СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

імені ВОЛОДИМИРА ДАЛЯ

(м. Сєвєродонецьк)

Факультет Інформаційних технологій та електроніки
(повне найменування факультету)

Кафедра Програмування та математики
(повна назва кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломної роботи

освітньо-кваліфікаційного рівня магістр
(бакалавр, спеціаліст, магістр)

напряму підготовки 122 Комп'ютерні наукі та інформаційні технології
(шифр і назва напряму підготовки)

спеціальності Інформатика
(шифр і назва спеціальності)

на тему Застосування генеративних нейронних мереж з метою поліпшення точності розпізнавання образів на зображеннях

Виконав: студент групи ІНФ-16дм

Шамшін С.О. .....

(прізвище, та ініціали) (підпис)

Керівник Даніч В.М. .....

(прізвище та ініціали) (підпис)

Завідувач кафедри Лифар В. О.. .....

(прізвище та ініціали) (підпис)

Рецензент _____ .....

(прізвище та ініціали) (підпис)

Сєвєродонецьк - 2018

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

імені ВОЛОДИМИРА ДАЛЯ

(м. Сєвєродонецьк)

Факультет Інформаційних технологій та електроніки

Кафедра Програмування та математики
Освітньо-кваліфікаційний рівень магістр

(бакалавр, спеціаліст, магістр)

Спеціальність Інформатика

(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____
В.О.Лифар
“____” _____ 2017
року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТУ

Шамшіну Сергію Олександровічу

(прізвище, ім’я, по батькові)

1. Тема проекту (роботи) Застосування генеративних нейронних мереж з метою поліпшення точності розпізнавання образів на зображеннях

керівник проекту (роботи) Даніч В.М.,

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від “08” листопаду 2017 року
№_____

2. Строк подання студентом проекту (роботи)_____

3. Вихідні дані до проекту (роботи) Матеріали науково-дослідної практики

_____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1) Аналіз існуючих моделей нейронних мереж;

2) Розробка моделі згорткової нейронної мережі;

3) Розробка алгоритму навчання нейронної мережі;

4) Втілення моделі в програмному продукті;

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Актуальність теми і постановка завдань дослідження.	02.10.17-10.10.17	
2	Аналіз існуючих моделей нейронних мереж	11.10.17-12.11.17	
3	Розробка моделі згорткової нейронної мережі	13.11.17-25.11.17	
4	Розробка алгоритму навчання нейронної мережі	25.11.17-30.12.17	
5	Втілення моделі в програмному продукті	31.12.17-04.01.18	
6	Оформлення пояснювальної записки	05.01.18-8.01.18	

Студент _____ Шамшин С.О.

(підпис) (прізвище та ініціали)

Керівник проекту (роботи) _____ Даніч В.М.

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту: 78**с.,** 37 **рис.,** 34 **посилань.**

Предмет дослідження: механізми процесінгу цифровіх зображень із вікорістанням нейронних мереж.

Мета роботи: дослідження можливості використання технік машинного навчання, таких як глибокі нейронні мережі у сфері обробки цифрових зображень та в завданні підвищення роздільної здатності зображень.

В роботі описані методи обробки цифрових зображень, а саме підвищення роздільної здатності, за допомогою методів глибоко навчання, зокрема глибоких згортальних нейронних мереж, з метою поліпшення якісних показників роботи алгоритмів розпізнавання.

Актуальність роботи обумовлена зростаючою кількістю завдань, де з успіхом застосовуються ті чи інші алгоритми розпізнавання, і одночасно з цим обмеженістю можливості їх застосування внаслідок недосконалості джерел даних для розпізнавання.

Ключовим фактором, який впліає на ймовірність успішного розпізнавання образу на зображенні, крім самого алгоритму розпізнавання, є наявність на цифровому зображенні достатньої кількості інформації, що можливо тільки при достатній роздільної здатності цифрового зображення.

Робота полягає в розробці програмного продукту, в основі якого лежать генеративні змагальні нейронні мережі, який би зміг значно збільшувати роздільну здатність цифрових зображень, таким чином якісно поліпшувати результати роботи алгоритмів розпізнавання.

ЦИФРОВІ ЗОБРАЖЕННЯ, РОЗПІЗНАВАННЯ ОБРАЗІВ, ГЛІБОКЕ НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ, ГЕНЕРАТІВНІ ЗМАГАЛЬНІ МЕРЕЖІ.

Умови отримання дипломного проекту
93400 м. Сєвєродонецьк , вул. Донецька, 41.

ABSTRACT

Explanatory note to the diploma project: 78 sec., 37 Fig., 34 links.

Subject of study: digital image processing using deep neural networks

Objective: To explore the opportunity of using machine learning, such as deep neural networks in the field of digital image processing and in image super-resolution task.

The paper describes methods for processing circus images, namely, super-resolution, using deep learning techniques, in particular deep convolutional neural networks, in order to improve the qualitative performance of recognition algorithms.

The urgency of the work based on increasing number of tasks, where algorithms of recognition are successfully applied, and at the same time with the limited possibility of their application due to the imperfection of data sources for recognition.

The key factor that influences the likelihood of successful image recognition on the image, apart from the recognition algorithm itself, is the presence of sufficient information on a digital image, which is only possible with sufficient resolution of the digital image.

The main goal is developing prototype of a software product, based on Generative-Adversarial Networks that could significantly improve the resolution of digital images, improving the performance of recognition algorithms.

DIGITAL IMAGE, IMAGE RECOGNITION, DEEP LEARNING, NEURAL NETWORKS, GENERATIVE-ADVERSARIAL NETWORK.

For manuscript contact
93400 Ukraine, Lugansk reg., Severodonetsk,Donetcka, 41

Зміст

! "#$%

Область інформатики та математики, присвячена штучним нейронних мереж, швидко розвивається останнє десятиліття. Штучні нейронні мережі - математичні моделі, організовані за образом і подобою організації та реалізації мереж нервових клітин ссавців в цілому, і нервової системи людини зокрема. Дослідження в цій області знаходять своє практичне застосування у величезній кількості прикладних задач. Це розпізнавання візуальних образів, подібно до того як людський мозок розпізнає візуальну інформацію, штучні нейронні мережі розпізнають образи і іншу інформацію на цифрових зображеннях і відео. Так само своє застосування вони знаходять у перетвореннях і трансформації зображень і відео, крім цього вони вже сьогодні з успіхом застосовуються для обробки мов, апроксимації та символьної регресу функцій, прогнозування та передбачення деяких явищ в майбутньому, грунтуючись на зібраних до цього моменту даних. Глибокі та надглібокі штучні нейронні мережі вже сьогодні застосовуються для машинного перекладу з однієї природної мови на іншу, передбачення результатів спортивних матчів, розпізнавання і сегментації об'єктів на відео в режимі реального часу, що знаходить своє застосування з прикладних розробках безпілотних засобів пересування. Однак така велика кількість різноманітних завдань, які можуть вирішувати нейронні мережі, призводить до того що на сьогоднішній день неможливо створити універсальне потужне рішення, здатне однаково добре виконувати будь-яке завдання, змушуючи дослідників і розробників розробляти більш цілеспрямовані рішення, що працюють за певними алгоритмами і виконують конкретну задачу. При цьому, кількість прикладних програмних продуктів, заснованих на нейронних мережах, лавиноподібно росте з року в рік останні роки. Одним з перших прикладних застосувань стало розпізнавання поштових кодів на пошті США за допомогою згорткової мережі. З тих пір, завдяки зростанню обчислювальної потужності, яка доступна дослідникам, було запропоновано і покращено величезну кількість рішень і алгоритмів, в тому

числі в сфері розпізнавання образів в цілому і людей в частномті.

У 2011 році компанії-гіганти анонсували системи розпізнавання людей, якісно поліпшуючі досвід корістувача. Завдяки їм стало можливим автоматичний пошук по фотографії людини в соціальній мережі facebook, або в месенджері Skype компанії Microsoft. При цьому з року в рік системи стають все краще і все менш схильні до таких факторів як зміна освітленості, кута зору, зачіски людини, наявності бороди або окулярів на обличчі і так далі. У 2017 року компанія Apple представила свій новий смарфтон iPhone X, корторий широко використовує нейронні мережі для розпізнавання свого власника, міміки обличчя і так далі, на чому побудований кардинально новий досвід користувача.

Говорячи про розпізнавання образів, мається на увазі можливість визначити що саме представлено на растровому цифровому зображенні - яке має певну кількість точок і несе в собі певну кількість цифрової інформації. Для забезпечення достатньої точності розпізнавання зображення повинно мати достатнью роздільну здатність та нести в собі достатню кількість інформації. Роздільною здатністю називається кількість точок зображення поділена на одиницю площі. Досить багато завдань комп'ютерного зору залежать від того, що вхідне зображення має достатню роздільну здатність.

Збільшення кількості точок зображення з мінімальною втратою інформації є важливою частиною великої кількості завдань комп'ютерного зору, так як дозволяє поліпшити якість роботи алгоритмів комп'ютерного зору, а саме алгоритмів розпізнавання образів, що в свою чергу покращує досвід користувача, якщо мова йде про прикладне програмноме забезпечення, знижує фінансові витрати і покращувати якість бізнес-процесів, якщо використовується в промисловості, покращує рівень безпеки і захищеності від несанкціонованого вторгнення, якщо використовується в охоронній системі, і так далі. Цей процес отримав назву суперрезолюції (англ. super-resolution).

Розпізнавання образів - це віднесення вихідних даних до певного класу за допомогою виділення істотних ознак, що характеризують ці дані, із

загальної маси несуттєвих даних. Таким чином, маючи деяку кількість класів, ми можемо оцінити якість роботі алгоритма розпізнавання по тому, наскільки високу ймовірність приналежності до потрібного класу зможе забезпечити деякий алгоритм розпізнавання. У даній роботі, ми будемо оцінювати, наскільки точно деякий алгоритм класифікації буде відносити до потрібного класу об'єкт на зображенні низького дозволу, зображення збільшеного за допомогою бикубической інтерполяції, а так само зображення отриманого в результаті роботи запропонованої нейронної мережі, а так само еталонного зображення, при цьому якість роботи алгоритму буде оцінюватися як різниця між результатом розпізнавання об'єкта на еталонному зображенні і зображенні, яке було отримано в результаті роботи запропонованої системи, таким чином встановити наскільки алгоритм здатний відтворювати вихідне зображення.

Мета роботи полягає в дослідженні можливості використання технік машинного навчання, таких як глибокі нейронні мережі у сфері обробки цифрових зображень та в завданні підвищення роздільної здатності зображень з метою відновлення висикочастотної інформації, що не є можливим у разі застосування існуючих алгоритмів та програмних засобів. Як конкретний приклад застосування був обраний алгоритми розпізнавання осіб, а так само зображення, які містять людські обличчя. Незважаючи на те що запропонований алгоритм універсальний і може застосовуватись для будь-яких зображень, що несуть будь-яку інформацію, саме розпізнавання осіб було обрано в якості прикладного прикладу застосування запропонованих алгоритмів, так як саме розпізнавання осіб найбільш чутливим до нестачі інформації на цифровому зображенні.

Задачі дипломної роботи:

1. Аналіз існуючих моделей нейронних мереж.
2. Розробка моделі згорткової нейронної мережі.
3. Розробка алгоритму навчання нейронної мережі.
4. Втілення моделі в програмному продукті, розробленому за допомогою

найновіших технологій паралельних обчислень.

Об’єкт дослідження. Згорткові нейронні мережі, генеративні змагальні мережі, алгоритми розпізнавання образів.

Предмет дослідження. Механізми процесінгу цифрових зображень із використанням нейронних мереж.

Методи дослідження. Для вирішення поставлених задач та досягнення поставленої мети використовувався комплекс методів дослідження: методи теорії розпізнавання образів, методи дискримінантного аналізу, метод градієнтного спуску, метод зворотного поширення помилки

Наукова новизна, в рамках дослідження, полягає в наступному:

1. Запропановано використовувати генеративно змагальні нейронні мережі з метою полипшення якості розпізнавання образів, на прікладі людських обліч.

2. Впроваджено програмну сістему, що успішно віконує поставлену задачу.

Практичне значення одержаних результатів. Розроблена в дипломній роботі модель нейронної мережі та її програмна реализація дають змогу якісно поліпшити будь яку сістему розпізнавання образів.

Статистика роботи. Дипломна робота складається з вступу, трьохрозділів і висновків, списку використаних джерел, викладених на 78 сторінках машинописного тексту. Матеріали дипломної роботи містять 37 малюнків. Список використаних джерел складається з 34 найменувань.!

1%&' ( )* '+, -./

"#01 %)#0112. 3&#0 . ,0-04. -+"/.-5&112

1.1. Основні поняття

Збільшення роздільної здатності зображення має свої особливості, які обмежують застосування загальних методів інтерполяції, таких як бікубічна інтерполяція, саме по-цьому для вирішення завдання суперрезлдюції необхідна розробка нових і спеціалізувалися алгоритмів. Такими особливостями є:

1. Наявність специфічної апріорної інформації про вміст зображення
2. Важливість суб'єктивної оцінки людиною якості роботи алгоритму, так як не існує об'єктивних і достатніх метрик якості зображення.
3. У більшості випадків сверхразрешеіе полягає в переході з більш грубою сітки пікселів на більш дрібну - цей процес отримав назву ресамплінг (англ. - resample), а відношення кроку великої сітки до кроку дрібної сітки - коефіцієнтом збільшення зображення
4. Для використання в режимі реального часу важливо забезпечити мінімально можливу обчислювальну складність.

Аналіз якості збільшення роздільної здатності відбувається з використанням спеціальних метрик, які враховують як близькість отриманого зображення з еталонним зображенням, так і суб'єктивну оцінку якості і фотореалістичності зображення. Так само в застосуванні до задачі розпізнавання образів критеріем оцінки алгоритму є різниця вихідних значень обраного алгоритму зображення до і після застосування до нього алгоритмів суперрезолюції. Глибокі згорткові нейронні мережі показали свою ефектівность в завданні збільшення роздільної здатності. Моделі глибоких і надглибоких штучних згорткових нейронних мереж стали темою досліджень останніх років і показали відмінні результати, що не були можливі з використанням класичних ме-

тодів інтерполяції. Моделі, засновані на згорткових нейронних мережах, випереджають інші підходи з точки зору показників якості зображення, такі як пікове відношення сигнал / шум і структурна подібність [1].

1.2 Проблема розпізнавання образів та облич

Практичним застосуванням теорії розпізнавання образів є розділ комп'ютерного зору, присвячений розпізнаванню та ідентифікації осіб на цифрових зображеніях або відео. Функція ідентифікації особи по обличчю сьогодні активно використовується у багатьох сферах - соціальний мережі, охоронні системи, мобільні пристрої і так далі.

Розпізнавання облич стало одним з перших практичних застосувань теорії машинного зору, яка стимулювала зростання і розвиток теорії розпізнавання образів в цілому.

Є декілька образів, які можливо виділити, вирішуючи завдання розпізнавання:

- об’єкти, якими можна маніпулювати (ключі, годинник і т.д.);
- об’єкти, якими можна частково маніпулювати (автомобілі, матеріали і т.д.);
- нерухомі об'єкти (дерева, будівлі і т.д.);
- люди;
- міміка облич;
- живі об'єкти (тварини, фігура людини);
- букви, символи, знаки тощо (літери, символи, знаки);
- рукописний текст;
- світло, його кут падіння і нахилу, яскравість і так далі (місяць, сонце).

Інтерес до механізму зору і розпізнавання завжди був істотним, особливо в зв'язку зі збільшенням практичних потреб. Незважаючи на те, що людина досить добре розпізнає і ідентифікує особу людини по зображенню його об-

личчя, процес перенесення цієї здатності людини в програмну реалізацію і наділення комп'ютера тими ж навичками, досить складний і неочевидний.

Завдання виділення особи людини в природною або штучною обстановці і подальшої ідентифікації завжди перебувала в ряду найбільш пріоритетних завдань для дослідників, що працюють в області систем машинного зору і штучного інтелекту. Незважаючи на близькість завдань і методів, використовуваних при розробці альтернативних систем біометричної ідентифікації людини таких, як ідентифікація за відбитками пальців або по зображенню райдужної оболонки, системи ідентифікації по зображенню особи істотно поступаються перерахованим вище системам.

Серйозною проблемою, що стоїть перед системами комп'ютерного зору, є велика мінливість візуальних образів, пов'язана зі змінами освітленості, забарвлення, масштабів, ракурсів спостереження. Однак найбільш складним завданням комп'ютерного зору є проблема усунення неоднозначності, що виникає при проектуванні тривимірних об'єктів реального світу на плоскі зображення. Колір і яскравість окремих пікселів на зображенні також залежить від великої кількості важко прогнозованих факторів. У число цих факторів входять:

- число і розташування джерел світла;
- колір і інтенсивність випромінювання;
- тіні або віддзеркалення від навколишніх об’єктів.

У найзагальнішому випадку алгоритм вирішення задачі виявлення та ідентифікації людини по зображенню його особи складається з наступних очевидних кроків:

- виявлення факту присутності людини на аналізованої сцені;
- виділення фігури людини;
- виділення голови;
- визначення ракурсу спостереження голови (анфас, профіль);
- виділення особи;
- порівняння з еталонами і ідентифікація.

Залежно від конкретних умов структура і реалізація окремих кроків алгоритму можуть відрізнятися. У найбільш складному випадку, при використанні системи виявлення та ідентифікації людини по зображенню його обличчя в змінюваних умовах, з великим потоком вхідних даних (робота на міських вулицях з інтенсивним рухом, в метро, аеропортах і т. Д.), Потрібне використання максимальної кількості доступною інформації для досягнення задовільних результатів роботи алгоритму. Потрібні камери з високою роздільною здатністю і хорошою оптикою для забезпечення можливо більшої дальності достовірної ідентифікації.

Вибір алгоритму, який використовується для ідентифікації людини по зображенню його особи, залежить від конкретних умов його застосування. Наприклад, із завданням розпізнавання в строго обмеженому колективі легко справляється багатошарова нейронна мережа. У той же час завдання виявлення конкретної людини в натовпі (з невизначеним складом) вимагає застосування витончених методів для зниження рівня помилкових тривог. Швидше за все, в цьому випадку буде потрібно багаторівнева система, що містить безліч аналізаторів, що працюють в різних просторах, з прийняттям рішення методом голосування. Однією зі складових такої складної системи міг би стати запропонований алгоритм, який би після виділення всіх осіб на фото збільшував би роздільну здатність кожного зображення з обраним особою, для того щоб підвищити точність ідентифікації людини. Це могло б істотно поліпшити якість роботи всієї системи в цілому, оскільки швидше за все зображення, що містять знайдені особи, які не будуть мати високу роздільну здатність.

1.3 Проблема нестачі інформації на зображенні низької якості

При вирішенні задачі розпізнавання образів критичну роль відіграє якість і роздільна здатність вхідного зображення. Чим вище якість і роздільна здатність, тим вище ймовірність того що алгоритм розпізнавання, в даному

випадку розпізнавання та ідентифікації особи, спрацює на належному рівні і з мінімальною ймовірністю помилки. Чим нижче якість і роздільна здатність, тим вище ймовірність помилки і некоректної ідентифікації. На жаль, безліч пристроїв, які використовуються як джерела зображень, не володіють достатньою роздільною здатністю (камери відеоспостереження, камери для прихованого носіння, вбудовані пристрої і так далі). Так само, при аналізі осіб на сцені, наприклад, загального плану або натовпу так само виникає така проблема. Навіть якщо вдається знайти особи на зображенні, при їх аналізі помилка теж може бути неприпустимо низькою. Подібна проблема з тим чи іншим успіхом раніше вирішувалася за допомогою «склеювання» зображення з високою роздільною здатністю з n зображень низького. Однак такий підхід має досить багато зображень - n джерел повинні одночасно захоплювати об'єкт (особа), при цьому бути правильно розташованими в просторі.

Інтерполяція так само не вирішує дану проблему - так як при інтерполяції неможливо отримати ті дані, яких не було в оригінальному документі.

Судячи з наукових робіт, опублікованих в останні роки, генеративні нейронні мережі можуть досягати набагато більших успіхів у вирішенні цієї проблеми [2][13]. Суть в тому, що вони не просто збільшують деталі на фото, вони відтворюють їх фактично заново так, якими вони могли б бути на цьому зображенні в високій якості «на думку» нейронної мережі. Таким чином, домогтися 100% в точності відновленої інформації представляється мало можливим, однак навіть та точність яку може забезпечити подібний алгоритм недосяжна для всіх не інтелектуальних алгоритмів. Точність повинна бути досягнута в процесі навчання мережі на підготовленому зборі даних, що містить в собі зображення у високому і низькому дозволі, таки чином щоб мережа робить знімки максимально близькими до вихідних зображень з високою роздільною здатністю.

1.4 Висновки за розділом. Мета і завдання дослідження

1. Беручи до уваги кількість систем, що використовують інтелектуальні алгоритми розпізнавання образів на зображеннях, з'являється необхідність отримувати якомога більше інформації з зображення перед застосуванням алгоритмів розпізнавання, для отримання якомога кращого результату. Класичні алгоритми інтерполяції практично не здатні вирішити це завдання.

2. З огляду на кількість інформації у вигляді зображень в сучасному світі, алгоритм який міг би справлятися із завданням збільшення роздільної здатності зображення без втрати інформаації міг би не тільки в кращу сторону вплинути на точність систем розпізнавання осіб в тому числі і образів в цілому, але і знайти застосування в різноманітних сферах.

Викладене визначило мету дослідження – розробка інтелектуального алгоритму, здатного вирішувати проблему збільшення роздільної здатності цифрових зображень без втрати інформації.

Для досягнення мети необхідно вирішити основні завдання, що включають:

- аналіз існуючих моделей нейронних мереж
- розробка моделі згорткової нейронної мережі
- розробка алгоритму навчання нейронної мережі.
- втілення моделі в програмному продукті, розробленому за допомогою найновіших технологій паралельних обчислень.

Схема вирішення згаданих завдань: будується модель генеративної змагальної нейронної мережі, проводиться опис моделі на мові програмування Python за допомогою фреймворку Tensorflow та навчання моделі.

2 - '$6)* '+,-./

#&+'&#)41. !.-+3+"#.

2.1. Згорткові нейронні мережі!

Так як поставлена задача являє собою підзадачу комп'ютерного зору, розглянемо основні типи нейронних мереж, які знаходять своє застосування в комп'ютерному зорі.

Згорткові нейронні мережі (англ.- CNN, Convolutional neural networks) стають найбільш часто використовуваними в задачах комп'ютерного зору. Зокрема, згорткові мережі широко використовуються для високорівневих задач бачення, таких як класифікація зображень (наприклад, AlexNet, VGG, Inception, ResNet і так далі). Так само, після того як в ході декількох років експериментів було точно встановлено, що згорткові нейронні мережі успішно вирішують завдання класифікації зображень, через зростання інтересу до штучного інтелекту в цілому і глибокому навчанню зокрема, почалися пошуки інших застосувань згорткових мереж - зокрема генерації нових зображень, перетворення одного зображення в інше, тексту в зображення, зображення в текст і так далі, і з більшістю з цих задач CNN справляються більш ніж успішно, за рахунок механізму їх роботи, який був натхненний механізмом зору ссавців.

На найперших нейронах, пов'язаних з сітківкою ока, розпізнаються тільки прості лінії і градієнти яскравості. Це результат роботи першого шару згорткової нейронної мережі [3].

Умовно, ми можемо говорити про те що наш мозок являє собою десятишарову згорткову нейронну мережу, при цьому проходження кожного шару від сітківки ока до останнього шару займає 10-20 мс, і через 100 мс, коли зображення з сітківки досягає останнього шару, наростаюча складність розпізнаних образів досягає такого рівня, що ми визначаємо, що перед нами знаходиться людина, або автомобіль, а не собака або мотоцикл, наприклад.

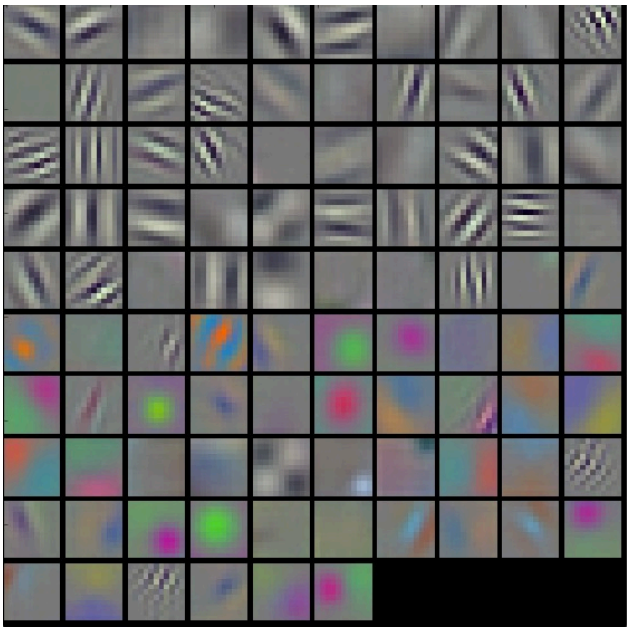

Рис 2.1 - Приклад роботи першого шару згорткової нейронної мережі

Приклад фільтрів першого шару сзгорткової нейронної мережі зображено на рісунку 2.1.

Подивимось на експеримент Девіда Хьюбела і Торстена Визеля, нобелівських лауреатів 1981 року (рисунок 2.2). Вони отримали премію за робо-

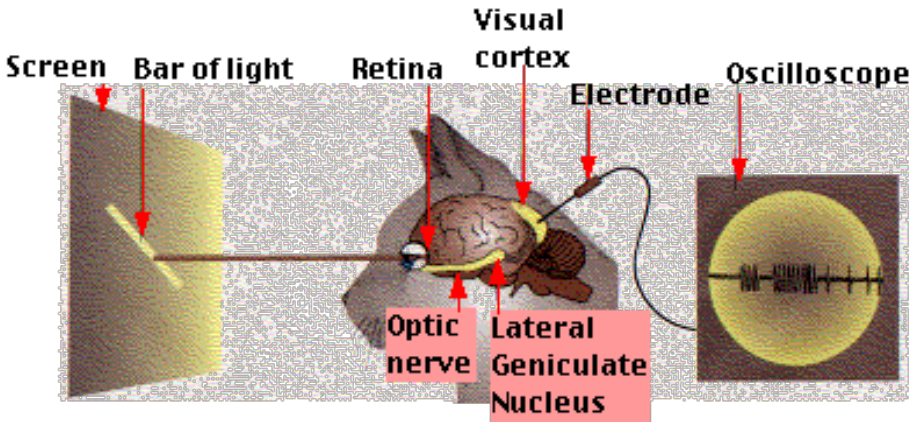

Рис 2.2 - Експеримент Девіда Хьюбела і Торстена Визеля

ту, проведену в 1959 році [4], приблизно в той же час Розенблатт працював над моделлю перцептрону, який ліг в основу класичних повнозв'язних нейронних мереж. Премія отримана за «роботи, що стосуються принципів переробки інформації в нейронних структурах і механізмів діяльності головного

мозку». У роботі описаний експеримент над кішками. Суть експерименту зображена на рісунку 2.2:

Коту на темному екрані під різними кутами демонструється яскравий витягнутий рухомий прямокутник; електрод осцилографа приєднаний до потиличної частини головного мозку, де у ссавців знаходиться центр обробки візуальної інформації. В процесі експерименту вчені спостерігали такі ефекти:

1. Деякі частини зорової кори виявляють активність тоді, коли лінія спроектована на певну частину сітківки;

2. Активність нейронів області змінюється при зміні кута нахилу прямокутника;

3. Деякі області активуються тільки тоді, коли об'єкт рухається в певному напрямку.

Одним з результатів дослідження стала топографічна карта, що представляє собою деяку модель зорової системи ссавців, яку зображено на рісунку 2.3, з наступними властивостями:

1. Сусідні нейрони обробляють сигнали з сусідніх областей сітківки;

2. Нейрони утворюють ієрархічну структуру (рисунок 2.3), де кожен наступний рівень виділяє все більше і більше високорівневі ознаки;

3. Нейрони організовані в так звані колонки - обчислювальні блоки, які трансформують і передають інформацію від рівня до рівня."

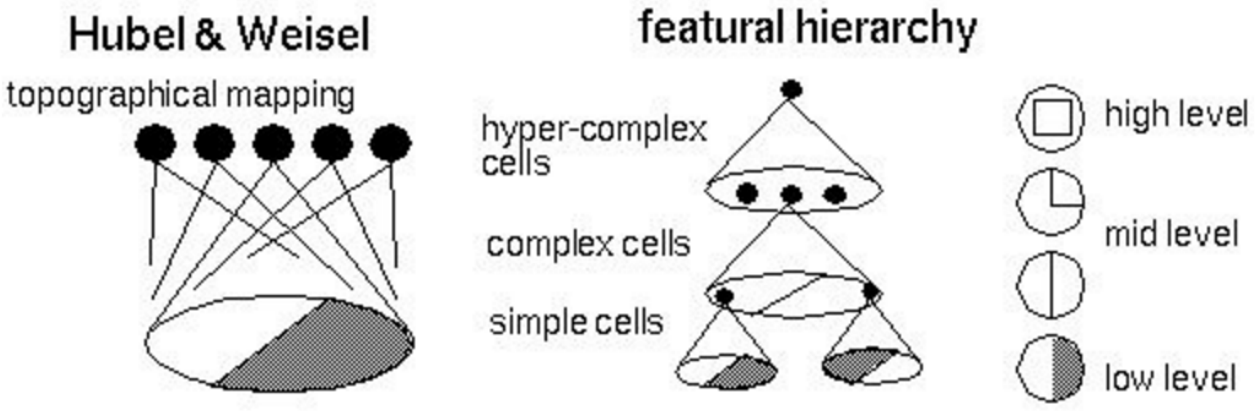

Рис. 2.3 - Модель зорової системи

Згорткова нейронна мережа в процесі навчання, подібно механізму зору людини, вчиться виділяти все більш і більш складні ознаки, які характеризують об'єкт на вході. Імовірно. так само як і людське око, перші шари згорткових мереж вчиться виділяти найбільш низькорівневі ознаки зображення - Вертикальні, горизонтальні лінії. На наступному шарі мережу намагається витягти складніші ознаки - кути, криві лінії і так далі. Таким чином, чим глибше шар нейронної мережі, тим більше абстрактні і складні ознаки він здатний виділити з вхідного зображення.

Куніхіко Фукусіма в період з 1975 по 1980 рік намагався реалізувати ідеї роботи Хьюбела і Визеля в програмному коді, він запропонував дві моделі - когнітрон і неокогнітрон [5]. Вони повторювали модель біологічного зору, і складалися з простих клітин і комплексних клітин. Сьогодні прості клітини в нейронних мережах і глибокому навчанні називаються згортками, а комплексні клітини - субдискертизацією: це як і раніше основні будівельні блоки глибоких згорткових мереж. Модель навчали не алгоритмом зворотнього поширення помилки, а оригінальним евристичним алгоритмом в режимі без вчителя. Можна вважати, що саме ця робота стала початком нейромережевого комп'ютерного зору.

Через багато років, в 1998 році, Ян ЛеКунн, публікує роботу (в співавторстві з іншими дослідниками в галузі штучного інтелекту), в якій використовує ідеї згорток і субдискретизації зі зворотним поширенням помилки, в результаті отримуючи першу працюючу згорткову нейронну мережу [6]. Вона була впроваджена на пошті США з метою розпізнавання індексів. Ця архітектура використовувалася в побудові згорткових мереж аж до недавнього часу: згортка чергується з субдискетизацією кілька разів, потім кілька повнозв'язних шарів. Мережа містила 60 тисяч параметрів. Основні будівельні блоки - згортки з фільтрами 5×5 та зі зсувом 1 і субдискретизація (макс-пулінг) $2 \times$

2 із зсувом 2. Як було сказано вище, згортки грають роль виділення ознак об'єкта, а макс-пулінг використовують для зменшення розмірності, використовуючи той факт, що зображення мають властивість локальної коррелированности пікселів - сусідні пікселі, як правило, не сильно відрізняються один від одного. Таким чином, якщо з декількох сусідніх отримати будь-якої агрегат, то втрати інформації будуть незначними. LeNet-5 зображена на рисунку 2.4. Наступна мережа AlexNet (2012 рік) зображена на рісунку 2.5.

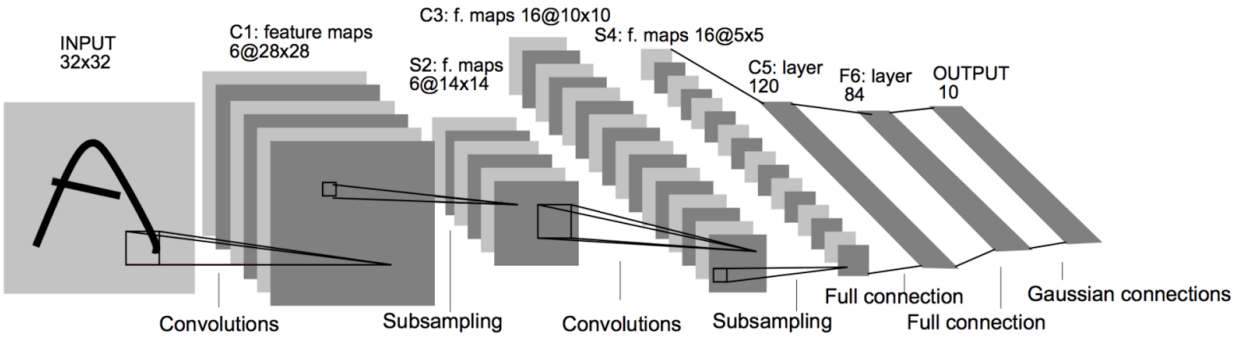

Рис 2.4 - Згорткова мережа LeNet-5

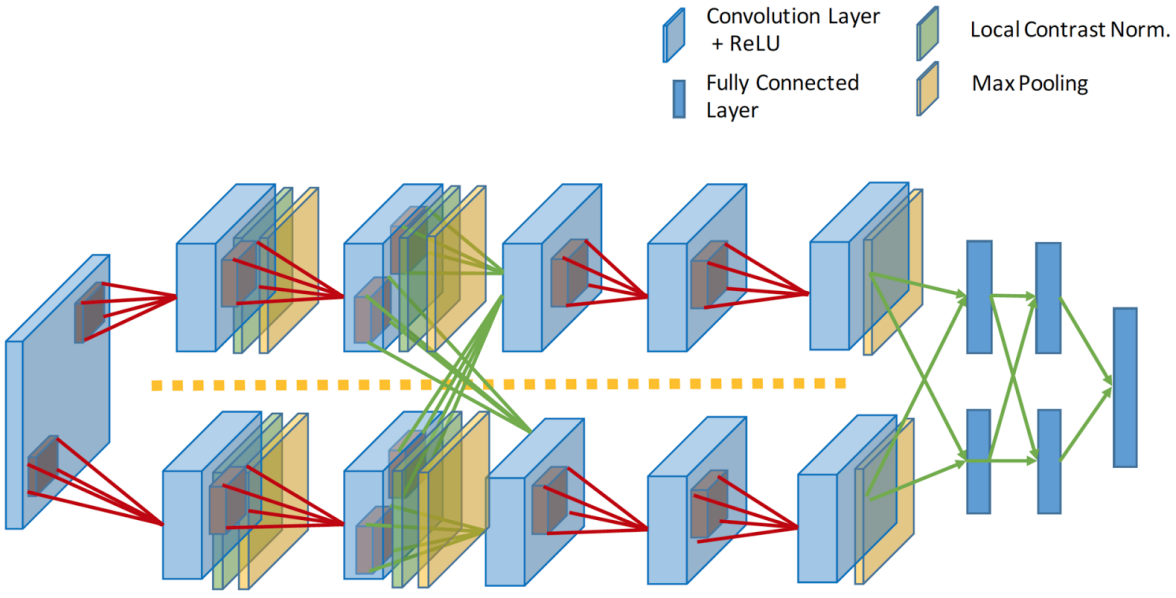

Рис 2.5 - Згорткова мережа AlexNet

Через 14 років, Алекс Крижевський з тієї ж лабораторії, де працював вищезгаданий ЛеКун, поліпшив результати розпізнавання і якість роботи згорткових нейронних мереж [7]. Він використовував більше даних для навчання, побудована мережа мала в тисячу разів більше параметрів, і для навчання використовувалися GPU. Саме GPU дозволило значно збільшити кількість параметрів, які знаходяться в процесі навчання мережі.

З точки зору топології мережі це майже той же LeNet, просто збільшений в тисячу разів. Додали ще кілька згорткових шарів, а розмір фільтрів згортки зменшується від входу мережі до виходу.

Згорткові нейронні мережі створюються як набір різних шарів, що чергуються один з одним, які трансформують вхідні дані у вихідні (зображення в значення класу, до якого воно належить, зображення в зображення і так далі). Зазвичай використовуються такі шари:

1. Шар згортки - є основним "будівельним блоком" згорткової мережі. Шар складається з набору так званих фільтрів, які представляють собою матрицю n x n. Фільтри мають невелике рецептивне поле, проте проходять по всьому об’єму вхідних даних. Під час прямого проходу кожен фільтр шару робить прохід по ширині і висоті вхідний матриці, обчислюючи скалярний добуток параметрів фільтра і даних входу і формує двомірну карту ознак. В ході навчання параметрів фільтрів вони підлаштовуються таким чином, щоб вони активувалися, коли на вхід надходить певна ознака в певному просторовому положенні на вході.

2. Шар субдискретізації (англ. - pooling) - являє собою нелінійний спосіб зниження розмірності. Існує кілька типів пулінгу, найбільш часто використовуваний - max-pooling, і avg-pooling. В обох з них вхідне зображення розбивається на прямокутники, що не перекривають один одного, і далі з кожного вибирається максимальне значення в разі макс-пулінгу, або обчислюється середнє значення в разі середнього пулінгу.

3. Шар активації - приймаючи на вхід матрицю (тензор), застосовує до

кожного значення деяку нелінійну функцію активації. Можуть використовуватися різні функції активації - сигмоїда, гіперболічний тангенс, випрямляч (англ. - ReLU, Rectifier Linear Unit), випрямляч з витоком (англ. - LeakyReLU) і так далі. Функція активації підсилює нелінійність функції нейронної мережі.

Навчання нейронної мережі полягає в підборі параметрів (ваг) усіх шарів згортки таким чином, щоб функція втрат (англ. - loss function), вона ж функція оцінки, ставала мінімальною. Функція оцінки же в свою чергу вибирається виходячи з завдання яку вирішує мережа

Зазвичай, при вирішенні задачі класифікації, після деякої кількості шарів, в яких чергуються згортки, активації і субдискетизації, застосовується декілька повнозв'язних шарів - таких, в яких кожен нейрон з'єднаний з усіма нейронами попереднього шару. При цьому, в завданні класифікації, кількість нейронів останнього шару дорівнює кількості класів, до яких може належати вхідне зображення. Таким чином нейронна мережа кодує вихідний вектор так, щоб вихід нейронів, до яких не відноситься вхідне зображення був мінімальний (наближався до нуля), а того класу до якого належить - був максимальний (наближався до одиниці).

Якщо ж вирішується не завдання класифікації, а трансляції зображення в зображення, то повнозв'язні шари зазвичай відсутні - замість них останнім шаром мережі є шар згортки з кількістю фільтрів, що дорівнює кількості колірних каналів, які ми бажаємо отримати у зображення. Наприклад, якщо на виході ми очікуємо чорно-біле зображення, що останній шар згортки матиме один фільтр, якщо очікуємо RGB, наприклад якщо стоїть завдання розмальовки чорно-білого зображення, то фільтрів буде 3.

2.2. ResNet - мережі

Зі збільшенням глибини мережі точність набуває насичення, а потім швидко погіршується. Несподівано, така деградація не виникає через пере-

навчання мережі, а додавання більшої кількості шарів до відповідної глибини моделі призводить до вищої помилки навчання.

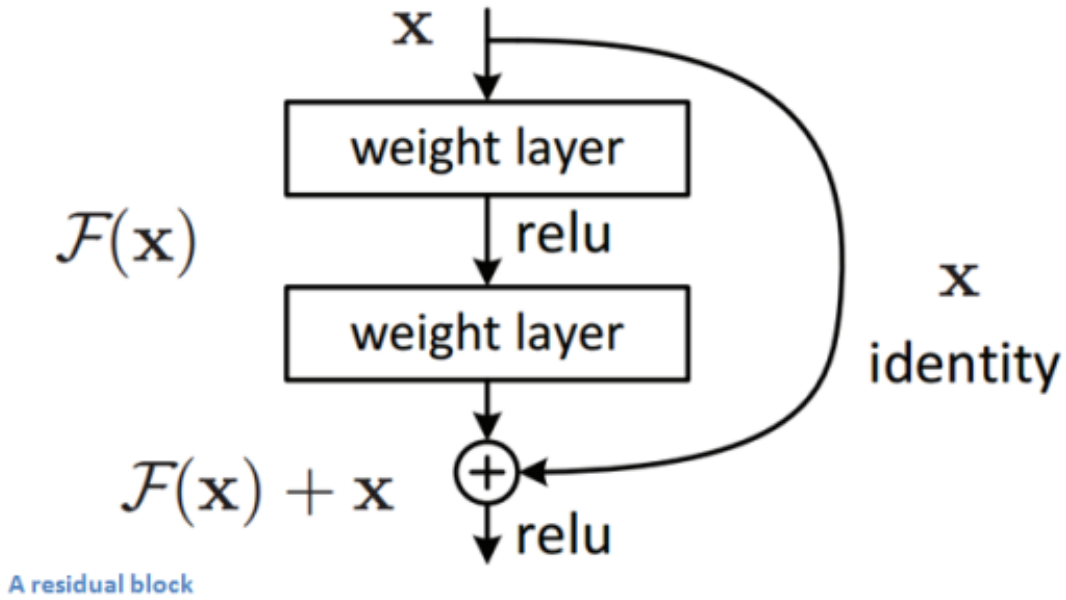

Рис 2.6 - Приклад шару мережі ResNet

Одним з варіантів вирішення цієї проблеми є використання залишкових шарів нейронної мережі (Residual Network, далі ResNet) [8]. Дане рішення було запропоновано дослідниками з Microsoft, і дозволило вирішити проблему зникання градієнта в дуже глибоких нейронних мережах. Крім того, незважаючи на те, що розроблена така архітектура виключно з метою вирішення проблем насичення градієнтів, ResNet показав свою ефективність у завданнях трансляції зображення в зображення завдяки своїй архітектурі. У ResNet пропонується використовувати замість класичного для згорткових нейронних мереж підходу "згортка > субдискретизація > функція активації" використовувати такий: вхід проходить через згортку, потім функцію активації, потім знову згортку та знову функція активації, після чого конкатенується з початковим вхідним значенням (показано на малюнку), та після цього опціонально шар субдискретизації (рисунок 2.6). Значною мірою, саме завдяки правильному використанню такого підходу можливо домогтися приго-

ломшливий результатів в завданні суперрезолюції.

2.3. Генеративні змагальні мережі

Генеративні змагальні мережі - найбільш багатообіцяюча розробка останнього часу в сфері глибокого машинного навчання. GAN, представлений Яном Гудфелоу (Ian Goodfellow) [9] в 2014, пропонує вирішення проблеми "навчання без вчителя" глибоких нейронних мереж за допомогою тренування двох глибоких нейронних мереж, які називаються Генератором і Дискримінатором відповідно, які змагаються і кооперуються один з одним. В процесі тренування обидві нейронні мережі в результаті вчаться виконувати завдання. Головна ідея полягає в тому, що обидві частини мережі конкурують один з одним. Коли дискримінатор кращає, генератор теж повинен стати краще, інакше він більше не зможе перемогти дискриминатора. Аналогічно, коли генератор стає краще, дискримінатор повинен стати краще, інакше він втратить здатність відрізняти підробку від реального контенту.

В останні роки контрольоване навчання (так зване навчання з вчителем) зі згортковими мережами стало широко розповсюджене в додатках для комп'ютерного зору. Порівняно, процес неконтрольованого навчання (так званий процес навчання без вчителя) зі згортковими мережами отримав менше уваги. Для подолання цього розриву, був введений клас згорткових мереж, званий глибокими згортковими генеративними змагальними мережами (Deep Convolution Generative Adversarial Network, далі DCGAN [10]), які демонструють, що вони є сильним кандидатом на неконтрольоване навчання та вирішення завдання суперрезолюції у певних доменах. Навчаючи такі мережі різними наборами даних зображень, були показані переконливі докази того, що така глибока згорткова змагальна пара вивчає ієрархію уявлень від частин об'єкта до сцен як в генераторі, так і в дискримінаторі. DCGAN пропонує один із способів побудови зображень шляхом навчання генеративних змагальних мереж (англ. - GAN, Generative Adversarial Network). Можна додатко-

во стверджувати, що їх навчальний процес і відсутність евристичної функції вартості (такий як піксельна незалежна середньоквадратична похибка) робить їх привабливими для дослідження. Змагальні мережі, як відомо, нестабільні для навчання, часто призводять до генераторам, які виробляють безглузді результати.

Генеративні моделі добре вивчені і діляться на дві категорії: параметричні і непараметричні. Непараметричні моделі часто зіставляються з базою даних існуючих зображень, часто порівнюючи патчі зображень і були використані в синтезі текстури, суперрезолюції. Параметричні моделі для генерації зображень були широко досліджені (наприклад, на MNIST цифрах або для синтезу текстури). Однак, у завданні генерації природних образів реального світу не мали великого успіху до недавнього часу. Варіаційний підхід (VAE) до генерації зображень мав певний успіх, але зразки часто страждають від розмитості. Інший підхід генерує зображення з використанням ітеративного процесу прямої дифузії. Підхід з використанням рекурентної мережі і мережевий підхід розгортки також нещодавно показав певні успіхи у створенні природних образів. Однак вони не використовували генеративні моделі для контрольованих завдань. Генеративні моделі є класом моделей навчання без вчителя("Unsupervised Machine Learning Models"), які використовуються для створення деяких даних.

Вони використовують спільний розподіл ймовірності над спостереженнями. Хоча генеративні моделі мають короткострокові програми, але в підсумку вони фактично володіють потенціалом і здатністю автоматично вивчати функції з набору даних, категорій або значень або чого-небудь ще і генерувати дані.

Генеративні змагальні мережі практично завжди пояснюються на прикладі фальшивомонетника (генератор) і поліцейського (дискримінатор). Спочатку фальшивомонетник показує поліцейському фальшиві гроші, поліцейський говорить що вони фальшиві. Поліцейський пояснює чому вони фальшиві. Фальшивомонетник намагається зробити нові гроші грунтуючись на зворот-

ному зв'язку від поліцейського таким чином, щоб їх було складніше відрізнити від справжніх. Цикл продовжується до тих пір поки поліцейський не стане нездатний відрізнити фальшиві гроші від справжніх.

Так як ідея генеративних змагальних мереж досить проста в теорії, досить складно побудувати працюючу модель.

В GAN, дві глибоких нейронних мережі пов'язаних разом роблять процес зворотного поширення помилки ще більш складним. Глибокі згорткові генеративні змагальні мережі (Deep Convolutional GAN, DCGAN) - одна з моделей, яка демонструє створення практичної реалізації GAN, здатної тренуватися самостійно для того щоб генерувати нові зображення.

Дискрімінатор говорить наскільки реалістично (тут і далі під реалістичним розуміється фотореалістичне зображення, позбавлене артефактів, які можуть сказати про те що зображення створене штучно). Дискримінатор зазвичай являє собою згорткову нейронну мережу або згортковий автоенкодер. Сигомоідальна функція активації на виході позначає можливість того що зображення реальне (вихід 0.0 означає що зображення абсолютно підроблене, 1.0 - що зображення абсолютно реалістичне). Активація кожного шару - Leaky-ReLU.

Генератор синтезує штучні зображення з випадкового розподілу (в загальному випадку - 100-мірний вектор з випадковими числами від -1.0 до 1.0), або з вхідного зображення, за допомогою операції зворотної згортки (розгортки). У класичному DCGAN пропонувалося використовувати збільшення розмірності за допомогою Fractionally-strided згортки. Функція активації leakyReLU. На останньому шарі - сигмоїда, яка видає зображення.

Оскільки ми можемо бачити процес навчання Генератора - спочатку Генератор генерує гучні зображення з низькою роздільною якістю, але через зворотне поширення помилки - з часом вони дійсно навчаються генерувати кращі і кращі зображення, які виглядають абсолютно реалістичними.

Ці нейронні мережі дійсно вивчають, як виглядає візуальний світ. Зазвичай ці моделі мають близько мільйона параметрів. Це допомагає виявити

найбільш характерні риси даних: наприклад, ймовірно, вони дізнаються, що пікселі поблизу, ймовірно, мають один і той же колір, або що світ складається з горизонтальних або вертикальних країв або крапель різних квітів. Зрештою, з часом модель може виявити безліч більш складних закономірностей і функцій з даних: існують певні типи фону, об'єкти, текстури, які вони зустрічаються в певних ймовірних механізмах або що вони з часом змінюються в відео, і т.п.

2.4 Функція втрат

Багато класичних проблеми можуть бути описані як завдання перетворення зображень, де система отримує деяке вхідне зображення і перетворює його в вихідний образ. Приклади обробки зображень включають шумозаглушення, суперрезолюцію і розмальовку, де вхід являє собою погіршене зображення (гучне, низький дозвіл або відтінки сірого), а вихід - високоякісне кольорове зображення. Приклади комп'ютерного бачення включають семантичну сегментацію і оцінку глибини, де вхід являє собою кольорове зображення, а вихідне зображення кодує семантичну або геометричну інформацію про сцену.

Один з підходів до вирішення завдань перетворення зображень полягає в тому, щоб контролювати надійну згорткову нейронну мережу використовуючи функцію втрат на рівні пікселів для вимірювання різниці між вихідними і еталонним зображенням. Такий підхід ефективен при тестуванні, вимагаючи тільки одноразової передачі через навчену мережу. Проте, втрати на рівні пікселів, що використовуються цими методами, не фіксують відмінностей сприйняття між вихідними і справжніми зображеннями. Наприклад, розгляньте два однакових зображення, зміщених одне від одного на один піксель; незважаючи на їх візуальне схожість, вони були б зовсім іншими, якщо їх виміряти на рівні пікселів. Паралельно недавня робота показала, що високоякісні зображення можуть генеруватися з використанням функцій сприйняття

втрат, заснованих не на розбіжностях між пікселями, а замість цього на відмінностях між уявленнями характеристик зображення високого рівня, витягнутими з попередньо навчених згоркових нейронних мереж.

Зображення генеруються шляхом мінімізації функції втрат. Цей підхід створюють високоякісні зображення, але повільні, так як висновок потребує вирішення проблеми оптимізації. У цій роботі тренуємо мережу прямого перетворення для задач перетворення зображень, замість того, щоб використовувати тільки функцію втрат на рівні пікселів, що залежать тільки від інформації про значення конкретних пікселів, ми тренуємо наші мережі з використанням функцій втрати на рівні пікселів і перцептивної втрати, яка залежать від високорівневих карт ознак, витягнутих при проходженні зображення через попередньо навчену згорткову нейронну мережу. Під час навчання такі функції вимірюють схожість зображень більш надійно, ніж втрати на пікселному рівні. Це було зазначено в работах [11][12][13].

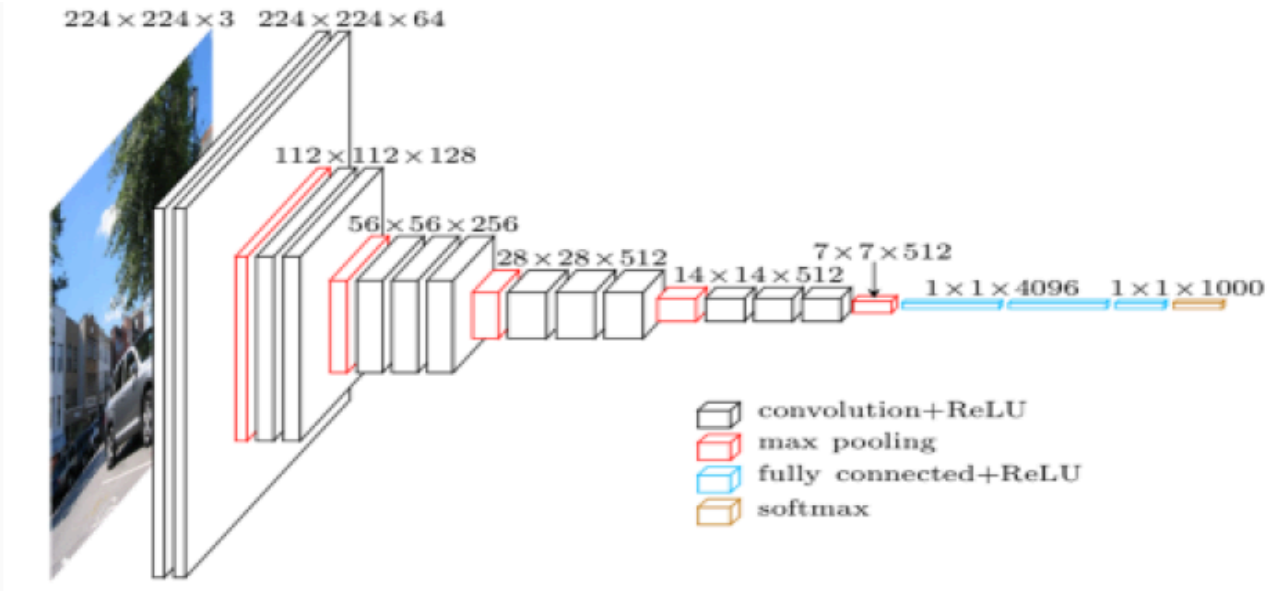

Рис 2.7 - Мережа VGG16

При суперрезолюції з одним зображенням завдання полягає в створенні вихідного зображення з високою роздільною здатністю з входу з низьким дозволом. Це по своїй суті некоректна проблема, оскільки для кожного зоб-

раження з низьким дозволом існують кілька зображень з високою роздільною здатністю, які могли б генерувати його.

Невизначеність стає більш екстремальною в міру зростання фактора суперрезолюції; для великих факторів (× 4, × 8) дрібні деталі зображення з високою роздільною здатністю можуть мати мало або взагалі не мати доказів в його версії з низьким дозволом. Щоб подолати цю проблему, ми тренуємо мережі з високою роздільною здатністю, які не використовують втрати на піксель, але замість цього з перцептуальной функцією втрат, щоб дозволити передачу семантичних знань з попередньо тренованої мережі VGG16 в мережу-генератор.

Таким чином, перша функція втрат генератора являє собою суму втрат на рівні пікселів, яка обчислюється як середньоквадратична різниця між генерованим і еталонним зображенням, і середня абсолютна різниця між виходами певного шару попередньо тренованої мережі VGG16 (рисунок 2.7).

Другим складовим функції помилки генератора є вихід дискримінатора, що оцінює зображення з точки зору реалістичності.

2.5 Оптимізація втрат

Метод найшвидшого градієнтного спуску є одним з найпопулярніших алгоритмів для оптимізації і, безумовно, найбільш поширеним способом оптимізації нейронних мереж.

Метод найшвидшого градієтного спуску - це спосіб мінімізації цільової функції нейронної мережі, параметризрваної вагами моделі, шляхом поновлення параметрів в зворотному напрямку градієнта цільової функції в частних похідних до ваг. Швидкість навчання визначає розмір кроків, які ми робимо для досягнення (локального) мінімуму. Іншими словами, ми слідуємо напрямку нахилу поверхні, створеної цільовою функцією вниз, поки не досягнемо долини.

Існує три варіанти градієнтного спуску, які відрізняються тим, скільки даних ми використовуємо для обчислення градієнта цільової функції. Залежно від обсягу даних ми робимо компроміс між точністю поновлення параметрів і часом, який потрібен для виконання оновлення.

Груповий градієнтний спуск [14].

Груповий градієнтний спуск обчислює градієнт функції втрат в похідних до параметрів для всього набору навчальних даних.

Оскільки нам потрібно обчислити градієнти для всього набору даних, щоб виконати тільки одне оновлення, спуск групового градієнта може бути дуже повільним і важковирішуваними для наборів даних, які не вписуються в пам'ять. Груповий градієнтний спуск також не дозволяє нам оновлювати нашу модель онлайн, тобто новими прикладами. Для заздалегідь визначеного числа епох ми спочатку обчислюємо вектор градієнта функції втрат для всього набору даних в приватних похідних до параметрів. Сучасні бібліотеки глибокого навчання забезпечують автоматичне диференціювання, яке ефективно обчислює градієнт.

Потім ми оновлюємо наші параметри в напрямку градієнтів з частотою навчання, який визначає, наскільки велике виконується оновлення. Гарантується, що спуск градієнта партії сходиться до глобального мінімуму для опуклих поверхонь помилок і до локального мінімуму для неопуклих поверхонь.

Стохастичний градієнтний спуск [15].

Стохастичний градієнтний спуск (SGD) виконує оновлення параметрів мережі для кожного прикладу навчання.

Пакетний градієнтний спуск виконує надлишкові обчислення для великих наборів даних, оскільки він перекомпонований градієнтами для подібних прикладів перед кожним оновленням параметра. SGD усуває цю надмірність, виконуючи одне оновлення за раз. Тому він зазвичай набагато швидше і може також використовуватися для вивчення р режимі реального часу.

SGD виконує часті оновлення з високою дисперсією, які змушують

функцію втрат сильно коливатися.

У той час як спуск групового градієнта сходиться до мінімуму басейну, в який входять параметри, коливання SGD, з одного боку, дозволяють йому перейти до нових і потенційно кращих локальних мінімумів. З іншого боку, це в кінцевому рахунку ускладнює конвергенцію до точного мінімуму, оскільки SGD буде продовжувати перерегулювання. Однак було показано, що коли ми повільно зменшуємо швидкість навчання, SGD демонструє теж саме сходження цільової функції, що і груповий градієнтний спуск, майже напевно сходиться до локального або глобального мінімуму для не опуклої і опуклої оптимізації відповідно.

Міні-пакетний градієнтний спуск [16]

Міні-пакетний градієнтний спуск, нарешті, бере найкраще з обох світів і виконує оновлення для кожної міні-серії прикладів навчання нейронної мережі.

Таким чином, це:

- зменшує дисперсію поновлення параметрів, що може привести до більш стабільної збіжності;

- можуть використовувати високо оптимізовані оптимізаційні матриці, загальні для сучасних бібліотек глибокого навчання, які роблять обчислення градієнта в приватних похідних міні-пакета дуже ефективний. Загальні розміри міні-пакета варіюються від 50 до 256, але можуть відрізнятися для різних додатків. Міні-пакетний градієнтний спуск, як правило, є алгоритмом вибору при навчанні нейронної мережі, і термін SGD зазвичай використовується також при використанні міні-пакетів.

Виходячи з усього вищевикладеного, був обраний саме міні-пакетний градієнтний спуск, як поєднує в собі швидку збіжність цільової функції і помірне споживання обчислювальних ресурсів.

Однак класичний міні-пакетний градієнтний спуск сам по собі не гарантує хорошої збіжності, але пропонує кілька проблем, які необхідно вирішити:

Вибір правильної швидкості навчання може бути утруднений. Швидкість навчання, яка є дуже маленькою, призводить до повільної збіжності, в той час як занадто велика швидкість навчання може перешкоджати збіжності і викликати коливання функції втрат навколо глобального мінімуму або навіть локального мінімуму.

Можна використовувати графік темпів навчання, намагаючись скорегувати швидкість навчання під час навчання, наприклад отжиг. Зниження швидкості навчання відповідно до заздалегідь визначеним графіком або коли зміна мети між епохами падає нижче порога. Однак ці графіки і порогові значення повинні бути визначені заздалегідь і, отже, не здатні адаптовуватись до характеристик набору даних.

Крім того, та ж швидкість навчання застосовується до всіх оновлень параметрів. Якщо наші дані розріджені, а наші функції мають дуже різні частоти, ми, можливо, не захочемо оновлювати їх все в однаковій мірі, але виконуємо більше оновлення для функцій, які рідко зустрічаються.

Ще одна ключова задача мінімізації високонепреривних функцій помилок, спільних для нейронних мереж, полягає в тому, щоб уникнути попадання в їх численні субоптимальних локальні мінімуми. В роботі [19] дослідники стверджують, що труднощі виникають, по суті, не з локальних мінімумів, а з сідлових точок, тобто точок, де одна розмірність нахиляється вгору, а інша схиляється вниз. Ці сідлові точки зазвичай оточені плато з тієї ж помилкою, що змушує SGD зупиняти оновлення параметрів, оскільки градієнт близький до нуля в усіх вимірах.

Таким чином, має сенс вибирати алгоритм оптимізації градієнтного спуску з більш просунутих, ніж класичний SGD. Неповний список існуючих алгоритмів:

1. Стохастичний градієнтний спуск з моментом
2. Прискорений градієнт Нестерова [17]
3. Адаптивний градієнт (Adagrad) [18]
4. RMSProp і Adadelta [19][20]

5. Adam (ADAptive Moment estimation) [21]

6. Adamax [21]

З них було обрано саме Adam (метод оцінки адаптивного моменту). Він поєднує в собі і ідею накопичення руху і ідею слабшого поновлення ваг для типових ознак. Автори Adam стверджують, що алгоритм виступає краще або приблизно так само, як і всі попередні алгоритми на широкому наборі данних за рахунок початкового калібрування. Так само, в процесі вивчення наукових робіт з'ясувалося, що Adam з успіхом застосовується для вирішення даного завдання іншими дослідниками. Немає гарантії, що жоден з перерахованих вище алгоритмів працює стовідсотково гірше, однак, виходячи з усього перерахованого вище, є підстави стверджувати що в більшості ситуацій Adam працює або краще, або так само добре як інші алгоритми.

Learning rate (швидкість навчання) був обраний 10-3 для генератора і 5-4 для дискриминатора з дворазовим зменшенням після 10 ^ 5 епох навчання.

2.6 Висновки за розділом

В даному розділі описані теоретичні відомості про техніки машинного навчання, які застосовуються для вирішення посталенних завданнь, а саме глибокі згорткові нейронні мережі, генеративні змагальні нейронні мережі, описана функція втрат, а також алгоритм її оптимізації.

3 #'&#I* '+ , -I/

%'07#)410 40"#)10

3.1 Вибір апаратних засобів

Фахівці з обробки та аналізу даних як в промисловості, так і в наукових колах використовують GPU в сфері машинного навчання, щоб дістатись значних удосконалень в широкому спектрі додатків, включаючи додатки для класифікації зображень, аналізу відеоданих, обробки мови і обробки текстів на природній мові. Глибоке навчання, тобто використання складних, багаторівневих нейронних мереж для створення систем, які можуть виявляти ознаки з великого обсягу немаркованих даних, - саме та область, в якій ведуться активні дослідження і інвестиційна діяльність. [22]

Хоча машинне навчання існує вже десятки років, дві відносно нові тенденції привели до його широкомасштабного використання: доступність великого обсягу даних, а також продуктивність і ефективність паралельної обробки даних, яка можлива завдяки обчислень на GPU. GPU використовуються для навчання цих глибоких нейронних мереж за допомогою набагато більших навчальних послідовностей в більш стислі терміни, з використанням меншої інфраструктури ЦОД. GPU також використовуються, щоб відтворювати ці навчальні моделі машинного навчання для виконання завдань класифікації та прогнозування на хмарі. При цьому графічні процесори дозволяють працювати з даними більшого обсягу і з більш високою продуктивністю, споживаючи менше енергії і на базі меншою інфраструктури.

До числа тих, хто вперше застосував графічні прискорювачі для вирішення завдань машинного навчання, відносяться багато великих веб компанії і соціальні мережеві сервери, поряд з науково-дослідними інститутами високого рангу в області обробки та аналізу даних і машинного навчання. Завдяки тисячам обчислювальних ядер і збільшення продуктивності додатків в 10-100

разів у порівнянні з CPU, GPU стали процесорами, які вибирають фахівці з обробки даних для роботи з даними великого обсягу.

Глибоке навчання включає в себе величезну кількість матричних і векторних операцій, в основному множення, які можуть бути масово распаралелені, що має на увазі прискорення роботи на графічних процесорах. Це пов'язано з тим, що графічні процесори були розроблені для паралельної обробки цих матричних операцій, оскільки це необхідно для комп'ютерних ігор, наприклад, 3D-ефектів, рендерингу землі і т. д. З іншого боку, одноядерний процесор буде виконувати матричну операцію послідовно, по одному елементу за раз. Один GPU може мати сотні або тисячі ядер, в той час як у процесора зазвичай є не більше кількох ядер (від двох до восьми).

Як GPU для розробки був використаний відеоприскорювач NVIDIA GTX 1060. Характеристики обраного апаратного засобу:

Графічний чіп GeForce GTX 1060
Частота пам'яті 8000 МГц
Обсяг пам'яті 6 ГБ
частота ядра
Базова частота: 1506 МГц
У режимі розгону: 1708 МГц
Система охолодження Радіатор + Вентилятор
Розрядність шини пам'яті 192 біт
Тип пам'яті GDDR5

3.2 Опис програмних засобів

CUDA і CuDNN

CUDA [23] є архітектуру паралельних обчислень, розроблену компанією NVIDIA і реалізовану на її ж апаратних рішеннях, яка дозволяє кардинально збільшити обчислювальну ефективність систем з використанням

графічних процесорів (англ. GPU).

На сегодняшній день продажу апаратних засобів з використанням технології CUDA досягли мільйонів, а розробники прикладного програмного забезпечення, вчені і дослідники в області машинного навчання і глибоких нейронних мереж використовують технології CUDA в різних областях, включаючи обробку зображень і відео, обчислювальну хімію і біологію, моделювання фізичних процесів, обробка результатів медичних досліджень, аналіз природних катаклізмів і так далі і тому подібне.

Завдяки CUDA математичні обчислення переносяться з центрального процесора на графічний процесор. Для реалізації подібної парадигми NVIDIA і розробила CUDA, яка сьогодні представлена в графічних процесорах Geforcec, Ion, Quadro і Tesla, а так само реалізована в програмному забезпеченні яке надається компанією.

Різні галузі наукових досліджень також з великим ентузіазмом зустріли цю технологію, наприклад сьогодні CUDA прискорює програму моделювання молекулярної динаміки AMBER, яка використовується більш ніж шістдесяти тисяч дослідників і вчених, а також фармацевтичними компаніями для успоренія проектування нових лікарських засобів.

Також показником ефективності використання CUDA прискорення є використання графічних процесорів Tesla в GPU обчисленнях. На даний момент компанії з Fortune 500 використовують понад 700 обчислювальних кластерів по всьому світу в енергетичному секторі, банківському секторі і так далі.

Бібліотека NVIDIA CUDA® Deep Neural Network (cuDNN) [24] являє собою бібліотеку примітивів для глибоких нейронних мереж з прискоренням GPU. cuDNN забезпечує налаштовані реалізації для стандартних підпрограм, таких як форвардні та зворотні згортки, рівні об'єднання, нормалізації та активації. cuDNN є частиною SDK NVIDIA Deep Learning.

Дослідники і розробники фреймворків в усьому світі покладаються на cuDNN для високопродуктивного прискорення GPU. Це дозволяє їм зосере-

дитися на навчанні нейронних мереж і розробці програмних додатків, а не на часі на настройку продуктивності графічного процесора низького рівня. cuDNN прискорює широко використовуються фреймворками глибокого навчання, включаючи Caffe2, MATLAB, Microsoft Cognitive Toolkit, TensorFlow, Theano і PyTorch.

На відміну від NVIDIA CUDA, яка забезпечує можливість виконання на GPU будь-яких обчислень, бібліотека cuDNN спеціально розроблена для навчання глибоких нейронних мереж. Вона містить оптимізовані для GPU реалізації згорткових і рекурентних мереж, різних функцій активації (напівлінійних, сигмоїдальна, гіперболічний тангенс, softmax), алгоритму зворотного поширення помилки і т.п. cuDNN дозволяє навчати нейронні мережі на GPU в кілька разів швидше, ніж просто CUDA.

Вибір мови програмування Python

Розглянемо загальну популярність мов, які використовуються для задач машинного навчання. Python очолює рейтинг, 57% вчених-розробників і дослідників машинного навчання використовують його, а 33% виділяють його для розробки. Не дивно, з огляду на всю еволюцію в глибоких навчальних системах Python за останні 2 роки, включаючи випуск TensorFlow і широкий вибір інших бібліотек. Python часто порівнюють з R, але вони ніде майже не можуть порівнюватись за популярністю: R займає четверте місце в загальному використанні (31%) і п'яте за пріоритетністю (5%). R насправді є мовою з найнижчим співвідношенням пріоритетів і використання серед п'яти, причому тільки 17% розробників, які використовують його для визначення пріоритетів. Це означає, що в більшості випадків R є додатковою мовою, а не першим вибором. Те ж саме ставлення для Python становить 58%, що насправді найвищий результат серед п'яти мов, явно вказує на те, що тенденції використання Python є повністю протилежними тим, що відносяться до R. Не тільки Python є найбільш широко використовуваним мовою, а й основним вибором для більшості його користувачів. C / C ++ є наступним за Python, як у використанні (44%), так і при визначенні пріоритетів (19%). Java уважно стежить за

C / C ++, в той час як JavaScript займає п'яте місце в використанні, хоча з трохи кращою оцінкою пріоритетів, ніж R (7%). Julia, Scala, Ruby, Octave, MATLAB і SAS опускаються нижче 5% -ну позначку пріоритетності і нижче 26% використання. Тому ми зосередили нашу увагу на 5 мовах. Дані показують, що найбільш рішучим фактором при виборі мови для машинного навчання є тип проекту, - область застосування. Вчені, що працюють над аналізом настроїв, оцінюють Python (44%) і R (11%) більше, а JavaScript (2%) і Java (на 15%) менше, ніж розробники, що працюють в інших областях. Навпаки, Java приділяє більше уваги тим, хто працює над мережевою безпекою / кібер-атаками і виявленням шахрайства, двома областями, де Python є найменш пріоритетним. Алгоритми виявлення мережі та виявлення шахрайства будуються або використовуються головним чином у великих організаціях, і особливо в фінансових установах, де Java є фаворитом більшості внутрішніх команд розробки. В областях, які менш орієнтовані на підприємства, такі як обробка природної мови (NLP) та аналіз настроїв, розробники вважають за краще Python, який пропонує більш простий і швидкий спосіб створення високопродуктивних алгоритмів через великий набір спеціалізованих бібліотек, які поставляються з ним.

Так як процес розробки не пов'язаний з корпоративним сектором, мережевою безпекою, а також з огляду на відсутність команди і прийнятих в ній стандартів розробки, а також з огляду на те що продуктивність не є абсолютним пріоритетом була вибрана мова розробки Python, як найбільш поширена в області. Python має простий і зрозумілий синтаксис, що дозволяє швидше описувати досить складні концепції, що лежать в основі алгоритмів глибокого навчання. Другим за важливістю чинником, який вплинув на вибір мови, є величезна кількість бібліотек для роботи з даними і зображеннями (scipy, numpy, pandas і так далі), які спрощують роботу. Так само Python використовується як мова написання коду більшістю фреймворків, які використовуються для досліджень в області глибокого навчання.

При старті розробки проекту в сфері машинного навчання (ML) вибір

фреймворку може бути складним. Оскільки ця сфера і навколишні її технології є відносно новими, досить складно явно виділити найкращий інструмент.

TensorFlow [25] був розроблений командою Google Brain для проведення досліджень в області машинного навчання і глибоких нейронних мереж. Багато дослиднікив описують TensorFlow як більш сучасну версію Theano.

TensorFlow досить простий для настройки і пропонує навчальні посібники, спрямовані на новачків, які охоплюють теоретичні основи і практичне застосування нейронних мереж. TensorFlow працює повільніше, ніж Theano і Torch, але в даний час це обговорюється Google і співтовариством з відкритим вихідним кодом. TensorBoard - це модуль візуалізації TensorFlow, який забезпечує інтуїтивне уявлення конвеєра обчислень.

Keras, бібліотека глибокого навчання, недавно була перенесена на TensorFlow, що означає, що будь-яка модель, написана в Keras, тепер може працювати на TensorFlow. Нарешті, варто згадати, що TensorFlow може працювати на самих різних апаратних засобах.

Прискорення GPU: Так

Мови / інтерфейси: Python, Numpy, C ++

Платформа: крос-платформа

Підтримка: Google

Theano

Theano [26] виникла в 2007 році в Університеті Монреаля в широко відомому Інституті вивчення алгоритмів. Theano є потужним, надзвичайно швидким і гнучким, але зазвичай розглядається як низкоуровневая структура (наприклад, повідомлення про помилки є особливо загадковими / марними). Таким чином, у необробленому Theano - це скоріше платформа досліджень і екосистема, ніж бібліотека глибокого навчання. Він часто використовується як базова платформа для бібліотек абстракції вищого рівня, які надають прості API-обгортки в Theano. Деякі з найбільш популярних бібліотек включають Keras, Lasagne і Blocks.

Прискорення GPU: Так

Мови / інтерфейси: Python, Numpy

Платформа: Linux, Mac OS X і Windows

Підтримка: лабораторія MILA в Університеті Монреаля

Microsoft CNTK

Microsoft Cognitive Toolkit [27], також відомий як CNTK, являє собою систему глибокого вивчення Microsoft з відкритим вихідним кодом. CNTK краще відомий в мовному співтоваристві, ніж в загальному співтоваристві глибокого навчання, хоча CNTK також може використовуватися для навчання зображень і текстів. CNTK підтримує широкий спектр алгоритмів, таких як Feed Forward, CNN, RNN, LSTM і послідовність в послідовность. Він працює на багатьох різних апаратних засобах, включаючи кілька графічних процесорів.

Прискорення GPU: Так

Мови / інтерфейси: Python, C ++, C # і CLI

Платформа: Windows, Linux

Підтримка: Microsoft Research

Torch не розглядається через те, що використовує мову Lua, PyTorch ж не має особливих переваг перед вищезгаданими. Caffe використовує C ++ в якості основного мови, з цієї причини теж не розглядається. З тієї ж причини - мовою розробки є не Python - відкинуті Deeplearning4j, MXNet і інші.

З Tensorflow, Theano і CNTK був обраний саме Tensorflow. Сам по собі він пропонує більш високорівневе API, ніж Theano, проте обидва фреймворка маю можливість використовувати API Keras [28]. Маючи схожу продуктивність на момент написання роботи, Tensorflow розвивається найбільш активно, має найбільшу підтримку спільноти і використовується найбільшим числом дослідників.

3.2 Набір даних для навчання

Вибір набору даних для навчання є одним з найбільш важливих етапів побудови системи. Саме в процесі навчання на цих даних будуть підбиратися параметри нейронної мережі, і в кінцевому підсумку саме від них залежіть досягнутий результат. Так як для тренування потрібно величезна кількість даних, було прийнято рішення взяти готовий і публічно доступний для дослідників набір даних «CelebA» [29] - набір з більш ніж двохсот тисячами зображень з обличчами знаменитостей (приклад зображень наведено на рисунку 3.1). При цьому перед початком навчання проводиться додаткова обробка - з кожного зображення вирізається 10 випадкових ділянок розміру 128x128 пікселів. Це допомагає розширити вибірку, і таким чином ми отримуємо більше двох мільйонів зображень. Тепер цей набір даних розбивається на тестовий і валідаційні - на першому відбуватиметься навчання мережі, другий потрібен для того щоб оцінювати здатність мережі генералізувати - знаходити рішення для тих зразків, яких не було в навчальній вибірці. На жаль, через обчислювальної складності та обмежених обчислювальних ресурсів навчання на весь обсяг даних зайняло б занадто багато часу, тому було прийнято рішення використовувати тільки його частину.

Рис 3.1 - Приклад зображень із набору даних для навчання мережі

При цьому для кожного зображення формується 2 версії - виского дозволу (128 x 128) і низького (32 x 32), другі будуть подаватися на вхід мережі, а перші будуть використовуватися в обчисленні функції втрат як еталон. Та-

ким чином ми отримуємо збільшення роздільної здатності в 4 рази.

Також, крім CelebA, для тренування і тестування моделі було прийнято рішення використовувати набори даних більш загального плану, що містять крім облич людей фотографії природи, архітектури і так далі. Були обрані набори даних: Set14 [30], BSD300 [31] і DIV2K [32].

Так само завжди необхідно перевіряти стабільність моделі машинного навчання. Під перевіркою стабільності мається на увазі що неможливо просто підганяти модель до даних навчання і припускати, що вона буде точно працювати для реальних даних, які вона ніколи не бачила раніше. Необхідна якась впевненість в тому, що ваша модель має більшу частину патернів з правильних даних і не дуже сильно набирає шум, або, іншими словами, її низький рівень перетренированности і відхилення. Валідація це процес визначення того, чи є кількісні результати кількісної оцінкою гіпотетичних відносин між змінними прийнятними як описи даних. Як правило, оцінка помилки для моделі проводиться після навчання. У цьому процесі проводиться чисельна оцінка різниці в передбачених і вихідних відповідях, також звана помилкою навчання. Однак це дає нам уявлення про те, наскільки добре наша модель працює з даними, які використовуються для її навчання. Тепер можливо, що модель недообучена або перетренована на тренувальних даних. Таким чином, проблема з цією методикою оцінки полягає в тому, що вона не дає інформації про те, наскільки добре модель буде узагальнювати і генералізувати, тобто наскільки добре вона працює на тих даних на яких вона ніколи не навчалася і яких ніколи "не бачила". Отримання цієї ідеї про нашу моделі називається крос-валідація. Існує кілька похдодов до реалізації крос-валідації.

Валідація методом утримання

Суттю даного методу є видалення частини даних з набору даних для тренування і використання її для отримання прогнозів від моделі, навченої рештою даних. Потім функція втрат повідомляє, як наша модель працює з раніше використалися даними або набором даних перевірки. Хоча цей метод не потребує будь-яких витрат для обчислень і краще, ніж традиційна перевір-

ка, він як і раніше страждає від проблем з високою дисперсією. Це пов'язано з тим, що не визначено, які точки даних виявляться в наборі перевірки, і результат може бути зовсім іншим для різних наборів.

Валідація K-Fold Cross

Оскільки даних для навчання моделі недостатньо, видалення частини її для перевірки становить проблему недоподготовкі. Зменшуючи дані навчання, ми ризикуємо втратити важливі шаблони / патерни в наборі даних, що, в свою чергу, збільшує похибку, викликану зміщенням. Отже, нам потрібен метод, який надає достатні дані для навчання моделі, а також залишає достатні дані для перевірки. K Fold cross validation робить саме це.

У перехресній перевірці "K Fold" дані діляться на k підмножин. Тепер метод утримання повторюється k разів, так що кожен раз один з k підмножин використовується в якості набору валідаційних даних, а решта k-1 підмножини об'єднуються для формування навчального набору. Оцінка похибки усереднюється за всіма k випробувань, щоб отримати повну ефективність нашої моделі. Як можна бачити, кожна точка даних потрапляє в перевірку, встановлену рівно один раз, і потрапляє в навчальний набір k-1 раз. Це значно зменшує зсув, оскільки ми використовуємо велику частину даних для тренування, а також значно зменшує дисперсію, так як більша частина даних також використовується в наборі валідації. Зміна наборів даних також підвищує ефективність цього методу.

З огляду на те, що обчислювальні ресурси дуже обмежені і навчання моделі на дійсно великій кількості даних зайняло б багато часу, було вирішено взяти вельми обмежений набір даних, і розділити його на 4 частини - таким чином, щоб в 1 момент часу з набору даних використовувалися як тренувальний, і один - як набір даних для валідації, при цьому кожні n епох навчання набори об'єднуються в один, частина даних відкидається і береться нова частина обрана випадковим чином, і знову розбирається на 4 частини. Такий підхід показав значно більшу швидкість наближення моделі до глобального мінімуму, порівняно до класичного підходу з формуванням тестово-

го і набору даних валідації методом утримання, за рахунок ротації даних при попаданні моделі в локальний мінімум зміна даних дозволяла ефективно з нього вийти і продовжувати рух в сторону глобального мінімуму.

3.3 Архитектура та модель мережі

Основними блоками генератора і дискримінатора, які представляють собою згорткові нейронні мережі, є шари згортки і шари активації, шари ResNet і повнозв'язні шари.

Шар згортки містить в собі N фільтрів, кожен з яких представляє собою матрицю ваг розміром m x n. На кожному шарі згортки кожен фільтр послідовно проходить по матриці розміром H x W, що представляє із себе зважену суму вхідного тензора, при цьому відповідна вага множиться на відповідне значення цієї матриці. Таки чином, на виході шар згортки видає тензор, розмірністю H x W x N. Передбачається, що в процесі тренування ваги фільтрів згорткових шарів підбираються таким чином, щоб виділяти з вхідний матриці найбільш важливу інформацію, з точки зору мережі.

Шар активації приймає на вхід тензор значень, і застосовує до кожного з них деяку нелінійну функцію активації, також звану передавальною функцією. Історично, функцією активації в повнозв'язних нейронних мережах (перцептронах), були сигмоїда і гіперболічний тангенс. Однак, в згорткових нейронних мережах вони виявилися мало застосовні, через високу обчислювальну складність, і їм на зміну прийшла функція активації «випрямляч» (англ. Rectifier, за аналогією з випрямлячем в електротехніці), яка приймає деяке значення, і якщо воно більше нуля - повертає без змін, якщо менше нуля - повертає 0. Графік функції ReLU [33] показаний на рісунку 3.2. Застосування даної функції активації не тільки зменшує обчислювальну складність, а й підвищує швидкість збіжності стохастичного градієнтного спуску, вважається, що це пов'язано з лінійним характером і не насиченням даної функції. Однак, подальші дослідження показали що ReLU не завжди

надійно працює, наприклад, великий градієнт, що проходить через ReLU, може привести до такого оновлення ваг, що даний нейрон ніколи більше не активується.

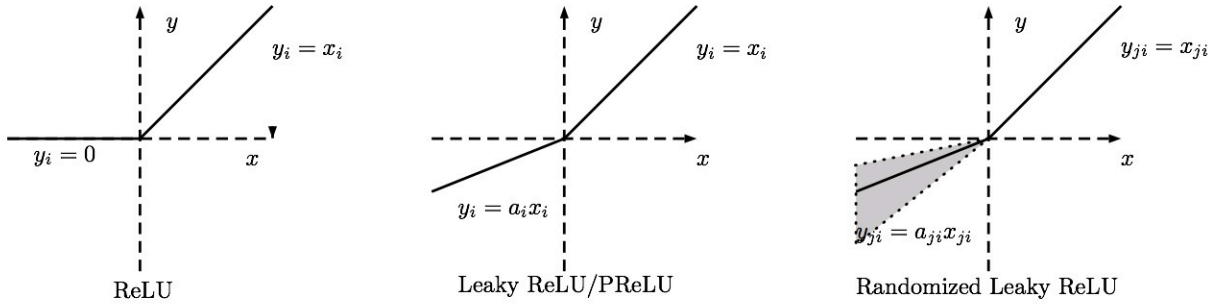

Рис 3.2 - Графіки функцій ReLU

Сьогодні на зміну ReLU прийшло ціле сімейство схожих функцій, в даній роботі була обрана функція активації LeakyReLU [34] для всіх шарів, крім:

- останнього шару ResNet блоку (замість використовується лінійна передавальна функція)

- останнього шару генератора (використовується ReLU, це пов'язано з тим що далі вихід даного шару буде перетворений в колірний простір, де існують тільки значення від 0 до 255)

- повнозв’язних шарів дискримінатора (гіперболічний тангенс і сигмоїда відповідно)"

ReLU з «витоком» (leaky ReLU, LReLU), рисунок 3.2, являє собою одну зі спроб вирішити описану вище проблему виходу з ладу звичайних ReLU. Звичайний ReLU на інтервалі x <0 дає на виході нуль, в той час як LReLU має на цьому інтервалі невелике від'ємне значення (кутовий коефіцієнт близько 0,1). Графік LeakyReLU зображений на малюнку.

Так як ми очікуємо на виході дискримінатора дійсне число в діапазоні від 0 до 1, яке відображає реалістичність зображення, на думку дискримінатора, для останнього шару дискриминатора була обрана сигмоїда як функції

активації. Її графік зображений на рісунку 3.3.

Для надання більшої нелінійності дискриминатора, передостаннього шару дискриминатора, який представляє собою повнозв'язну шар, був обраний гіперболічний тангенс як фунции активації - рісунок 3.4.

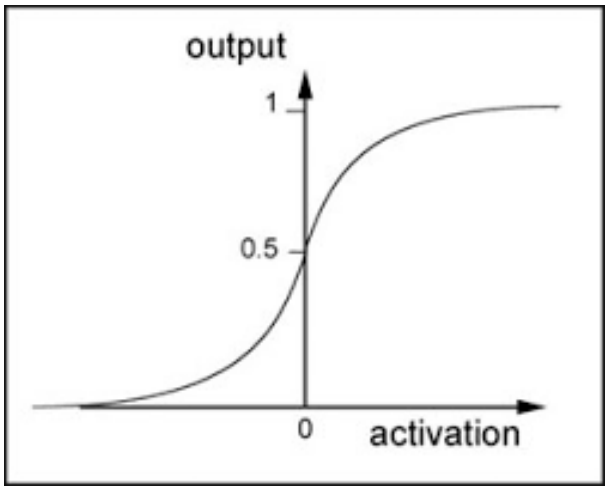

Рис 3.3 - Графік сігмоїдальної функциї актівациї

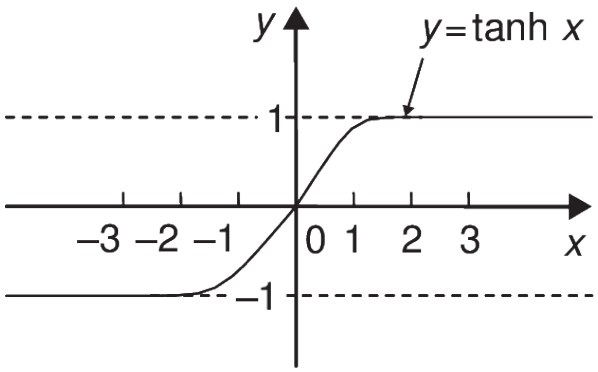

Рис 3.4 - Графік гиперболічного тангенсу

Для підвищення якості роботи мережі, грунтуючись на роботах інших дослідників, було вирішено частину шарів згортки замінити на так звані блоки ResNet, які представляють собою шар згортки, шар активації, знову шар згортки, після чого ми складаємо карти ознак, отримані після верств згортки, з вхідним тензором. По-перше, це дозволяє передавати від більш ранніх

шарів до більш глибоким вхідні дані, по-друге це допомагає вирішити проблему "вимивання градієнта", коли при передачі градієнта помилки від виходу мережі до її входу, через кілька шарів градієнт ставав занадто близьким до нуля і ваги ранніх шарів мережі не оновлювалися. ResNet блоки дозволяють ефективно вирішити цю проблему, і прискорити навчання, хоча таке рішення вимагає набагато більше пам'яті для зберігання даних під час навчання. Схематичне зображення ResNet блоку приведено на рисунку 3.5.

Повнозв'язні шари використовуються тільки в якості останніх шарів дискримінатора, для зниження розмірності і переходу від тривимірного тензора до вектору значень.

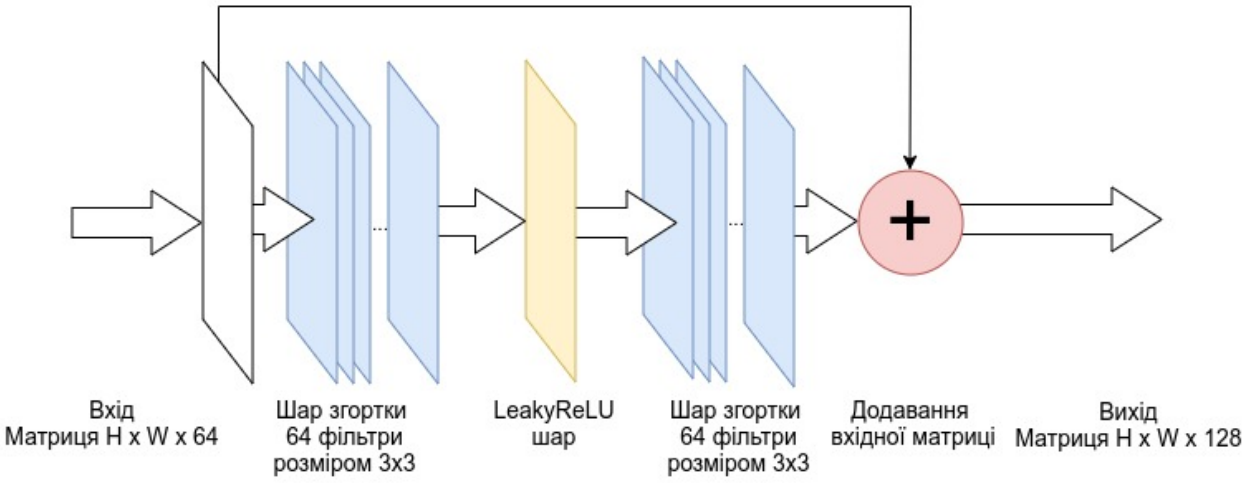

Рис 3.5 - ResNet блок

Архитектура генератора

Генератор є основною нейронною мережею, саме він збільшує розмір вхідного зображення, застосовуючи до нього деяку нелінійну функцію, яка має близько мільйона параметрів, які і підбираються в процесі навчання нейронної мережі.

При дизайні нейронної мережі - генератора, були використані деякі техніки, які були описані дослідниками в області машинного навчання в своїх роботах.

Умовно генератор можна розділити на кілька структурних блоків, кожен з яких повинен вирішувати свою певну задачу.

Ці блоки представляють собою комбінацію шарів згортки, шарів активації і ResNet блоків. Умовно ці блоки представляють собою:

1. Умовно неглибока мережу, яка збільшує вхідне зображення в 4 рази. Містить шар згортки, шар розгортки, шар згортки, шар розгортки і шар згортки. На всіх шарах функція активації ReLU. Аналогічна блоку деконволюции глибокої мережі, однак має всього 32 фільтра в третьому шарі згортки, а також не має LeakyReLU активацій.

2. Екстракція ознак вхідного зображення - один шар згортки, шар активації "випрямляч з витоком", блок ResNet, шар згортки і знову шар активації. Завдання цього блоку - виявити низькорівневі ознаки вхідного зображення. Його зображено на рісунку 3.6.

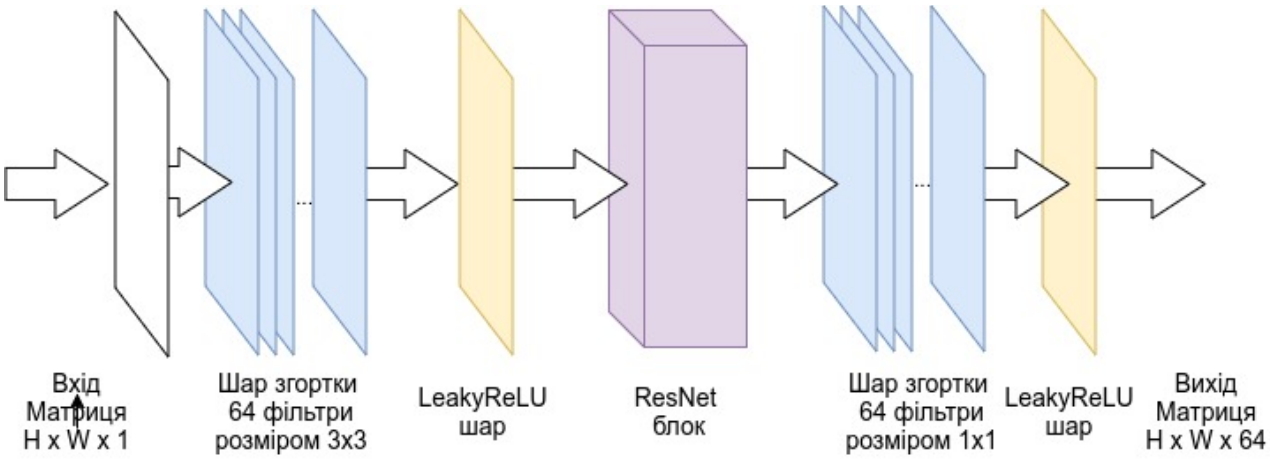

Рис 3.6 - Другий блок мережі-генератора

3. Збільшення розмірності зображення. Тут використовуються спеціальні блоки розгортки (деконволюции), завданням яких є збільшення горизонтальної та вертикальної розмірності вхідних даних. Саме тут відбувається збільшення дозволу картинки в 4 рази, для цього застосовуються два таких шару, і шари функції активації і простий згортки між ними. У шарах деконво-

люции застосовуються фільтри розмірності 14x14. Його зображено на риснку 3.7.

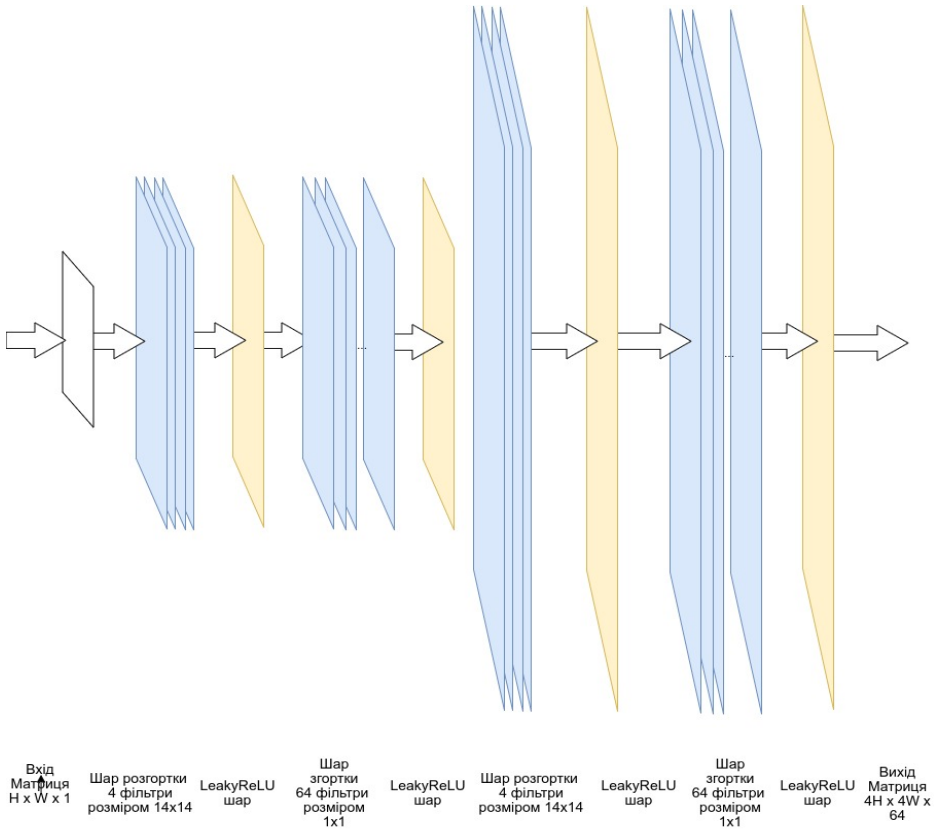

Рис 3.7 - Блок збільшення розмірності зображення

4. Блок первинної генерації ознак збільшених зображень. Передбачається, що саме цьому блоці нейронна мережа спробує виділити на генерованому зображенні ознаки, знайдені в збільшеному зображенні. Даний блок складається з чотирьох паралельних шарів згорток і активацій, які пізніше конкатенуються в один тензор.

Топологію блока зображено на рисунку 3.9.

5. Блок вторинної генерації ознак і формування вихідного зображення. Імовірно, дана частина нейронної мережі повинна змінювати ознаки, отри-

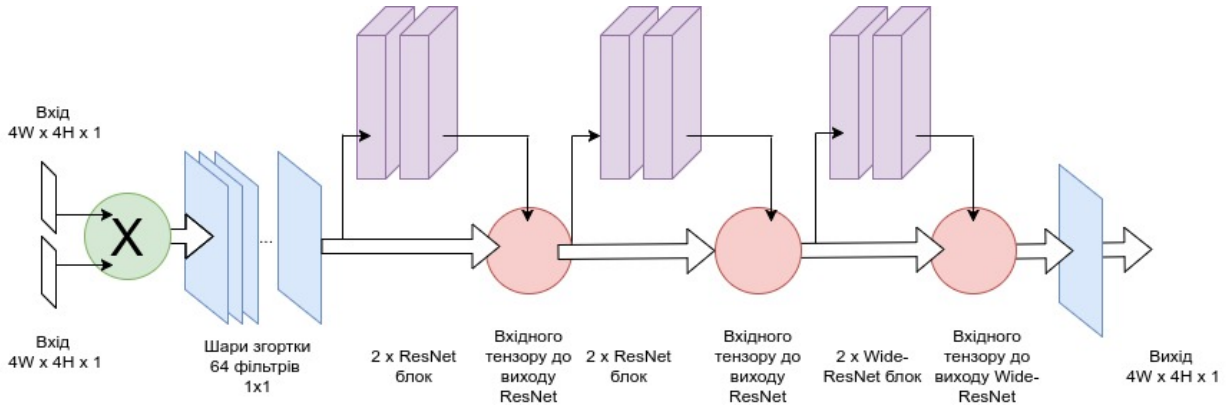

Рис 3.8 - Останній блок мережі-генератора

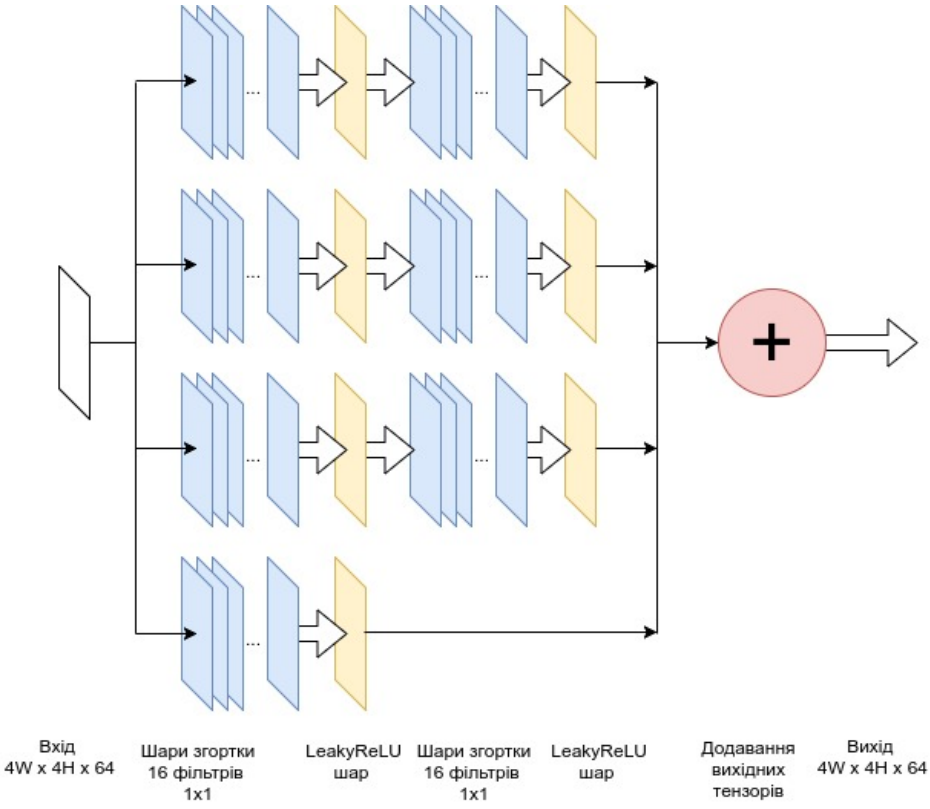

Рис 3.9 - Четвертий блок мережі-генератора

мані на попередньому етапі таким чином, щоб забезпечити максимальну подібність вихідного зображення до еталону і одночасно максимальну схожість з тим зображенням, яке було подано на вхід. Даний блок складається з шарів згортки, блоків ResNet і Wide-ResNet, і на виході має шар згортки з 1 фільтром розмірністю 1x1 і функцією активації ReLU. Блоки Wide-ResNet відрізняються від ResNet тим, що кожен шар згортки має 128 фільтрів замість

64, а так само другий шар згортки має фільтри розмірністю 1x1 замість 3x3, що дозволяє зробити функцію всієї мережі більш нелінійної. Блок приймає на вхід і складає результати, отримані з глибокої і неглибокої мережі. Блок зображено на рісунку 3.8.

Архітектура дискримінатора

В процесі тренування генератора, дискримінатор тренується разом з ним. Його завдання - визначати реалістичність зображень, що подаються на вхід. На відміну від генератора, дискримінатор не бере участі в процесі роботи мережі, він потрібен лише в процесі тренування.

Справа в тому, що в процесі тренування генератора, генератор намагається мінімізувати складну функцію втрат, яка відображає різницю між генерованим зображенням і справжнім, однак мережа-генератор не має уявлення від тому, наскільки реалістично будуть виглядати вихідні зображення. Саме для цього вводиться ще одна додаткова нейронна мережа - дискримінатор, яка тренуючись паралельно з генератором, отримує на вхід випадково вибрані реальні зображення з набору даних, марковані як реальні, і зображення, пропущені крізь мережу генератор з його поточним станом ваг, марковані як фальшиві. Таким чином ми тренуємо діскрімінатор відрізняти реальні зображення від фальшивих. Далі, в процесі тренування генератора, до функції втрат інформації (content loss), додається параметр, що представляє собою змагальну функцію втрат (adversarial loss), яка приймає на вхід генеровані в процесі тренування зображення, пропускає через дискриминатор з його поточним станом ваг, і повертає результат в діапазоні від 0 до 1, де 0 означає що дискримінатор вважає зображення абсолютно фальшивим, а 1 - абсолютно реальним.

Таким чином, генератор намагається максимізувати цю функцію, і мінімізувати функцію втрат інформації, а дискримінатор в свою чергу намагається її мінімізувати - так як зображення фальшиві, його вихід повинен прагнути до нуля. Через те, що генератору дуже просто підібрати такий стан,

в якому дискримінатор буде маркувати всі зображення як реальні, ці дві нейронних мережі повинні тренуватися паралельно, перебуваючи в рівновазі. Дискримінатор являю собою послідовність шарів згортки, ResNet-блоків, функцій активації LeakyReLU, і ці шари нічим концептуально не відрізняються від таких в генераторі, крім того що діскрімінатор не має верств деконволюции. Однак істотно відрізняються останні два шару (рисунок 3.10) - вони являють собою повнозв'язні шари, з 1024 і 1 нейроном відповідно. Кожен з 1024 нейронів першого повнозв'язну шару приймає на вхід кожне значення кожної карти ознак останнього згорткового шару, підсумовує і застосовує функцію активації - гіперболічний тангенс.

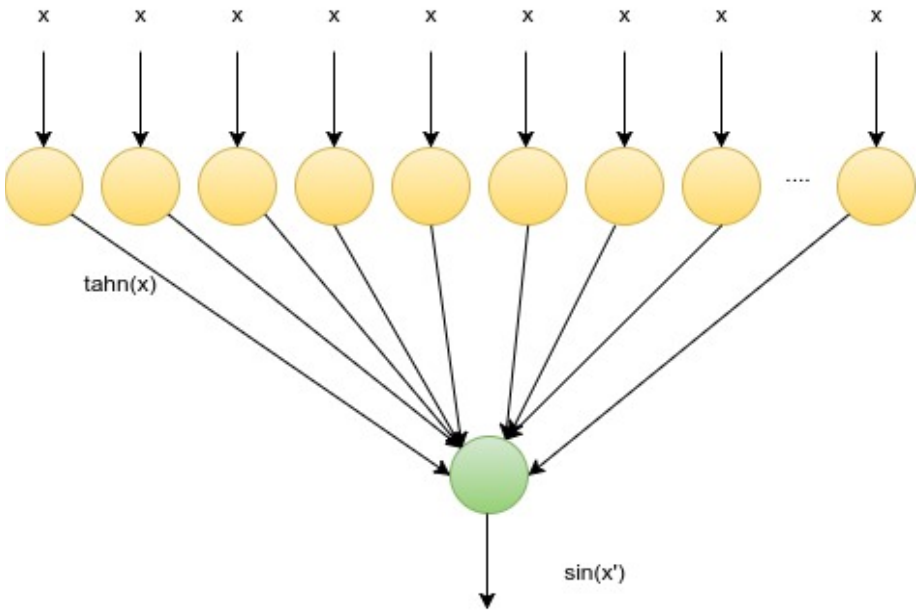

Рис 3.10 - Останні два шари мережі-дискримінатора

Наступний повнозв'язний шар приймає на вхід вихід функції активації попереднього шару, підсумовує, застосовує сигмоїдальну функцію активації. Таким чином беручи на вхід зображення розмірності 4W x 4H x 1, ми маємо на виході число від 0 до 1, що відбиває реальність зображення на думку дискриминатора. Повну схему дискримінатора зображено на рисунку 3.11.

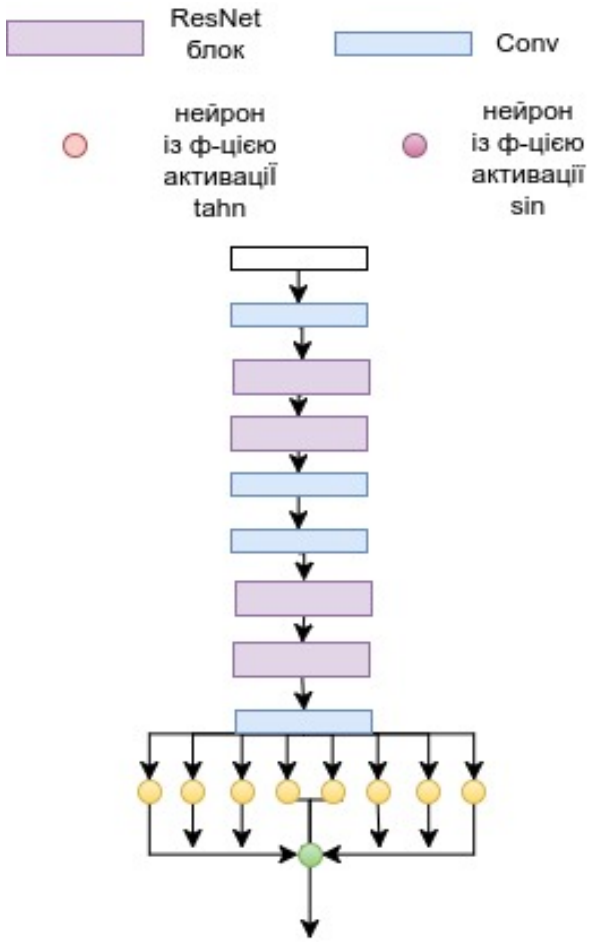

Рис. 3.11 - Діскріминатор

3.4 Препроцесінг набору даних

Далі необхідно провести попередню обробку набору данних. Суть попередньої обробки полягає в тому, щоб взяти кожне зображення з директорії з набором даних, перевести її з колірного простору RGB в YCrCb, відкинувши канали Cr і Cb, так як вони не так важливі для нашого завдання, а також розділимо значення Y каналу на 255 , щоб нормалізувати дані в проміжку від 0 до 1, натомість 0-255. Таким чином, ми отримали двовимірний масив розмірністю H x W, де H і W - ширина і висота вихідного зображення. Далі, ми випадковим чином вибираємо ділянку розміром 128 x 128. Робиться це по-

перше для того, щоб підвищити різноманіття даних для тренування мережі, а по-друге - зменшити необхідні обчислювальні ресурси для тренування, тому що чим більше початковий розмір зображень, тим більше параметрів доведеться тримати в пам'яті GDDR по час прямого і зворотного проходу даних через нейронну мережу. Далі, ми створюємо копію отриманого масиву 128 x 128, і зменшуємо її до 32 x 32 - так як в даному випадку обраний фактор масштабування 4. Таким чином ми отримали дві версії зображення - зменшену і нормального дозволу, зменшена буде подаватися на вхід в мережу, і вихід мережі буде порівнюватися з нормальною версією зображення 128 x 128. Тепер збережемо отримані масиви в файл dataset.h5 за допомогою бібліотеки h5, зображення 32 x 32 запишемо в поле data, а 128 x 128 - в поле label. Ми зберігає всі зображення в один файл для того, щоб звести до мінімуму витрати часу на читання файлів. Таким чином, через деякий час ми отримали файл dataset.h5 з даними для тренування мережі. Проробимо ту ж процедуру з частиною файлів і збережемо як validation_set.h5 - ці дані ми будемо використовувати для того, щоб оцінити працездатність мережі на даних, яких не було в навчальній вибірці.

3.5 Алгоритм навчання мережі

З метою демонстрації генеративно-змагальної архітектури було вирішено розбити тренування мережі на два основних етапи - тренування з дискримінатором і без. На першому етапі ми тренуємо нейронну мережу використовуючи лише функцію втрат контенту, яка є сумою середньоквадратичної різниці між передбаченим зображенням і справжнім, підготовленим на етапі підготовки наборів даних.

1. Алгоритм може бути описаний такою послідовністю дій:

2. Прочитати набір даних з жорсткого диска, який збережений у форматі h5 і записати його в 2 змінних - data і label, де data є зменшені до 32 x 32 пікселів зображення, а label - їх еталонний варіант вирішення 128 x 128. Ана-

логічно в змінні data_val і label_val прочитати валідаційні набір даних.

3. Проініціалізувати нейронну мережу випадковими вагами.

4. Подати на вхід мережі пакет даних, що містить 8 зображень. Розмір може бути будь-яким, проте 8 зображень - це максимальна кількість, яка не викликало переповнення пам'яті при тренуванні мережі в ході експерименту.

5. Отримавши на виході мережі передбачене зображення, порахувати помилку мережі.

6. За допомогою зворотного поширення помилки передати градієнт помилки від виходу мережі до її входу, оновивши ваги всіх верств.

Дії повторюються для всього набору даних, прохід по набору даних називається епохою. Дії повторюються протягом 100 епох, або до тих пір поки помилка на валідаційні наборі даних не припинить падати - в такому випадку слід зупинити навчання і змінити набір даних або параметри мережі і запустити заново. Швидше за все функція нейронної мережі потрапила в локальний мінімум, і подібні дії швидше за все змусять її залишити його і рухатися в бік глобального мінімуму.

Після тренування нейронної мережі, що виконує роль генератора, виконується тренування всієї системи генеративно-змагальної системи генератор-дискримінатор. Для початку проведемо попереднє навчання дискримінатора наступним чином:

1. Аналогічних попередньому алгоритму тренування генератора чином завантажимо набір данних.

2. Візьмемо пакет з 8 зображень і подамо на вхід генератора, отримавши на виході пакет з 8 передбачених зображень. Далі будемо називати їх синтетичними, або фальшивими.

3. Виберемо випадковим чином із змінної label з еталонними версіями зображень також 8 зображень. Далі ці зображення ми будемо називати реальними.

4. Створимо 2 вектора з кількістю елементів, що дорівнює кількості елементів в пакетах на попередніх пунктах - 8. Один з них заповнимо випад-

ковими числами в проміжку від 0.1 до 0.3, інший - в проміжку від 0.8 до 1.2. Вони будуть векторами очікуваного виходу мережі дискриминатора - перший для синтетичних зображень, другий - для реальних. В цілому, в оригінальній роботі по генеративно-змагальним мереж, використовувалася маркування фальшивих зображень як 0, а реальних - як 1, проте в ході експериментів було встановлено що випадкові числа, що знаходяться в деякій області навколо 0 і 1 відповідно, змушують дискримінатор швидше сходитися.

5. Припустимо пакет з синтетичними зображеннями через мережу-дискримінатор, на виході отримавши вектор чисел в діапазоні від 0 до 1 - одне число для кожного синтетичного зображення, що відображає ймовірність того наскільки зображення "реалістично" на думку дискриминатора. Порахуємо середню помилку між виходом дискриминатора і значеннями, згенерували на попередньому кроці, і пройшовши в зворотному напрямку від виходу мережі до її входу і вважаючи градієнт помилки оновимо ваги мережі.

6. Аналогічним чином подамо на вхід пакет реальних зображень, і порахуємо помилку між виходом мережі і очікуваним виходом з кроку 4, порахуємо градієнт і оновимо ваги всіх верств мережі.

Алгоритм циклічно повторюється для всіх зображень в наборі даних 1 епоху, або більше якщо за 1 епоху помилка мережі не стала досить малою (меньш ніж 0.01). Далі, маючи вже попередньо тренований генератор та дискримінатор будемо виконувати тренування всієї системи, що складається з генератора і дискримінатора:

1. Випадковим чином вибрати 8 зображень з набору даних і подати на вхід генератора, отримавши на виході 8 синтетичних зображень.

2. Випадковим чином вибрати 8 еталонних зображень.

3. Отримавши 8 синтетичних і 8 реальних зображень, промаркувати їх так само, як при попередній тренуванні дискриминатора - синтетичні випадковими числами в проміжку між 0 і 0.3, реальні - випадковими числами в проміжку між 0.8 і 1.2, подати на вхід дискримінатора спочатку один з них , порахувати помилку на виході, оновити ваги дискримінатора, потім інший.

4. Після чого вибрати випадковим чином 8 зображень (з контейнера data), подати на вхід повної мережі, що містить генератор і дискримінатор, таким чином щоб вихід генератора подавався на вхід дискримінатора. Повна мережа має 2 виходи - синтетичне зображення, передбачене генератором, і дійсне число в діапазоні від 0 до 1 - ймовірність того що зображення реальне, а не генерированное на думку дискриминатора.

5. Порахувати помилку між еталонним зображенням (з контейнера label) і синтетичним зображенням - отримати помилку втрати інформації.

6. Порахувати помилку, що представляє собою різницю між значенням на виході дискримінатора і константою, що представляє абсолютно реальне зображення - 1.

7. Скласти дві помилки - помилки втрати інформації і помилку дискримінатора, отримавши загальну помилку генератора, порахувавши градієнт помилки на виході мережі пропустити помилку від виходу до її входу, через генератор і дискримінатор таким чином, попередньо "заморозивши" ваги дискримінатора - це необхідно для того, щоб вони не оновлювалися під час проходу градієнта, так як на даному етапі нам необхідно оновити тільки параметри генератору.

Процес повторюється протягом 100 000 пакетів по 8 зображень, або ж до тих пір поки візуально і якісно зображення на виході генератору не перестануть поліпшуватись. Причина в тому, що через те що дві мережі тренуються одночасно один з одним, при цьому помилка кожної з них залежить від іншої мережі, чисельне вираження функцій втрат обох мереж може не відображати реальну "якість" роботи мереж, тому проводити ранню зупинку навчання , або ж навпаки продовження навчання, варто грунтуючись на суб'єктивній оцінці прикладів зображень на виході генератора. При цьому на кожній ітерації дискримінатор тренується на N пакетах даних, а генератор - на k * N пакетах даних - причина в тому, що теоретично генератору "складніше" покращитися за N проходів настільки, щоб "обдурити" дискримінатор, при цьому дискримінатору досить просто за N проходів, через меншу глибину

мережі, оновити свої ваги таким чином щоб досить точно відрізняти реальні зображення від синтетичних. Саме для того, щоб мережі постійно перебували в рівновазі, і жодна мережа не була набагато "сильніше" інший - вводиться коефіцієнт k, який підбирається емпірично - в діапазоні від 2 до 5. Причина полягає в тому, в що якщо генератор буде набагато "перевершувати" дискримінатор - дискримінатор на кожне синтетичне зображення буде видавати відповідь, близький до 1 (абсолютно реальне зображення), через що помилка буде прагнути до 0 - відповідно така помилка не вноситиме зміни в параметри генератора. У свою чергу, якщо дискримінатор буде набагато "перевершувати" генератор - і буде на кожне генерированное зображення видавати результат близький до 0 (абсолютно реалістичне зображення), помилка буде завжди максимальною, і буде неможливо порахувати градієнт такої помилки щоб рухати ваги генератора - генератор або зупиниться на одному місці, або буде видавати на виході випадковий шум.

3.6 Алгоритм роботи із тренованою мережею

Коли тренування завершено, мережа може бути використана у режимі передбачення.

1. Завантажити зображення, що потребує збільшення роздільної здатності.

2. Перевести зображення із RGB до YCrCb, відкинувши Cr та Cb, за допомогою бібліоткеки cv2, розділити всі значення отриманого масиву за 255 для того щоб отримати масив нормалізований в межах [0;1].

3. Подати на вхід нейронної мережі-генератора, денормалізувати - помножити всі значення виходу на 255 для того, щоб отримати масив значень від 0 до 255.

4. Взяти Cr та Cb канали зображення із п.2, та збільшити іх в необхідну кількість раз за допомогою бікубічної інтерполяції з бібліотеки cv2.

5. Конкатенувати вихід мережи із Cr та Cb, перевести в RGB за допомо-

гою бібліотеки cv2.

Блок-схема алгоритму зображено на рисунку 3.12.

Рис 3.12 - Алгоритм роботи із мережею-генератором

3.7 Аналіз отриманих результатів

Як було зазначено вище, ми навчаємо спочатку генератор без дискримінатора. Після 50 епох навчання проаналізуємо отримані результати. Як зображень будемо використовувати зображення з набору даних Set14. Для порівняння будемо використовувати еталонне зображення, зображення отримане методом бикубической інтерполяції з зменшеного в 4 рази вихідного зображення, і зображення отримане за допомогою нейронної мережі SRCNN [1].

Рис. 3.13 - Результат роботі мережі на даних Set 14

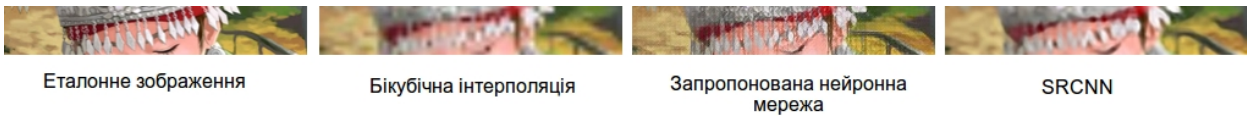

Рис. 3.14 - Результат роботі мережі на даних Set 14

Із рисунку 3.13 та рисунку 3.14 видно, що зображення виглядає краще ніж зображення, отримане методом бікубічної інтерполяції, принаймні запропанована глибока нейронна мережа краще відновлює дрібні деталі.

Порівнюючи з SRCNN, слід зазначити, що незважаючи на те що запропонована мережа краще відновлює дрібні деталі, вона так само генерує набагато більше високочастоних шуму, а так само "намагається" відновити висо-

кочастотні деталі там, де їх не було в оригінальному зображенні, на відміну від SRCNN, яка однак в свою чергу генерує більше розмиту картинку.

Імовірно, генерація мережею непотрібної вісокочастотної інформації пов'язана з набором даних, на яких вона навчалася, а також з функцією помилки, яка обчислюється як сума між середньоквадратичної розносять еталонного зображення і отриманого на виході нейронної мережі і середньої абсолютної різницею між еталонним зображенням, пропущеним через перші 7 шарів мережі VGG16, тренованою на наборі даних ImageNet, і генерованим зображенням, пропущеним через ті ж 7 шарів тієї ж мережі. Таким чином, ми порівнюємо не зображення самі по собі, а порівнюємо карти ознак виділені мережею VGG16. Як відомо, згорткові мережі виділяють все більш абстракні карти ознак (англ. feature maps) в по мірі даних від входу мережі, до її виходу. Інший пріклад зображено на рисунку 3.15.

Рис. 3.15 - Результат роботі мережі на даних Set 14

Протестуємо нейронну мережу на обличчях людей, випадковим чином вибраних з набору данних CelebA (рисунок 3.16). На тестових зображеннях досить помітно, що запропонована нейронна мережа недостатньо добре відновлює високочастотну інформацію - частини осіб, очі, волосся і так далі. По-друге, вона так само генерує непотрібний високочастотний шум, ймовірна причина появи якого описана вище. Ймовірно це пов'язано з тим, що в наборі даних, на якому мережа тренувалась 50 епох, людські обличчя становили всього близько 30%. Однак слід зазначити, що зображення виглядають набагато краще, ніж отримані методом бікубічної інтерполяції.

Рис. 3.16 - Результат роботі мережі на даних CelebA

Протестуємо нейронну мережу на частині випадковим чином вибраного зображення з набору даних DIV2K (рисунок 3.17).

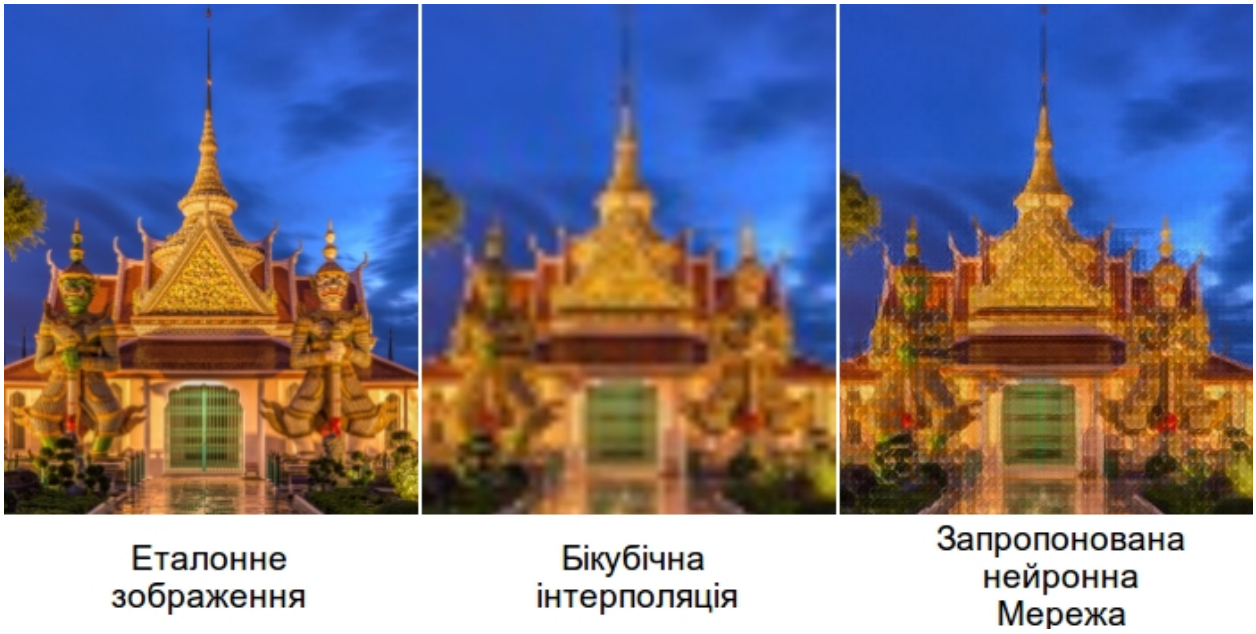

Рис. 3.17 - Результат роботі мережі на даних DIV2K

Незважаючи на те, що зображення, що створюється нейронною мережею виглядає менш розмитим, ніж зображення отримане методом бикубической інтерполяції, високочастотний шум і артефакти настільки явно виражені, що повністю позбавляє зображення фотореалістичності. З отриманих результатів випливає, що:

1. Необхідно змінити функцію втрати інформації помилки таким чином, щоб порівнювати менш абстрактний карти ознак - імовірно, слід брати їх не з сьомого згорткового шару VGG16, а, наприклад, з п'ятого згорткового шару.

2. необхідно додатково тренувати мережу на наборі даних, який містить тільки людські обличчя для отримання більш реалістичної картинки.

3. Для підвищення реалістичності необхідно тренувати мережу з використанням мережі-дискримінатора.

Після внесення необхідних змін в алгоритм тренування таким чином, щоб для обчислення помилки використовувався п’ятий шар мережі VGG16,

та 50 додаткових епох навчання, були отримані наступні результати.

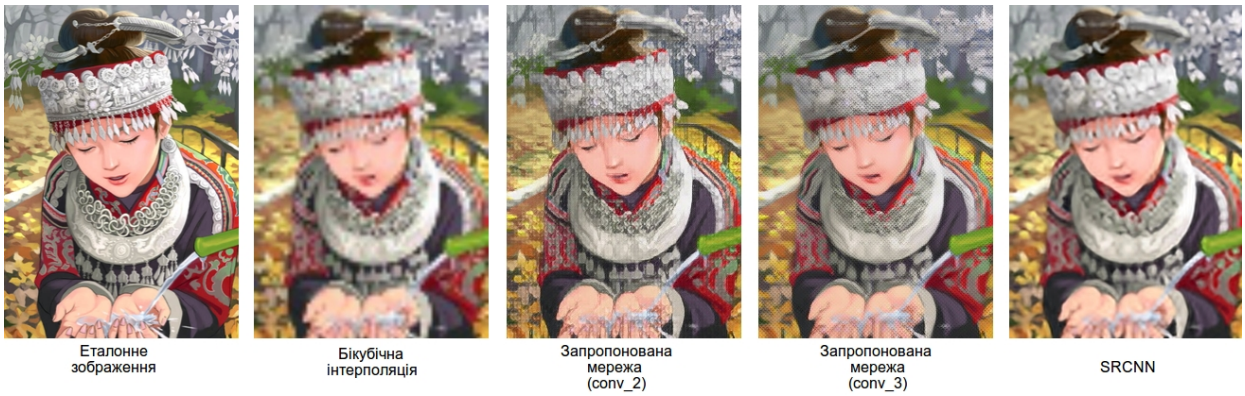

Рис. 3.18 - Результат роботі мережі на даних Set 14 після внесення змін

Рис. 3.19 - Результат роботі мережі на даних Set 14 після внесення змін

Із зображень (рисунок 3.18, 3.19) очевидно, що мережа почала генерувати більш розмиті зображення. Імовірно, це пов’язано із не достатньою кількостю епох додаткового тренування, так як після 50 епох помилка продовжувала зменшуватись, а також із застосуванням між третім та четвертим блоком генератора так званого “вузького місця” (англ. - bottlenecking), коли карти признаков із умовно неглибокої мережі та умовно глибокої мережі складаються в одну карту признаков, яка після цього у наступних ResNet блоках знов розкладається на 64 мапи ознак - імовіно, це зменшує швидкість проходження градієнту через мережу та робить навчання більш повільним.

Розглянемо інший приклад із набору даних Set 14 (рисунок 3.20, рисунок 3.21).

На ділянці зображення із квітами дуже добре видно, що замість високочастотного шуму, який був присутній при використанні сьомого шару мережі

VGG16, в разі використання п’ятого шару високочастотний шум відсутній, замість нього мережа генерує точки зниженої яскравості - що робить зображення в цілому меньш чітким.

Рис. 3.20 - Результат роботі мережі на даних Set 14 після внесення

Рис. 3.21 - Результат роботі мережі на даних Set 14 після внесення

Отже, потрібно додаткове тренування мережі для запобігання розмитості зображень. Але, незважаючи на це, слід зазначити що зображення, які генеруються мережею, виглядають краще та більше реалістично, ніж зображення отримані методом бікубічної інтерполяції.

Далі, будемо додатково тренувати мережу на даних із CelebA датасету, для того щоб покращити роботу алгоритму на зображеннях із людськими обличчями.

На рисунку 3.22 зображено інтерфейс процесу тренування, в консолі виводиться вся необхідна інформація - поточна епоха тренування, загальна кількість епох тренування, кількість пройдених зображень із обраного набору

даних, середня помилка на тренувальних даних, середня помилка на валідаціоних даних.

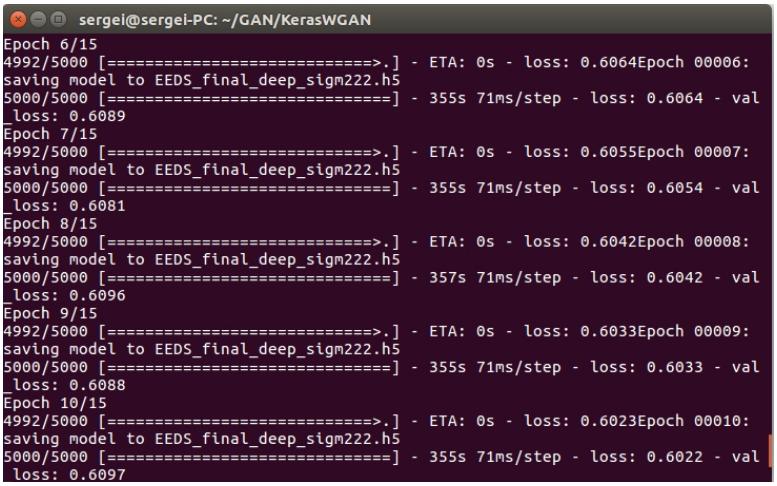

Рис. 3.22 - Інтерфейс процесу тренування

Після цього ми можемо візуальньо оцінити роботу алгоритму, який додатково тренувався тільки на даних із людськими обличчями на рісунку 3.23.

Рис. 3.23 - Додаткове тренування на CelebA

Після 30 епох тренування із дискримінатором перевіримо роботу генератора на даних, яких не було в датасетах.

Також перевіримо, як підвищується якість розпізнавання осіб. Для цього були знайдені деякі фото із відомими людьми. В якості алгоритма роспізнавання будемо використовувати API sightengine.com. В даному випадку не варто мета розробки алгоритму розпізнавання осіб, тому будемо використовувати вже готовий API, при цьому абсолютно не важливо як він працює всередині - важливо те, що він працює однаково на всіх прикладах.

Як зазначено в першому розділі, розпізнавання образів - це віднесення вихідних даних до певного класу за допомогою виділення істотних ознак, що характеризують ці дані, із загальної маси несуттєвих даних. Таким чином, маючи деяку кількість класів, ми можемо оцінити якість розпізнавання по тому, наскільки високу ймовірність приналежності до потрібного класу зможе забезпечити деякий алгоритм розпізнавання. У даній роботі, ми будемо оцінювати, наскільки точно деякий алгоритм класифікації буде відносити до потрібного класу об'єкт на зображенні низького дозволу, зображення збільшеного за допомогою бикубической інтерполяції, а так само зображення отриманого в результаті роботи запропонованої системи, а так само еталонного зображення, при цьому якість роботи алгоритму буде оцінюватися як різниця між результатом розпізнавання об'єкта на еталонному зображенні і зображенні, яке було отримано в результаті роботи запропонованої системи, таким чином встановити наскільки алгоритм здатний відтворювати вихідне зображення.

Розглянемо рісунок 3.24.

Зліва - еталонне фото, далі - бікубічна інтерполяція та останнє фото - результат роботи генератора.

Людину на еталонному фото було успішно ідентифіковано із вірогідністю 99%. На жаль, на двох інших фото ідентифікація не була

успішною, тобто алгоритм не був спроможен встановити класс до якого відноситься зображення, але мі можемо візуально оцінити поліпшення якісті ціфрового зображення у порівнянні з бикубічною інтерполяцією.

Рис. 3.24 - Робота мережі на реальних даних

Далі протестуємо на іншому фото (рисунок 3.25).

На еталонному фото було ідентифіковано чотири особи, всі із вірогідністю 99%.

На фото після бікубічної інтерполяції було вірно ідентифіковано алгоритмом лише одну людину із вірогідністю 82%.

На останньому фото, результаті роботи нейронної мережі, було ідентифіковано двох осіб із вірогідністю 86% та 90% відповідно.

Рис. 3.25 - Робота мережі на реальних даних

На третій серії фото, зображено деяку особу на церемонії вручення нагород премії «Оскар». На першому фото - еталоні - особу було ідентификовано вірно із вірогідністю 99%, тобто алгоритм розпізнавання точно візначив клас до якого відноситься особо на зображенні. На другому фото - бікубічна інтерполяція - із вирогідністю 89%. Останнє фото - результат роботи нейронної мережі - 99%. Таким чіном, мі отрімалі на 10% бліжчу до еталона точність розпізнавання. Експеримент зображено на рісунку 3.26.

Рис. 3.26 - Робота мережі на реальних даних

Далі на рісунку 3.27 зображено три фото ведучих автошоу. Зверху вниз:

- еталонне фото - ідентифіковано трьох осіб із вірогідністю 99%.

- бікубічна інтерполяція - ідентифіковано одну особу із вірогідністю 88%

- результат роботи нейронної мережі - ідентифіковано ту ж саму особу із вірогідністю 94%.

Таким чином, мі отримали результат на 6% ближчий до еталону у порівнянні із бікубічною інтерполяцією.

На рисунку 3.28 зображено ведучого, якого було ідентифіковано на попередній серії фото.

На всіх фото його було ідентифіковано із вирогідністю 99%.

Рис. 3.27 - Робота мережі на реальних даних

Рис. 3.28 - Робота мережі на реальних даних

На рисунку 3.29 фото зображено особу."

На еталонному фото зліва - особу віднесено до вірного класу з імовірністю 99%. На фото після застосування бикубической інтерполяції (посередині) - не ідентифікований. На останньому фото - результат роботи нейронної мережі - актора ідентифіковано успішно зі ймовірністю 94%. Тобто використання нейромережевого алгоритму зробило можлівім вірне розпізнавання особи, яке було неможливо у разі вікористання алгоритму бікубічної інтерполяції.

Рис. 3.29 - Робота мережі на реальних даних

На рисунку 3.29 три особи. Із-за артефактів та шуму, алгоритм розпізнавання облич не був спроможен ідентифікувати будь кого.

Рис. 3.30 - Робота мережі на реальних даних

Імовіно, це наслідок невеликої кількості епох тренування, та невеликої навчальної виборці даних. Проте, окрім останньої групи фото, нейронна мережа демонструє кращі результати, ніж бікубічна інтерполяція - як у відсотках ймовірності віднесення особи до вірного класу, який був визначений при використанні алгоритму розпізнавання на еталонному зображенні, так і візуально зображення виглядають більш фотореалістично ніж після бикубической інтерполяції. У всій випадках фактор масштабування дорівнює 4.

3.8 Висновки за розділом

В даному розділі детально описані програмні і апаратні засоби, що застосовуються для вирішення даного завдання, а так само обгрунтування, чому були обрані саме вони. Описано набори даних і їх підготовка для навчання нейронної мережі, описана архітектура використовуваних нейронних мереж, їх частини, їх алгоритм навчання, а також алгоритм роботи з мережею після навчання. Далі наведені отримані результати, порівняння з методом бикубической інтерполяції, спочатку на зображеннях з обраних наборів даних, далі на випадкових зображеннях з мережі інтернет, а так же продемонстровано покращену розпізнавання образів при застосуванні нейронної мережі в сравенніі з методом бикубической інтерполяції.

!) " 1 + ! 7)

В магістерській роботі досліджені можливості використання технік машинного навчання, таких як глибокі нейронні мережі у сфері обробки цифрових зображень та в завданні підвищення роздільної здатності зображень з метою відновлення висикочастотної інформації, що не є можливим у разі застосування існуючих алгоритмів та програмних засобів. Як конкретний приклад застосування був обраний алгоритми розпізнавання осіб, а так само зображення, які містять людські обличчя.

Незважаючи на те що запропонований алгоритм універсальний і може застосовуватись для будь-яких зображень, що несуть будь-яку інформацію, саме розпізнавання осіб було обрано в якості прикладного прикладу застосування запропонованих алгоритмів, так як саме розпізнавання осіб найбільш чутливим до нестачі інформації на цифровому зображенні.

Було здїйснено аналіз існуючих алгоритмив пидвіщеня розподильної здатності, доведено що без інтелектуальних алгоритмив, до яких видносяться глибоки нейронні мережі, досіть важко досягти результатів у завданні пидвищення роздільної здатності ціфровіх зображень в цілому та зокрема стосовно до задачі розпізнавання образів.

Проведено дослідження роботи згорткових нейронних мереж і можливості їх застосування в задачах процесингу зображень, до яких відноситься задача підвищення роздільної здатності. Також, проведено аналіз принципів роботи генеративно-змагальних нейронних мереж, показано, як їх застосування здатне поліпшити якість роботи існуючих алгоритмів навчання глибоких нейронних мереж.

В результаті:

1. Проаналізовано існуючі моделей нейронних мереж.
2. Розроблено модель згорткової нейронної мережі.
3. Розроблено алгоритм навчання нейронної мережі.
4. Модель нейронної мережі втілено в програмному продукті, розробленому за допомогою найновіших технологій паралельних обчислень.

Програмний продукт показав свою придатність і здатність виконувати поставлені цілі.!

"%)"+7 !)7+')"#01)8 -5&'&/

1. C. Dong, C. Change Loy, K. He, X. Tang, “Image Super-Resolution Using Deep Convolutional Networks”, URL http://mmlab.ie.cuhk.edu.hk/projects/SRCN-N.html, 2014.
2. Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, Wenzhe Shi “Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network” PDF https://arxiv.org/pdf/1609.04802.pdf
3. “CS231n Convolutional Neural Networks for Visual Recognition” URL http://cs231n.github.io/understanding-cnn/
4. Hubel, D. H.; Wiesel, T. N.“Receptive fields of single neurones in the cat's striate cortex”. The Journal of Physiology. 148 (3): 574–591.
5. Fukushima K., Miyake S., Takayuki I. “Neocognitron: A neural network model for a mechanism of visual pattern recognition”, IEEE Transaction on Systems, Man and Cybernetics SMC–13(5):826–34. — 1983.
6. Y. LeCun, L. Bottou, Y. Bengio and P. Haffner “Gradient-Based Learning Applied to Document Recognition”, Proceedings of the IEEE, 86(11):2278-2324, November 1998
7. Krizhevsky, Alex. “ImageNet Classification with Deep Convolutional Neural Networks” PDF, Retrieved 17 November 2013.
8. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun “Deep Residual Learning for Image Recognition” URL https://arxiv.org/abs/1512.03385
9. Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair , Aaron Courville, Yoshua Bengio, “Generative Adversarial Nets”, URL https://arxiv.org/pdf/1406.2661.pdf, 2014.
10. Alec Radford, Luke Metz, Soumith Chintala, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, URL https://arxiv.org/abs/1511.06434, 2015.

11. Justin Johnson, Alexandre Alahi, Li Fei-Fei, ”Perceptual Losses for Real-Time Style Transfer and Super-Resolution” URL https://arxiv.org/abs/1603.08155
12. Hossein Talebi, Peyman Milanfar “Learned Perceptual Image Enhancement” PDF https://arxiv.org/pdf/1712.02864.pdf
13. Bingzhe Wu, Haodong Duan, Zhichao Liu, Guangyu Sun, Peking Univeristy “SRPGAN: Perceptual Generative Adversarial Network for Single Image Super Resolution”, PDF https://arxiv.org/pdf/1712.05927.pdf 16 Dec 2017
14. Bottou, Léon “Online Algorithms and Stochastic Approximations” . Online Learning and Neural Networks. Cambridge University Press. ISBN 978-0-521-65263-6
15. “Optimization: Stochastic Gradient Descent” URL http://ufldl.stanford.edu/tutorial/supervised/OptimizationStochasticGradientDescent/
16. Hinton, Geoffrey. “Overview of mini-batch gradient descent” PDF pp. 27–29, 27 September 2016.
17. “Nesterov Accelerated Gradient and Momentum” URL https://jlmelville.github.io/mize/nesterov.html
18. John Duchi, Elad Hazan, Yoram Singer “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization” PDF http://www.jmlr.org/papers/volume12/duchi11a/duchi11a.pdf
19. Tieleman, Tijmen and Hinton, Geoffrey “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude” PDF https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf, 2012
20. Zeiler, Matthew D. “ADADELTA: An adaptive learning rate method”, PDF https://arxiv.org/pdf/1212.5701.pdf, 2012
21. Diederik, Kingma; Ba, Jimm “Adam: A method for stochastic optimization” PDF https://arxiv.org/pdf/1412.6980v8.pdf, 2014
22. “NVIDIA GPUs - The Engine of Deep Learning” , URL https://developer.nvidia.com/deep-learning
23. “CUDA Toolkit” URL” https://developer.nvidia.com/computeworks
24. “NVIDIA cuDNN” URL https://developer.nvidia.com/cudnn

25. “An open-source software library for Machine Intelligence” URL https://www.tensorflow.org/
26. Theano documentation URL http://deeplearning.net/software/theano/
27. Microsoft/CNTK URL https://github.com/Microsoft/CNTK
28. “Keras: The Python Deep Learning library” URL https://keras.io/
29. “Large-scale CelebFaces Attributes (CelebA) Dataset” URL http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html
30. R. Zeyde URL https://sites.google.com/site/romanzeyde/research-interests
31. “The Berkeley Segmentation Dataset and Benchmark” URL https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/
32. Eirikur Agustsson, Radu Timofte “NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study” PDF http://www.vision.ee.ethz.ch/~timofter/publications/Agustsson-CVPRW-2017.pdf
33. Yann LeCun, Leon Bottou, Genevieve B. Orr and Klaus-Robert Müller, “Efficient BackProp” PDF http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf
34. Andrew L. Maas, Awni Y. Hannun, Andrew Y. Ng “Rectifier Nonlinearities Improve Neural Network Acoustic Models”, 2014