КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра теорії та технології програмування

Кваліфікаційна робота

на здобуття ступеня магістра
за спеціальністю 122 Комп'ютерні науки

на тему:

ГЕНЕРАЦІЯ СИНТЕТИЧНИХ ЗОБРАЖЕНЬ НА ОСНОВІ ТЕКСТОВОГО ОПИСУ МЕТОДАМИ МАШИННОГО НАВЧАННЯ

Виконала студентка 2-го курсу магістратури
Назаркевич Ганна Ярославівна _____
(підпис)

Науковий керівник:
доцент, кандидат фізико-математичних наук
Панченко Тарас Володимирович _____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____
(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри теорії та технології програмування
«____» _____ 2021 р.,
протокол № ___
Завідувач кафедри
М. С. Нікітченко _____
(підпис)

Київ – 2021

РЕФЕРАТ

Обсяг роботи 64 сторінок, 18 ілюстрацій, 4 таблиці, 25 джерел посилань.

ГЕНЕРАЦІЯ СИНТЕТИЧНИХ ЗОБРАЖЕНЬ НА ОСНОВІ ТЕКСТОВОГО ОПИСУ, МАШИННЕ НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ, ГЕНЕРАТИВНІ ЗМАГАЛЬНІ МЕРЕЖІ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, МАТЕМАТИЧНА СТАТИСТИКА

Об’єктом дослідження є методи розв’язання задачі синтезу зображень з текстового опису. Предметом дослідження є архітектура GAN, модель, що генерує зображення з їх текстового опису.

Метою роботи є створення моделі генерації зображень з текстового опису.

Інструменти розробки: Python, Google Colab, бібліотека PyCharm.

Результати роботи: проведено огляд існуючих архітектур нейронних мереж для розв’язку поставленої задачі, проаналізовано їх переваги та недоліки, виконано порівняння цих нейронних мереж за допомогою показників Inception Score. Також використано для порівняння FID.

Натреновано нейронну мережу з архітектурою DF-GAN та DM-GAN, проаналізовано її IS та FID. Для тренування використано датасет CUB-200-2011, що містить зображення пташок та їх текстових описів. Впродовж тестування розглянуто, як змінюється IS, FID для цих моделей на різних етапах, побудовано графіки залежностей..

В результаті роботи було розглянуто нову модель DM-GAN з використанням DF-блоків. Порівняно її з іншими описаними у літературі моделями. Обчислено для неї IS та FID.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

GAN	– Generative adversarial network, генеративна змагальна мережа;
НМ	– нейронна мережа;
FID	– Frechet Inception distance;
IS	– Inception Score
ШНМ	– Штучна нейронна мережа
РНМ	– Рекурентна нейронна мережа
ДКЧП	– Довга короткочасна пам’ять
ResNet	– Residual Network, залишкова нейронна мережа
AttnGAN	– Attention GAN
Stack GAN	– stacked GAN, стекова GAN

ВСТУП

З розвитком машинного навчання та штучного інтелекту було розв’язано багато складних задач, які не можливо було розв’язати до того. Більше того, появились нові задачі, які раніше вважались майже не розв’язними. Однією з таких задач є задача синтезу правдоподібних зображень, якщо нам дано їх текстовий опис.

З розвитком нейронних мереж GAN [1] ця проблема стала популярною серед дослідників і в ній багато-хто бачить величезний потенціал. За останні п’ять років появилось багато архітектур нейронних мереж, що б розв’язують задачу генерації зображень з їх текстового опису [2,3,4] – це StackGAN, StackGAN++, Dalle, DF-GAN, AttnGAN, . Задача синтезу зображень з текстового опису – це одна з найбільш складних задач комп’ютерного зору. Багато років вважалось, що комп’ютер не здатний до творчого мислення, але останні архітектури, що розв’язують цю задачу є ілюстрацію того, наскільки розумним може бути штучним інтелект та наскільки він здатний до абстрактного мислення.

Актуальність. Задача генерації зображень з текстового опису є актуальною, оскільки її застосування включає в себе редагування фотографій, генерацію медіа-контенту для фільмів, мультфільмів, коміксів, тощо; також ця задача може використовуватись для інших задач комп’ютерного зору, наприклад, пошуку зображень та генерації відео.

Об’єкт дослідження: методи розв’язання задачі синтезу зображень з текстового опису.

Предмет дослідження: архітектура GAN.

Мета дослідження: дослідити методи генерації зображень з їх текстового опису, наприклад DF-GAN та DM-GAN та розглянути їх на прикладі конкретних датасетів. Розглянути інші можливі модифікації існуючих архітектур.

Наукова новизна дослідження: розглянуто нову архітектуру, побудовану на основі DM-GAN з використанням DF-блоків. Показано її переваги та недоліки в порівнянні з DF-GAN, DM-GAN та іншими нейронними мережами.

РОЗДІЛ 1.
ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

У цьому розділ дано визначення основних понять, тверджень та теорем, що використовуються у подальшому.

1.1 Генеративні змагальні мережі (Generative adversarial networks)

В 2014 році Ян Гудфелов в праці [1] запропонував новий фреймворк для оцінки генеративних моделей шляхом одночасного тренування двох моделей: генератора G, що оцінює розподіл даних та дискримінативної моделі D, що оцінює ймовірність того, що зразок належить тому ж розподілу, що і тренувальний датасет.

Нехай генератор вивчає розподіл p_g над даними x. Визначимо $p_z(x)$, що є відображенням на простір даних $G(z;\theta_g)$,де G – диференційовна функція, що є багатошаровим перцептроном з параметром θ_g. Також визначимо інший багатошаровий перцептрон $D(x;\theta_d)$, що набуває скалярних значень.

Тренуватимемо D для того, щоб досягти максимуму ймовірності правильного присвоєння ярлика для об’єктів з тренувального датасету і для зразків отриманих з мережі G.

Можна сказати, що D та G грають мінімаксну гру з функцією цінності $V(D,G)$:

$$V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}\left[\log\left(1 - D\left(G(z)\right)\right)\right]$$

Формулювання цієї задачі зводиться до пошуку $\min\limits_{G}\max\limits_{D} V(D,G)$.

Ця задача нагадує задачі з теорії ігор. І як показано у [7] зводиться до пошуку розв’язку варіаційних нерівностей, а саме пошуку вектора x^* для якого буде виконуватись

$$\langle A(x^*), x - x^*\rangle \geq 0 \ \forall x' \in \mathrm{C}\,,$$

де $F(x)$ –ліпшицевий та монотонний оператор зі сталою Ліпшиця $L > 0$

Ця задача є досить відомою, оскільки багато задач математичної економіки зводяться до пошуку точок рівноваги. В свою чергу ці задачі формулюються у вигляді пошуку розв'язків варіаційних нерівностей. Зокрема, Nagurney A. у працях [5, 6] розглядає задачі пошуку точок рівноваги у просторовій, транспортних, міграційних, олігополістичних, Вальрасіанській моделях, тощо, які формулюються у вигляді варіаційних нерівностей.

Тренування GAN тим не менше є досить складним. По-перше, хоча задача пошуку наближеного розв'язку $\min_G \max_D V(D,G)$ вже, як показано, досліджувалась досить давно, все ж дуже часто алгоритми для розв'язку цієї задачі не дають достатню швидкість збіжності, а також часто починають осцилювати. В результаті модель не сходиться до оптимального розв'язку.

Інша проблема полягає в тому, що генератор починає видавати одні і ті ж картинки, які з одного боку виглядають реально, з іншого боку немає варіативності вибірки. Дизбаланс між генератором та дискримінатором суттєво погіршує результат.

1.2 IC (Inception Score) та FID (Frechet Inception distance) як ефективні засоби оцінки генеративних змагальних мереж

Одна з найбільш очевидних метрик для оцінювання синтетичних зображень – є оцінка людиною. Суттєвим недоліком є час збору інформації, відсутність людських ресурсів. Також сам процес запровадження системи оцінки GAN людиною займає багато часу.

Тому більшу популярність серед дослідників завоювали аналітичні методи оцінки моделей. Один з таких методів - Inception Score як метод оцінювання якості синтетичних зображень був запропонований у [2] та

показано, що ця метрика корелює з результатами оцінки людьми. Картинки, що мають високий

$$IS(G) = exp(\mathbb{E}_{x\sim p_g} D_{KL} p(y|x)||(p(y)))$$

, де $x\sim p_g$ означає, що x – це картинка з розподілу p_g. $D_{KL}(p||g)$ – KL дивергенція або ж відстань Кульбака-Лейблера.

$$D_{KL}(p||g) = -\sum_{x\in\chi} P(x)\log\left(\frac{Q(x)}{P(x)}\right)$$

$p(y|x)$ – умовна ймовірність. Дана метрика дозволяє оцінити, чи об’єкт на згенерованій картинці відрізняється від наявних в тренувальному датасеті, та чи на картинка містить зображення одного чіткого об’єкту. [3]

У роботі [4] запропоновано використання метрики FID для визначення якості зображень створених мережами GAN. Суть оцінки генеративних моделей полягає в тому, щоб знайти ефективну метрику, що характеризуватиме відстань між реальними даними та синтетичними картинками. Якщо представити всі згенеровані та реальні картинки як багатовимірний гаусівський розподіл, то відстань FID (Frechet Inception distance) обчислюється як відстань Фреше між цими розподілами:

$$d^2\big((m,C),(m_w,C_w)\big) = \|m-m_w\|_2^2 + Tr(C+C_w-2(CC_w)^{\frac{1}{2}})$$

1.3 Поняття перцептрона, типи нейронних мереж

Перцептроном – математична модель, запропонована Ф. Розенблатом, що описується перетворенням $R^n \to R$ за допомогою формули

$$v = \sum_{i=1}^{m} \omega_{ij} x_i$$

де $j = 1, \dots, n$; ω_{ij} – вага перцептрона; x_i – значення вхідних сигналів. Далі від отримання результату, до отриманого значення у v застосовується функція активації f. Отримане значення порівнюється з порогом спрацювання θ і приймається рішення.

Навчання перцептрона полягає пошуку вагових коефіцієнтів. Нехай є набір пар векторів $(x^n, y^n), \alpha = 1, \dots . p,$ який називається навчальною вибіркою. Будемо називати нейронну мережу навченою на заданому навчальному датасеті, якщо при подачі на входи мережі кожного вектора x^n на виходах кожен раз отримуємо відповідний вектор y^n.

Розпізнавання, обробку текстів, людської мови, музики, зображень, відео, 3D об’єктів, табличних даних визначення об'єктів на фото та відео, класифікація обёєктів, навіть синтез текстів, зображень та відео - все це є задачами, що з легкістю вирішуються за допомогою нейронних мереж.

Запропонований Розенблаттом метод навчання полягає в ітераційній підстановці матриці ваг, що дозволяє послідовно зменшувати помилку у результуючих векторах. Алгоритм має наступні кроки:

- початкові значення всіх ваг нейронів $W(t = 0)$ ініціалізуються випадковими значеннями.
- Якщо отримуємо вхідний вектор x^n, в результаті отримаємо $\tilde{y}^n \neq y^n$.
- Обчислюється вектор помилки $\delta^n = (y^n - \tilde{y}^n)$, здійснюється мережею на виході. Зміна вектора вагових коефіцієнтів в області малих похибок повинна бути пропорційна похибці на виході; і вона повинна бути рівна нулю, якщо похибка рівна нулю.

- Вектор ваг модифікується за формулою:

$$W(t + \Delta T) = W(t) + \eta x^n \cdot (\delta^n)^T,$$

Де $0 < \eta < 1$ – темп навчання.

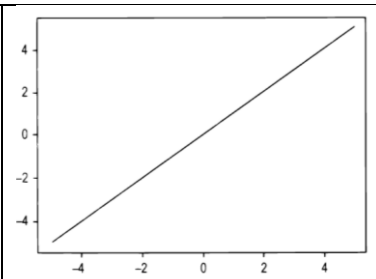
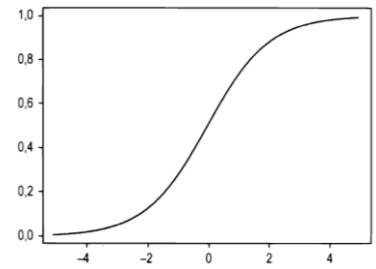
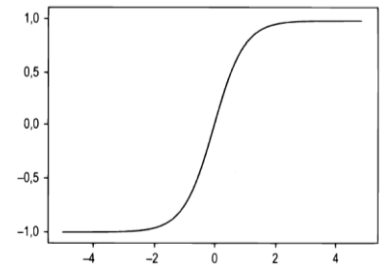
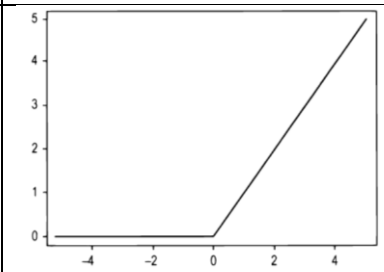
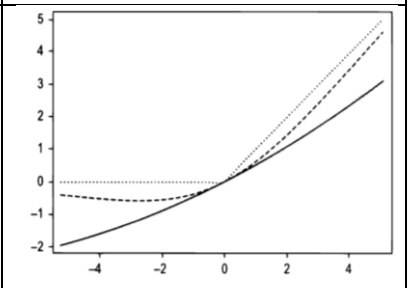
- Попередні кроки повторюються для всіх векторів, які навчаються.

Один цикл послідовного розгляду всієї тренувальної вибірки називається епохою. Навчання завершується по закінченні визначеної кількості епох: або коли метод зійдеться, тобто вектор перестане змінюватися, або коли повна, порахована за значеннями всіх векторівабсолютна похибка стане меншою від деякого ϵ.

Глибоке навчання ґрунтується на складних системах, що складаються з величезної кількості нейронів. В одній нейронній мережі можуть бути мільярди таких структурних одиниць як нейрони або ж перцептрони. Відповідно існує безліч способів структурувати їх. І в залежності від того, як вони будуть пов’язані між собою можна розрізняти різні архітектури нейронних мереж.

В залежності від типів активаційних функцій нейронна мережа може по-різному тренуватись. Тому розглянемо основні з них у табл. 1.1.

Таблиця 1.1 - Активаційні функції

Одиничне перетворення	$f(z) = z$	
Сігмоїда	$f(z) = \frac{1}{1 + e^{-z}}$	
Гіперболічний тангенс	$f(z) = \tanh(z)$	
Relu	$f(z) = \max(0, z)$	
Swish	$f(z) = z\sigma(\beta z)$, де β – параметр, що вивчається	

На практиці майже завжди використовуються тільки дві активаційні функції: сигмоїда і ReLU. З обома можна досягти хороших результатів, і, з огляду на часто досить складну архітектуру, обидві функції здатні апроксимувати будь-яку нелінійну функцію.

Задача тренування нейронної мережі полягає у тому, щоб мінімізувати Cost функцію.

1.4 Оптимізація параметрів НМ

Згадаємо як виглядає один з найпростіших алгоритмів оптимізації – градієнтний спуск:

$$w_{p+1} = w_p - \eta \nabla Cost(w_p)$$

Також теоретично обгрунтовано, що оптимальним алгоритмом для пошуку мінімуму буде алгоритм Нестерова

$$\Delta p + 1 = \gamma \cdot \Delta p + \eta \cdot \nabla Cost(wp - \gamma \Delta p),$$

$$wp + 1 = wp - \Delta p + 1.$$

Тут використовується, свого роду, уточнююча операція для знаходження градієнту. Тому метод інколи називають пришвидшений градієнт Нестерова. Однак попри те, що доведена верхня оцінка часу збіжності для цього алгоритму є мінімальна, він не знайшов свого практичного застосування. Річ у тому, що верхня оцінка – не завжли співпадає з середньою. І в випадку цього алгоритму, середня оцінка часу збіжності наближається до верхньої оцінки. Отже, алгоритм переважно працює повільно.

Найбільш популярний алгоритм оптимізації у машинному навчання – це є метод Адам (Adaptive Moment Estimation - адаптивна оцінка моментів). Цей алгоритм показує найшвидшу збіжність для більшості нейронних мереж.

$$mp = \beta_1 m_p - 1 + (1 - \beta 1) \cdot \nabla Cost(wp - 1)$$

$$vp = \beta_2 v_p - 1 + (1 - \beta 2) \cdot \left(\nabla Cost(wp - 1)\right)^2$$

Метод дозволяє зберігати зсув параметрів з попередньої ітерації.

$$\widehat{m_p} = \frac{mp}{1 - \beta_1^p}$$

$$\widehat{v_p} = \frac{vp}{1 - \beta_2^p}$$

Формула остаточної зміни має вигляд

$$w_p = w_{p-1} - \frac{\eta \widehat{mp}}{\epsilon + \sqrt{\widehat{v_p}}}$$

Для алгоритмів Нестерова, градієнтного спуску, Адам існує багато специфічних для машинного навчання способів оптимізувати швидкість цих алгоритмів. Розглянемо деякі з них.

1.4.1 Пакетний градієнтний спуск

Звичайний алгоритм градієнтного спуску, описаний вище, обчислює зміщення для кожного спостереження, але виконує оновлення ваг і зміщень лише після того, як були обчислені всі спостереження, або, іншими словами, після так званої епохи. Цикл по всьому набору даних називається епохою.

Перевага:

менше число оновлень ваг і зсуву означає більш стабільний градієнт, що в свою чергу призводить до більш стабільного сходження.

Недоліки:

- алгоритм реалізований таким чином, що всі набори даних повинні перебувати в пам'яті

- Цей алгоритм, як правило, є дуже повільним для дуже великих наборів даних.

1.4.2 Стохастичний градієнтний спуск

Стохастичний градієнтний спуск (stochastic gradient descent, SGD) обчислює градієнт $Cost$ функції, а потім оновлює ваги і зміщення для кожного спостереження в наборі даних.

Переваги цього методу:

- часті оновлення дозволяють легко перевіряти хід самонавчання моделі (вам не потрібно чекати до тих пір, поки всі набори даних будуть розглянуті);
- в деяких завданнях цей алгоритм працює швидше, ніж пакетний градієнтний спуск;
- модель має власний шум, що дозволяє уникнути локальних мінімумів при спробі знайти абсолютний мінімум вартісної функції.

Недоліки цього методу:

- у великих наборах даних цей метод є досить повільним, оскільки через постійні оновлень він вимагає інтенсивних обчислень;
- той факт, що алгоритм має свої шуми, може ускладнити йому найти мінімум $Cost$ функції, і сходження може виявитися не таким стабільним, як очікувалося.

1.4.3 Мініпакетний градієнтний спуск

При такому варіанті градієнтного спуску набори даних розбиваються на певне число малих (звідси і термін "міні") груп спостережень (іменованих

пакетами), а ваги і зміщення оновлюються тільки після того, як кожен пакет був поданий в модель. Цей метод використовується в сфері глибокого навчання найбільш часто.

Переваги:

- Частота оновлення моделі вище, ніж при пакетному градієнтному спуску, але нижча ніж в стохастичному градієнтному спуску. Отже, цей варіант допускає більш надійне сходження;
- Потрібно менше обчислень і ресурсів;
- Цей варіант, безумовно, є найшвидшим

1.5 Згорткові нейронні мережі, Рекурентні нейронні мережі

Теоретично, одношарова мережу може апроксимувати більшість функцію, але число необхідних нейронів може бути дуже великим, і, тому, модель стає набагато складнішою. Суть полягає в тому, що здатність апроксимувати функцію не означає, що мережа здатна навчитися робити це за допомогою градієнтного спуску чи іншого алгоритму оптимізації. Тому було придумано багато різних комбінацій нейронів, призначених для різних задач. Однією з таких комбінацій – є згорткова нейронна мережа. Вона призначена для обробки картинок. Її назва походить від операції згортки, що часто застосовується в задачах комп’ютерного зору. Ця операція застосовується також при пошуку значення згорткової нейронної мережі на зображенні. Якщо ми передамо зображення в нейронну мережу за допомогою вектора, то нам доведеться його сплюснути і довжина цього векттора буде досить великою, також втратиться інформаці про краї зображення. Також не зрозуміло як використати інформацію про кольори. Конволюційний шар застосовує операцію згортки до зображення з зазначеним вікном. Операція

згортки сумує значення інтенсивності в певному вікні помножені на певні коефцієнти – таким чином ми отримуємо ознаки, що будуть характеризувати зображення.

Для обробки текстової інформації та інформації, що містить в собі збереження послідовних зв'язків використовуються рекурентні нейронні мережі.

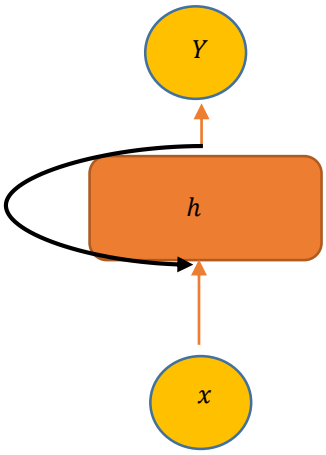


Рис.2.1 – Рекурентна нейронна мережа

Рис.2.1 ілюструє просту рекурентну нейронну мережу. Y – це вихід нейронної мережі. Прихований шар – h, x – вхідний вектор. На кожному етапі ми домножуємо на h попередній отриманий результат на нейронній мережі (позначений чорною стрілкою) та сумуємо з домноженим на h теперішнім вхідним вектором x. Таким чином ця нейронна мережа має свого роду пам'ять на всі попередні вхідні значення. Причому останні вхідні значення "запамятовуються" з більшим коефіцієнтом.

1.6 Основні визначення та формули для теорії обробки зображень

Інтенсивність або ж яскравість обчислюється як

$$Ip = 0.2126 \cdot r + 0.7152 \cdot g + 0.0722 \cdot b$$

де $w(x,y)$ – функція вікна – функція, що прямує до нуля поза певним інтервалом, та симетрична відносно середини розглянутого інтервалу. Для двовимірного вікна використовується добуток одновимірних функцій вікна. Приклад функції вікна:

$$w(n) = sin(\pi nN).$$

1.7 Ознаки на зображеннях

Для задачі пошуку схожих точок на зображеннях розроблено багато алгоритмів. Переважно вони всі працюють у три етапи: пошук точок, їх опис, пошук схожих точок. Найбільш популярні та відомі алгоритми: SIFT, Harris, SURF, FAST, Good Feature to Track Detector, MSER, ORB, BRISK. Найбільш точним є алгоритм SURF, найбільш швидким, але таким, що поступається в точності є ORB.

ORB – це комбінація декількох алгоритмів. Для визначення точок в алгоритмі ORB викоритсовується алгоритм FAST, для сортування їх за значимістю використовується Harrison response. На останній стадії алгоритм ORB шукає характеристики BRIEF для кожної з точок. Зазвичай, це 256 бітний вектор. Цей вектор можна порівнювати за допомогою відстані Хемммінга з іншими векторами, що характеризують особливі точки.

Fast-алгоритм

- Виберіть піксель p на зображенні, який слід перевірити, чи це як цікавий пункт чи ні. Нехай його інтенсивність буде Ip.
- Виберемо параметр - порогове значення t.

- Розглянемо коло з 16 пікселів навколо досліджуваного пікселя. (Це коло Брезена радіусом 3)

Тепер піксель p - кутова, якщо в колі існує набір n послідовних пікселів (16 пікселів), які усі або яскравіше $I_p + t$, або темніше $I_p - t$. (Переважно використовують коло з $n = 12$ точок)

В роботі використано модифікований алгоритм. На першому етапі порівнюємо інтенсивність пікселів 1, 5, 9 та 13 кола з I_p. Щонайменше три з цих чотирьох пікселів повинні задовольняти пороговому критерію, щоб існувала кутова точка. Далі порівнюємо, як описано вище, усі решта точки з масиву.

Повторюємо описані вище кроки для всіх інших пікселів.

Алгоритм ORB

1. Визначаємо точки за алгоритмом FAST, що описаний вище.

2. Сортуємо точки за детектором Harrison response і вибираємо перші 10 000 точок.

3. Для кожної з них обчислюємо BRIEF. [17]

1.8 Задача генерації синтетичних зображень на основі текстового опису

Однією з найбільш амбітних задач штучного інтелекту є задача побудови зображень з текстового опису. Ця задача потребує як і розвинутих інструментів комп'ютерного зору так і інструментів розпізнавання текстів, також більшість дослідників цієї задачі користуються класичними методами машинного навчання.

Розв'язання цієї задачі дозволить полегшити роботу спеціалістів, що працюють з медіа-ресурсами – таких як дизайнерів, аніматорів, художників-

мультиплікаторів. Задача може застосовуватись для розумного редагування фото, створення фотографій, прототипів дизайну, мультфільмів, фільмів, коміксів та іншого медіа-контенту. Також розвиток цієї задачі стане поштовхом для розвитку аналогічних задач в 3d-графіці та 3d-моделюванні, таких як моделювання 3d реальності за текстовим описом, створення ігор, 3D реконструкція об’єктів.

Розвиток цієї задачі і застосування її на величезних датасетах так, як у DALLE показує, що штучний інтелект здатний не лише до знаходження закономірностей, використання багатьох факторів, а також він здатний до узагальнення та абстракції [18]. Проте отримана авторами модель дає погані показники якості IS та FID, тому вона не розглядається в цій роботі.

На мою думку, використання великих датасетів завжди супроводжується падінням показників IS та FID, хоча якість зображень не є суттєво гіршою, ніж для моделей, що натреновані на маленьких датасетах. Це підтверджує той факт, що задача оцінки та тестування таких моделей ще теж недостатньо досліджена.

РОЗДІЛ 2.
МЕТОДИ ГЕНЕРАЦІЇ СИНТЕТИЧНИХ ЗОБРАЖЕНЬ НА ОСНОВІ ТЕКСТОВОГО ОПИСУ

Задача синтезу зображень з їх текстового опису почала розвиватись відносно недавно. Не зважаючи на це, уже існує безліч методів та натренованих мереж. У цьому розділі описано вже існуючі нейронні мережі, їхні відмінності, переваги.

2.1 Генеративні змагальні мережі GAWWN

Спочатку вбудовування тексту (показано зеленим кольором) копіюється, щоб сформувати масив $M \times M \times T$, а потім приводиться до нормального розподілу. Далі застосовуються операції згортання та об’єднання для зменшення просторового виміру назад до 1×1. Інтуїтивно, цей вектор функції кодує просторову структуру на зображенні. На наступному етапі він об'єднується з вектором шуму z. Згодом генератор розгалужується на локальну та глобальну стадії обробки. Глобальний шлях - це просто серія деконволюцій для збільшення просторових розмірів від 1×1 до M × M. До локального шляху, досягнувши просторового розміру M × M, застосовується операція маски так, що регіони за межами поля обмеження об'єкта вважаються рівними нулю. В результаті, локальний та глобальний шляхи об'єднуються операцією конкатенації. Останні шари деконволюцій використовується для досягнення фінального просторового виміру. В останньому шарі ми застосовуємо $\tanh$, щоб обмежити результати на проміжку $[-1, 1]$.

У дискримінаторі текст подібним чином просторово відтворений, утворюючи тензор $M \times M \times T$. Так само зображення обробляється локальним та глобальним шляхами. У локальному шляху зображення зменшується за допомогою згорткової нейронної мережі до просторової

розмірності $M \times M$, після чого він конкатенується з вектором, що характеризує текст. Отриманий тензор просторово обрізається у визначених межа, а потім обробляється згортково, поки просторова розмірність не становитиме 1×1. В кінці, локальний та глобальний вектори вихідних шляхів поєднуються адитивно і подаються у кінцевий шар, формуючи скалярну оцінку дискримінатора.

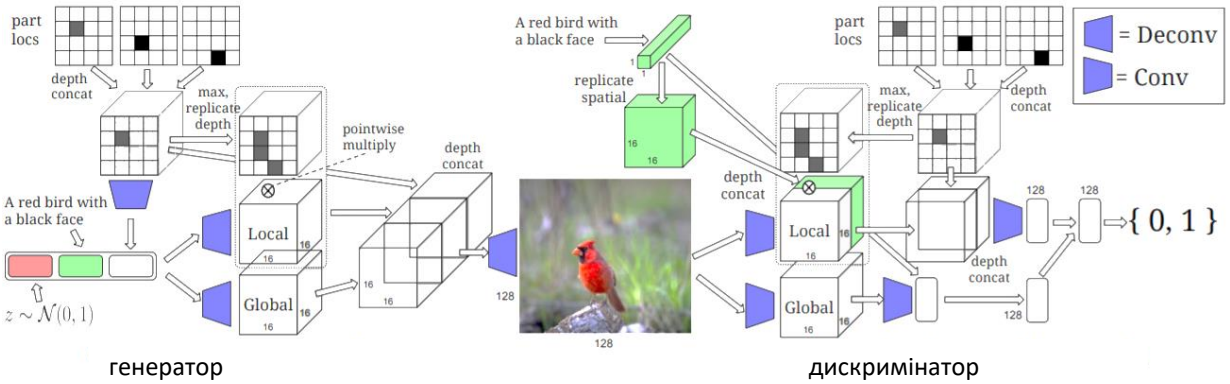


Рис.2.1 – Ілюстрація архітектури GAWWN або ж (Generative Adversarial What-Where Network)

2.2 Генеративні змагальні змагальні мережі StackGAN

У [8] Han Zhang1 , Tao Xu, Hongsheng Li , Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, Dimitris Metaxas запропонували метод синтезу зображень на основі покрокового уточнення деталей в процесі виконання генерації зображень. На першому етапі GAN намагається намалювати свого роду ескіз, з низькою роздільчою здатністю, використовуючи вказані кольори та форми. На другому етапі інший GAN бере результат отриманий на першому етапі та текстовий опис та домальовує реалістичні деталі. Друга нейронна мережа може виправити дефекти отримані на першому етапі.

На кожному кроці ми отримує все кращі зображення.

Рис. 2.2 – Ілюстрація того, як працює перший та другий етап для Stack GAN. Згенеровано зображення для фрази: “A white bird with a black crown and yellow beak”. У першому рядку – зображення, отримані на першому етапі, у другому рядку - зображення, отримані на другому етапі

На рис. 2.2 показано картинки на різних етапах формування картинок StackGAN.

2.3 Генеративні змагальні змагальні мережі StackGAN++

Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, Dimitris N. Metaxas згодом запропонували покращену архітектуру, що використовує в собі StackGAN. Було запропоновано використовувати декілька генераторів та декілька дискримінаторів. Картинки різних розмірів відповідають різним віткам дерева побудованого з натренованих генераторів та дискримінаторів.

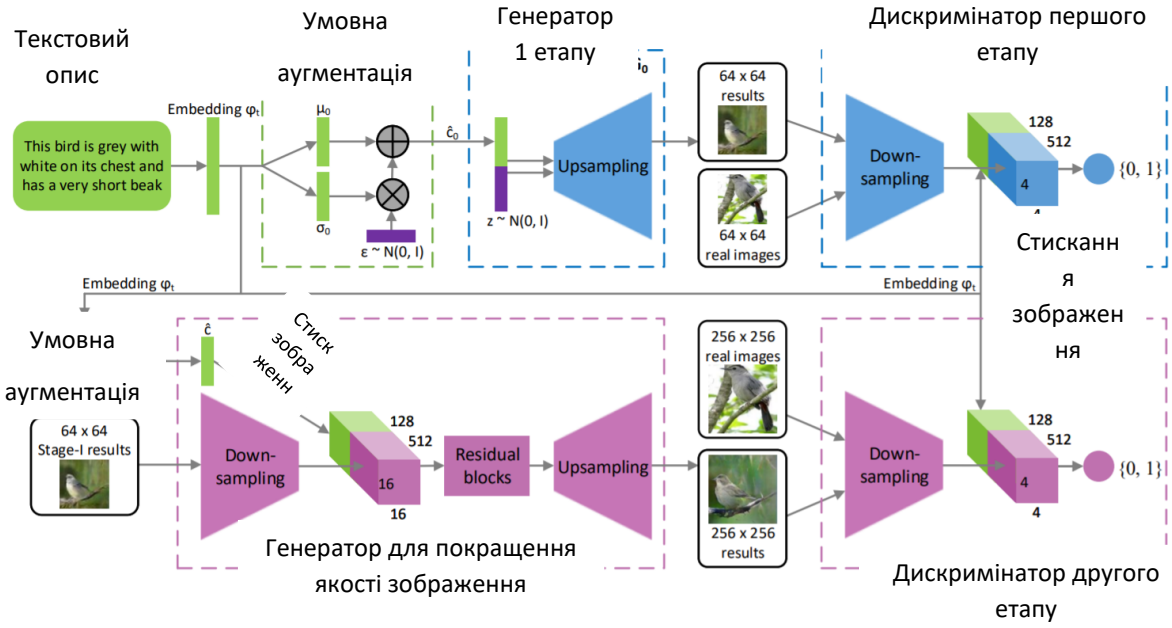


Рис.2.3 – Ілюстрація архітектури StackGAN++

Рис. 2.3 ілюструє архітектуру нейронної мережі, запропоновану авторами StackGAN ++.

Спочатку text embedding (укр.. вкладання слів) (вектор, що характеризує певне речення, отриманий за допомогою інших моделей машинного навчання)подається на генератор. Наступний в архітектурі є блок збільшення умовності. Це блок, запропоновай вперше саме авторами цього методу. Він дозволяє збільшити різноманіття можливих картинок на виході, використовуючи вектор з нормального розподілу. Відповідно це полегшує тренування самої нейронні мережі.

Далі як видно з *Рис.2.3* отриманий результат конкатенується з вектором, що належить нормальному розподілу. Цей підхід дозволяє додати регуляризацію до моделі і вперше був запропонований робзробниками

InfoGAN. Оскільки InfoGAN – одна з найкращих моделей для генерації картинок (без текстового опису), то схожий підхід або його модифікаціях застосовувався в усіх подальших архітектурах. Далі отриманий вектор проходить через мережу збільшення розмірності картинки та ми отримуємо зображення розмірністю – 64x64.

Отримане зображення подається на дискримінатор – де текстовий опис, ще раз конкатенується до вектору картинки, який показано зеленим кольором. Це робиться для того, щоб дискримінатор не просто навчався відрізняти фейкові картинки від реальних, а також, щоб він міг розрізняти картинки, що не відповідають текстовому опису від тих, що йому належать.

Далі застосовується ще одна мережа, що має архтектуру GAN. Генератор другого етапу має структуру схожу на генератор першого етапу. Тобто ми подаємо отриманий вектор на блок збільшення умовності, далі отриманий вектор подаємо на нейронну мережу, що зменшує розмірність картинки. З’єднуємо операцією конкатенації отриманий вхід на генератор з тим, що ми отримали з блоку збільшення умовності. Далі використовується нейронна мережа запрпонована розробниками ResNet - Residual Block. Традиційні нейронні мережі передбачають послідовну передачу векторів з шару до шару. Проте мережа з Residual Block(Залишковий блок) дозволяє передавати вектор не лише на наступний шар, а й на шар на декілька рівнів вище. Архітектура залишкового блоку ілюстрована на рис. 2.4.

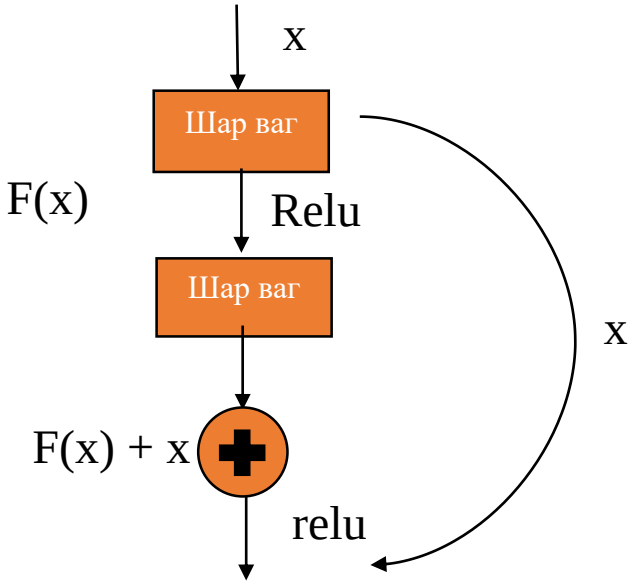


Рис.2.4 – Ілюстрація архітектури Residual Block

Залишковий блок – це блок, що ввели з міркувань того, що нейронна мережа може інколи не натренуватись щоб відображати навіть такі прості функції як одиничне відображення. Через до прикладу відому проблему зникання градієнту, може зупинитись процес оптимізації коефіцієнтів, і точність нейронної мережі буде залишатись малою, хоча відображення може бути досить простим. Для цього придумати залишковий блок – він дозволяє пропустити більшість коефіцієнтів, які можуть бути неправильними і натренуватись правильно. Залишкові блоки додають, коли одношарові нейронні мережі тренуються швидше, ніж багатошарові або ж коли багатошарові нейронні мережі спочатку тренуються швидко, а потім процес оптимізації зупиняється.

Опісля залишкового блоку є шар, що збільшує роздільчу здатність зображення до 256x256.

В порівнянні з вище описаними моделями, StackGAN дозволяв генерувати картинки з більшою кількістю деталей.

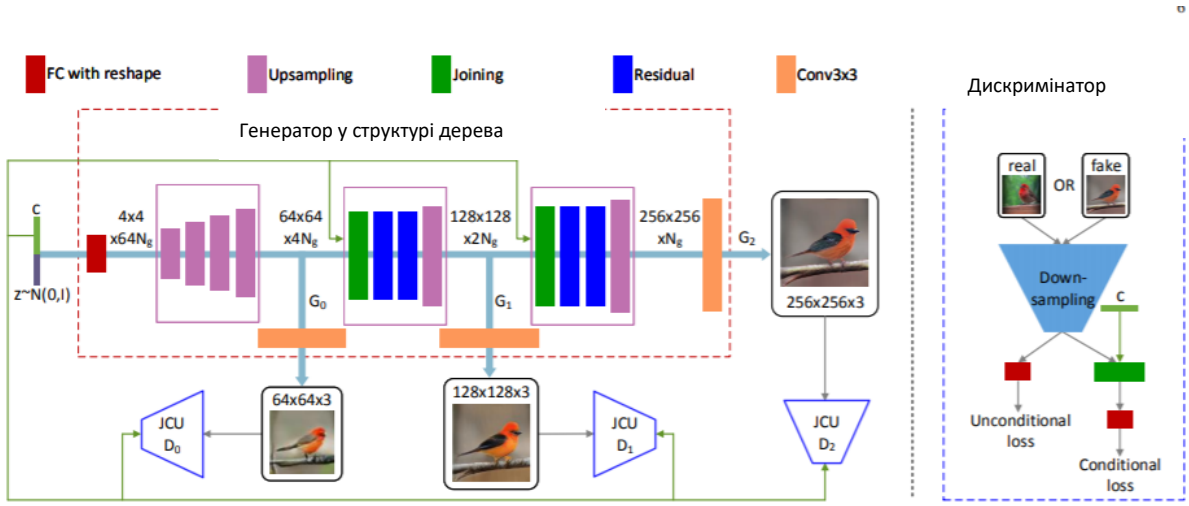


Рис. 2.5 – Архітектура StackGAN++, де c – це вектор умовних змінних, що може бути порахований з ярлика класу.

2.4 Генеративні змагальні змагальні мережі AttnGAN

Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, Xiaodong He у своїй статті [10] розглянули Attentional Generative Adversarial Network (AttnGAN) Модель містить два механізми, що дозволять їй покращити попередні результати. По-перше, автори запропонували використовувати на кожному етапі генерації зображень вектор, що відповідає слову речення. Спочатку формується зображення з низькою роздільчою здатністю. Потім поєднується вектор, що відповідає словам та вектор зображення та додаються нові деталі на кожному етапі. По-друге, автори ввели поняття Мультимодальної моделі подібності (DAMSM), що дозволяє обчислити подібність між зображеннями та реченнями.

Модель AttnGAN намагається знайти залежність між кожним пікселем картинки та словом у тексті опису.

Рис. 2.5 ілюструє нову модель для тренування, запропоновану авторами AttnGAN.

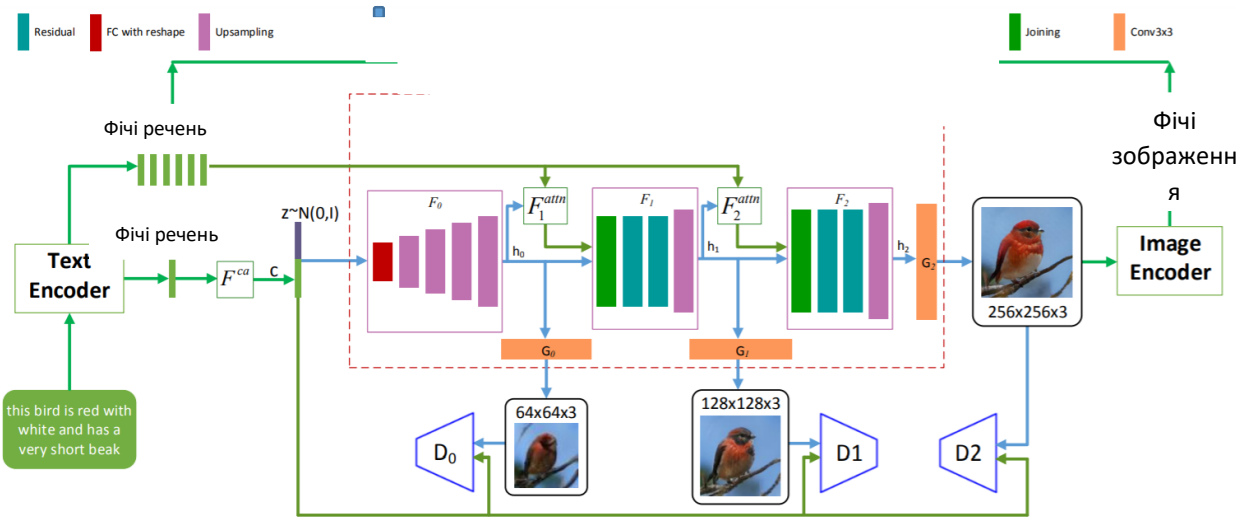


Рис. 2.5 – Архітектура AttnGAN

Модель AttnGAN використовує у фіолетовому блоці – механізм уваги або ж Attention Mechanism.

Механізм уваги ввели вперше в [13] Dzmitry Bahdanau , KyungHyun Cho, Yoshua Bengi. Механізм уваги – це архітектура типу автокодувальник. Вона містить двохзв'язну РНМ та декодер, що намагається емулювати пошук по реченню. Даний механізм можна застосовувати до машинного перекладу.

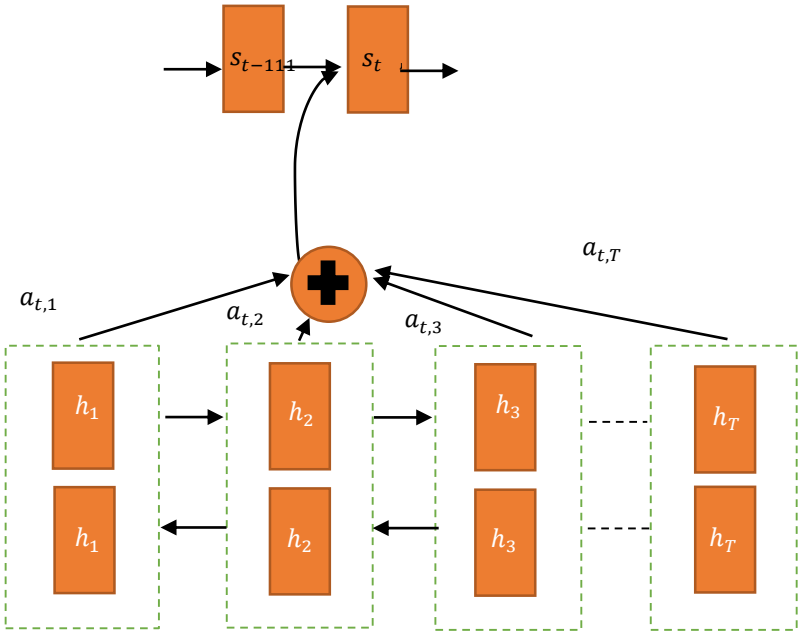


Рис. 2.6 – Архітектура механізму уваги

Рис. 2.6 ілюструє архітектуру механізму уваги. Вектор h_j - це є конкатенація прихованих станів при проході вперд та назад.

$$h_j = [\overrightarrow{h_j}; \overleftarrow{h_j}]$$

Вектори h_j – це є, по суті, представлення кожного слова в вхідному реченні.

$$c_j = \sum_{j=1}^{T_x} a_{i,j} h_j$$

Ваги $a_{i,j}$ рахуються за допомогою софтмаксу за формулою

$$a_{i,j} = \frac{e^{(e_{ij})}}{\sum_{k=1}^{T_x} e^{(e_{ik})}},$$

де

$$e_{ij} = a(s_{i-1}, h_j),$$

e_{ij} – це є вихід нейронної мережі, що описана функцією $a()$.

$a_{i,j}$ – показує як добре розташовані слова в тексті.

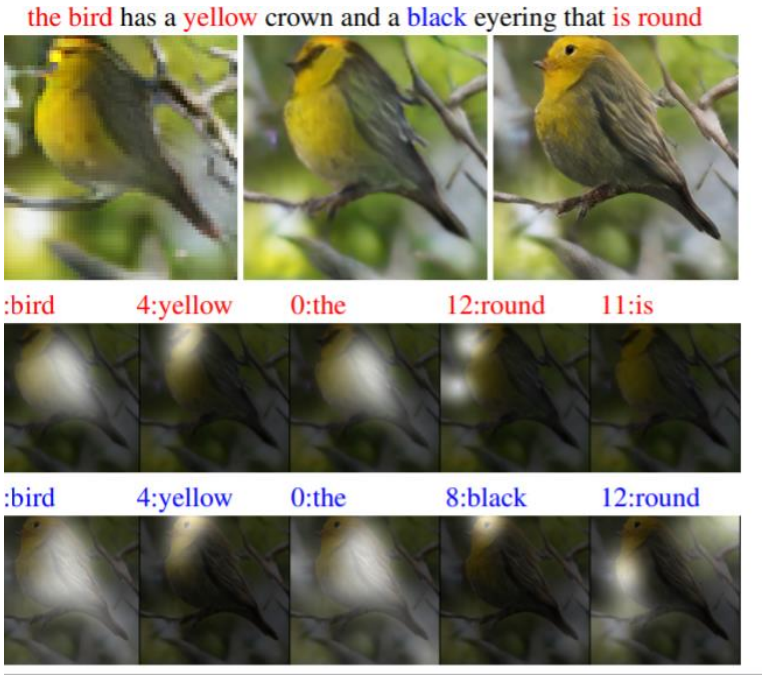


Рис. 2.7 – Ілюстрація застосування механізму уваги

Автори AttnGAN ввели нову версію механізму уваги. Для підрахунку використовувлась матриця всіх пар слів та частин малюнка.

2.5 Генеративні змагальні змагальні мережі DF-GAN

Ming Tao, Songsong Wu, Hao Tang, Nicu Sebe, Xiaoyuan Jing, Bingkun Bao, Fei Wu, запропонували модель, що є досить простою для тренування та дає хороші результати. Автори запропонували ввести DFBlock, що буде

ефективно “змішувати” ознаки отримані з текстового опису та із зображення. Matching-Aware Gradient Penalty (MA-GP) дозволяє вдосконалити дискримінатор без введення додаткових нейронних мереж. Використано натреновану авторами AttnGAN модель обробки текстів.

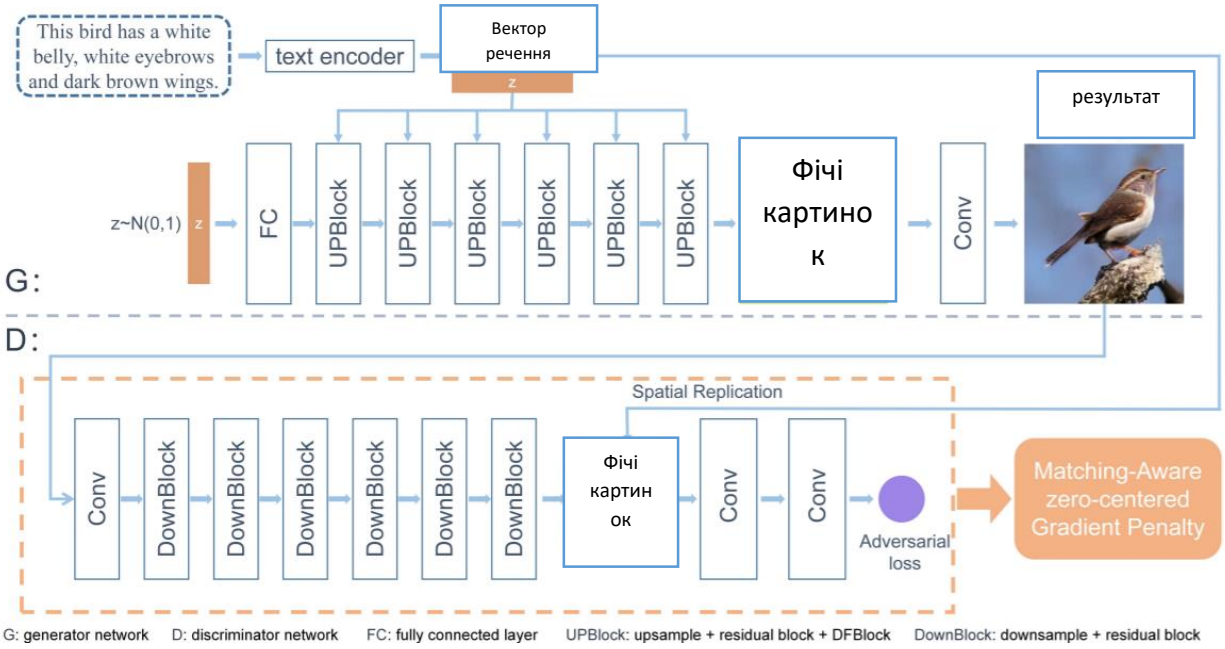


Рис. 2.6 – Модель DF-GAN

2.6 Порівняння показників FID та IS для існуючих моделів, що генерують зображення з текстового опису на прикладі датасету CUB

Для порівняння моделей використано показники FID та IS, описані у розділі 1. Вони дозволяють оцінити якість згенерованих зображень. Чим більший IS або чим менший FID тим краще зображення. За останні роки в наукових статтях майже не наводиться FID, оскільки було показано, що він не є найкращою характеристикою якості зображення.

Таблиця 2.1 – Порівняння показників IS та FID для розглянутих моделей

Модель	IS	FID
DF-GAN	4.86	
DM-GAN	4.75	
ATTNGAN	4.36	
MirrorGAN	4.56	
StackGAN	3.7	
StackGAN++	3.82	15.3
GAWWN	3.62	67.22

В таблиці 2.1 наведено, наявні в літературі показники FID та IS для датасету CUB.

РОЗДІЛ 3.
ЗАСТОСУВАННЯ МЕТОДУ DF-GAN

В результаті виконання магістерської кваліфікаційної роботи було натреновано нейронну мережу з архітектурою DF-GAN, оскільки ця нейронна мережа дає найкращі результати на обраному датасеті CUB-200-2011. В цьому розділі є опис процес тренування нейронні мережі, її тестування, опис нейронної мережі.

3.1 Елементи архітектури DF-GAN

Автори методу DF-GAN запропонували покращити модель для того, щоб уникнути проблем того, що фінальна картинка виглядає як комбінація деталей та розмитих плям, що нагадують форму описуваних об’єктів. Для цього автори запропонували використовувати архітектуру, що матиме лише один генератор та один дискримінатор, і генеруватиме зображення за один етап. Також було створено MA-GP – нову модель дискримінатора, що на думку авторів, повинно бути способом регуляризації для дискримінатора. Автори запропонували поміняти архітектуру блоків нейронної мережі, що збільшують роздільчу здатність зображення на нові та більш ефективні. Таким чином, стало можливим швидше змішувати ознаки з тексту та з картинок. *Рис. 3.1* ілюструє DF-блоки запропоновані авторами методу DF-GAN.

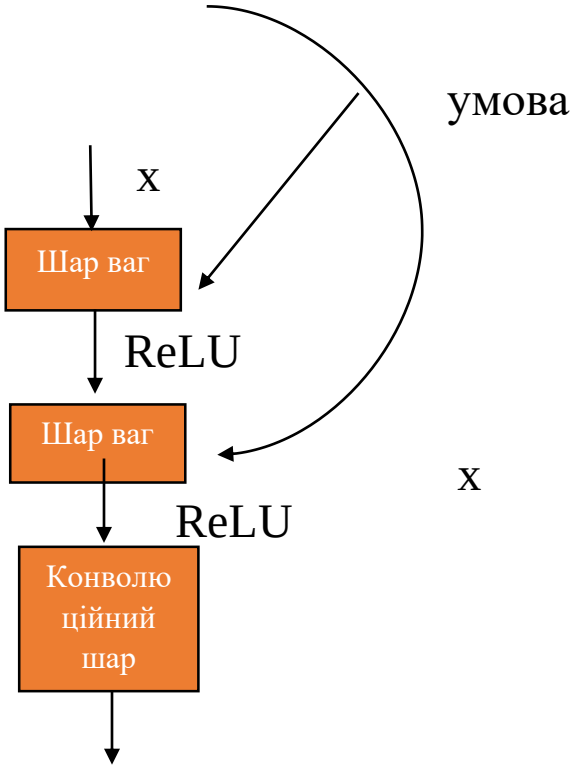


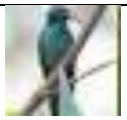
Рис. 3.1 – Архітектура DF-block у DF-GAN.

3.2. Датасети для задачі генерації картинок з їх текстового опису

Загалом, задача збору датасету для задач такого типу не надто складна, але оскільки в цій роботі була ціль порівняти різні архітектури для нейронних мереж, тому доцільніше використовувати відомі пораховані значення показників якості для порівняння зі своїми. Отже, використання вже відомого датасету спрощує роботу.

У табл. 3.1. проаналізовано число об’єктів, співвідношення кількості тренувальних прикладів до тестових, кількість видів об’єктів у різних датасетах.

Табл. 3.1 Відомі датасети для вирішення задачі генерації зображень з їх текстового опису

Назва датасету	Чисельність	Відношення тренувальних прикладів до тестових	Кількість видів об’єктів	Приклад
COCO[19]	~123000	118:5	91 вид фонових об’єктів та 80 видів речей	
CUB	~12000	1:1	1 вид (лише птахи)	
Oxford 102 Flowers[21]	~8000	6:2	1 вид (лише квіти)	
Multi-Modal-CelebA-HQ[22]	~30 000	347:81	1 вид (лише люди)	
Fashion-Gen[23]	~293000	13:3	48 видів одягу	

Проаналізувавши різні наявні датасети було вирішено обрати за основу датасет з картинками пташок та їх словесними описами – CUB-200-2011. На відміну від інших датасетів він має малу кількість прикладів, також він є досить складним для генерації зображень, оскільки зображення птахів мають багато дрібних деталей. Опис природньою мовою для картинок цього датасету містить щонайменше 10 слів без будь-якої інформації про підкатегорії та дії. Це є найбільший датасет, що стосується однієї категорії об’єктів, разом з тим він не є надто великим і містить 11,788 картинок 200 підкатегорій, що належать птахам. 5994 – для тренування, 5794 – для тестування [19] Також цей датасет є досить популярним серед дослідників задачі генерації картинок з текстового опису, тому можна було використати результати, що опубліковані в наукових статтях для порівняння зі своїми результами.

У літературі описані різні архітектури нейронних мереж для генерації картинок. У подальшому для порівняння цих архітектур використано їх Inception Score, знайдений у відповідній літературі (див. Розділ 2). Більший їх Inception Score означає вищу якість синтезованих зображень. В результаті огляду літератури було вирішено розглянути DF-GAN.

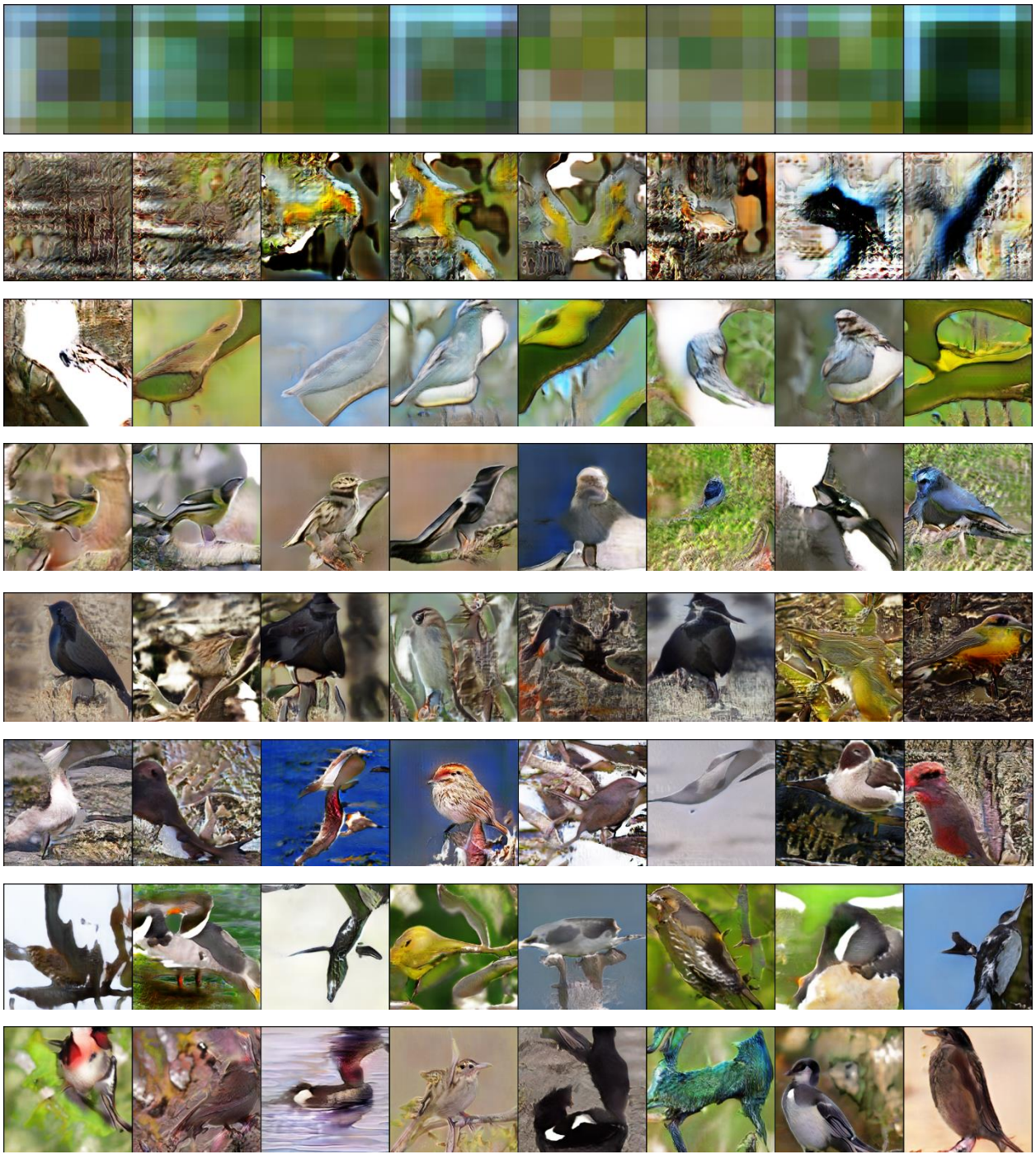
Було створено адаптацію цієї архітектури для менш потужних відеокарт, ніж ті з якими писали автори досліджень методу DF-GAN також підібрано відповідні параметри. В результаті навчання мережі – отримано результати, які наближаються до результатів отриманих авторами-розробниками DF-GAN. Також проаналізовані показники Frechet Inception Distance (FID) для отриманої нейронної мережі. Виявилось, що на відмінувід Inception Score FID не завжди є ілюстрацією якості картинки.

Для тренування мережі використано середовище Google Colab. Таким чином можна було натренувати нейронну мережу швидше, ніж це дозволяє більшість ПК без спеціальних потужних відеокарт. Оскільки, під час роботи

використовувалась бібліотека PyTorch, то можливим було лише використання відеокарт від Nvidia. Google Colab дозволяє скористатись однією з відеокарт Nvidia K80s, Nvidia T4, Nvidia P4s, Nvidia P100s, Nvidia K80s, Nvidia T4s, Nvidia P100sB. Відеокарта вибирається випадковим чином, не може буде ідентифікована, і як я зауважила може змінюватись під час одного сеансу роботи на іншу. Це дуже суттєво ускладнює оцінку швидкості тренування, тому вона не наведена в роботі.

Результати тренування на різних епохах показано у Табл. 3.2.

Таблиця 3.2 - Ілюстрація довільних зображень, отриманих на деяких етапах тренування нейронної мережі



У таблиці таблиці 3.2 показано перший ряд 1-ша епоха, другий ряд -10-та епоха, третій ряд -20-та епоха, четвертий ряд – 30-а епоха, п’ятий ряд – 50-а епоха, шостий ряд – 50-а епоха, сьомий ряд – 70-а епоха, восьмий ряд – 80-а епоха, дев’ятий ряд – 90-а епоха

Для аналізу показників якості нейронної мережі використано бібліотеку torch-fidelity та функцію calculate_metrics. Це дозволяє отримувати результати у вигляді словника

{'frechet_inception_distance': 119.9746678408907,
'inception_score_mean': 4.7903676931936,
'inception_score_std': 0.12391703982956095}

На останній епосі вдалось отримати результат 4.79. Автори статті отримали результат 4.85, але вони тренували свою нейронну мережу впродовж 600 епох. Тренування мережі в роботі зайняло чимало часу, наступна після GAN мережа – DMGAN мала дещо гірший результат, ніж 4.85, тому було прийнято рішення припинити подальше тренування.

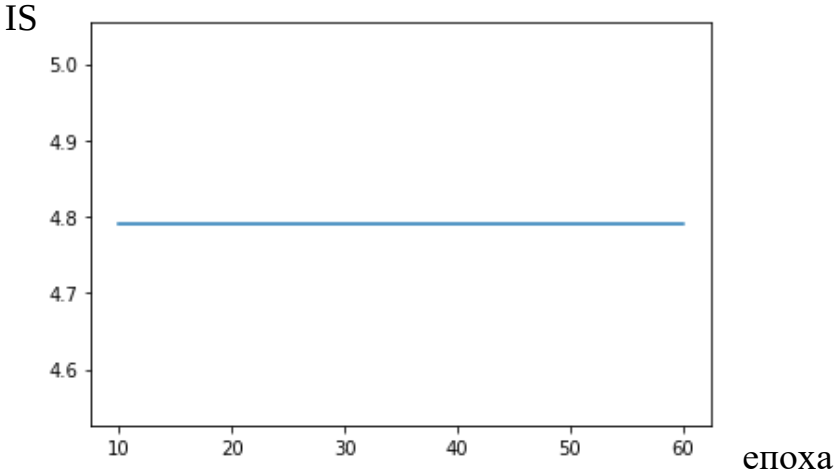


Рис. 3.2 – Залежність IS від номеру епохи для моделі DF-GAN.

З рис. 3.2 видно, як змінюється IS впродовж тренування нейронної мережі. На рис 3.3 зображено графік залежності FID від номеру епохи. Видно, що нейронна мережа покращує свій FID і на останніх ітераціях він майже не змінний. Графік зображений для кожної 10 ітерації. FID та IS побудовані за допомогою, вже згаданого torch-fidelity.

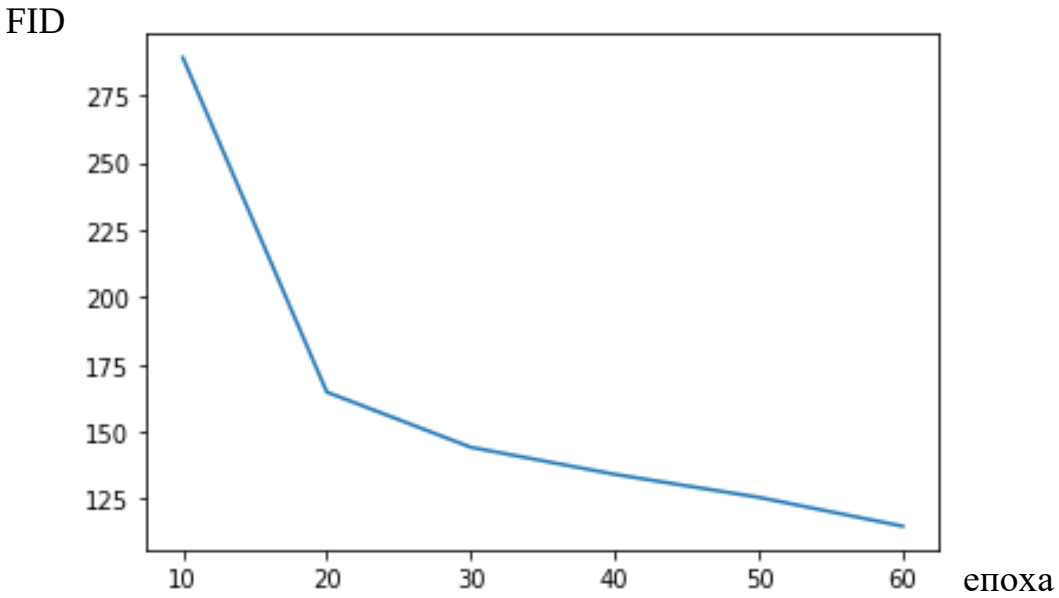


Рис.3.3 – Залежність FID від номеру епохи для моделі DF-GAN.

Отримані результати на рис. 3.3 дозволяють припустити, що модель при подальшому тренуванні майже не буде змінювати свої показники.

РОЗДІЛ 4.
ЗАСТОСУВАННЯ МЕТОДУ DM-GAN

У цьому розділі описано архітектуру DM-GAN, та процес тренування нейронної мережі DM-GAN.

4.1 Елементи архітектури DM-GAN

Архітектура DM-GAN зображена на *рис. 4.1*. Вона є двохетапною так як і StackGAN, StackGAN++, AttnGAN. Тобто спочатку ми генеруємо початкове зображення, а потім покращуємо його якість шляхом введення ще одного генератора та дискримінатора. Для другого етапу автори методу запропонували механізм “динамічної пам’яті”.

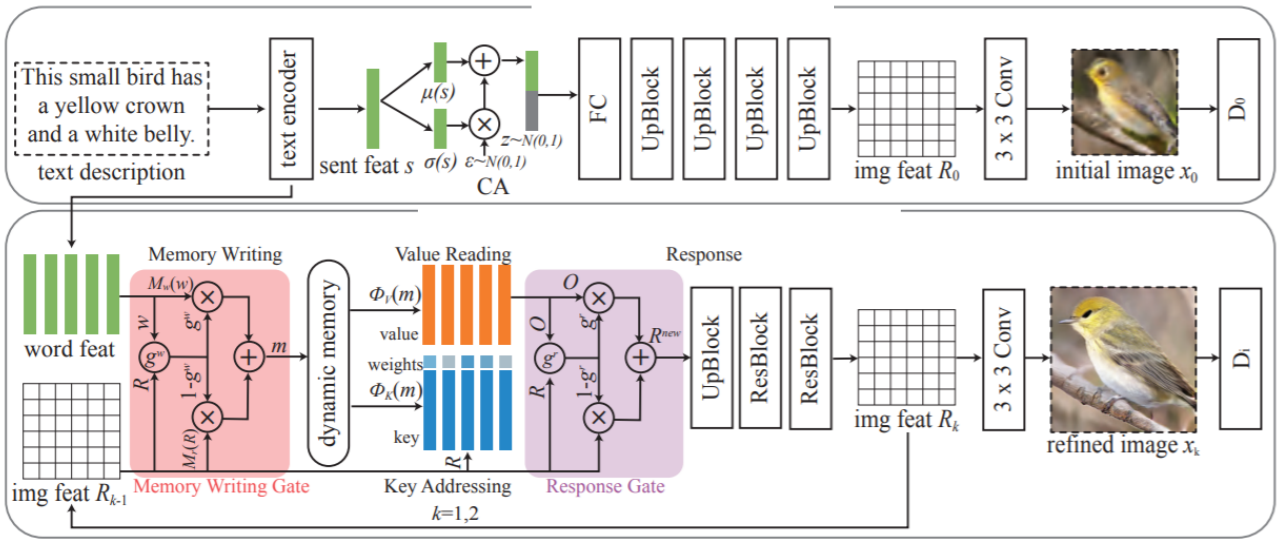


Рис. 4.1 – Архітектура DM-GAN.

4.2 Тестування моделі DM-GAN

Аналогічно, як і для DF-GAN для тренування використовуватимемо GoogleColab. Натренуємо модель для 70 ітерацій для того, щоб порівняти її з DF-GAN.

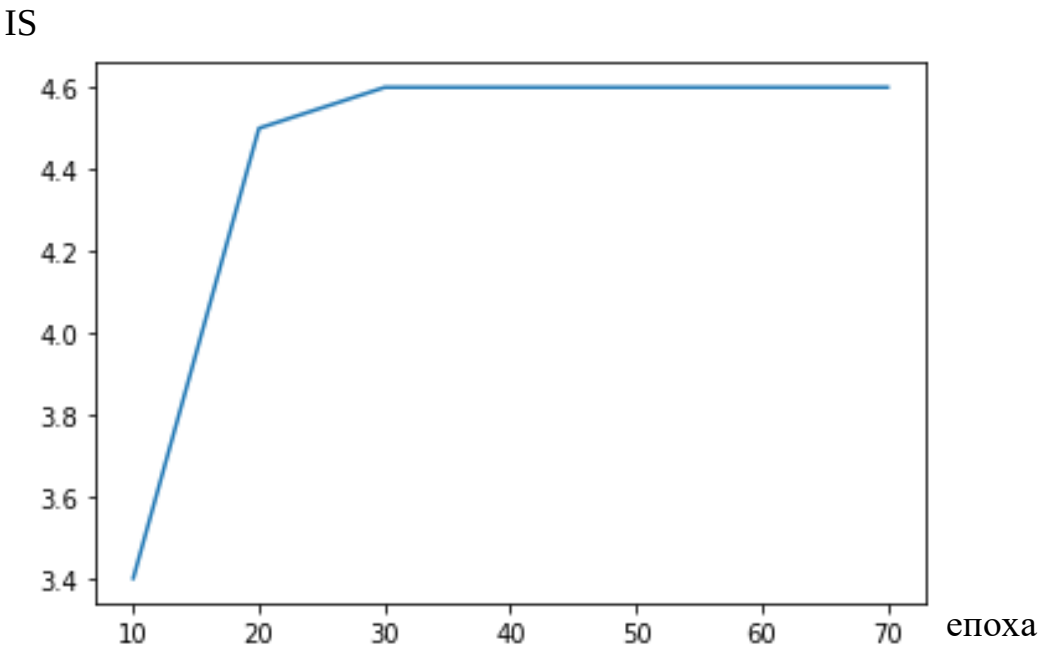


Рис.4.2 – Залежність IS від номеру епохи для моделі DF-GAN.

Аналогічно, як і в попередньому розділ проаналізуємо як веде себе IS впродовж тренування нейронної мережі. На *рис. 4.2* показано на осі абсцис – номер епохи, а на осі ординат - IS . Графік побудовано для кожної десятої ітерації. На *рис 4.3* зображено графік залежності FID від номеру епохи. Видно, що нейронна мережа покращує свій IS та FID і на останніх ітераціях він майже не змінний, проте отримані значення є гіршими, ніж для моделі DF-GAN, що ще раз підверджує результат отриманий авторами архітектури DF-GAN у [5].

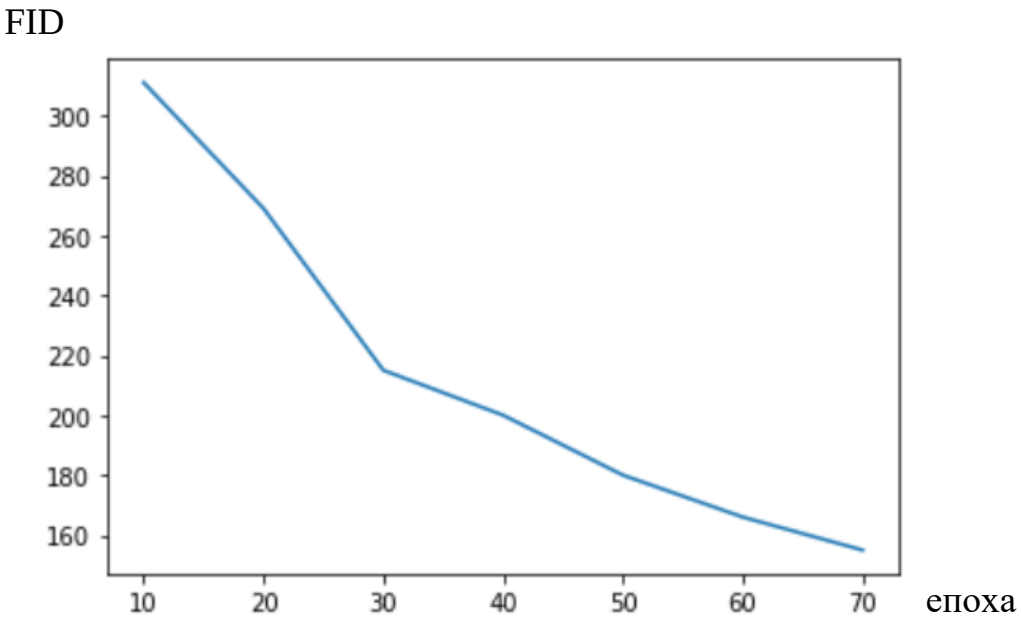


Рис.4.3 – Залежність FID від номеру епохи для моделі DM-GAN.

РОЗДІЛ 5.
DM-GAN З ВИКОРИСТАННЯМ DF-БЛОКІВ

У цьому розділі запропоновано нову архітектуру нейронних мереж, що дозволяє покращити вже отримані результати.

5.1 Архітектура DM-GAN з використанням DF-блоків

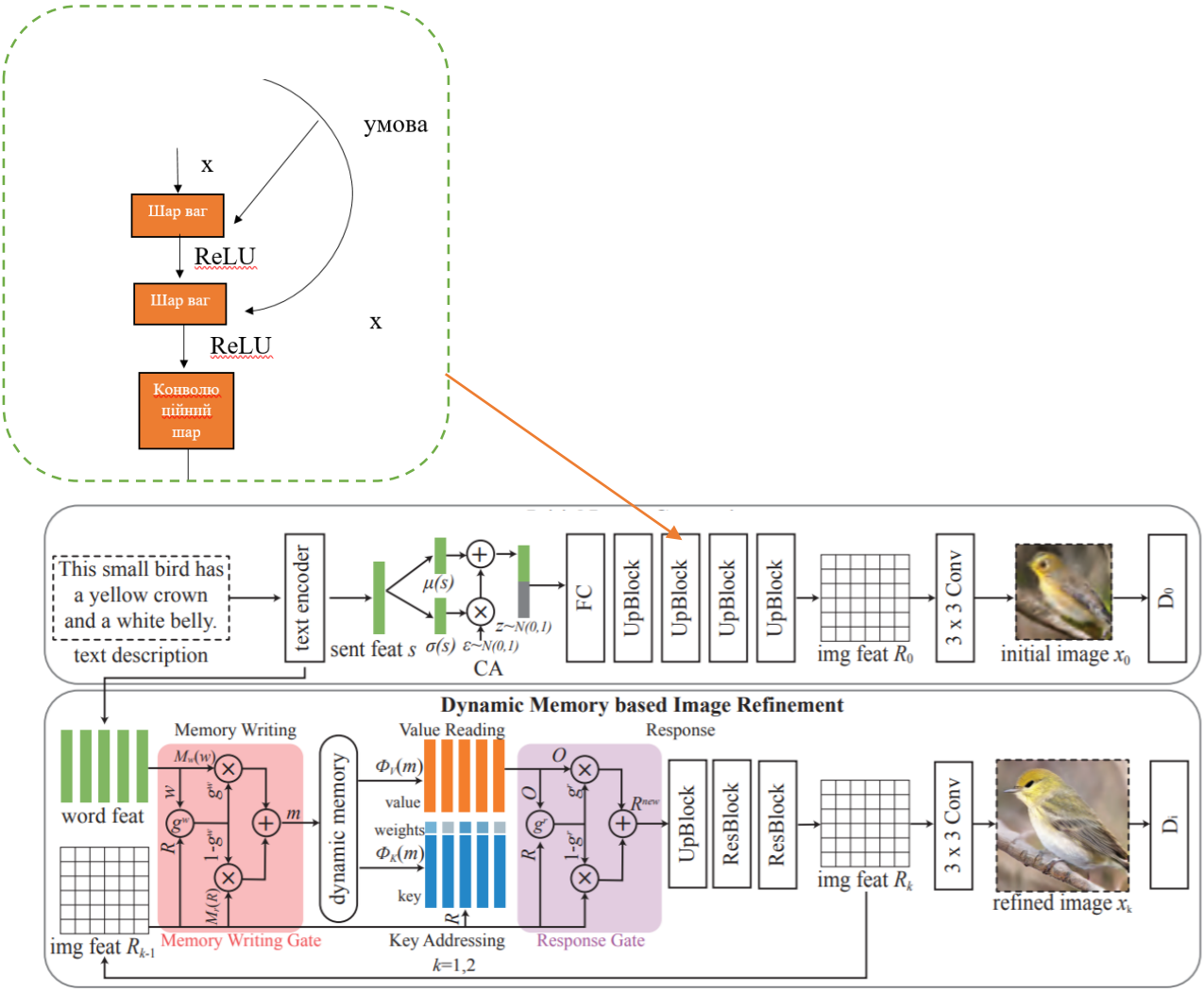


Рис. 5.1– Архітектура DM-GAN з додаванням DF блоків.

Використаємо DF-блоки, описані в третьому розділі для того, щоб покращити результати DM-GAN, додавши їх в генератор моделі DM-GAN. На *Рис. 5.1* зображено, як виглядає нова архітектура.

5.2 Тестування моделі DM-GAN з використанням DF-блоків

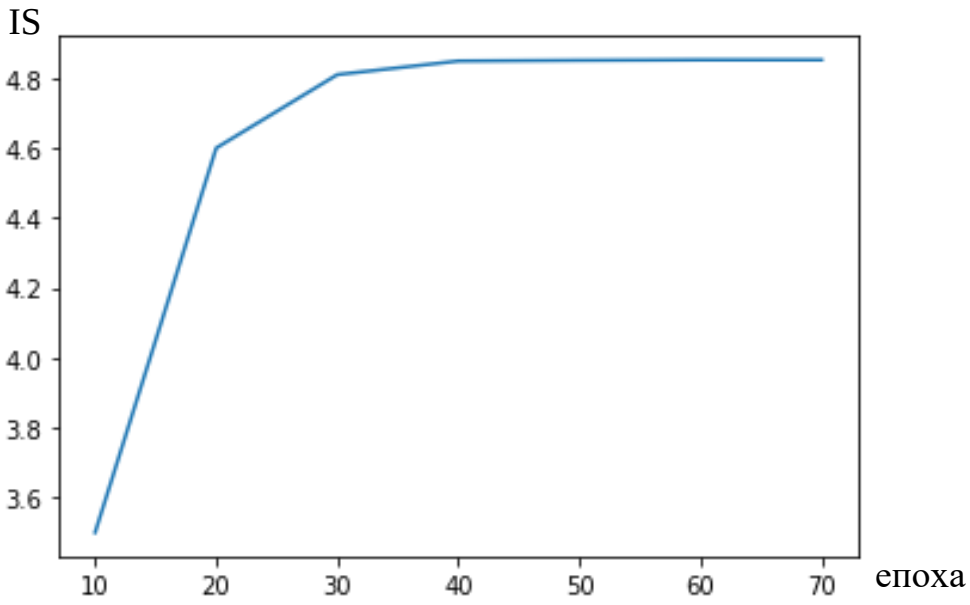


Рис.5.2 – Залежність IS від номеру епохи для моделі DF-GAN.

На *рис. 5.2* на осі абсцис – номер епохи, а на осі ординат - IS. Графік побудовано для кожної десятої ітерації. На рис 5.3 зображено графік залежності FID від номеру епохи. Графік теж побудовано для кожної десятої ітерації.

Впродовж тренування проміжні файли моделей зберігались як файли з розширенням .pth для того, щоб у разі виходу за ліміт часу передбачений Google Colab відновити роботу. Також це дозволяло згодом побудувати графіки залежностей IS від епохи.

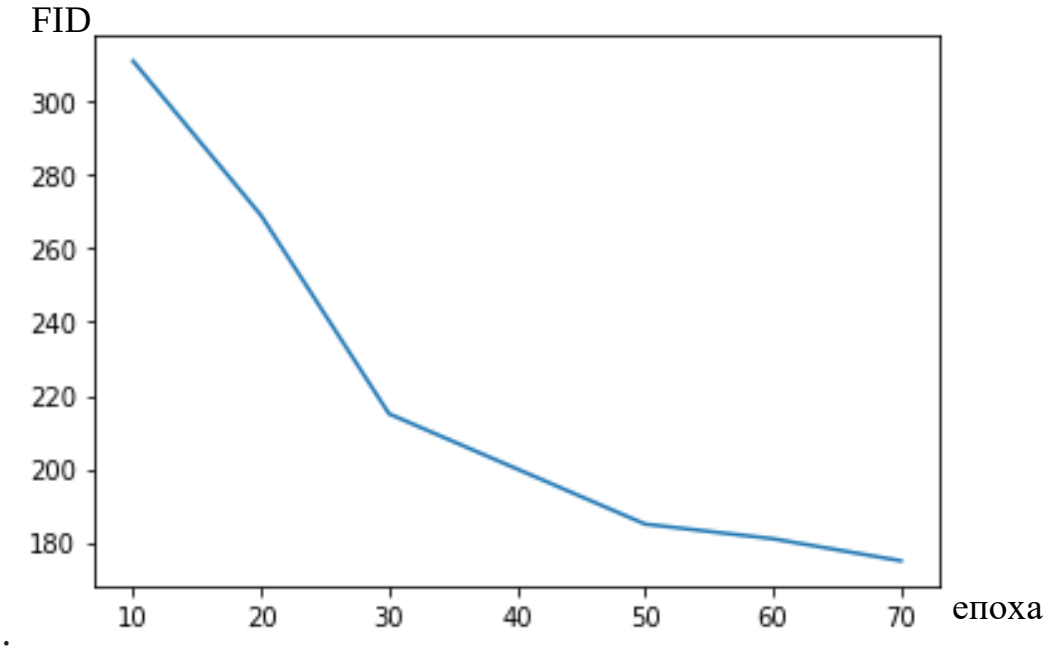


Рис.5.3 – Залежність FID від номеру епохи для моделі DF-GAN.

5.2 Порівняння розглянутої моделі з моделями DM-GAN та DF-GAN

Видно, що отримані значення FID є дещо гіршими, ніж для моделі DM-GAN, проте у статті про DF-GAN [5] автори стверджують, що показник FID не є настільки достовірним як показник IS. Досягнуто значення 4.846 показника IS, що показує, що отримана нейронна мережа генерує більш якісні зображення.

Час тренування займає досить багато часу, тому з отриманих результатів за 70 епох не можна стверджувати, що ця архітектура є кращою чи гіршою за інші архітектури. Автори DF-GAN та DM-GAN тренували свої моделі впродовж 300 епох. На жаль, складність обчислень не дозволяє відтворити подібний експеримент на відеокарті в Google Colab за розумний час. З іншого

боку, графік залежності IS від епохи показує, що модель ще може покращити свої результати.

Попри невелику кількість епох усі моделі, що були натреновані досягли результатів IS та FID більших, ніж ті моделі, що були створені до DMGAN та DFGAN – наприклад, AttnGAN, StackGAN ++, MirrorGAN і тд.

У додатках А, Б, В, Г, Д, Е, Ж наведені візуалізації нейронних мереж, розглянутих у попередніх розділах. Для візуалізації було використано Netron.

ВИСНОВКИ

Було проведено аналіз існуючих методів розв’язування задачі генерації картинок з текстового опису, обгрунтовано, чому той чи інший метод працював краще чи гірше. Проілюстровано архітектури нейронних мереж GAN, які використовуються для генерації зображень з текстів – GAWWN, StackGAN, StackGAN++, AttnGAN, DMGAN, DFGAN. Розглянуто елементи цих архітектур, показано принципи їх роботи.

У роботі запропоновано удосконалити уже відому архітектуру мережі DM-GAN шляхом додаванням DF-блоків. Обгунтовано доцільність вибору такої архітектури. Пораховано для реалізованої моделі показники IS та FID, використовуючи датасет CUB-200-2011, та порівняно результати з вже описаними у літературі.

Практичною частиною диплому було тренування нейронної мережі DF-GAN, DF-GAN та DM-GAN з додаванням DF-блоків для генерації зображень з текстового опису. Показано, як працює ця нейронна мережа для датасету CUB-200-2011, що містить картинки пташок та їх текстові описи. Показано, як змінюється IS та FID в залежності від номеру епоху для всіх перечислених архітектур.

Попри високу складність обчислень та довгий час тренування мережі вдалось досягти результатів наближених до тих, що отримали розробники архітектур DF-GAN та DM-GAN. Більше того, отримані результати є кращі, ніж результати усіх моделей, що створені раніше, ніж DF-GAN та DM-GAN.

Всі обчислення здійснювались з використанням Google Colab, GPU, хмарних сервісів для зберігання та обробки даних. Це дало змогу суттєво пришвидшити час тренувань.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ian J. Goodfellow Generative adversarial nets. [Text] / Jean Pouget-Abadie, Mehdi Mirza, Bing Xu [et al] — Proceedings of the 27th International Conference on Neural Information Processing Systems Volume 2 NIPS'14. — USA: MIT Press, Cambridge, 2014,- 2672–2680 2
2. Zhang H., Xu, Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. [Text] / T., Li, H., Zhang, S., Wang, [et al]. — Proceedings of the IEEE international conference on computer vision, 2017 - 5907-5915
3. Scott Reed Learning what and where to draw. [Text] / Zeynep Akata, Santosh Mohan, Samuel Tenka [et al] - 30th International Conference on Neural Information Processing Systems NIPS'16 - USA: Curran Associates Inc., Red Hook, NY, 2016 - 217–225.
4. Xu, T Attngan: Fine-grained text to image generation with attentional generative adversarial networks [Text] / Zhang, P., Huang, Q., Zhang, H., Gan, Z. [et al]. - X. Proceedings of the IEEE conference on computer vision and pattern recognition, 2018 - 1316-1324.
5. Tao, M., Df-gan: Deep fusion generative adversarial networks for text-to-image synthesis. / Tang, H., Wu, S., Sebe, N. [et al]. - 2020 arXiv preprint arXiv:2008.05865
6. Zhu, M Dm-gan: Dynamic memory generative adversarial networks for text-to-image synthesis. / Pan, P., Chen, W., Yang, Y. - Proceedings of

the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019 - 5802-5810

7. Salimans, T. Improved techniques for training gans. / Goodfellow, I., Zaremba, W., Cheung, V., Radford [et al] - 2016, arXiv preprint arXiv:1606.03498.
8. Barratt, S., (2018). A note on the inception score. /Sharma, R- 2018, arXiv preprint arXiv:1801.01973.
9. Heusel, M Gans trained by a two time-scale update rule converge to a local nash equilibrium. /., Ramsauer, H., Unterthiner, T., Nessler, B [et al] - 2017 arXiv preprint arXiv:1706.08500.
10. Nagurney, A. Network economics: A variational inequality approach (Vol. 10). - USA: Springer Science & Business Media. 2013
11. Корпелевич Г. М. Экстраградиентный метод для отыскания седловых точек и других задач. - Экономика и математические методы. 1976. Т. 12, № 4
12. Gidel A variational inequality perspective on generative adversarial networks./ G., Berard, H., Vignoud, G., Vincent, P [et al] 2018 arXiv preprint arXiv:1802.10551.
13. Горбань А. Н. Обобщенная аппроксимационная теорема и вычислительные возможности нейронных сетей - Архивная копия от 27 января 2012 на Wayback Machine Сибирский журнал вычислительной математики, 1998, т. 1, № 1. — 12—24.
14. Миркес Е. М. Логически прозрачные нейронные сети и производство явных знаний из данных Нейроинформатика / А. Н. Горбань, В. Л. Дунин-Барковский, А. Н. Кирдин [и др.] — Новосибирск: Наука. Сибирское предприятие РАН, 1998. — 296 с. — ISBN 5-02-031410-2.

15. Bahdanau, D., Neural machine translation by jointly learning to align and translate. — 2014 arXiv preprint arXiv:1409.0473.

16. Микелуччи, У. Прикладное глубокое обучение. Подход к пониманию глубоких нейронных сетей на основе метода кейсов. Cho, K., & Bengio, Y. — 2020 Пер. с англ., СПб.: БХВ-Петербург.

17. Le Cun Y Deep learning Nature/ Bengio Y., Hinton — G.521 2015, 436-444, 10.1038/nature14539

18. Ramesh, A. Zero-shot text-to-image generation/ Pavlov, M., Goh, G., Gray [et al] — 2021, arXiv preprint arXiv:2102.12092.

19. Wah, The Caltech-UCSD Birds-200-2011 Dataset. / Branson S., Welinder P., Perona P. — Computation & Neural Systems Technical Report, CNS-TR-2011-001.

20. Lin, T. Y Microsoft coco: Common objects in context. / Maire, M., Belongie, S., Hays, [et al] — European conference on computer vision— USA: Springer, Cham. 2014 — 740-755

21. Nilsback, M. E Automated flower classification over a large number of classes/ Zisserman, A. — 2008 Sixth Indian Conference on Computer Vision, 2008— USA:IEEE, Graphics & Image Processing — 722-729.

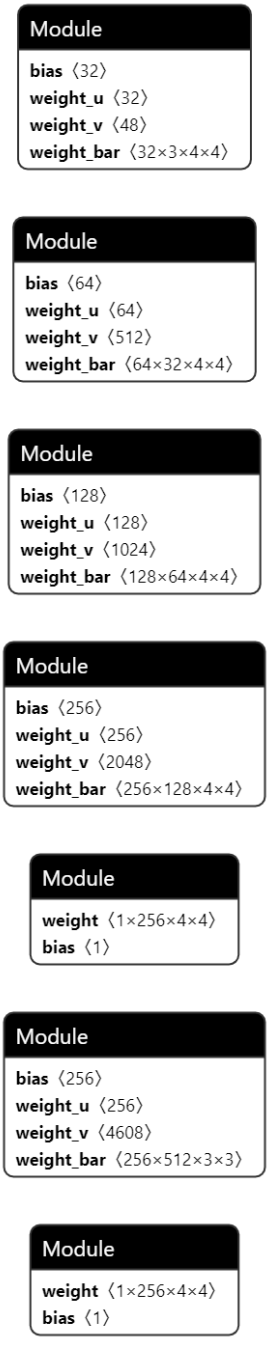
22. Xia, W. TediGAN: Text-Guided Diverse Image Generation and Manipulation./ Yang, Y., Xue, J. H., Wu, B — 2020, arXiv preprint arXiv:2012.03308.

23. Rostamzadeh, N., Fashion-gen: The generative fashion dataset and challenge./ Hosseini, S., Boquet, T., Stokowiec — 2018, arXiv preprint arXiv:1806.08317.

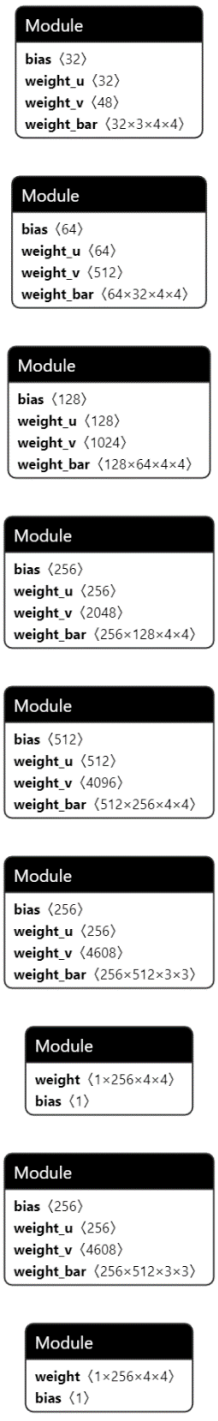
24. Xia, W. Towards Open-World Text-Guided Face Image Generation and Manipulation. / Yang, Y., Xue, J. H., & Wu, B. — 2021, arXiv preprint arXiv:2104.08910.
25. Harris, E. Torchbearer: A model fitting library for PyTorch./ Painter, M., & Hare, J. — 2018 arXiv preprint arXiv:1809.03363.

ДОДАТКИ

ДОДАТОК А.
ДИСКРИМІНАТОР 1 МОДЕЛІ DM-GAN



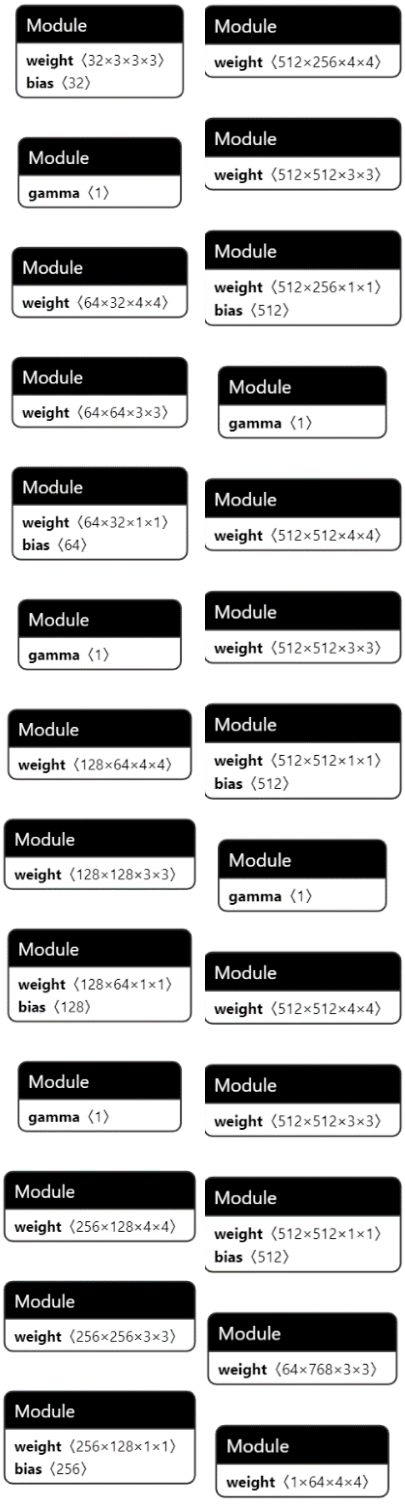
ДОДАТОК Б.
ДИСКРИМІНАТОР 2 МОДЕЛІ DM-GAN



ДОДАТОК В.
ДИСКРИМІНАТОР З МОДЕЛІ DM-GAN

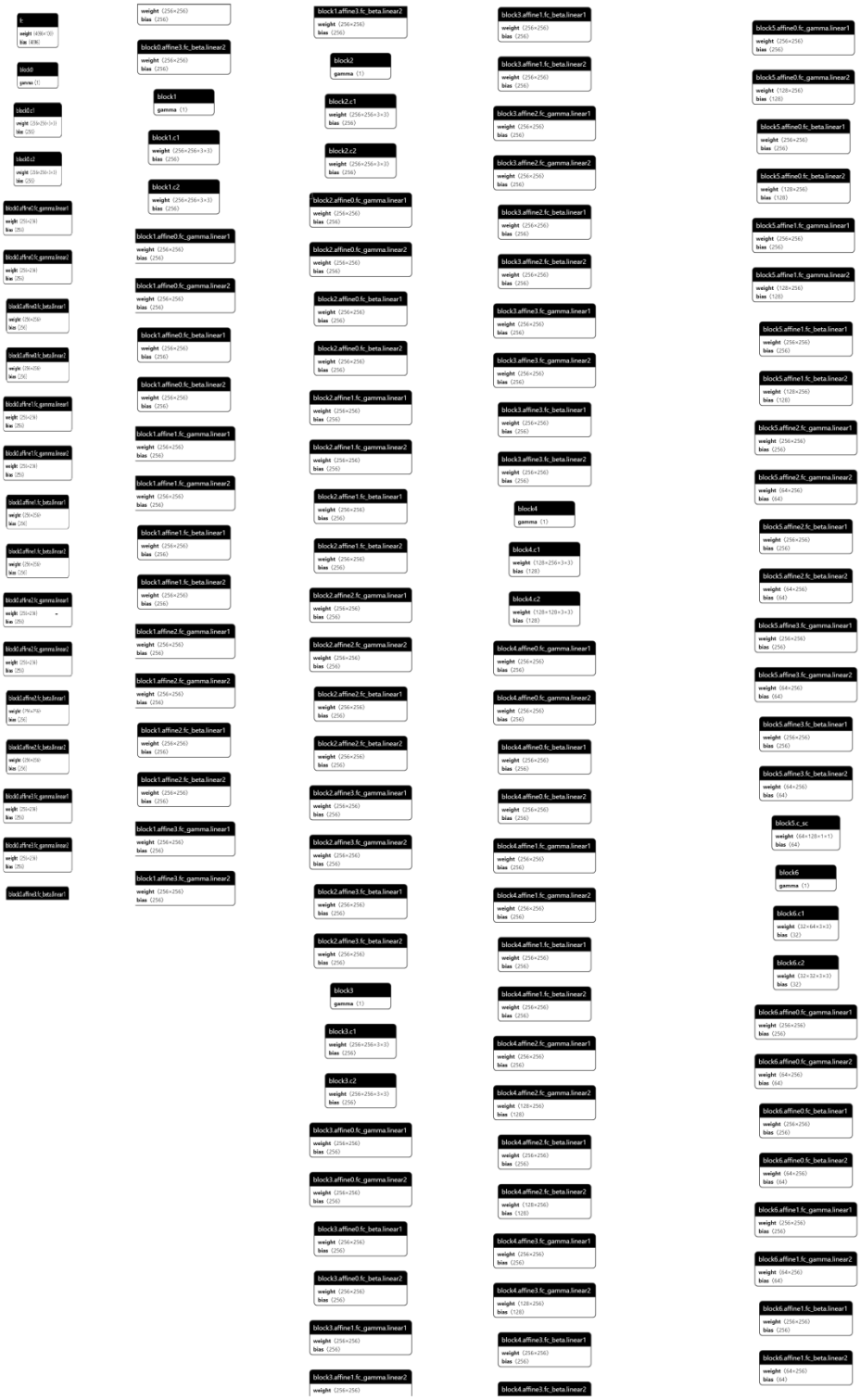
ДОДАТОК Г.
ГЕНЕРАТОР МОДЕЛІ DM-GAN

ДОДАТОК Д.
ДИСКРИМІНАТОР МОДЕЛІ DF-GAN



ДОДАТОК Е.

ГЕНЕРАТОР МОДЕЛІ DF-GAN



ДОДАТОК Ж.

ГЕНЕРАТОР МОДЕЛІ DM-GAN З ВИКОРИСТАННЯМ DF-БЛОКІВ