КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
МЕХАНІКО-МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
КАФЕДРА ТЕОРІЇ ЙМОВІРНОСТЕЙ,
СТАТИСТИКИ ТА АКТУАРНОЇ МАТЕМАТИКИ

ДИПЛОМНА РОБОТА
НА ЗДОБУТТЯ СТУПЕНЯ МАГІСТРА
НА ТЕМУ:

"ПЕРЕНОС СТИЛЮ НА ЗОБРАЖЕННЯХ ЗА ДОПОМОГОЮ ГЕНЕРАТИВНИХ ЗМАГАЛЬНИХ НЕЙРОННИХ МЕРЕЖ"

Виконав:
студент 2 курсу магістратури
механіко-математичного факультету
спеціальності «Статистика»
Ширченко Максим Вікторович

Науковий керівник:
кандидат фізико-математичних наук,
доцент кафедри теорії ймовірностей,
статистики та актуарної математики
Голомозий Віталій Вікторович

Допустити до захисту ДЕК
Протокол No _____ від _____ р.

Зав. кафедрою
_____ Мішура Ю.С.

КИЇВ
2021

Зміст

1 Вступ

На початку 21 століття комп'ютери могли робити безліч складних речей, деколи навіть переважаючи людей. Комп'ютер міг перемогти вас в шахи, допомогти знайти щось корисне, чи навіть допомогти тримати в безпеці ваш будинок. Все це за допомогою машинного навчання (ML, скорочено від machine learning). Алгоритми машинного навчання дуже добре справляються з двома задачами - класифікації (присвоєння данним мітки класу) та регресії (передбачення данних). Все це через їх здібність до "запам'ятовування" шаблонів в данних, переданих для навчання. Але данний клас алгоритмів стикається з проблемою, коли потрібно згенерувати якісь данні. З їх допомогою можна передбачити ціни, знайти об'єкти на зображеннях, але будь-яка спроба згенерувати будь-який контент, для прикладу, зображення або ж звук, закінчиться невдачею. Тому потрібне було нове рішення - здатне справитись з такого типу завданнями.

В 2014 році Ян Гудфелоу, на той час аспірант Монреальського університету, створив модель генеративних змагальних нейронних мереж (Generative Adversarial Networks; GAN). Ці моделі були не першими в області генерації данних, але суттєво перевершували всі наявні по якості генерації контенту. За допомогою тренування двох нейронних мереж, які конкурують між собою, з'явилась можливість генерувати зображення, якості, не гіршої як у фотографій, перетворювати нарис в повноцінну картину, пейзаж в фотографію і так далі.

Рис. 1: Еволюція генерації портретних зображенньь за допомогою GAN

Ціль данної роботи дослідити процес генерації зображенньь та переносу стилю за допомогою генеративних змагальних нейронних мереж. Для цього потрібно розібрати як архітектуру моделі, так і архітектуру нейронних мереж, які можна використовувати для генерації. Важливо зрозуміти процес навчання цієї моделі, як він проходить і чому це працює. Також в цій роботі я спробую різні моделі, порівняю використання ними ресурсів, швидкість їх навчання та генерації зображенньь. Не менш важливою частиною будуть блоки з аналізом нюансів архітектури, як кожна зміна впливає на генерацію та які може викликати проблеми. Спробуємо різні моделі для переносу стилю на зображення, намагаючись надати нового стилю як всьо-

му зображенню, так і певним його частинам.

Актуальність роботи полягає в широкому застосуванні генерованого контенту в нашому житті. Ми кожен день зустрічаємось з ним в соціальних мережах, приміряючи нові фільтри на наші фотографії. Зараз є багато нетривіальних використань генеративних моделей, таких як збільшення роздільної здатності зображення або ж відновлення пошкоджених частин зображення. Використовуючи це ми можемо отримати чіткіше зображення з мікроскопу або телескопу. Приклад зображено на рисунку 2. Вражає, чи не так?

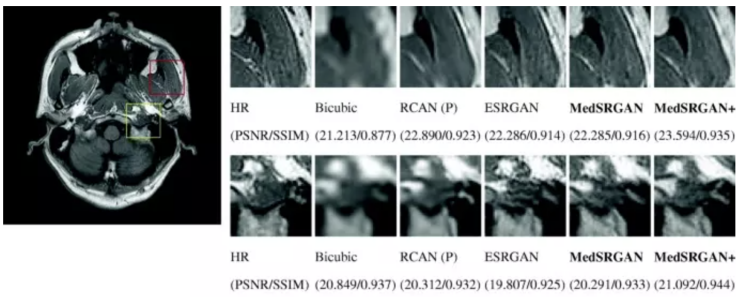

Рис. 2: Приклад збільшення роздільної здатності зображення за допомогою GAN в медицині

Одним із яскравих прикладів використання генеративних моделей є додаток Canvas від немаловідомої компанії Nvidia. Основний його функціонал - перетворення нарису в повноцінний пейзаж. Ви можете додати різні типу об'єктів - гори, поля, озера, вибрати пору року або ж навіть час доби. Результати вражають.

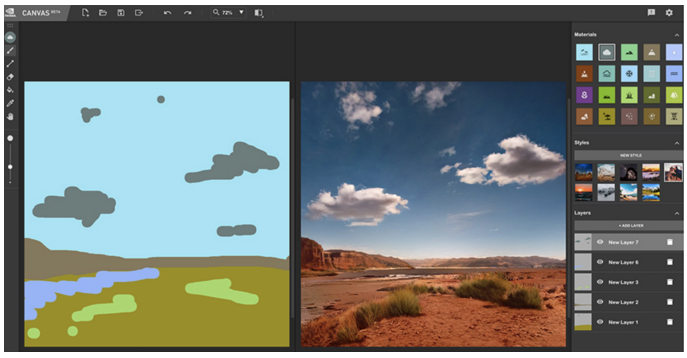

Рис. 3: Результи роботи додатку Canvas від Nvidia

Але для досягнення таких результатів потрібно чітко розуміти як працюють генеративні моделі. Є дуже багато вузьких місць в яких потрібно розібратись, що можливо лише при достатніх знаннях теоретичної частини.

2 Теоретична частина

2.1 Основна ідея моделі

Генеративні змагальні нейронні мережі - це клас алгоритмів машинного навчання, що працює за допомогою двох нейронних мереж. Завдання першої генерувати контент із випадкового вектору певного розподілу - відповідно називається вона генератор. Завдання іншої відрізняти згенерований контент від реального - далі будемо називати її дискримінатор.

Слово "генеративні" означає основне завдання моделі - генерувати данні. Наважливо чи це звук, картинки, відео чи щось інше. Все що ви передасте в якості набору данних для тренування.

Слово "змагальні" описує процес навчання моделі. Під час навчання генератор і дискримінатор намагається конкурувати один з одним - генератор вчиться породжувати контент який би дискримінатор не зміг відрізнити від реального, поданого на вхід для навчання. Дві нейронні мережі постійно змагаються. Для того щоб відрізнити згенерований контент від реального, породженого кращим генератором, покращується дискримінатор і навпаки.

І нарешті слово "мережі" позначає клас моделей, які використовуються для генератора та дискримінатора. В залежності від складності моделі мережі можуться варіюватись від простої повнозв'язної мережі до складних згорткових моделей та інших.

Як це все працює? Якщо говорити більш технічними термінами, генератор намагається породжувати контент з характеристиками, які він знаходить в контенті, взятому з тренувального наборого данних. Його можна розглянути як класифікатор навиворіт. Класифікатор намагається знайти певні характеристики в наборі данних, тоді як генератор намагається ці характеристики відтворити.

Навчання генератора контролюється згідно результів дискримінатора. Якщо дискримінатор хибить і позначає згенерований контент як реальний, тоді модель генератору отримає хороший знак про те що породженні данні дуже схожі на реальні. І навпаки, якщо дискримінатор правильно класифікує всі зображення як згенеровані, генератор отримує відгук про те що данні не схожі на реальні і моделі потрібно покращуватись.

Також постійно покращується і дискримінатор. Йому передаються для навчання поміченні данні про реальний і згенерованний контент. Тому кожен об'єкт, породжений генератором і неправильно класифікований дискримінатором робить другого краще.

2.2 Постановка задачі генерації зображення

Одна із основних задач данної роботи є генерація зображень, схожих на зображення, які були подані на вхід моделі.

Більш формально, нехай X - набір зображень (далі будемо позначати зображення літерою x), який буде використовуватись для тренування моделі. Якщо ми маємо n зображень і збираємочь генерувати кольорові зображення, кожне розміру (w, h), тоді, враховуючи те, що кожен піксель в зображенні відповідає трьом цифрам - ступеню насиченості червоного, зеленого і синнього кольорів (якщо зображення збережено в кольоровій схемі RGB), X буде тензором розміру $(n, w, h, 3)$. Для генерації чорно-білих зображень розмір тензору X буде (n, w, h), так як кожен піксель буде відповідати тільки одній цифрі - ступеню насиченості сірого. Далі введемо ймовірнісний простор Ω, та випадкову величину $z : \Omega \to X$ таку, підмножина X, на якій щільність $p(z)$ ненульова - це певний клас зображень, який ми збираємось генерувати, наприклад фотографії облич. Тому маємо на вхід вибірку випадкових незалежних однаково розподілених величин, зо означають певний клас зображень $\{x_i, i \in [1, N], x_i \sim p(x)\}$. Далі введемо ймовірнісний простір $Z = R^n$ і випадкову величину $z : \Phi \to Z$, з розподілом ймовірностей, що має щильність $q(z)$. $D : X \to (0, 1)$ - функція-дискримінатор, вона показує з якою ймовірністю, зображення x, взяте з X, відноситься до класу зображень, який ми збиражмось генерувати. $G : Z \to X$ - функція-генератор, вона приймає $z \in Z$ і повертає як результат об'єкт з простору X, в нашому випадку зображення. Припустимо ми маємо ідеальний дискримінатор, який для кожного $x \in X$ показує вірогідність входження в x_i. Нам потрібно навчитись "обманювати" ідеальний дискримінатор, тобто максимізувати вірогідність його відгуку на згенерованих данних. Тобто ідеальний генератор знаходиться як

$$G_{ideal} = arg \max_G E_{z \sim q(x)} D(G(z))$$

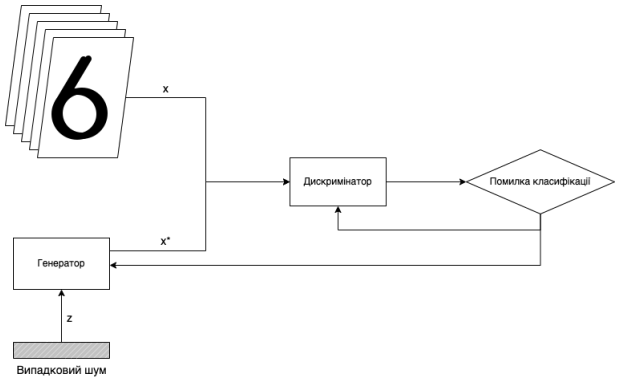

Рис. 4: Схема генерації зображення за допомогою GAN

Під час аналізу різних моделей для виконання цієї задачі ми розглянемо такі питання:

1. Якість генерації зображень.
2. Час та складність тренування.
3. Вибір оптимізатора та архітектури.
4. Використання ресурсів та час генерації зображень.

2.3 Навчання генеративних змагальних мереж

В попередньому розділі ми вже ввели всі базові поняття, тому продовжимо працювати з ними. В реальному світі ідеального генератору зазвичай знайти майже неможливо. Так як завдання дискримінатора в тому, щоб передавати сигнали для навчання генератора, то ідеального дискримінатора не потрібно, достатньо функції яка чітко розділяє згенеровані об'єкти від об'єктів з навчальної вибірки x_i. Тобто в термінах машинного навчання, нам потрібно натренувати бінарний класифікатор - модель, яка максимізує ймовірність правильної класифікації для об'єктів з x_i. Для її тренування маємо нескінченну кількість данних. Вибірка складається з об'єднання скінченної x_i та поповнюється кожну ітерацію новими згенерованими об'єктами, при цьому в кожного є мітка, згенерований він чи ні. Опишемо процес тренування дискримінатора

Маємо вибірку $\{(x_i, 0), i \in [1, N], x_i \in X, p(x_i) > 0\} \cup \{(G(z), 0), z \sim q(z)\}$. Визначимо щільність розподілу $f(\xi|\eta = 1) = D(\xi), f(\xi|\eta = 0) = 1 - D(\xi)$. Так як $f(\xi|\eta)$ - переформулювання дискримінатора, який видає ймовірність справжнього об'єкту в вигляді розподілу ймовірностей на інтервалі $(0, 1)$.

Так $D(\xi) \in (0,1)$, тоді визначення задає щільність розподілу. З цього випливає, що оптимальний дискримінатор можна знайти, як

$$D^* = f^*(\xi|\eta) = arg\max_f f(\xi_1, ...|\eta_1, ...) = arg\max_f \prod_i f(\xi_i|\eta_i)$$

Згрупувавши множники для $\eta_i = 0$ і $\eta_i = 1$, маємо:

$$D^* = f^*(\xi|\eta) = arg\max_f \prod_i f(\xi_i|\eta_i = 1) \prod_i f(\xi_i|\eta_i = 0) =$$

$$arg\max_D \prod_{x_i \sim p(x)} D(x_i) \prod_{z_i \sim q(z)} D(G(z_i)) =$$

$$arg\max_D \sum_{x_i \sim p(x)} logD(x_i) + \sum_{z_i \sim q(x)} D(G(z_i))$$

І якщо спрямувати розмір вибірки до нескінченності:

$$D^* = arg\max_D E_{x_i \sim p(x)} logD(x_i) + E_{z_i \sim q(z)} logD(G(z_i))$$

Тоді маємо наступний ітераційний процес:

1. Встановлюємо випадковий початковий $G_0 z$.
2. Починається k-та ітерація , $k = 1...K$
3. Шукаємо оптимальний для наявного генератора дискримінатор.

$$D_k = arg\max_D E_{x_i \sim p(x)} logD(x_i) + E_{z_i \sim q(z)} logD(G_{k-1}(z_i))$$

4. Покращуємо генератор, використовуючи оптимальний дискримінатор.

$$G_k = arg\max_D E_{z_i \sim q(x)} D_k(G(z))$$

5. Задача навчання генератора вважається вирішеною, коли $D(x) = 1/2$, для кожного x.

В оригінальній статті (Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio - Generative Adversarial Networks) весь алгоритм складається в одну формулу, що задає мінімакс-гру між генератором і дискримінатором:

$$\min_G \max_D L(D, G) = E_{\sim p(x)} logD(x) + E_{z \sim q(z)} D(G(z))$$

Як D, так і G можуть будуть представленні нейронними мередами, тому всі оптимізації зводяться до алгоритму зворотнього розповсюдження помилки та стохастичного градієнтного спуску. Також нейронна мережа

є універсальним оптимізатором, тому не потрібно витрачати час на вибір розподілу $q(z)$. Може бути використаний будь-який в розумних рамках, наприклад нормальний $N(0, 1)$, або рівномірний на інтервалі $[-1, 1]$.

Декілька слів про вирішення задачі навчання. Як згадувалось раніше формула для навчання задає мінімакс-гру між генератором і дискримінатором. Коли одна модель стає краща, інша стає гірша. В цьому налаштуванні можна побачити гру нульової суми, термін з теорії ігор, в якому описується змагання в якому виграш одного з гравців дорівнює суми програшів інших. Кожна така гра має точку рівноваги Неша - така сукупність стратегій та виграшів, при якій жоден із учасників не може збільшити виграш, змінивши вибір стратегії в односторонньому порядку, коли інші учасники не змінюють свого вибору. Названа іменем відомого американського математика та економіста, спеціаліста в галузі теорії ігор, лауреата Нобелівської премії з економіки Джона Форбса Неша, який запропонував цей термін і зробив вагомий внесок у розробку формалізованого опису конфліктних ситуацій, зокрема у визначення формули рівноваги. Для генеративниї нейронних мереж ця рівновага досягається при виконання двох умов:

1. Генератор породжує об'єкти, які не можна відрізнити від реальних об'єктів.

2. Дискримінатор випадково дає мітки реальним данним (1/2 вірогідність того, що об'єкт реальний)

На практиці знайти точно рівноваги для генеративних змагальних мереж майже нереально, через великі трудності з досягненням конвергенції для неопуклих ігор. Взагалі, питання конвергенції для GAN залишається відкритим для дослідження.