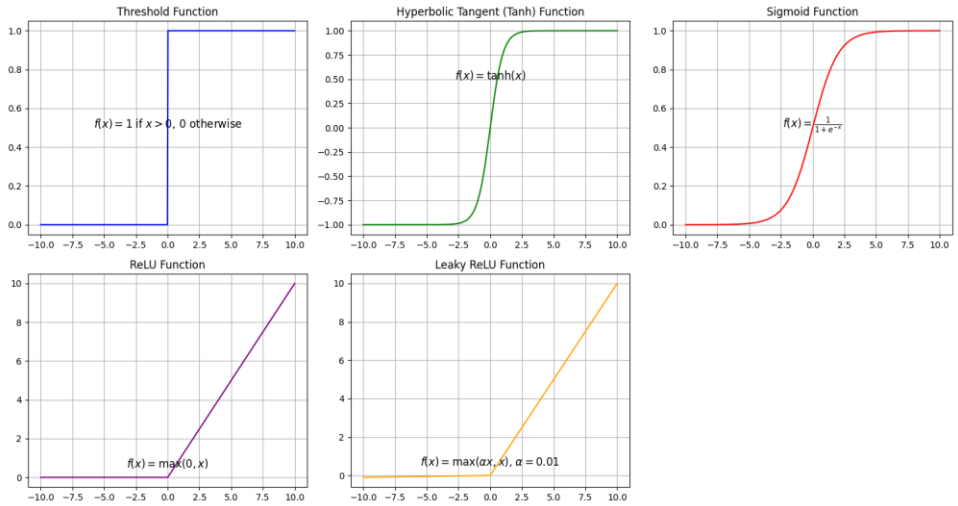


Рис. 1.5. Графіки порогової функції, гіперболічного тангенсу, сигмоїдної функції, ReLU, Leaky ReLU

Softmax – функція активації, яка часто використовується на вихідних шарах нейромереж, які займаються задачами класифікації. Вона перетворює набір входжень у ймовірності, які відповідають певним класам, сума цих ймовірностей дорівнює 1.

1.3. Алгоритми оптимізації та функції втрат

Алгоритми оптимізації спрямовані на підвищення ефективності роботи алгоритмів навчання, для запобігання явищу перенавчання та для виправлення інших проблем які можуть виникати при навчанні нейромереж чи їхній роботі.

Розглянемо алгоритми, які застосовують для запобігання «застрягання» градієнтного спуску в локальних мінімумах.

Метод моментів (імпульсів). Базова ідея цього методу – додати певний множник γ, який лежить в межах $0 < \gamma < 1$, перед градієнтом вирахуваним на попередньому кроці (формула 1.2). Це дозволить збільшувати градієнт, що може допомогти йому «проскочити» певні локальні мінімуми.

Метод Нестерова – розвиває ідею методу моментів і полягає в тому, що градієнт рахується не у точці W, а у точці W- γW^{t-1} (формула 1.2) [9].

Adagrad – базова ідея полягає у тому, що певні параметри можуть швидше досягати свого оптимуму, ніж інші. Ті параметри, які близькі до оптимуму треба змінювати із меншим кроком, а інші – із більшим. Крок для параметрів залежить від величини їхніх флуктуацій (коливань), – чим більші коливання, тим більший крок. Недоліком такого методу є постійне зменшення кроку навчання і складність підбору гіперпараметру швидкості навчання [9].

Adadelta – алгоритм, який був представлений для покращення роботи Adagrad. Головна ідея – це збереження не усієї історії обчислень, як в Adagrad, а лише її частини. При чому в цій історії зберігаються квадрати всіх параметрів. У цьому алгоритмі існує спеціальний параметр – дисконтуючий множник (ρ), в межах $0< \rho <1$, який обмежує вплив градієнтів, які вирахувані давно порівняно із поточним кроком [9].

RMSProp – схожий на Adadelta, але замість зберігання історії квадратів він бере корінь із середнього квадрату градієнта по всім параметрам [9].

Adam – чергова модифікація Adagrad, яка використовує згладжені версії середнього і середньоквадратичного градієнтів. Ця функція добре себе зарекомендувала для пошуку мінімуму функції втрат [9; 11].

Dropout – алгоритм для запобігання перенавчанню моделі. Його ціль – зменшити спеціалізацію кожного нейрона. На кожній ітерації навчання з певною вірогідністю (p) відкидається частина нейронів. Це призводить до того, що нейрони стають ширше спеціалізованими. Рекомендовано починати із p=0.5, але ця величина підбирається експериментально. При відкиданні

нейронів сума на нейроні в наступному шарі стане значно меншою, від тієї, яка мала б бути. Це може призводити до некоректної роботи нейромережі, тому потрібно коригувати суму. Для цього слід визначити скільки в середньому буде виключених нейронів, це математичне очікування, яке слід обчислити. Опишемо роботу алгоритму формулами [10]:

$$\mathrm{M\{x\}} = \sum_{i=1}^{n} x_i p_i \tag{1.8}$$

де, p_i – вірогідність виключення, x_i – число нейронів, до якого застосовується дана ймовірність. Це число завжди буде $= 1$, n – кількість нейронів.

Отже, отримаємо:

M{x}=np – середнє число виключених нейронів в поточному шарі

n-np=n(1-p)=nq – середнє число включених нейронів

q=1-p – вірогідність того, що нейрон не виключений

Із цього випливає:

$$\frac{x_q}{x} \sim \frac{nq}{n} = q \Rightarrow\ x_q \cdot \frac{1}{q} \sim x$$

Щоб «x_q» стала пропорційною «x» потрібно домножити її на $\frac{1}{q}$. Ці сигнали стануть пропорційними (співставними), ніби нічого і не відкидалося.

Алгоритм BatchNormalization, в якому стандартизуються дві характеристики, – математичне очікування (формула 1.9) – до нуля і дисперсія (формула 1.10) – до одиниці. Розробники цього методу рекомендували проводити нормування перед функцією активації (для суми), але теперішні дослідження показують, що цей блок нормалізації можна ставити і після функції активації (для виходів нейронів), оскільки іноді цей варіант працює краще. В цьому алгоритмі статистики визначаються для величини V – вектора суми кожного спостереження із mini-batch [11].

$$m_\vartheta = M\{V\} = \frac{1}{m}\sum_{i=1}^{m} \vartheta_k^i \tag{1.9}$$

де, m – розмір mini-batch, V - вектор суми кожного спостереження із mini-batch, ϑ_k^i – складова вектора V

$$\sigma_\vartheta^2 = M\{(V - m_\vartheta)^2\} = \frac{1}{m-1}\sum_{i=1}^{m}(\vartheta_k^i - m_\vartheta)^2 \approx \frac{1}{m}\sum_{i=1}^{m}(\vartheta_k^i - m_\vartheta)^2 \tag{1.10}$$

Тепер, щоб вектор V мав нульове середнє значення і одиничну дисперсію, кожен елемент цього вектора підлягає наступному перетворенню:

$$Z_k^i = \frac{\vartheta_k^i - m_\vartheta}{\sqrt{\sigma_\vartheta^2 + \varepsilon}} \tag{1.11}$$

де, ε – мала позитивна величина, щоб уникнути ділення на нуль, якщо σ=0 чи $\sigma \approx 0$.

Вектор Z являтиме собою випадкову величину з нулевим математичним очікуванням і дисперсією близькою до одиниці. Але такого перетворення недостатньо. При надходженні на вхід функції активації такого вектора виникають певні проблеми. Перше – будуть втрачатися природні статистичні характеристики між mini-batch, тобто будуть невеликі зміни в середніх значеннях і дисперсіях. Наприклад, коли в batch1 машини чорного кольору, а в batch2 – червоного, то при такому нормуванні всередині нейромережі усі машини будуть представлені як машини одного кольору. А нам хотілося б зберегти цю різницю. Друге – нулеве математичне очікування означатиме, що в середньому ми будемо опинятися в точні А (рис. 1.6.). На цьому рисунку представлено функцію активації, наприклад сигмоїдну чи гіперболічний тангенс. Сенс у тому, що в такій ситуації ми опиняємося на ділянці функції, яка близька до лінійної, тобто нейромережа втрачає нелінійність [11].

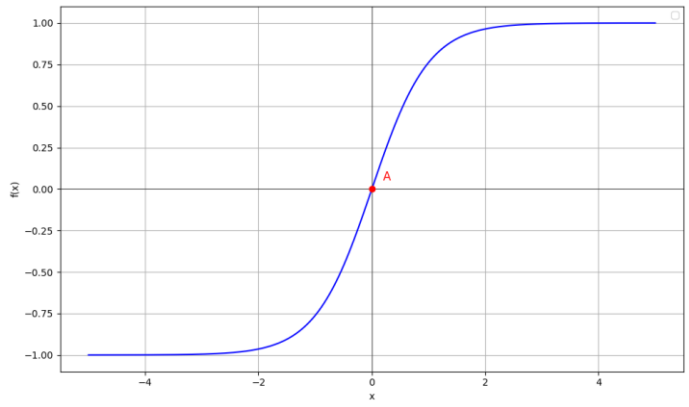


Рис. 1.6. Представлення функції активації

Щоб виправити проблеми, описані вище, до вектору Z додатково застосовується формула 1.12. Спершу γ=1, β=0, в процесі навчання вони підлаштовуються за допомогою алгоритму градієнтного спуску. Тепер

нейромережа матиме додаткові налаштовувані змінні, окрім ваг чи bias. Результуюча величина y_k^i подається на вхід функції активації [11].

$$y_k^i = \gamma Z_k^i + \beta \quad (1.12)$$

де, γ, β – додаткові налаштовувані параметри, початково γ=1, β=0.

Алгоритм BatchNormalization може як покращити, так і погіршити роботу нейромережі. Серед можливих покращень: більша незалежність навчання для кожного шару нейронів, можливість збільшення кроку навчання (λ), в певній мірі запобігання перенавчанню, менша чутливість до початкової ініціалізації вагових коефіцієнтів. Рекомендовано початково будувати нейромережі без BatchNormalization чи Dropout і додавати їх лише за необхідності, але не обидва алгоритми разом. Хоча іноді їх використовують разом, але це рідкісні випадки.

Перейдемо до ознайомлення із функціями втрат. Класично в алгоритмі Back Propagation як функція втрат використовується середній квадрат помилок (формула 1.13). Ця функція втрат може застосовуватись у всіх моделях, які будуть розглянуті у розділі 2.

$$\mathrm{E} = \frac{1}{N}\sum_{i=1}^{N}(d_i - y_i)^2 \quad (1.13)$$

де, d – бажаний вихід, y – реальний вихід.

Ще одна класична функція втрат, яка застосовується у багатьох нейромережах, – це бінарна кросс-ентропія (формула 1.14). Ця функція втрат використовується коли є пара значень за якими доцільно визначати відхилення роботи нейронної мережі від бажаної. Вона добре себе показує у трансформерах, де використовуються токенізовані зображення [14]. Також широко використовується у генеративно-змагальних нейромережах [20].

$$E = -\frac{1}{N}\sum_{i=1}[d_i \log(y_i) + (1 - d_i)\log(1 - y_i)] \quad (1.14)$$

Дивергенція Кульбака-Лейблера – функція, яка часто використовується у різних автоенкодерах (формула 1.15).

$$D_{KL} = \frac{1}{2}\left(tr(\sum_N^{-1} \cdot \sum_G) + (\mu_N - \mu_G)^T \cdot \sum_N^{-1} \cdot (\mu_N - \mu_G) - k + \log\left(\frac{det\sum_N}{det\sum_G}\right)\right) \quad (1.15)$$

де, *tr*() – сума значень елементів головної діагоналі матриці, *det*() – добуток значень матриці на головній діагоналі (детермінатор матриці), μ_N – нормальне математичне очікування, μ_G – фактичне математичне очікування, $\sum_G$ – фактична матриця, $\sum_N$ – очікувана матриця.

1.4. Засоби створення нейронних мереж

Середовище для розробки нейромереж грає критичну роль у створенні моделей глибокого навчання, від впровадження та налаштування архітектури до тренування та оптимізації. Огляд різних середовищ розробки й популярних бібліотек дозволить оцінити, які інструменти найбільше підходять для конкретних завдань, залежно від їхніх можливостей, гнучкості, продуктивності та підтримки.

Jupyter Notebook – інтерактивне середовище для роботи із різними мовами програмування. Воно надає можливість виконувати код поетапно, що спрощує аналіз проміжних результатів, дебаг, а також документування роботи у реальному часі. Ключовою особливістю Jupyter Notebook є здатність відображати не лише текстовий вивід, але й графіки та зображення, що є особливо корисним для аналізу даних і машинного навчання. Завдяки своїй зручності, Jupyter Notebook є популярним інструментом для навчання та досліджень, а також підходить для швидкого прототипування моделей машинного навчання. Його широкі можливості інтеграції з бібліотеками для візуалізації даних, такими як Matplotlib і Seaborn, а також з бібліотеками для нейронних мереж TensorFlow, PyTorch, дозволяють виконувати комплексні обчислення та аналіз у рамках одного документа.

AWS SageMaker – це хмарний сервіс від Amazon, спеціально розроблений для навчання та розгортання моделей машинного навчання.

SageMaker надає повний набір інструментів для розробки нейронних мереж. Цей сервіс дозволяє масштабувати обчислення за рахунок використання потужностей GPU та TPU, а також забезпечує інтеграцію з іншими сервісами AWS. Однією з переваг AWS SageMaker є можливість автоматичного налаштування гіперпараметрів, що дозволяє значно спростити процес пошуку оптимальних налаштувань для моделей. Також цей сервіс дозволяє масштабувати та оновлювати моделі в реальному часі без необхідності управління інфраструктурою. Інструменти для моніторингу продуктивності моделей роблять цей сервіс особливо корисним для організацій, що працюють з великими обсягами даних та потребують високої надійності і продуктивності.

Google Colab – це хмарна платформа для написання і виконання коду. Містить і безкоштовну частину, якої цілком вистачає для більшості необхідних обчислень та маніпуляцій. Colab підтримує Jupyter Notebook, що дозволяє об’єднувати код, текст, графіку, математичні формули в єдиному документі, що особливо корисно для наукових досліджень. Основна перевага Google Colab полягає в доступі до апаратних прискорювачів (GPU та TPU), що дозволяє обробляти великі об’єми даних та тренувати складні моделі нейронних мереж швидше, ніж на звичайному процесорі. Платформа також інтегрується з Google Drive, що спрощує роботу із файлами і даними.

NumPy – це бібліотека Python для проведення обчислень, яка надає потужні можливості для роботи із багатовимірними масивами та матрицями. Ця бібліотека забезпечує ефективну реалізацію математичних операцій, таких як лінійна алгебра, статистичні обчислення, випадкове моделювання, що робить її основою для багатьох інших бібліотек у Python. Завдяки своїй швидкості та продуктивності NumPy є стандартом у машинному навчанні, обробці сигналів і зображень, а також у обробці великих обсягів даних.

MXNet – це високопродуктивна, гнучка бібліотека для глибокого навчання. Часто використовується у поєднанні із сервісами AWS. MXNet має архітектуру, яка підтримує паралельне виконання на декількох GPU, що

робить її підходящою для тренування великих нейронних мереж. Вона надає можливість побудови моделей з різним рівнем деталізації: від високорівневих функцій до низькорівневого контролю для оптимізації продуктивності. Відзначається підтримкою динамічного обчислювального графа. Ця бібліотека широко застосовується в комерційних рішеннях і підтримує різноманітні мови програмування.

TensorFlow – це бібліотека машинного навчання та глибинного навчання, розроблена компанією Google, її робота заснована на обчислювальному графі. Вона є одним із найпопулярніших інструментів для розробки нейронних мереж, оскільки має потужний інструментарій для навчання і використання моделей у різних масштабах – від невеликих дослідницьких проєктів до повноцінних комерційних додатків, включаючи можливості для мобільних і вбудованих пристроїв (TensorFlow Lite) та веб-браузерів (TensorFlow.js). TensorFlow підтримує широкий спектр моделей і алгоритмів, включаючи нейронні мережі, обробку природної мови та комп’ютерний зір.

Keras – це високорівнева бібліотека машинного навчання, що є частиною Tensorflow. Вона розроблена для швидкої й легкої побудови нейронних мереж, надаючи інтуїтивний інтерфейс, який робить процес створення і тренування моделей простішим і доступнішим. Keras підтримує багато типів нейронних шарів і функцій, таких як згорткові та рекурентні шари, що дозволяє використовувати її для побудови як простих, так і складних архітектур нейронних мереж. Keras також має велику спільноту користувачів і гарно написану документацію, що полегшує роботу із нею.

Caffe – це бібліотека для глибинного навчання, розроблена зокрема для задач комп’ютерного зору. Вона була створена для обробки великого обсягу зображень з високою швидкістю, що робить її особливо корисною для задач класифікації та детекції об’єктів. Підтримує навчання на GPU. Основний недолік полягає в обмеженій гнучкості: створення складних моделей або налаштування індивідуальних архітектур може бути непростим. Ця бібліотека

більше підходить для спеціалізованих задач, де потрібна висока швидкість навчання й немає потреби у динамічних архітектурах.

PyTorch – це бібліотека для глибокого навчання, розроблена компанією Facebook, яка здобула популярність завдяки своїй простоті та гнучкості. Вона відома зручним інтерфейсом та «динамічним графом обчислень», що означає, що обчислювальні графи створюються та змінюються в реальному часі під час виконання коду, а це є зручним для досліджень та тестування нових підходів. PyTorch широко використовується в академічній спільноті завдяки своїй легкості у використанні та підтримці передових методів у машинному навчанні. Бібліотека підтримує тренування як на CPU, так і на GPU, що робить її ефективним інструментом для обробки великих обсягів даних і створення складних моделей. PyTorch забезпечує інтеграцію з іншими бібліотеками, такими як TotchVision і Hugging Face Transformers, що робить його популярним вибором для досліджень у сфері комп’ютерного зору й обробки природної мови.

Fast.ai – це бібліотека, побудована на основі PyTorch. Вона надає інтуїтивний інтерфейс для простого та швидкого створення моделей. Ця бібліотека орієнтована на початківців, оскільки надає набір готових до використання архітектур і функцій, що значно полегшує процес розробки та оптимізації моделей. Основний акцент зроблено на спрощення роботи з машинним навчанням для тих, хто не має значного досвіду програмування. Fast.ai надає доступ до навчальних матеріалів і курсів, що забезпечують розуміння основних концепцій машинного навчання. Вона дозволяє досягти високих результатів із мінімальною кількістю коду та часто використовується в академічних дослідженнях.

Таким чином, вибір бібліотеки та середовища розробки значною мірою залежить від конкретних вимог проєкту, досвіду розробника, потреб у гнучкості й продуктивності. Tensorflow і PyTorch залишаються найбільш універсальними і популярними бібліотеками, тоді як інші, такі як MXNet і Fst.ai, часто обирають для специфічних завдань. Середовища розробки, як-от

Jupyter Notebook та Google Colab, забезпечують зручність при наукових дослідженнях, створенні тестових моделей, розробці нейромереж різного рівня складності, тоді як AWS SageMaker більше підходить саме для великих корпоративних проєктів.

Висновки до розділу 1

З математичної точки зору, генеративні моделі використовують ймовірнісні підходи для опису розподілу даних. Генеративні моделі дозволяють формувати нові дані, схожі на навчальні, базуючись на відновленні латентного простору або змагальному принципі.

Один з основних алгоритмів навчання є зворотне поширення помилки (Back Propagation), в основі якого алгоритм градієнтного спуску. Цей алгоритм має проблеми – «застрягання» в локальному мінімумі та проблему затухаючих градієнтів, при використанні на глибоких нейромережах. Ці проблеми вирішуються застосуванням різних оптимізаційних алгоритмів та різних функцій активації. Найпопулярнішими алгоритмами є Adam, моменти Нестерова та інші. Найпопулярніші функції активації – ReLU, Softmax, Sigmoid, Tenh та їхні варіації. Особлива увага приділяється запобіганню перенавчанню, через алгоритм Dropout та стандартизації характеристик на виходах нейронів через алгоритм BatchNormalization.

Вибір функції втрат залежить від специфіки завдання, яке потрібно вирішувати, а також від архітектури нейромережі. Найпопулярнішими функціями втрат є бінарна кросс-ентропія, середній квадрат помилок та дивергенція Кульбака-Лейблера.

Засоби створення нейромереж охоплюють різноманітні програмні засоби, такі як бібліотеки, фреймворки, наприклад Tensorflow, PyTorch, Keras, NumPy, хмарне середовище Google Colab. В Google Colab користувачі можуть використовувати GPU і TPU, що дозволяє значно прискорити процес навчання.

2. АНАЛІЗ ОСНОВНИХ АРХІТЕКТУР ГЕНЕРАТИВНИХ НЕЙРОМЕРЕЖ

2.1. Аналіз сучасних тенденцій у генеруванні зображень

Сьогодні технології генерування зображень займають важливе місце у розвитку машинного навчання. Вони використовуються в різних галузях: від мистецтва та дизайну до медицини та науки. Завдяки швидкому розвитку таких інструментів, як генеративно-змагальні мережі (GAN) та дифузійні моделі, створення реалістичних зображень з високим ступенем деталізації стало доступним для широкого кола користувачів.

Трансформери, спочатку розроблені для задач перекладу текстів, зараз активно використовуються у комп’ютерному зорі. Самі по собі трансформери погано працюють із генерацією зображень, тому їх використовують в якості доповнення до інших моделей. Трансформери можуть забезпечувати широкий спектр додаткових можливостей, основними серед яких є аналіз текстового опису для генерації зображень, або покращення якості на завершальних етапах генерації.

Автоенкодери також здебільшого використовуються як частина інших архітектур. Наприклад, варіаційні автоенкодери або лише декодери можна використовувати як генератор у генеративно-змагальних мережах. Хоча автоенкодери рідко використовуються безпосередньо для генерації зображень, через обмеженість якості генеративного процесу, все ж вони активно розвиваються у даному напрямку.

Генеративно-змагальні мережі стали революційною технологією у сфері генерування зображень. GAN складається із двох нейронних мереж – генератора і дискримінатора, які навчаються у конкуренції один із одним. Генератор намагається створити якомога реалістичніше зображення, дискримінатор намагається відрізнити згенероване зображення від реального.

Технологія дифузійних моделей базується на поступовому перетворенні шуму в зображення через обернений процес дифузії. Дані моделі демонструють високу якість та контрольованість результатів.

Можна підсумувати, що на даний момент спостерігається тенденція до використання GAN та дифузійних моделей, а також інших моделей, як доповнення до них. Що ж стосується складності генеративних моделей, то постійно зростає їхня глибина, а отже зростають вимоги до апаратного забезпечення, через потребу в обробці все більшої кількості навчальних даних, та створення зображень все кращої якості та деталізації. Використовуються генеративні нейромережі переважно дизайнерами, художниками та у сфері розваг, в меншій мірі науковцями для створення навчальних даних та в медицині для покращення якості медичних знімків.

2.2. Згорткові нейромережі

Згорткові нейронні мережі (CNN) – це особливий вид нейромереж, робота яких заснована на використанні спеціальних згорткових шарів. Даний вид нейромереж є одним з найпопулярніших на сьогоднішній день у сфері роботи із зображеннями. Вони використовуються для різних задач комп’ютерного зору, зокрема класифікації зображень, виявлення об’єктів, сегментації. Самі по собі CNN використовуються для задач пов’язаних із комп’ютерним зором, а не генерацією зображень. Ми їх розглядаємо тому, що принципи їхньої роботи є базовими для нейромереж у сфері роботи із зображеннями. А також часто згортки в тому чи іншому вигляді застосовують у генеративних моделях.

Згорткові нейромережі використовують набагато менше налаштовуваних параметрів, наприклад вагові коефіцієнти та зміщення (bias), а це дозволяє зменшити потребу в обчислювальних потужностях та прискорити процес навчання. Незважаючи на це CNN показує кращі результати, якщо порівнювати їх із нейромережами, які мають велику кількість налаштовуваних параметрів, наприклад повнозв’язні нейромережі.

Що стосується розмірів CNN, команда Алекса Крижевськи провела дослідження, які показують, що глибина згорткової нейромережі має значення. Вони виявили, що при вилученні навіть одного згорткового шару

якість роботи такої мережі знижується [12]. Здавалося б, постійне збільшення глиби CNN буде призводити до зростання їхньої продуктивності. Воно так і є, але лише до певної межі. Глибокі нейромережі набагато важче навчати через недосконаліть сучасних алгоритмів. Зокрема, при навчанні методом зворотного поширення помилки (Back Propagation) виникає проблема затухаючих градієнтів. Це призводить до погіршення загальної ефективності роботи нейромережі. Як приклад ефективних глибоких згорткових нейромереж можна навести VGG-16 і VGG-19. Вони мають по 16 і 19 шарів відповідно [13]. Завдяки своїй архітектурі вони можуть працювати із більш складними елементами зображення, що сприяє підвищенню продуктивності.

Як можна побачити із статтей [12; 13], для CNN класичною є архітектура коли послідовно стоїть певна кількість згорткових шарів, разом із Pooling шарами, а далі йде кілька повнозв'язних шарів із нелінійною функцією активації (найчастіше це ReLU), завершувати цю архітектуру, залежно від задач нейромережі, може, наприклад, повнозв'язний шар із функцією активації Softmax, якщо мова йде про класифікацію.

Розглянемо детальніше роботу CNN та різних її складових.

Спочатку розглянемо загальну будову згорткового шару і припустимо, що робота ведеться над одноканальним зображенням (в градаціях сірого). Цей особливий шар складається із n-груп нейронів і кожна з цих груп містить m нейронів. Кожна група нейронів має своє ядро фільтра (на малюнку позначено червоним), зазвичай це ядро формує область 3x3. Для кожної позиції в ядрі є своя визначена вага, а також один bias загалом для всього ядра і ці параметри є незмінними для цієї групи нейронів. Таким чином, для однієї групи нейронів існує 10 налаштовуваних параметрів – 9 вагових коефіцієнтів і 1 bias. Крім того також задається крок цього фільтра на зображенні, зазвичай він рівний одиниці.

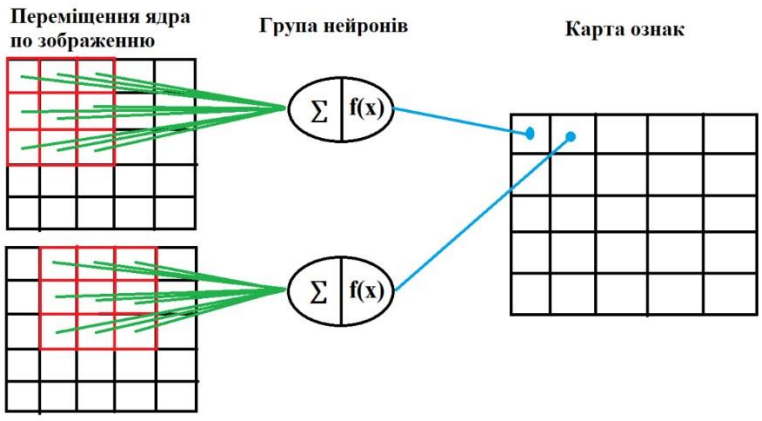


Рис. 2.1. Переміщення ядра, формування виходів нейронів

Кожен нейрон приймає на вхід область, рівну ядру фільтра, тобто у нашому випадку 3x3, виконує операцію згортки, пропускає результат згортки через функцію активації та повертає певний результат, який виступає окремим значенням на карті ознак (рис. 2.1). Далі відбувається зміщення фільтра, всі операції повторюються та повертається ще одне значення карти ознак. І так відбувається доки не будуть сформовані всі карти ознак. Формула згортки має вигляд:

$$\vartheta_{kg} = \sum_{i=1}^{3}\sum_{i=1}^{3} x_{i+k,j+g} \cdot \omega_{ij} + \omega_0; k \qquad (2.1)$$

де, g=1,2,3,… , $x_{i,j}$ – піксель зображення, $\omega_{i,j}$ – ваговий коефіцієнт ядра, ω_0 – bias, $O_{kg} = f(\vartheta_{kg})$ – окреме значення на карті ознак, $f(x)$ – функція активації, ϑ_{kg} – значення згортки для конкретної області зображення.

З виходів кожної групи нейронів формується своя карта ознак, тобто на виході згорткового шару ми отримуємо n карт ознак. Значення цих карт ознак визначають фільтри, вони дозволяють виділяти характерні ділянки на зображенні (один фільтр – одна характеристика). Завдяки цьому нейрони кожної групи активуються коли на зображенні з’являється фрагмент підходящий для ваг відповідного фільтра. Отже, кожна величина на карті ознак вказує на наявність певної ознаки у певному місці зображення. При поданні цих карт ознак на наступний згортковий шар вони будуть виступати як багатоканальне зображення, де кожна карта ознак окремий канал. На першому згортковому шарі карти ознак характеризують наявність якихось

простих форм, наприклад горизонтальна лінія, нахилена лінія. На подальших (прихованих) згорткових шарах карти ознак все більше узагальнюються і представлять все складніші форми та характеристики.

Досі розглядалася робота над одноканальним зображенням на вході, тепер розглянемо як відбувається робота із багатоканальним зображенням на вході. Аналогічно відбувається обробка багатоканальних зображень (рис. 2.2), що складаються із карт ознак та подаються на приховані згорткові шари. При роботі із кольоровим зображенням кожен кольоровий канал обробляється своїм окремим фільтром і для кожної кольорової компоненти створюється своя карта ознак. Далі ці карти ознак поелементно сумуються, додається зміщення (bias). Так ми отримуємо значення ϑ_{kg} і пропустивши його через функцію активації отримуємо O_{kg} – окреме значення на карті ознак. Та сама ситуація і на подальших, прихованих, шарах. Він (прихований шар) отримує набір з n карт ознак (каналів), кожен канал «проганяється» через свій фільтр, вихідні значення сумуються, додається bias, ця сума пропускається через функцію активації, формується новий набір карт ознак та подається на вхід наступному прихованому шару.

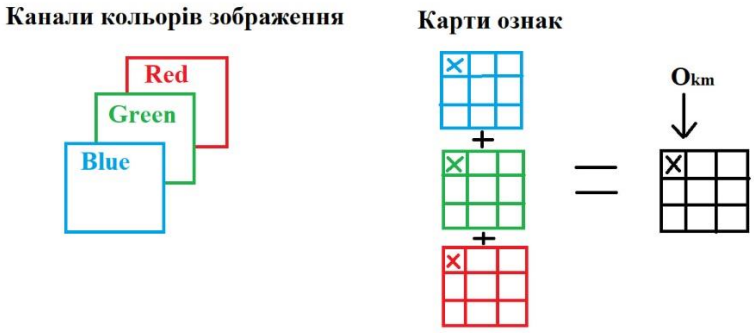


Рис. 2.2. Робота із багатоканальним зображенням

Залежно від побудови архітектури карти ознак можуть бути того самого розміру що й зображення або меншими. Якщо дано зображення розміром 32x32 пікселі, то щоб карта ознак мала такий самий розмір потрібно змістити фільтр за край зображення, залежно від кроку цього фільтра. При цьому значення $x_{i,j}$ для області по-за зображенням слід приймати рівним нулю (рис. 2.3).

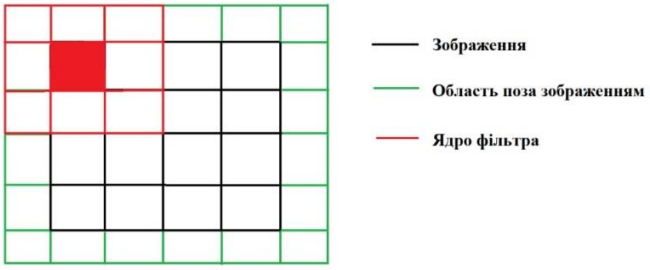


Рис. 2.3. Зміщення ядра відносно країв зображення

Окрім згорток в CNN існує ще одна важлива операція – Pooling (масштабування). Існує три типи цієї операції:

- MaxPooling – вибір найбільших значень;
- MinPooling – вибір найменших значень;
- AveragePooling – вибір середніх значень.

Найчастіше обирається саме MaxPooling, бо більші значення, як правило, свідчать про присутність певної ознаки, а менші, в свою чергу, про відсутність цієї ознаки. Тому, відбираючи максимальні числа, ми відбираємо ознаки, для аналізу їх на більшому масштабі, завдяки чому нейрони наступного згорткового шару можуть відбирати більш загальні характеристики.

Як це діє? Припустимо, ми маємо карту ознак як на рисунку 2.4. Аналізуємо цю карту ознак за допомогою вікон, які не перетинаються, на відміну від операції згортки. Припустимо, що розмір вікна буде 2x2, тоді зміщення такого вікна буде рівним 2, а розмір карт ознак буде зменшено вдвічі по осі Ox та по осі Oy. Із кожного вікна відбирається максимальне (якщо використовуємо MaxPooling) значення.

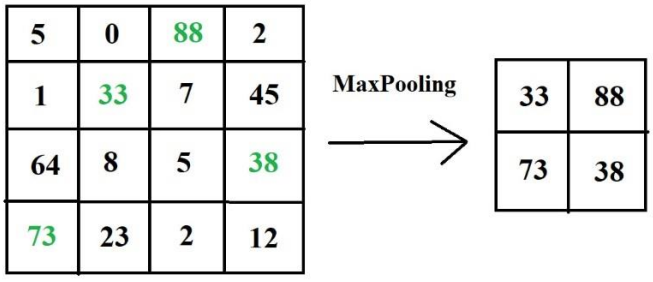


Рис. 2.4. Max Pooling

Щодо особливості архітектури, коли кілька згорткових шарів розміщені один за одним, то така архітектура використовується у нейромережах VGG-16, VGG-19. Якщо збільшити розмір ядра (фільтра) до, наприклад 5x5, то можна за один раз захопити більшу область зображення, відповідно це позитивно вплине на продуктивність роботи згорткового шару. Проте таке збільшення розміру ядра несе за собою значне збільшення кількості налаштовуваних параметрів. Дві згортки із ядрами 3x3, які стоять одна за одною якраз будуть еквівалентні одній згортці із ядром 5x5. В такому випадку сумарне число параметрів для двох згорток 3x3, буде менше, ніж для однієї згортки 5x5. Даний механізм був описаний у статті [13].

2.3. Трансформери

Ще один цікавий клас нейронних мереж для роботи із зображеннями – це трансформери. Вони добре працюють із просторовими залежностями, тому їхня основна сила в екстракції ознак. Це означає, що вони добре підходять для задач комп’ютерного зору, але в генеративних задачах здебільшого є допоміжними в дифузійних моделях чи GAN. В цих моделях вони можуть значно допомогти в разі генерації зображення на основі інших зображень чи окремих об’єктів на них або із покращенням якості та деталізації на завершальному етапі генеративного процесу. Ще трансформери використовують за прямим призначенням, для аналізу тексту, коли потрібно згенерувати зображення на основі опису користувача. Тому тут не буде детально розглядатися їхня архітектура, а лише принцип роботи. Звернемо увагу на такі моделі як ViT, Swin, Swinv2.

Початково дані нейромережі створювалися для перекладу текстів. На момент створення трансформерів ці задачі виконувати рекурентні нейронні мережі (RNN). RNN здійснювали послідовний доступ до слів, як наслідок неможливість обчислення значень для подальших слів доки не обчислено значення для першого слова. Це спричиняло неточності перекладу та низьку його швидкість. У статті [14] автори стверджують, що можна на кожне окреме

слово сконцентрувати свою «увагу» (Attention), а це дозволяє проводити незалежні обчислення для кожного слова у тексті. В зображеннях це працює так само – «увага» розосереджена на кожен фрагмент зображення [4].

Розглянемо складові частини трансформерів та основні принципи їх роботи. Ембендинг – це спеціальний вектор, обчислений із конкретного фрагменту зображення, у нього також кодується інформація про розташування цього фрагмента на зображенні. Ембендинги часто використовують для представлення різних об’єктів в якості числових векторів. Найважливіша частина це механізм Attention, він дозволяє нейромережі фокусуватися на частинах зображення, що мають ключові деталі. Для кожного фрагменту зображення створюються три вектори Query, Key, Value за допомогою трьох матриць ваг W^Q, W^K, W^V шляхом множення ембендингу фрагмента на відповідну матрицю ваг. На основі цих векторів обчислюється Attention за формулою 2.2. Self-Attention – це механізм в якому заключається основна потужність трансформерів, він покликаний помічати зв’язки між різними фрагментами зображення. Залежності виявляються завдяки обчисленим Attention [14]. При створенні зображень даний механізм є критично важливим, адже дозволяє додавати деталі ґрунтуючись на контексті [4].

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) \cdot V \text{ ; [14]} \qquad (2.2)$$

де, $d_k = \frac{Hidden\ Size\ D}{Heads}$, ці величини можна знайти у відповідній таблиці статті [15].

Перейдемо до загальної архітектури, основна складова якої – енкодер, який перетворює вхідне зображення у векторне представлення. Для прикладу модель ViT_Base містить 12 шарів енкодера [15]. Кожен такий шар складається із блоків Multi-Head Attention і Multy Layer Persiptron. Ці блоки мають наскрізне підключення, як мережах ResNet [16]. Multi-Head Attention використовується для захоплення різних аспектів візуальної інформації, так модель одночасно обробляє кілька зв’язків між фрагментами. ViT_Base містить 12 Heads, тобто 12 незалежних потоків, кожен з яких має свій набір

матриць ваг W^Q, W^K, W^V. Інші моделі мають інше число потоків, інформацію про це наведено у таблиці [15]. Multy Layer Persiptron являє собою повнозв'язні шари, які нелінійно перетворюють вхідні дані. Цей блок складається із повнозв'язного шару із 768 входами і 3072 виходами, шару активації із функцією активації Gelu, шару Propout і ще одного повнозв'язного шару із 3072 входами і 768 виходами [4].

Розглянемо особливості архітектур Swin та Swin v2 [17; 18]. Дані архітектури краще оптимізовані для задач комп'ютерного зору, ніж ViT. Основні відмінності це менший розмір фрагментів і наявність так званих вікон. Вікно об'єднує певну кількість фрагментів, це дозволяє застосовувати механізм Self-Attention лише до фрагментів всередині вікна, а не до усього зображення одразу. Це суттєво скорочує кількість обчислень і дозволяє нейромережі більше звертати увагу на певні локальні особливості, не втрачаючи при цьому глобальних. Для кодування позицій фрагментів використовуються матриця зміщень (bias) [17], в такому випадку Attention обчислюється за формулою 2.3. Swin v2 – це той самий Swin з кількома відмінностями [18]:

- в енкодері нормалізація виконується після блоків Multi-Head Attention і Multy Layer Persiptron, це вирішує проблему суттєвого росту значення активації;
- множення матриць Q і K не відбувається. Замість цього використовується формула 2.4, щоб тримати значення активації в діапазоні оптимальних значень;
- змінено кодування позицій фрагментів, тепер для цього використовуються log-spaced координати. Це вирішує проблему сильного просідання точності навченої моделі зі збільшенням розміру вхідного зображення.

Дані покращення дозволили зробити Swin v2 найбільш підходящим із розглянутих трансформерів для генерації зображень. Swin v2 краще працює із високою роздільною здатністю, забезпечується стабільність навчання та гнучкість в інтеграції в генеративні структури [4].

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}} + B\right) \cdot V \text{ ; [13]} \quad (2.3)$$

де, B – матриця зміщень

$$\frac{cos(q_i k_j)}{\tau} + B_{ij} \text{ ; [18]} \quad (2.4)$$

де, B_{ij} – відносне позиційне зміщення між пікселями (bias), τ – навчальне значення для кожного шару і голови, яке встановлюються більшим, ніж 0,01.

2.4. Автоенкодери

Дані нейронні мережі є більш популярними у задачах реконструкції даних, згладжуванні зображень, перетворення даних. Також автоенкодери виступають складовими частина архітектур інший нейронних мереж. Для задач генерації, зокрема зображень, використовують особливу архітектуру автоенкодерів, таку як варіаційні автоенкодери (VAE). Розрізняють звичайні та варіаційні автоенкодери.

Автоенкодер – це нейронна мережа, яка кодує сигнал у певний прихований стан, цим прихованим станом часто є вектор прихованого стану. Далі із цього прихованого стану дані декодуються в новий стан, формуючи вихідний сигнал. Розмірності вхідних, вихідних даних і даних у прихованому стані можуть відрізнятися. Наприклад, для генерації зображень на основі інших зображень чи ще якихось даних або масштабування зображення.

Архітектура автоенкодерів (рис. 2.5), в загальному випадку, складається із кодера та декодера. Кодер відповідає за перетворення вхідних даних у вектор прихованого стану. Декодер відповідає за розгортання вектора прихованого стану у вихідні дані, із найбільш точним їх відтворенням відносно вхідних даних [19].

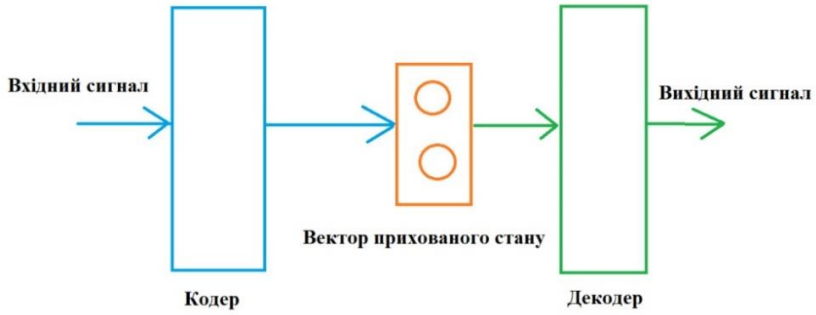


Рис. 2.5. Загальна схема автоенкодера

Вектор прихованого стану зазвичай не може вмістити всю інформацію про вхідні дані, тому він зберігає лише найбільш значущу інформацію. Розмірність вектора прихованого стану позначимо k [19].

Як відбувається кодування? Будь-яке зображення $n \times m$ можна представити як точку у $n \cdot m$ – вимірному просторі. Більшість точок цього простору відповідатимуть шумовим (беззмістовним) зображенням. Кодер в процесі навчання намагається вловити область визначення дійсних зображень у цьому багатовимірному просторі. Назвемо цю область визначення простором стану. В результаті кодер виділяє певний k-вимірний простір стану, в якому знаходяться образи дійсних зображень. В свою чергу, при декодуванні, декодер, на основі вектора прихованого стану, рухається по точкам простору стану і намагається найбільш точно відтворити зображення [19].

Щодо функцій активації, то для прихованих шарів кодера та декодера класично обирають ReLU або її варіації. Для вектора прихованого стану часто обирають лінійну функцію активації. Для вихідного шару декодера зазвичай приймають Sigmoid, але тоді дані на вході кодера і на вході вихідного шару декодера мають бути в межах від 0 до 1 [19].

Недоліком звичайних автоенкодерів є серйозні розділення простору прихованого стану, через що страждає точність створених декодером даних. Вирішенням цього є оптимізація простору прихованого стану, із цим допомагають VAE.

Варіаційні автоенкодери (VAE) дозволяють формувати компактний простір прихованого стану, без суттєвих розділень. Досягається це завдяки заданому закону розподілу. Часто приймають нормальний (Гаусівський) закон розподілу, бо він є найпростішим із точки зору проведення обчислень. Цей закон характеризується двома величинами – математичним очікуванням (μ_G) і дисперсією (σ_h^2). Вектор прихованого стану позначимо h [19].

Критерієм якості для визначення розбіжності між очікуваним і фактичним розподілом точок є дивергенція Кульбака-Лейблера (формула 1.15), для поточного випадку, коли за очікуваний результат сформований за нормальним законом отримаємо наступну формулу [19]:

$$D_{KL} = \frac{1}{2}\left(tr(\sum_G) - \mu_G^T \cdot (-\mu_G) - k + \log(\frac{1}{det\sum_G})\right) == \frac{1}{2}(tr(\sum\quad_G) + \mu_G^T \cdot\cdot\, \mu_G - -k - \log(det\sum_G)) \quad (2.5)$$

$\mu_N = [0,0,\dots,0_k]^2$ – нормальне математичне очікування

$\mu_G = [M\{h_1\}, M\{h_2\}, \dots, M\{h_k\}]^2$ – фактичне математичне очікування

$$\sum_G = \left\{\begin{bmatrix} h_1 \\ h_2 \\ h_k \end{bmatrix} \cdot [h_1, h_2, h_k]\right\} = \begin{bmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_2^2 & 0 \\ 0 & 0 & \sigma_k^2 \end{bmatrix}$$ – фактична матриця

$$\sum_N = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$ – очікувана матриця

h=$[h_1, h_2, \dots, h_k]$ – вектор прихованого стану

σ_h^2=$[\sigma_1^2, \sigma_2^2, \dots, \sigma_k^2]$ – дисперсія

В такому випадку у VAE кодер формуватиме не напряму вектор прихованого стану (h), а математичне очікування (μ_G) та дисперсію (σ_h^2), тобто два вектори. Далі, вже сам вектор прихованого стану генеруватиметься на основі векторів математичного очікування та дисперсії, за формулою:

$$h = \sigma_h \cdot N + \mu \quad (2.6)$$

де, N – число створене генератором нормальних випадкових величин

В процесі навчання VAE (рис. 2.6) будуть мінімізовуватися одразу два критерії. Перший критерій сприятиме створенню правильного вихідного

сигналу і це буде середній квадрат помилок, а другий критерій – дивергенція Кульбака-Лейблера. Результуючим критерієм якості буде їх сума.

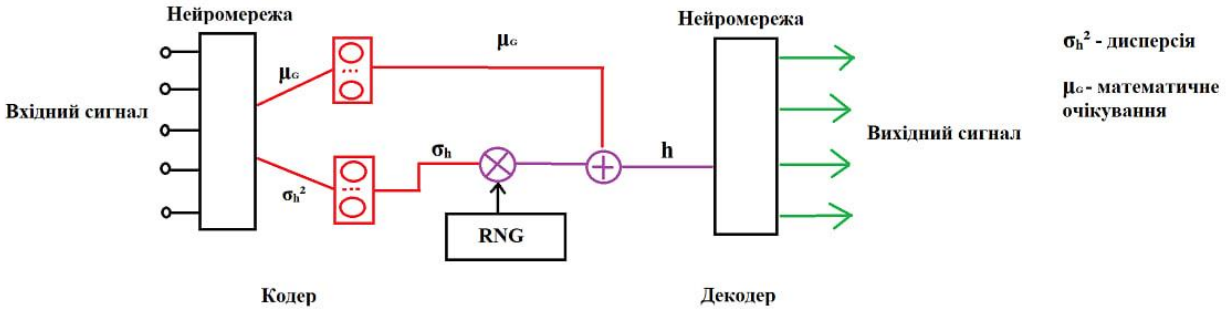


Рис. 2.6. Загальна схема VAE

2.5. Генеративно-змагальні мережі

Цей тип нейронних мереж один із тих, які найкраще підходять для задач генерації зображень. Але часто їх застосовують для масштабування зображень. Архітектурно генеративно-змагальні мережі (GAN) (рис. 2.7) складаються із двох нейромереж (як й автоенкодери) – генератора та дискримінатора. Тут одночасно навчаються генератор та дискримінатор, вони конкурують між собою. У більшості сучасних архітектур генератор і дискримінатор є глибокими згортковими мережами. Зазвичай обидві ці нейромережі використовують шар Batch Normalization. Дискримінатор намагається відрізнити реальне зображення (real) від згенерованого (fake). Генератор, у свою чергу, навпаки – намагається створити таке зображення, щоб дискримінатор не зміг його відрізнити від реального. Основним недоліком є нестабільність навчання та необхідність у значних обчислювальних ресурсах [20].

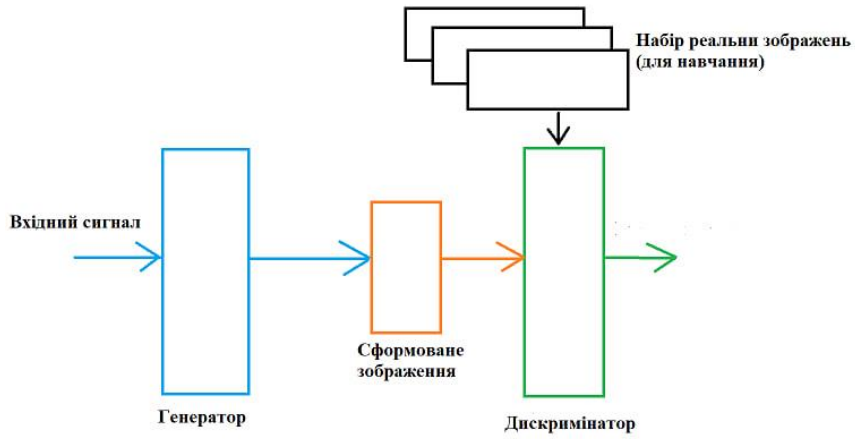


Рис. 2.7. Загальна схема GAN

Дискримінатор працює наступним чином: спочатку йому ззовні подаються реальні зображення близькі до тих, які має видавати генератор, і на виході він формує певне число (real), воно завжди прямує до 1. В інший момент часу на вхід дискримінатору подається згенероване зображення (fake), і знову формується певне число, яке завжди прямує до 0. Ці два числа виступають ступенем впевненості нейронної мережі в реальності зображення. Щоб покращити архітектуру можна замінити шари Pooling на згорткові із більшим кроком. Як функція активації на всіх шарах використовується Leaky ReLU. Показник якості візьмемо на основі бінарної кросс-ентропії (формула 1.14), оскільки має два значення, які є показником роботи нейромережі. Формула показника якості виглядатиме так [20]:

$$\text{loss\_dis} = -\log(\text{real}) - \log(1 - fake) \qquad (2.7)$$

Генератор, в свою чергу, видає згенероване зображення на яке дискримінатор повинен повернути певне число (fake), але генератор має так згенерувати зображення, щоб число fake прямувало до 1. Якщо fake → 1, то це означає, що дискримінатор не може відрізнити згенероване зображення від реального. Щодо архітектури, то доцільно використовувати деконволюційні шари. Щодо функції активації, то тут на усіх шарах окрім останнього використовується ReLU, а на останньому – Tanh. З бінарної кросс-ентропії (формула 1.14) отримуємо наступну формулу для критерію якості генератора [20]:

$$\text{loss\_gen} = -\log(fake) \quad (2.8)$$

Загалом процес навчання виглядає так: подаємо на дискримінатор реальні зображення, потім згенеровані, вираховуємо на основі виходів loss_dis, далі, за допомогою алгоритму back propagation, навчаємо дискримінатор, мінімізуючи loss_dis. Після цього подаємо на дискримінатор лише згенеровані зображення і по значенню fake навчаємо генератор, знову ж таки, через алгоритм back propagation. Отримуємо незалежне навчання цих двох нейромереж [20].

Існує модель, коли в якості генератора використовується варіаційний автоенкодер. При такій архітектурі для навчання VAE йому на вхід також потрібно подавати той самий набір реальних зображень, як і на вхід дискримінатора. Можна піти ще іншим шляхом і залишити від VAE лише декодер. В такому випадку схема архітектури виглядатиме як на рис. 2.7. Тоді на вхід генератора (декодера) будемо подавати нормальні випадкові величини (гаусівські величини) з нулевим математичним очікуванням і одиничною дисперсією.

2.6. Дифузійні моделі

Побудова та тренування дифузійних моделей ґрунтується на методах дифузії і зворотного процесу. Дифузійний (прямий) процес – це процес поступового перетворення реального зображення на випадковий шум. Зворотній процес [21] – на основі шуму модель навчається відновлювати зображення [22].

Дифузійні моделі використовуються здебільшого в генеративних задачах, але можуть використовуватися і для домальовування об’єктів на зображенні, перетворень, масштабувань зображень, покращення їхньої якості. Тож сфера застосування дифузійних моделей доволі широка.

Рис. 2.8. Схема прямого і зворотного процесів

Прямий процес можна описати формулою:

$$q(x_t|x_{t-1}) = N(x_t; \sqrt{1-\beta_t x_{t-1}}, \beta_t I) \quad (2.9)$$

де, $N()$ – багатовимірний нормальний розподіл із середнім $\sqrt{1-\beta_t x_{t-1}}$, β_t – параметр Гаусіана, задає рівень шуму (дисперсія шуму), визначає розмиття даних на кожному кроці, I – одинична матриця.

Зворотній процес можна описати формулою:

$$p_0(x_{t-1}|x_t) = N(x_{t-1}; \mu_0(x_t, t), \Sigma_0(t)) \text{ [21]} \quad (2.10)$$

де, $\mu_0(x_t, t)$ – середнє, що відповідає за реконструкцію.

$$\mu_0(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\overline{\alpha_t}}}\varepsilon_0(x_t, t)\right) \quad (2.11)$$

де, $\alpha_t = 1 - \beta_t$, $\overline{\alpha_t} = \prod_{s=1}^{t} \alpha_s$, $\varepsilon_0(x_t, t)$ – передбачений шум, ε – справжній шум $\varepsilon \sim N(0, I)$.

Переважно в подібних моделях використовується лінійне зашумлення (формула 2.12), але це не обов'язково і іноді нелінійне зашумлення буде кращим рішенням. Якщо взяти певну кількість кроків T, то лінійне зашумлення, у прямому процесі, зробить зображення нечітким за меншу кількість кроків, ніж нелінійне. Оскільки для генерації зображення виконуватиметься зворотній процес, то при нелінійному зашумленні ознаки зображення проявляться через меншу кількість кроків [22].

Лінійне зашумлення передбачає, що рівень шуму пропорційно зростає на кожному кроці, а нелінійне зростає за нелінійним законом. Наприклад, β_t у формулі 2.12 може бути квадратичною чи експоненційною функцією.

$$x_t = \sqrt{\beta_t x_0} + \sqrt{1-\beta_t \varepsilon} \quad (2.12)$$

Навчання таких моделей має наступний алгоритм: генерація шуму ε, виконання прямого процесу, моделювання зворотного процесу, мінімізація відмінностей між відновленим та оригінальним зображенням. Тобто, навчання здійснюється шляхом мінімізації різниці між ε і $\varepsilon_0(x_t, t)$. Як функція втрат використовується варіаційна втрата:

$$L = E_q[\|\varepsilon - \varepsilon_0(x_t, t)\|^2][22] \quad (2.13)$$

де, E_q – математичне очікування.

В реальних моделях робота із даними проводиться в латентному просторі, окрім того, зображення генерується невеликих розмірів (наприклад 32x32 пікселі), а потім воно масштабується з покращенням якості, для цього також використовуються нейронні мережі чи їхні окремі архітектурні складові.

Для розуміння загального підходу до побудови архітектури даного типу нейромереж наведемо схему Stable Diffusion (рис. 2.9).

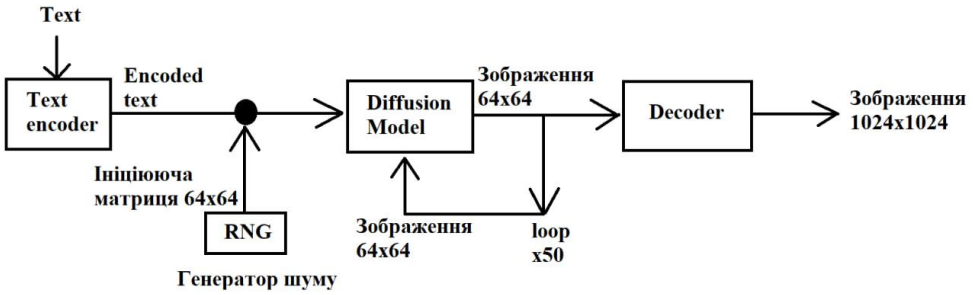


Рис. 2.9. Загальна схема Stable Diffusion

Текст із описом бажаного вихідного зображення подається на енкодер, де певним чином кодується. Далі RNG генерує вектор шуму 64x64, після цього кодований текст та шум об’єднуються. Вся робота відбувається в латентному просторі. Далі це все подається на дифузійну модель, де в циклі створюється зображення малого розміру (64x64). Потім це зображення йде на декодер, де виконується up scalling до більшого розміру. При збільшенні розміру також покращується якість і деталізація зображення.

Висновки до розділу 2

Згорткові нейромережі (CNN) є базовою технологією для роботи із зображеннями. Їхня особливість полягає у використанні згорткових шарів, які дозволяють ефективно працювати із зображеннями при невеликій кількості вагових коефіцієнтів.

Трансформери, першим призначення яких було обробка текстів, завдяки своєму потужному механізму Self-Attention чудово встановлюють просторові зв’язки між різними частинами зображення.

Автоенкодери є одним із перших підходів до генерації зображень, який ґрунтується на кодуванні вхідних даних у певний прихований стан і відновлення їх у вихідному форматі. Вдосконалені варіанти, такі як варіаційні автоенкодери дозволяють моделювати простір прихованого стану за допомогою певного закону розподілу, зазвичай нормального Гаусівського.

Особливістю генеративно-змагальних мереж є взаємодія між двома складовими моделями – генератором і дискримінатором, які шляхом «змагання» одне з одним вчаться генерувати реалістичні зображення та відрізняти реальні зображення від згенерованих.

У дифузійних моделях процес навчання базується на дифузійному процесі зашумлення реального зображення та потім на його реконструкції із шуму в зворотному процесі.

3. ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

3.1. Сфери використання нейромереж для генерації зображень

Нейромережі для створення зображень займають важливу нішу у сферах розваг, комерції, дизайну, архітектори, освітніх та наукових дослідженнях, реклами та маркетингу. Вони популярні, бо дозволяють швидко і з мінімальними затратами ресурсів створювати високоякісні візуальні матеріали. В мистецтві та індустрії розваг нейромережі дозволяють художникам та дизайнерам розширити свої творчі здібності, створюючи цифрові картини та розробляючи нові стилі. Наприклад, такі моделі, як Midjourney або DALL-E, використовуються для створення зображень на основі текстових описів, що дозволяє митцям легко ілюструвати ідеї та сюжети.

У відеоіграх та кінематографі нейромережі допомагають створювати реалістичні або стилізовані образи персонажів, фонові сцени та візуальні ефекти, які раніше потребували значних зусиль і часу. Зокрема, генеративно-змагальні мережі (GAN) дозволяють автоматично створювати безліч варіантів облич, об’єктів або локацій, що сприяє різноманітності у візуальному контенті. Ці інструменти використовуються також для «омолодження» або «старіння» акторів на екрані, а також для створення повністю цифрових персонажів із реалістичною мімікою та рухами.

Медицина – ще одна важлива сфера, де нейромережі показують високу ефективність. За допомогою нейронних мереж можна генерувати зображення, які використовуються для покращення діагностики, симуляції анатомічних структур. Наприклад, у радіології вони допомагають покращувати якість зображень з низькою роздільною здатністю або навіть реконструювати пошкоджені медичні зображення.

В рекламній та маркетинговій індустрії нейромережі дозволяють швидко створювати візуальні матеріали для рекламних кампаній. Це значно пришвидшує процес створення та персоналізації реклами, адже замість великої кількості фотосесій можна створити потрібні зображення штучним

шляхом. Персоналізація реклами також стає простішою – нейромережі можуть генерувати зображення на основі індивідуальних характеристик аудиторії або створювати варіанти для A/B тестування.

В освіті та науці генерація зображень допомагає створювати візуальні матеріали для пояснення складних понять або візуалізації даних. Нейромережі здатні не лише генерувати малюнки або схеми, але й створювати реалістичні моделі географічних структур та різного роду об’єктів, що сприяє кращому розумінню та засвоєнню матеріалу.

Для архітектури та дизайну нейромережі також є корисним інструментом, оскільки дозволяють швидко генерувати ідеї для інтер’єрів та екстер’єрів будівель, створюючи можливі варіанти оформлення на основі певних стилів чи конкретних вимог замовника. Це економить час на етапі проєктування та дозволяє замовнику побачити, як його ідеї можуть бути втілені у життя.

У сфері досліджень та розробок нейромережі для генерації зображень використовуються для тестування та симуляції, наприклад в автомобільній та робототехнічній галузях, де вони допомагають моделювати сценарії руху на дорозі або функціонування певних механізмів у віртуальному середовищі. Завдяки цьому можна заощадити на дорогих фізичних експериментах та скоротити час розробки, що в кінцевому підсумку сприяє прискоренню інновацій.

Загалом нейромережі для генерації зображень знаходять застосування у багатьох сферах, полегшуючи процеси створення контенту, оптимізуючи робочі процеси та підвищуючи ефективність у різноманітних галузях людської діяльності.

3.2. Огляд використаних програмних засобів

Для експериментальних досліджень, наведених далі у цьому розділі використовувалися найпопулярніші програмні засоби, призначені для машинного навчання.

Python – це високорівнева мова програмування, яка має зручний та простий синтаксис, велику кількість різноманітних бібліотек та дуже активну спільноту, що постійно розширює можливості Python. Ця мова часто використовується в наукових цілях, для обробки даних, штучного інтелекту, різноманітних обчислень та у багатьох інших напрямках. Перевагою Python є наявність великої кількості готових бібліотек та модулів, що дозволяють виконувати надзвичайно широкий спектр задач та значно прискорити розробку. Бібліотеки NumPy, Matplotlib, TensorFlow, Keras, PyTorch активно використовуються у сферах машинного навчання та обробки даних.

Google Colab – хмарне середовище на базі Jupyter Notebook, забезпечує доступ до GPU та TPU, отже дозволяє проводити всі необхідні обчислення. Крім того дозволяє зручно організовувати дані.

NumPy – бібліотека Python для роботи із багатовимірними масивами та матрицями, є основою для багатьох інших бібліотек, зокрема Tensorflow та PyTorch.

TensorFlow – бібліотека машинного навчання, яка надає широкий набір інструментів для створення та навчання нейромереж. В основі роботи лежить обчислювальний граф.

Keras – високорівнева бібліотека, частина TensorFlow, створена для швидкої й легкої побудови нейронних мереж, надає інтуїтивно зрозумілий інтерфейс.

PyTorch – бібліотека, яка надає зручний і зрозумілий інтерфейс для створення нейромереж. В основі роботи лежить динамічний обчислювальний граф.

Matplotlib – бібліотека для візуалізації даних у Python, яка дозволяє створювати різноманітні типи графіків, діаграм і візуальних представлень даних. Вона підтримує як прості лінійні графіки, так і складні 3D-візуалізації, гістограми, тепло карти та анімовані графіки. Matplotlib є важливим інструментом для аналізу даних, оскільки забезпечує наочне представлення інформації, що дозволяє зрозуміти й інтерпретувати дані.

3.3. Опис проведених експериментів

У даній роботі було проведено кілька експериментів для перевірки на практиці результатів досліджень. Дослідження проводилися із використанням бібліотек NumPy, Tensorflow, PyTorch, Matplotlib, мови програмування Python та хмарного середовища Google Colab. В експериментах брали участь по одній моделі кожного типу генеративних нейромереж StyleGAN2 (GAN), Stable Diffusion (Diffusion Model), DALL-E (трансформер), варіаційний автоенкодер (VAE). Розглянемо ці моделі, набори даних та метрики, які були використані в експериментальних дослідженнях.

StyleGAN2 – нейромережа, яка має генеративно-змагальну архітектуру, призначена для генерації реалістичних зображень. Використовує ефективний механізм нормалізації adaptive instance normalization. Stable Diffusion – дифузійна модель для генерації зображень на основі текстових описів. Ця модель компактна та може запускатися навіть на звичайних відеокартах. Також є можливість навчати дану модель на власних даних для специфічних потреб. DALL-E – трансформер, генерує зображення на основі текстових описів, завдяки використанню трансформерів може ефективніше аналізувати текст. Серед недоліків, – модель може створювати некоректні зображення з точки зору логіки або помилятися в складних запитах. VAE – здатен генерувати нові зразки завдяки тому, що додає випадковість до латентного простору, на відміну від класичних автоенкодерів. Серед недоліків – найменша точність у генерації високоякісних зображень.

Функції втрат, використані у експериментах. Для StyleGAN2 – Wasserstein Loss із регуляризацією. Регуляризація додається, щоб забезпечити гладкість функції дискримінатора; Hinge Loss запобігає надмірно великим значенням дискримінатора; LSGAN Loss (Least Squares GAN) зменшує проблему затухаючих градієнтів. Для Stable Diffusion – MSE (Mean Squared Error), середньоквадратична помилка; MAE (Mean Absolute Error), замість квадратичної різниці використовує абсолютну; Huber Loss (Smooth L1 Loss), поєднує переваги MSE і MAE, використовує квадратичну різницю для малих

значень помилки та абсолютну для великих. Для DALL-E – Cross-Entropy Loss; MSE; Perceptual Loss. Для VAE – ELBO, MSE + дивергенція Кульбака-Лейблера; MI Loss (Mutual Information), максимізує взаємну інформацію між вхідними даними та латентним представленням; WAE (Wasserstein Autoencoder).

Для навчання моделей використовувався один й той самий набір даних CIFAR-100. Цей датасет складається з невеликих кольорових зображень, розміром 32x32 пікселі. Зображення представлені у форматі RGB. Весь датасет розділений на 100 класів та 20 суперкласів. Розмір вибірки – 60 тисяч зображень, 50 тисяч – навчальна вибірка та 10 тисяч – тестова вибірка.

Метрики, використані для оцінки згенерованих зображень FID, IS, візуальна оцінка. FID – Frechet Inception Distance, вимірює відстань між розподілами реальних та згенерованих зображень у латентному просторі, визначеному мережею Inception, зазвичай використовується Inception-v3. Інтерпретувати цю метрику можна так: чим менше значення, тим ближчі згенеровані зображення до реальних. З недоліків: не завжди низький FID означає гарну якість, а також залежність від моделі Inception. IS – Inception Score, оцінює якість та різноманітність згенерованих зображень на основі класифікації, що виконана мережею Inception. Ця метрика перевіряє наскільки легко згенеровані зображення класифікуються, та чи різні класи генеруються з однаковою ймовірністю. Інтерпретація, високий IS означає, що згенеровані зображення мають високе різноманіття та чітко належать певним класам. Серед недоліків можна визначити, що зображення не порівнюються із реальними, залежність від моделі Inception. Серед переваг – легкість розрахунків.

Перший експеримент – це порівняння якості згенерованих зображень, яке проводилося між чотирма описаними вище моделями за допомогою описаних вище метрик. Як функції втрат використані:

- StyleGAN2 – Wasserstein Loss із регуляризацією;
- Stable Diffusion – MSE;

– DALL-E – Cross-Entropy Loss;

– VAE – ELBO.

Другий експеримент – це визначення впливу різних функцій втрат на якість генерації. Використовувалися ті самі моделі та метрики, використовувалося по дві нові функції втрат для кожної моделі, але також враховувалися результати попереднього експерименту. Використані функції втрат:

– StyleGAN2 – Hinge Loss, LSGAN Loss;

– Stable Diffusion – MAE, Huber Loss;

– DALL-E – MSE, Perceptual Loss;

– VAE – MI Loss, WAE.

Для VAE, DALL-E, Stable Diffusion експерименти проводилися зі значеннями розміру батчу та латентного простору по 128 та 2048. Для StyleGAN2 ці значення були 64 – розмір батчу та 128 і 1024 – розмір латентного простору.

3.4. Аналіз отриманих результатів

Перший експеримент – порівняння якості згенерованих зображень за допомогою моделей описаних у підрозділі 3.3. Результати викладемо у таблиці 3.1. Графіки втрат та згенеровані зображення у порівнянні із зображеннями з тренувального набору можна переглянути у Додатку А. Для StyleGAN2 Training Loss це помилка на генераторі, а Validation Loss – це помилка на дискримінаторі.

Таблиця 3.1

Порівняння якості згенерованих зображень експерименту 1

Модель та функція активації	Розмір батчу та латентного простору	FID	IS	Training Loss	Validation Loss
VAE (ELBO)	128, 128 2048, 2048	1545.6937 1288.5579	1.00 1.00	0.0050 0.0003	0.0051 0.0003
DALL-E (Cross-Entropy)	128, 128 2048, 2048	1085.1683 1179.9853	1.00 1.00	0.0042 0.0003	0.0042 0.0003

StyleGAN2 (Wasserstein Loss)	64, 128 64, 1024	1268.9309 1196.2641	1.00 1.00	17.7616 2.3081	-12.8966 -0.7176
Stable Diffusion (MSE)	128, 128 2048, 2048	1096.8337 1236.8875	1.00 1.00	0.00003 0.000005	0.00003 0.000005

Як можна побачити, параметр IS завжди був рівний одиниці, що свідчить про те, що згенеровані зображення слабко піддавалися класифікації і не були різноманітними. Параметр FID знижувався зі збільшенням розміру батча та латентного простору, але постійно був високим, а це означає, що згенеровані зображення були далекими від реальних. Якщо говорити про візуальні спостереження, то найкраще впоралася модель StyleGAN2. Спостерігалася чітка тенденція до формування окремих об’єктів. Модель VAE видала найгірший результат, оскільки там були лише набори пікселів або слабкі акценти на окремих ділянках зображення. Моделі DALL-E та Stable Diffusion генерували зображення з багатьма акцентами на різних ділянках зображення, але в них більше проглядалися текстури ніж окремі об’єкти. VAE, DALL-E навчалися приблизно однаковий проміжок часу, довше Stable Diffusion, StyleGAN2 навчалася найдовше. Детальніше по кожній моделі:

VAE – просто набір пікселів певних кольорів, результат трохи покращився зі збільшенням розміру батчу та латентного простору, з’явилися певні акцентовані ділянки зображення. При навчанні на значеннях 128 спостерігалася чітка різниця між значеннями помилок на валідаційній та тестовій вибірках. Проте, при значеннях 2048 навчання проходило чудово. Результати можна переглянути в додатку А.

DALL-E – при значеннях характеристик 128 спостерігалися яскраві кольори на зображеннях, зі збільшенням цих параметрів зображення кольори ставали менш яскравими, а також значно знизилась концентрація синього кольору. Загалом можна сказати про наявність певних текстур на згенерованих зображеннях, коли розмір батчу та латентного простору 2048. Навчання при значеннях 128 йшло гірше, оскільки до 60 епохи збільшувалась різниця між валідаційною і тестовою вибірками. При значеннях 2048 розбіжність була

мінімальна, але графік менш згладжений. Результати можна переглянути в додатку Б.

StyleGAN2 – при генерації спостерігалася велика кількість артефактів, тому було застосовано більше оптимізацій та перетворень, це дозволило трохи покращити результат. Зі збільшенням розміру латентного простору кількість артефактів значно знизилася, а також на згенерованих зображеннях з'явилися ділянки із більшою присутністю ознак, що свідчить про тенденцію моделі до створення повноцінних окремих об'єктів. Помилки генератора і дискримінатора мали чіткі тенденції до зниження, проте вони весь час сильно «стрибали», при чому при розмірі латентного простору 1024 ці «стрибки» посилились. Така поведінка характерна для цієї моделі оскільки дві нейромережі навчаються одна відносно одної. Дану модель навчати було найскладніше через постійну неоднорідність помилок та артефакти на зображеннях. Результати можна переглянути в додатку В.

Stable Diffusion – тут результати схожі з попередньою моделлю, але зображення є більш деталізованими. Навчання проходило чудово, при значеннях характеристик 2048 спостерігалося менше згладження графіка. Результати можна переглянути в додатку Г.

Другий експеримент – порівняння впливу різних функцій втрат на якість генерації зображень. Деякі моделі майже не змінювали свої результати із заміною функції втрат, але були й випадки коли результати сильно відрізнялися. Розглянемо детальніше кожну модель.

VAE – перша альтернативна функція це MI Loss, результати генерації із нею майже не відрізнялися від ELBO, а помилка збільшилась вдвічі. Із функцією WAE зображення набули чіткіших обрисів, із яскраво вираженими ділянками при розмірі батчу та латентного простору 128, а при значенні 2048 спостерігається схильність моделі до роботи із текстурами. При використанні WAE помилка майже така сама як при ELBO. Навчання добре проходило на усіх функціях втрат при значеннях характеристик 2048, а при 128 найкраще

навчання проходило на WAE, найгірше при MI Loss, тут найбільша різниця між помилками на двох вибірках. Результати можна переглянути в додатку А.

DALL-E – Perceptual Loss, на відміну від Cross-Entropy, при значеннях характеристик 128 кольори зображень набагато яскравіші, а також можна розрізняти окремі абстрактні об’єкти, а при значеннях 2048 згенеровані зображення просто сірі з рідкісними кольоровими пікселями. Отже, для даної функції потрібно підбирати менше значення латентного простору. При використанні MAE результати дуже подібні до Cross-Entropy, проте при значенні характеристик 128 присутньо більше рожевого та фіолетового кольорів. Найкраще проходило навчання для функції MSE при обох значеннях характеристик. При функції Perceptual Loss сильно зростала відмінність між тестовою і валідаційною вибірками при значеннях характеристик 128, різниця зменшилась при значеннях 2048. Результати можна переглянути в додатку Б.

StyleGAN2 – при усіх функціях, вже при розмірі батчу 64 та латентного простору 128 можна розгледіти чіткі текстури або певні об’єкти. Лише Wasserstein Loss на цьому етапі генерує багато артефактів. Найменше артефактів при цих значеннях характеристик (батч – 64, латентний простір – 128) видавала модель із функцією втрат Hinge Loss. LSGAN Loss також давав багато артефактів, але все одно розрізнялися об’єкти та текстури. При розмірі латентного простору 1024 (батч -64) текстури та об’єкти видно трохи чіткіше, але покращення не дуже сильне. На зображеннях згенерованих моделлю із LSGAN Loss знову ж таки присутні артефакти. Найкраща якість при Hinge Loss. Навчання загалом можна охарактеризувати дуже великими відмінностями між помилками, графіки помилок генератора та дискримінатора рідко перетинаються та неохоче наближаються одна до одної. При використанні LAGAN Loss спостерігалося цікаве явище при навчанні, графіки помилок на дискримінаторі та генераторі мінялися місцями у вертикальній площині. При використанні функції Hinge Loss графіки помилок майже не наближаються одне до одного і не перетинаються, кожен з них не покидає свою вертикальну площину. Результати експериментів в додатку В.

Stable Diffusion – при значеннях 128 на усіх функціях спостерігається набір яскравих кольорів, Huber Loss відрізняється тим що має більше зеленого кольору, а решта дві більше рожевого, фіолетового. На усіх таких зображеннях можна розрізняти певні абстрактні об’єкти. При значеннях 2048 при усіх функціях втрат спостерігалася тенденція до створення текстур, і всі вони приблизно однакові. Навчання на усіх трьох функціях мало гарну тенденцію, лише рідко з’являлися неоднорідності. Отже, ця модель найлегше навчається із представлених у цій роботі. Результати можна переглянути в додатку Г.

Підведемо підсумки. Модель VAE показує найкращі результати генерації при використанні функції втрат WAE разом зі значенням розміру батчу та латентного простору 2048. Навчання ефективне при значеннях параметрів 2048. Для моделі DALL-E найкращою є MSE, оскільки якість генерації така сама як і при Cross-Entropy, а навчання відбувається краще. StyleGAN2, найкраща генерація при використанні функції Hinge Loss, хоча при навчанні графіки помилок генератора та дискримінатора не наближаються одне до одного. Зі Stable Diffusion всі представлені функції втрат працюють доволі ефективно. По генерації різниці майже немає, а при навчанні найкраще себе показує функція втрат MAE. Порівняння результатів експерименту 2 представимо у табл. 3.2.

Таблиця 3.2

Порівняння результатів експерименту 2

Модель	Функція активації	Розмір батчу та латентного простору	FID	IS	Training Loss	Validation Loss
VAE	MI Loss	128, 128 2048, 2048	1899.2275 1544.5630	1.00 1.00	0.0130 0.0008	0.0132 0.0008
	WAE	128, 128 2048, 2048	1295.8879 1271.1597	1.00 1.00	0.0046 0.0003	0.0057 0.0003
DALL-E	Perceptual Loss	128, 128 2048, 2048	989.0345 1615.1552	1.00 1.00	0.0001 0.000012	0.0002 0.000015
	MSE	128, 128 2048, 2048	1164.9207 1235.8547	1.00 1.00	0.000017 0.000013	0.000017 0.000013
StyleGAN2	Hinge Loss	64, 128 64, 1024	1147.6701 1152.5728	1.00 1.00	-0.1922 -0.0853	-1.9633 1.7892
	LSGAN Loss	64, 128 64, 1024	1167.8127 1342.2578	1.00 1.00	0.2995 0.2735	0.4499 0.4368

Stable Diffusion	MAE	128, 128 2048, 2048	1095.1281 1187.2999	1.00 1.00	0.00003 0.00002	0.000035 0.00002
	Huber Loss	128, 128 2048, 2048	1171.9208 1213.4270	1.00 1.00	0.00002 0.000025	0.00002 0.000025

3.5. Можливості інтеграції із іншими технологіями

Інтеграція нейромереж для генерації зображень з іншими технологіями відкриває широкі можливості для вдосконалення багатьох сфер діяльності.

Технології віртуальної та доповненої реальності (VR/AR). Генеративні нейромережі здатні автоматично створювати реалістичні об’єкти, що спрощує процес розробки VR/AR контенту. Крім того це забезпечуватиме більш реалістичний ефект присутності завдяки реагуванню у реальному часі на зміну освітлення чи текстури віртуальних об’єктів. Також забезпечуються значні можливості персоналізації для кожного користувача, що позитивно впливатиме на користувацький досвід. Також це дозволяє зменшити витрати на 3D моделювання та художню і графічну роботу.

Інтеграція з технологіями IoT дає можливість використовувати нейромережі для генерації зображень в реальному часі на основі даних із сенсорів і пристроїв IoT. Наприклад, у сфері сільського господарства нейромережі можуть створювати візуалізації на основі показників, зібраних сенсорами, таких як рівень вологості, температура, стан ґрунту, для моніторингу стану полів і прогнозування врожайності. Водночас це сприяє оптимізації систем контролю і обслуговування складних промислових процесів.

Висока потреба в обчислювальних ресурсах робить хмарні платформи зручним середовищем для запуску складних моделей. Це забезпечить легкий доступ користувачам до потужних нейромереж. Також використання хмарних обчислень дозволить розподіляти процес навчання на кілька пристроїв, значно підвищуючи його ефективність і швидкість.

Інтеграція із текстовими та голосовими технологіями. Системи Text-to-Image та Image-to-Text дозволяють генерувати зображення на основі

текстового опису та навпаки – генерувати текстовий опис того, що знаходиться на зображенні. Це має значний потенціал для генерації вузькоспеціалізованих зображень. Те саме і з голосовими інтерфейсами. Це розширює можливості сервісів генерації зображень на основі генеративних нейромереж та підвищує зручність використання даних сервісів.

Технології обробки значних об’ємів даних. Тут знову ж таки випливає генерація персоналізованого контенту у, наприклад, соціальних мережах, на основі обробки значних масивів даних про користувача чи певну групу користувачів. Також можлива генерація візуальних даних для візуалізації аналітичних даних, наприклад створення інфографіки чи анімованих графіків на основі великих об’ємів даних.

Також це й інтеграція в повсякденні додатки (Photoshop, Canva). Нейромережі можуть значно підвищити область задач, які зможуть виконувати програми. Симбіоз вже наявних в подібних редакторах стандартних алгоритмів обробки зображень із генеративними моделями, чи з будь-якими нейромережами, значно підвищить швидкість виконання задач, знизить поріг входження нових користувачів, підвищить зручність у використанні досвідченими користувачами.

Висновки до розділу 3

Нейромережі для генерації зображень стали важливим інструментом у багатьох галузях, таких як розваги, дизайн, медицина, освіта, маркетинг та ін. Вони дозволяють автоматизувати створення візуального контенту, що підвищує ефективність роботи певних відділів та економить ресурси. Для реалізації задач порівняння роботи різних генеративних моделей використовувалися потужні інструменти: бібліотеки Tensorflow, Keras, NumPy, Matplotlib, хмарне середовище Google Colab. Щодо інтеграції з іншими технологіями, то найбільш перспективними є VR/AR, хмарні платформи, технології IoT, текстові та голосові технології. Генеративні моделі тут значно допоможуть із персоналізацією контенту. Також генеративні

моделі активно впроваджуються у програмні засоби для роботи із зображеннями (Photoshop, Canva). Це значно підвищує область задач, які зможуть виконувати програми.

За результатами експериментальних досліджень, наведеними у підрозділі 3.4 та у додатку А, можна зазначити, що найкраще із задачею генерації впоралися моделі StyleGAN2 із функцією втрат Hinge Loss та Stable Diffusion із функцією втрат MAE (Mean Absolute Error). У них спостерігається чітка тенденція до покращення якості згенерованих зображень. У StyleGAN2 на відміну від Stable Diffusion спостерігаються певні проблеми із навчанням, але це характерно для даного типу нейромереж. Слід зазначити, що для усіх моделей спостерігається низьке значення метрики IS, а це означає, що згенеровані зображення погано піддаються класифікації, отже є нечіткими. Метрика FID навпаки на усіх моделях є доволі високою, що свідчить про значне розходження згенерованих та оригінальних зображень, а це вказує на низькі можливості генерування конкретних образів. З огляду на дані результати можна стверджувати про позитивні тенденції зростання якості генерації із застосуванням різних конфігурацій даних моделей.

ВИСНОВКИ

Основною метою даної роботи було визначення і аналіз основних засобів та підходів до організації генеративних нейронних мереж. У ході виконання був здійснений комплексний аналіз сучасних підходів у сфері генерації зображень. У рамках теоретичного дослідження було охоплено ключові аспекти роботи генеративних моделей, пов’язані з математичною та архітектурною складовою. Що стосується математичної складової, то були розглянуті математичні основи функціонування генеративних нейромереж, методи навчання, оптимізації, підрахунку точності і коректності роботи моделей. Щодо архітектурних рішень, то розглянуто як ті, що застосовуються у генеративних моделях, так і ті, що часто використовуються як допоміжні, для виконання окремих функцій або використовуються як основа для роботи із зображеннями чи текстовими описами.

Одним з важливих аспектів роботи стало дослідження архітектур нейромереж. Розглянуто сучасні тенденції, з чого можна виділити, що на даний момент спостерігається чітка тенденція до використання генеративно-змагальних моделей (GAN) та дифузійних моделей. Інші ж моделі, такі як автоенкодери та трансформери є скоріше допоміжними. Окремо слід виділити згорткові нейромережі (CNN). Їхні згорткові шари використовуються у більшості моделей, пов’язаних із роботою над зображеннями, в тому числі й у генеративних моделях, у цих моделях також часто застосовуються деконволюційні шари. Постійно зростає глибина нейронних мереж, а це означає, що постійно зростають вимоги до апаратного забезпечення, через потребу в обробці все більшої кількості даних та створення зображень все кращої якості та деталізації.

Для створення нейромереж існує велика кількість програмних засобів та сервісів. Деякі із цих засобів були розглянуті і використані у даній роботі. Особливо слід відзначити бібліотеки Tensorflow та PyTorch.

Теоретичні дослідження супроводжувались практичними експериментами. У роботі здійснено порівняння роботи генеративних моделей та порівняння результатів із використанням різних функцій втрат. Крім того розглянуто також можливі сфери застосування даного типу нейромереж та сучасні тенденції їхнього розвитку.

Також у роботі здійснено оцінку можливостей інтеграції генеративних моделей із іншими сучасними технологіями. На мою думку найбільш перспективно виглядає інтеграція із системами VR/AR, де генеративні моделі дозволять забезпечити більш реалістичний ефект присутності завдяки високодеталізованому згенерованому контенту. Також це дозволить досягти високого рівня персоналізації, що важливо для покращення користувацького досвіду. На друге місце я постав інтеграцію із технологіями обробки значних обсягів даних. Тут генеративні нейромережі значно допоможуть зі створенням більш наочного візуального представлення даних. Також не слід забувати про сферу маркетингу, де дані нейромережі вже займають важливу нішу та інтеграцію із програмним забезпеченням для роботи із зображеннями (Photoshop, Canva).

Для кращого розуміння принципів та особливостей роботи генеративних нейромереж було проведено ряд експериментальних досліджень. Ці експерименти найкраще демонструють особливості роботи різних генеративних нейронних мереж та як може варіюватися результат, залежно від заданих гіперпараметрів чи вибору функції втрат. Така залежність також справедлива й для інших складових архітектури, таких як: функції активації, алгоритм навчання, функції оптимізації тощо. Детально опис проведених експериментів та аналіз отриманих результатів наведено у підрозділах 3.3 і 3.4.

Експериментальні дослідження показали важливість вивчати можливості застосування різних функцій втрат при розробці генеративних моделей. У такій самій мірі це стосується й інших складових генеративних нейромереж. Потрібно підбирати глибину архітектури, розміщення шарів, функції активації, різноманітні оптимізації, гіперпараметри тощо. Усі ці

елементи тісно взаємопов’язані і тому важко точно наперед передбачити результати навчання та роботи нейронної мережі. Крім основних елементів, потрібно також ретельно підбирати критерії оцінки якості роботи нейромережі, оскільки різні метрики будуть більш чи менш показовими для різних моделей. Існує кілька універсальних метрик, які і були використані у даній роботі (FID та IS). Також під час експериментів вдалося оцінити потреби у обчислювальних ресурсах, які є значними навіть для порівняно простих моделей, які були представлені у роботі. Не дивлячись на використання потужного «заліза» наданого Google Colab все одно навчання моделей потребувало значного часу, наприклад навчання моделі StyleGAN2 могло затягнутися до 2,5 год.

На основі результатів дослідження викладених у роботі можна виділити кілька рекомендацій для подальшого розвитку генеративних моделей. Доцільно зосередити увагу на оптимізації апаратної складової під час навчання, наприклад використання кластерів із декількох GPU або TPU для прискорення процесу навчання моделей. Для досягнення кращих результатів у генерації зображень рекомендується експериментувати із комбінуванням наявних архітектур, наприклад дифузійні моделі та GAN. Перспективною є інтеграція генеративних нейромереж із технологіями машинного аналізу текстів для створення мультимодальних моделей, які працюють одночасно з текстовими та візуальними даними. Окремо слід зазначити важливість розробки більш адаптивних функцій втрат, які могли б враховувати специфіку вхідних даних та потреби кінцевих користувачів. Не менш важливими є експерименти із використанням різних функцій активації та оптимізації, їх комбінування і підбір необхідних параметрів. Нарешті важливо розробляти інтерфейси, які полегшували б інтеграцію у вже існуючі системи, що дозволить розширити сферу їх використання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Методи та технології обчислювального інтелекту: Навчальний посібник для студ. Спеціальності 122 «Комп’ютерні науки» / І. В. Федорін; КПІ ім. Ігоря Сікорського. Київ КПІ ім. Ігоря Сікорського, 2022. 73 с.
2. Методи та системи штучного інтелекту: Навчальний посібник для студентів напряму підготовки 6.050101 «Комп’ютерні науки» / Уклад. : А. С. Савченко, О. О. Синельніков. Київ: НАУ, 2017. 109 с.
3. Микола КОНДРУШЕНКО. Ефективність глибоких згорткових нейронних мереж у розпізнаванні зображень. Збірник матеріалів наукової конференції за підсумками науково-дослідної роботи здобувачів вищої освіти фізико-математичного факультету Кам’янець-Подільського національного університету імені Івана Огієнка у 2023-2024 н.р., 9-10 квітня 2024 року [Електронний ресурс]. Кам’янець-Подільський : Кам’янець-Подільський національний університет імені Івана Огієнка, фізико-математичний факультет, 2024. С. 53-57.
4. Микола КОНДРУШЕНКО. Порівняльний аналіз нейромереж-трансформерів в задачах генерації зображень. Вісник Кам’янець-Подільського національного університету імені Івана Огієнка. Фізико-математичні науки. Випуск 17. Кам’янець-Подільський : Кам’янець-Подільський національний університет імені Івана Огієнка, 2024. С. 56-61.
5. Пилипюк Т., Кондрушенко М. Застосування згорткових нейронних мереж. Актуальні аспекти розвитку STEAM-освіти в умовах євроінтеграції: збірник матеріалів ІІ Міжнародної науково-практичної інтернет-конференції (м. Кропивницький, 26 квітня 2024 року). Кропивницький : ДонДУВС, 2024. С. 219-220. https://dnuvs.ukr.education/wp-content/uploads/2024/06/zbirnyk_tez_konferencziya_steam_26_04_2024.pdf
6. Sahil Sidheekh, Sriraam Natarajan. Building expressive and tractable probabilistic generative models: A Review.
7. Diederik P. Kingma, Jimmy Lei Ba. Adam: A method for stochastic optimization.

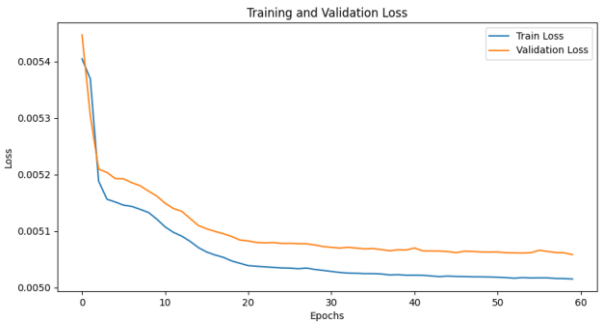
8. Review and comparison of commonly used activations functions for deep neural networks.
9. Soham De, Anirbit Mukherjee, Enayat Ullah. Convergence of adaptive gradient methods
10. Yarin Gal, Zoubin Ghahramani. Dropout as a Bayesian Approximation: representing model uncertainty in deep learning.
11. Sergey Ioffe, Christian Szegedy. Batch Normalization: accelerating deep network training by reducing internal covariate shift.
12. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classifications with deep convolutional neural networks. In Advances in neural information processing systems.
13. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional Networks for large-scale image recognition.
14. Attentions is all you need 12 Jun 2017 (перевидано 2 Aug 2023).
15. Animag is worth 16x16 words: tranformers for image recognitional at scale 22 Oct 2020 (перевидано 3 Jun 2021).
16. Deep residual learning for image recognition 10 Dec 2015.
17. Swin Transformer: hiererchical vision transformer using shifted windows 17 Aug 2021.
18. Swin transformer 2: scalling upcapacity and resolution. Apr 2022.
19. Diederik P. Kingma, Max Welling. Auto-Encoding Variational Bayes. 10 Dec 2022.
20. Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Shefjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. 10 Jun 2014.
21. Jonathan Ho, Ajay Jain, Pieter Abbeel. Denoising Diffusion Probabilistic Models. 16 Dec 2020.
22. Alex Nichol, Prafulla Dhariwal. Improved Denoising Diffusion Probabilistic Models. 18 Feb 2021.

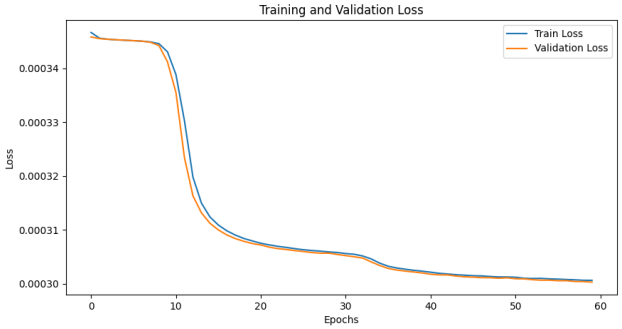
Додатки

Додаток А

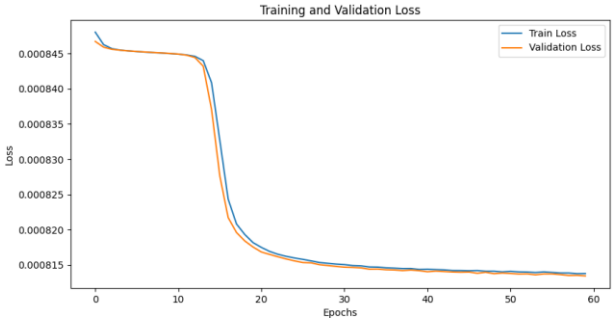
Навчання та згенеровані зображення за допомогою моделі VAE

Функція активації ELBO для батчу і латентного простору 128 і 2048

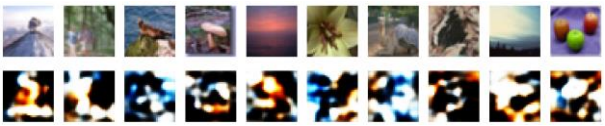


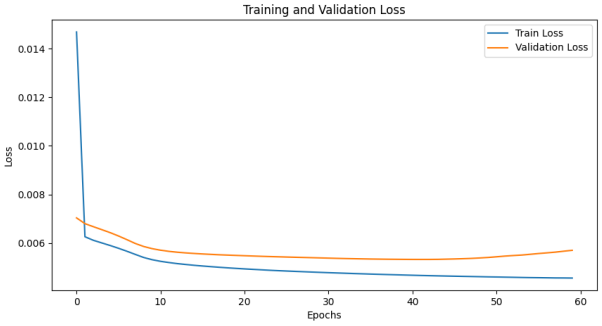


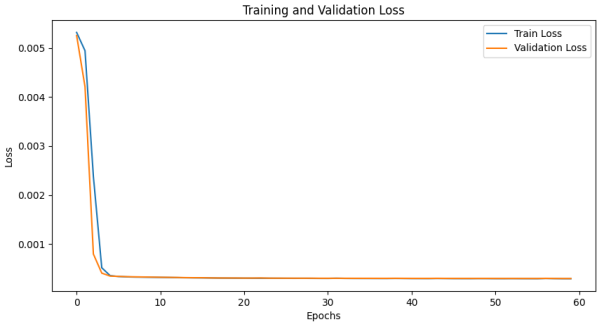
Функція активації MI Loss для батчу і латентного простору 128 і 2048



Функція активації WAE для батчу і латентного простору 128 і 2048



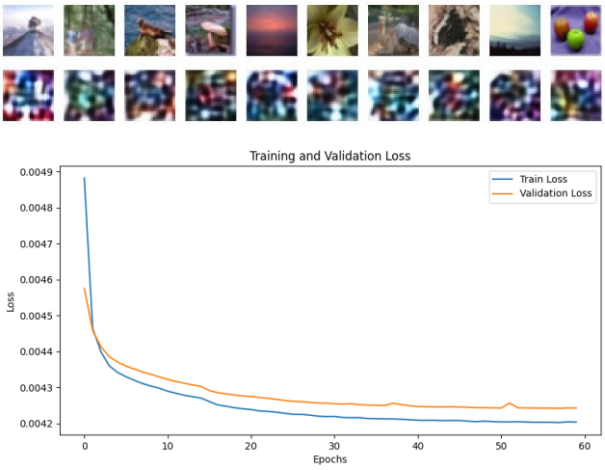


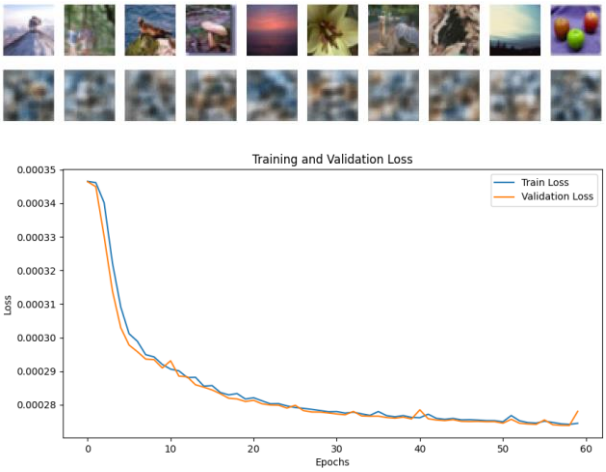


Додаток Б

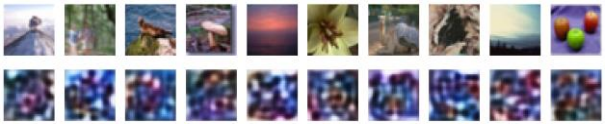
Навчання та згенеровані зображення за допомогою моделі DALL-E

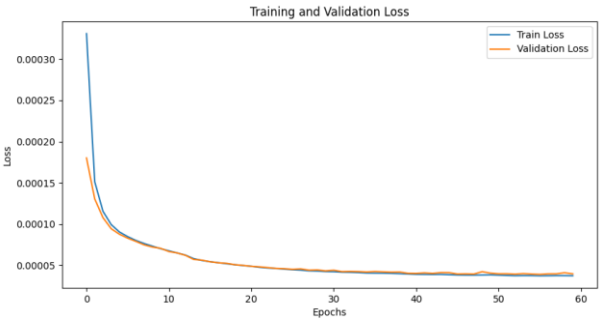
Функція активації Cross-Entropy для батчу і латентного простору 128 і 2048



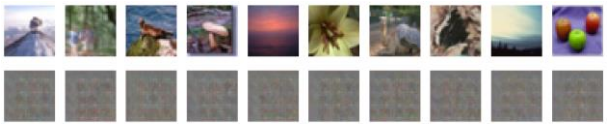


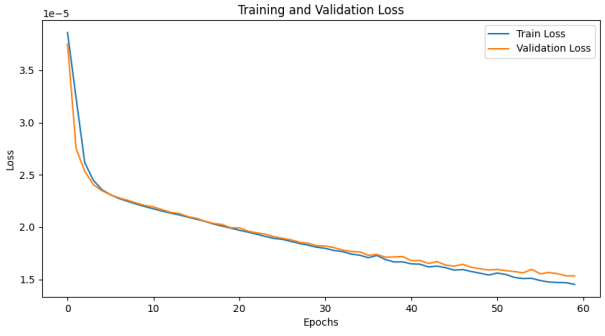
Функція активації MSE для батчу і латентного простору 128 і 2048





Функція активації Perceptual Loss для батчу і латентного простору 128 і 2048

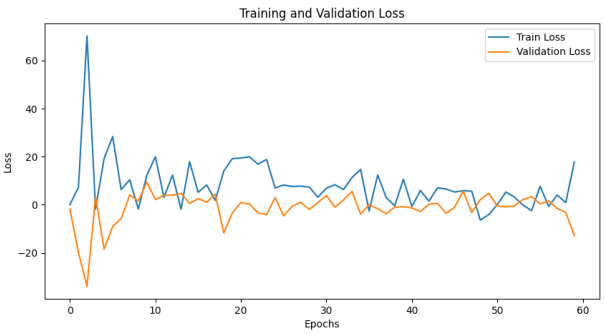


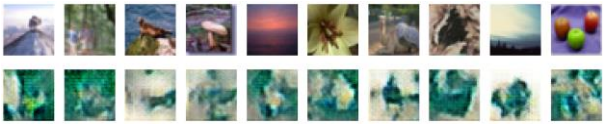


Додаток В

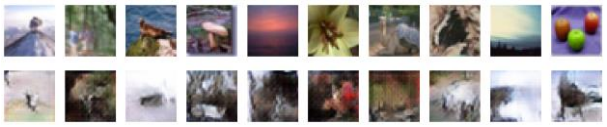
Навчання та згенеровані зображення за допомогою моделі StyleGAN2

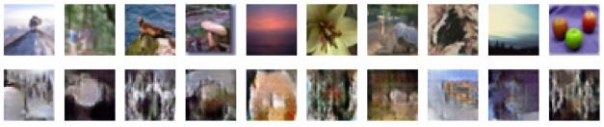
Функція активації Wasserstein для батчу 64 і латентного простору 128 і 1024



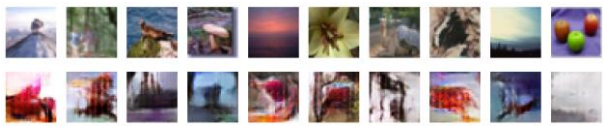


Функція активації Hinge Loss для батчу 64 і латентного простору 128 і 1024





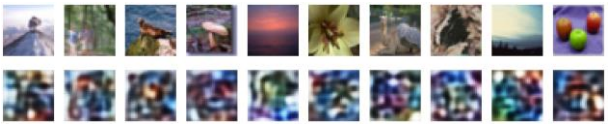
Функція активації LSGAN Loss для батчу 64 і латентного простору 128 і 1024

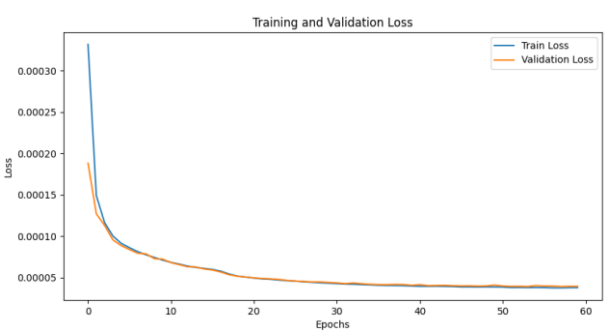


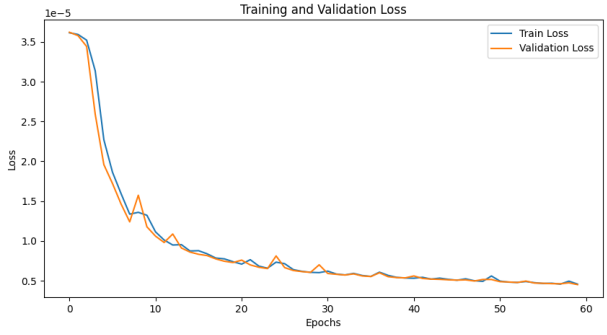
Додаток Г

Навчання та згенеровані зображення за допомогою моделі Stable Diffusion

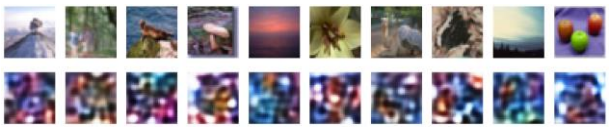
Функція активації MSE для батчу і латентного простору 128 і 2048

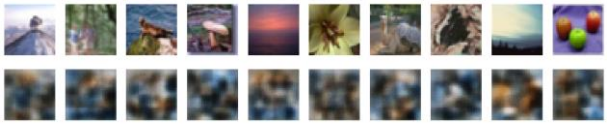






Функція активації MAE для батчу і латентного простору 128 і 2048





Функція активації Huber Loss для батчу і латентного простору 128 і 2048