КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра математичної інформатики

Кваліфікаційна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформатика»
за спеціальністю 122 Комп’ютерні науки
на тему:

ЗАСТОСУВАННЯ НЕЙРОННИХ МЕРЕЖ ДЛЯ ГЕНЕРАЦІЇ ДИЗАЙНУ ІНТЕРФЕЙСІВ КОРИСТУВАЧА

Виконала студентка 4-го курсу
Олена НАМАКА _____
(підпис)

Науковий керівник:
асистент, доктор філософії
Ярослав ТЕРЕЩЕНКО _____
(підпис)

Консультант:
доцент кафедри математичної інформатики
кандидат фізико-математичних наук
Олександр ДЕРЕВЯНЧЕНКО _____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри математичної інформатики
«___»_____2023 р.
протокол №____
Завідувач кафедри
Василь ТЕРЕЩЕНКО _____
(підпис)

Київ – 2023

РЕФЕРАТ

Обсяг роботи 42 сторінок, 7 ілюстрацій, 18 джерел посилань.

ДИЗАЙН САЙТУ, ГЕНЕРАЦІЯ КОДУ, ІНТЕРФЕЙС КОРИСТУВАЧА, НЕЙРОННІ МЕРЕЖІ, МАШИННЕ НАВЧАННЯ.

Об’єктом досліджень в даній роботі є методи генерації дизайну інтерфейсу користувача. Предметом досліджень є програмна реалізація генерації інтерфейсів цільової сторінки за допомогою нейронних мереж.

Метою роботи є аналіз наявних методів генерації дизайну інтерфейсу та їх порівняння на прикладі задачі генерації вебсторінки сторінки.

Методи розробки: порівняльний аналіз, огляд літературних джерел, методи побудови та навчання нейронних мереж. Інструменти розробки: середовище розробки Jupyter Notebook, мова програмування Python, фреймворк TensorFlow.

Результати обробки: проведено аналіз підходів, які використовуються для генерації інтерфейсів, проаналізовано переваги кожного з цих методів, застосовано наявні підходи для конкретної задачі генерації цільової сторінки та проаналізовано результати роботи.

Сфера застосування: результати можуть бути застосовані при виборі технології для створення цільової сторінки.

ЗМІСТ

ВСТУП

Огляд сучасного стану досліджень предметної області. Генерація інтерфейсу користувача полягає в створенні ефективного та зручного способу взаємодії користувача з програмою чи вебсайтом. Це є своєрідним мостом для співпраці між кінцевим користувачем і програмним забезпеченням. Від того наскільки якісним, легким у розумінні та ефективним буде інтерфейс, залежить лояльність користувачів. Хороший графічний інтерфейс користувача складний і займає багато часу навіть для професійних дизайнерів, оскільки процес проєктування повинен відповідати багатьом правилам і принципам проєктування.

Саме тому науковці вже більше як 8 років намагаються розв’язати дану задачу за допомогою машинного навчання. Першим таким кроком була робота Halbe та Joshi у 2015 році з генерації HTML сторінки з рукописного ескізу.[1] Белтрамеллі у своєму підході pix2code, пояснив, як глибоке навчання може трансформувати скриншоти графічного інтерфейсу користувача в комп’ютерні токени.[2] Chen et al. поєднав CNN та RNN для створення методу генерації скелета GUI з вхідних даних.[3]

Незважаючи на активний дослідницький прогрес, генерація інтерфейсу далека від ідеалу. Багато підходів зосереджені на дуже невеликій кількості елементів GUI. Наприклад, Aşıroğlu et al. і Robinson розглядають лише чотири різні типи елементів.[4] Це обмежує застосовність запропонованих моделей на практиці, оскільки для ефективності підходів необхідно виявляти всі види елементів. Також поширеною проблемою є відсутність достатньої кількості якісних датасетів, а також відсутність одної чіткої системи оцінювання результатів згенерованих інтерфейсів. І як наслідок багато моделей мають помилки в своїй генерації такі як: неправильне об’єднання елементів, неправильна

класифікація малих елементів, неправильна реалізація альтернативних елементів GUI та інше.

Актуальність роботи.У сучасному світі всі програми та сайти потребують якісного інтерфейсу, а ручне проєктування та розробка GUI може бути складним та трудомістким процесом, який вимагає значних знань та навичок. Генерація GUI за допомогою нейронних мереж дозволяє спростити та прискорити процес розробки інтерфейсу для програмних продуктів, забезпечити їх зручність та ефективність використання користувачами.

Попри те, що були проведені деякі дослідження з генерації інтерфейсів користувача, мало уваги було приділено саме генерації вибраній нами landing сторінці.

Мета й завдання роботи. Метою роботи є тестування моделей машинного навчання на основі створеного датасету, які допоможуть розв’язати задачу генерації інтерфейсу для лендинг сторінок. Для досягнення цієї мети поставлені наступні завдання:

- Дослідити наявні алгоритми та підходи до розв’язання такої чи подібної задачі
- Дослідити застосування різних архітектур нейронних мереж
- Підібрати датасет та моделі для нашої задачі
- Протестувати рішення

Об'єкт, методи й засоби розробки. Об'єктом дослідження є моделі машинного навчання, призначена для генерації коду інтерфейсу користувача на основі відповідних вхідних зображень. Методи, що застосовуються, включають алгоритми глибокого навчання для генерації коду, зокрема з використанням генеративно-протилежної мережі (GAN), яка забезпечує здатність генерувати реалістичний код на основі навчальних даних. Основним засобом розробки для цього проекту є мова програмування Python. Python пропонує широкий вибір бібліотек та

фреймворків для машинного навчання, таких як TensorFlow, PyTorch, Keras, що надають потужні інструменти для розробки та навчання глибоких нейронних мереж.

Можливі сфери застосування. Розробка даної моделі може використовуватися для генерації сторінок для продуктів-початківців, які за допомогою якісної landing сторінки зможуть залучати нових клієнтів

РОЗДІЛ 1

ОСНОВНІ ПОНЯТТЯ І СКЛАДОВІ ПРИ СТВОРЕННІ МОДЕЛЕЙ ГЕНЕРАЦІЇ GUI

1.1 Графічний інтерфейс

В сучасному світі використання комп'ютерів та іншої техніки є невід'ємною частиною життя. Одним із ключових аспектів у використанні цих пристроїв є взаємодія з ними.

Графічний інтерфейс користувача (GUI) - це тип інтерфейсу, що дозволяє користувачам взаємодіяти з електронними пристроями за допомогою графічних елементів, таких як кнопки, текстові поля, меню та інші. Він є найбільш поширеним типом інтерфейсу для більшості електронних пристроїв, таких як комп'ютери, смартфони, планшети, телевізори та інші. Окрім GUI, існують ще три інтерфейси комп'ютера: інтерфейс командного рядка, інтерфейс, заснований на меню, та інтерфейс сенсорного екрану. GUI та інтерфейси сенсорного екрану схожі, але останній є розвитком, який широко використовується в сучасному світі.[6]

Графічний інтерфейс користувача (GUI) є важливою складовою взаємодії з комп'ютером та має багато переваг. 1) Він робить роботу на комп'ютері більш інтуїтивно зрозумілою, спрощуючи використання та процес навчання. Графічний інтерфейс користувача простий у використанні, так як не вимагає від користувача використання будь-яких команд. 2) Зазвичай він надає користувачеві швидкий візуальний зворотний зв'язок про результат кожної дії та дозволяє відображати декілька програм або дій одночасно. 3) GUI дозволяє користувачеві виконувати декілька задач одночасно, тому що програми та вікна можуть

бути відкритими одночасно на екрані. Це дозволяє ефективніше використовувати час та підвищує продуктивність.
Крім того, GUI розробляється таким чином, щоб передбачати потреби користувачів, фіксувати та підтримувати їх увагу.

Система WIMP є одним з підходів до створення графічного інтерфейсу користувача (GUI), і використовується в багатьох операційних системах, таких як Microsoft Windows, macOS та Unix-подібних системах.

WIMP-інтерфейс складається з наступних елементів:

- Вікна - основний елемент графічного інтерфейсу, який відображає вміст програми або файлу. Вікна можуть бути переміщені, змінювати свій розмір та приховуватись за іншими вікнами.
- Іконки - маленькі зображення, які представляють програми або файли.
- Меню - перелік команд, які можна виконати в програмі або на робочому столі.
- Вказівник миші - візуальний елемент, який користувач використовує для взаємодії з графічним інтерфейсом.

Інші типи GUI можуть використовувати інші графічні елементи, такі як тачскрини, жести, голосові команди тощо, але WIMP є одним з найбільш поширених та знайомих підходів до створення GUI.

1.2 Комп'ютерний зір

Комп'ютерний зір - це наука про сприйняття та розуміння світу через зображення та відео шляхом побудови фізичної моделі світу, щоб система штучного інтелекту могла потім виконати відповідні дії.[5] Комп'ютерний зір відповідає за розробку алгоритмів та програм для обробки, аналізу та розуміння зображень та відео, отриманих з камер,

сенсорів або інших джерел. Іншими словами, це технологія, що дозволяє комп'ютерам "бачити" світ, подібно до того, як це роблять люди.

Насправді, комп’ютерний зір вже давно допомає кожному з нас і навіть в цю секунду покращує наше життя. Воно аналізує наші рентгенівські знімки і допомагає лікарям з діагнозом; водить за нас автомобіль: розпізнає транспортні засоби на дорогах, виявляє порушення правил дорожнього руху та автоматично керує сигналізацією. У виробництві, комп'ютерний зір використовується для автоматизації виробничих процесів, контролю якості продукції та виявлення дефектів на обладнанні. У розважальній та кінематографічній індустрії комп'ютерний зір використовується для створення реалістичних спецефектів та графіки.

Для того, щоб досягти якнайкращих результатів, комп’ютерний зір поєднує в собі такі науки: інформатику, фізику, математику, нейробіологію і навіть психологію. Протягом багатьох років сформувалося безліч методів і технік використання комп’ютерного зору, одними з найбільш популярнимим є класифікація зображень, виявлення об’єктів, трекінгу об’єктів, семантична сегментація, сегментація екземплярів і реконструкція зображення.

1.2.1 Класифікація зображень

Класифікація зображень — це завдання присвоєння мітки зображенню із попередньо визначеного набору категорій.[5] Основна мета цього завдання - навчити комп'ютер розпізнавати об'єкти на зображеннях та класифікувати їх на певні категорії або класи.

У загальному випадку, для класифікації зображення модель використовується як функція, яка приймає на вхід зображення і повертає ймовірності належності зображення до кожного з попередньо визначених класів.У машинному навчанні або комп’ютерного баченні, класифікація

зображень є «проблемою навчання під наглядом», яка вирішується шляхом визначення набору цільових класів на зображенні та навчання моделі за допомогою міток на зображеннях для тренувань, які дозволяють передбачати класи або набори зображень.[6] Зазвичай використовують глибокі нейронні мережі, зокрема згорткоі нейронні мережі (Convolutional Neural Networks або CNN).

Однак досконала класифікація часто неможлива. В основному це пов’язано з наявністю шуму (у вигляді тіней, перспективних спотворень тощо), викидів (наприклад, зображення з категорії «будівлі» можуть містити людей, тварин, будівлі чи категорії автомобілів), відсутність міток, доступність лише невеликих навчальних вибірок.

1.2.2 Детекція об’єктів на зображенні

Детекція об’єктів на зображенні - це технологія пов'язана з комп'ютерним зором та обробкою зображень, яка полягає у виявленні на зображенні екземплярів семантичних об'єктів певного класу, таких як люди, будівлі чи автомобілі. Це включає в себе дослідження областей, таких як локалізації кількох категорій, виявлення країв, помітних об'єктів, визначення пози, тексту на зображенні, обличчя та пішоходів.

Метою детекції об'єктів є розробка обчислювальних засобів, моделей та методів, що є ключовими складовими для програм комп'ютерного зору. Ці інструменти допомагають виявляти предмети на зображеннях та визначати їх місцезнаходження, що є однією з основних складових знань у галузі комп'ютерного зору. [7]

Основна різниця між детекцією об’єкта та класифікацією полягає в тому, що в задачі детекції необхідно визначити не лише клас зображення, але й точне положення та межі об'єктів на зображенні. Детекція об'єктів

використовується в різних сферах, таких як системи безпеки та відеоспостереження, медичні дослідження та індустрія робототехніки. Його застосовують для автоматичного аналізу великих обсягів даних.

Детекція об'єктів може використовуватися для автоматичної генерації GUI, наприклад, для локалізації елементів на сторінці веб-сайту, які можуть бути складовими графічного інтерфейсу. Основна структура підходів до детекції об’єктів, що лежить в основі глибокого навчання, складається з кодерів і декодерів. Кодер бере зображення та оцінює його за допомогою шарів і мереж, щоб проаналізувати його характеристики та відрізнити його від інших зображень. Зображення, класифіковане кодувальником, приймається декодером, який потім прив’язує та позначає зображення.[6]

1.2.3 Трекінг об’єктів

Трекінг об'єктів - це процес автоматичного визначення та відстеження руху конкретних об'єктів на відео або зображенні за допомогою комп'ютерного зору. Цей процес використовується в багатьох областях, таких як автоматичне водіння, безпека, медицина, моніторинг транспорту та багато іншого.

Підходи до вирішення проблеми трекінгу об'єктів використовують виявлення об'єктів один раз під час початкових етапів виявлення. Воно достатньо точне та ефективне для локації об'єкта, коли він виходить за межі. Алгоритм трекінгу об'єктів є стійким до затемнення та достатньо ефективним для відстеження об'єктів під час швидких рухів та переміщень.[6]

1.2.4 Семантична сегментація

Семантична сегментація - це процес обробки зображень, який полягає в присвоєнні кожному пікселю зображення певної мітки, що відповідає приналежності до класу. Зазвичай ці мітки представлені в вигляді кольорових пікселів, які відповідають певним об'єктам або регіонам зображення. Семантична сегментація застосовується в області комп'ютерного зору, глибокого навчання та штучного інтелекту, і є важливим етапом у вирішенні багатьох завдань, таких як розпізнавання об'єктів, розуміння сцени, робота з відеоданими тощо.

Сучасні системи семантичної сегментації часто побудовані на таких архітектурах, як мережа піраміди ознак (Feature Pyramid Network або FPN) (Lin, Dollar´ et al. 2017)[8], яка має зв’язки зверху вниз, щоб допомогти просочити семантичну інформацію до карти з вищою роздільною здатністю. Також популярними є застосування глибоких нейронних мереж, зокрема, згорткових нейронних мереж (Convolutional Neural Networks, CNN), методів розпізнавання образів, зокрема, алгоритмів кластеризації (наприклад, алгоритм K-means) [6]. Алгоритми семантичної сегментації спочатку тренувались та тестувалися на наборах даних, таких як MSRC (Shotton, Winn et al. 2009) та PASCAL VOC (Everingham, Eslami et al. 2015)

Семантична сегментація використовується в задачі генерації інтерфейсів для автоматичного розпізнавання елементів інтерфейсу на зображенні, таких як кнопки, текстові поля, функціональні елементи тощо, та їх подальшої розмітки в коді програмного забезпечення.

1.2.5 Реконструкція зображень

Відновлення старих або деформованих зображень називається відтворенням зображень (англ. Image Reconstruction), де моделі використовують поточні набори даних, щоб відновити знищені або пошкоджені частини зображення. Цей процес може включати в себе використання алгоритмів, що базуються на статистичних методах аналізу зображень, а також на технологіях глибокого навчання, які використовують нейронні мережі для відновлення пошкоджених або втрачених частин зображення.

1.3 Глибоке навчання

Навчання по прецедентах — це набір методів, які дозволяють будувати алгоритми вирішення задач, що автоматично виявляють ознаки і представлення даних, необхідні для виявлення або класифікації. Методи глибокого навчання – це методи навчання репрезентації з декількома рівнями репрезентації, отримані шляхом складання простих, але нелінійних модулів, кожен з яких перетворює представлення на одному рівні (починаючи з необроблених вхідних даних) у представлення вищого рівня абстракції.[9]

Глибоке навчання (англ. deep learning) використовується в багатьох галузях, включаючи:

1. Комп'ютерний зір: глибинні нейронні мережі використовуються для розпізнавання облич, обробки зображень, аналізу відео, автоматичної індексації та класифікації фотографій.

2. Обробка природніх мов: глибоке навчання використовується для розуміння тексту, генерації тексту, перекладу мов та аналізу емоцій в текстах.
3. Фінанси: глибоке навчання використовується для прогнозування фінансових ринків, аналізу кредитного ризику та виявлення шахрайства.
4. Медицина: глибоке навчання використовується для діагностики та прогнозування захворювань, виявлення нових методів лікування та аналізу медичних зображень.
5. Робототехніка: глибоке навчання використовується для навчання роботів та автономних систем працювати в різних умовах та виконувати різні завдання.’

Головний аспект глибинного навчання полягає в тому, що шари ознак не створені інженерами-людьми: вони вивчаються з даних за допомогою універсальної процедури навчання.

Історія глибокого навчання починається в середині 20 століття, коли науковці почали вивчати мозок і розумові процеси. У 1943 році Мак-Каллок та Піттс опублікували статтю про модель нейрона [10], яка згодом стала основою для розвитку нейронних мереж. Вони використовували комбінацію алгоритмів та математики, яку вони називали "поріговою логікою", щоб імітувати процес мислення. З того часу глибоке навчання регулярно розвивалося.

40 років тому сенсорні матриці були лише двох шарів глибиною, оскільки арифметично було неможливо побудувати більші матриці. У 1980-х роках з'явилися багатошарові нейронні мережі, які були здатні вирішувати складніші завдання. Проте, в той час не було ефективних методів навчання цих мереж, і глибоке навчання не знайшло широкого застосування.

У 2006 році Джеффрі Хінтон та його колеги опублікували статтю про глибокі нейронні мережі [11], які були навчені за допомогою алгоритму зворотнього поширення помилок. Це відкриття забезпечило революцію в глибинному навчанні і відновленні нейронних мереж в цілому. За допомогою різних сенсорних матриць у глибинному навчанні комп'ютери мають здатність бачити, вчитися та реагувати на складні умови так само або навіть краще, ніж люди

1.3.1 Generative Adversarial Network (GAN)

GAN (генеративно-змагальна нейронна мережа) є алгоритмом класичного машинного навчання без учителя, що поєднує в собі дві нейромережі - "генератор" та "дискримінатор". Основною метою генератора є створення образів певної категорії, тоді як завдання дискримінатора полягає в розпізнаванні та класифікації створених образів.

Можна вважати генератора аналогом художника, який фальсифікує картини, а дискримінатора - інспектора з живопису, що намагається виявити підробку. Дві нейромережі постійно намагаються обманути одна одну: чим краще генератору вдається створювати переконливі картини, тим краще дискримінатор повинен відрізняти реальні приклади від фальшивих.

На початку навчання генератор виявляє незадовільну точність у створенні образів, які значно відрізняються від реальних зображень. Дискримінатор, у свою чергу, проявляє недостатню кваліфікацію і має обмежену здатність розрізняти між справжніми та згенерованими зображеннями та визначати належні дії. Проте, дискримінатор має доступ до справжніх прикладів та після кожної спроби розпізнати, яке зображення є фальшивим, а яке є реальним, ми повертаємо результат дискримінатора.

Цим самим, кожного разу, коли дискримінатор помиляється, класифікуючи підроблене зображення як справжнє, генератор отримує зворотний зв’язок про свої успіхи, і навпаки - кожного разу, коли дискримінатор правильно відрізняє підроблене зображення від реального, генератор отримує зворотний зв’язок про необхідність покращити свою роботу.

Розглянемо як працює GAN на прикладі генерації рукописних чисел. Для цього використаємо набір даних MNIST (The Modified National Institute of Standards and Technology)[16]

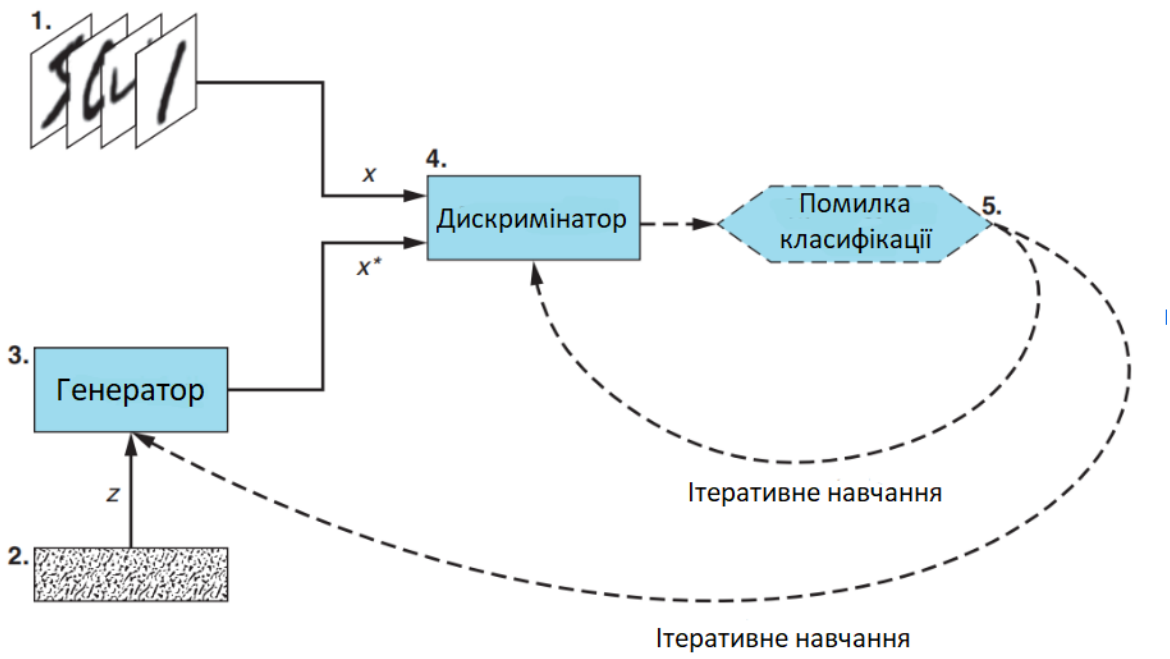

Рисунок 1 [12]

(1)Навчальний набір даних (2)Вектор випадкового шуму

(3)Генератор (4) Дискримінатор (5) Ітеративне навчання

На рисунку 1 ви можете бачити основні складові нашої моделі, а саме:

- Навчальний набір даних - набір даних реальних прикладів. У нашому випадку набір даних складається із зображень рукописних цифр. Цей набір даних служить входом (x) для дискримінатора.
- Вектор випадкового шуму - необроблений вхід (z) у мережу генератора, який складається з випадкових чисел. Він потрібен для гарантії того, що генератор не буде постійно генерувати одне і те саме зображення.
- Генератор - приймає вектор випадкових чисел (z) на введення та виводить фальшиві приклади (x^*). Його мета - зробити фальшиві приклади не відрізненими від реальних прикладів у навчальному наборі даних.
- Дискримінатор - приймає як вхідні дані реальний приклад (x) із навчального набору або підроблений приклад (x^*), створений генератором. Для кожного прикладу дискримінатор визначає й виводить ймовірність того, чи є приклад реальним.
- Ітеративне навчання. Для кожного прогнозу дискримінатора ми визначаємо, наскільки він хороший — так само як і для звичайного класифікатора — і використовуємо результати для ітеративного налаштування мереж дискримінатора та генератора :
- Вагові коефіцієнти та зміщення дискримінатора оновлюються, щоб максимізувати його точність класифікації (максимізація ймовірності правильного передбачення: x як реальне та x^* як підробка)

  Вагові коефіцієнти та зміщення генератора оновлюються, щоб максимізувати ймовірність, що дискримінатор неправильно класифікує x^* як дійсний.

Модель GAN є багатофункціональним та різноманітним інструментом, який має широкі можливості застосування в різних областях. Охорона здоров'я, бізнес, кібербезпека, анімація, переклад та

редагування є основними та найважливішими галузями, в яких модель GAN допомагає покращувати якість та продуктивність. Сектор охорони здоров'я є найбільш корисним використанням нейронних мереж та моделі GAN, оскільки вони допомагають у діагностуванні та радіаційній терапії, щоб отримати відповідні зображення та спостерігати рух внутрішніх частин організму.

Генерація віртуальних середовищ з використанням GAN є одним з найбільш інноваційних застосувань цієї технології. За допомогою GAN можна створювати віртуальні світи, які можуть використовуватися для різних цілей, таких як ігри, візуалізація даних, віртуальні тренажери тощо. Один з прикладів застосування GAN для генерації віртуальних середовищ - це створення віртуальних ігор. Наприклад, можна навчити генератор створювати нові зображення з різними об'єктами, текстурами, освітленням, та іншими характеристиками, які потрібні для створення віртуальних ігор. За допомогою цих зображень можна створювати віртуальні світи, які будуть відповідати заданим параметрам та створювати графічний ефект реалістичного світу. Ще один приклад застосування GAN для генерації віртуальних середовищ - це візуалізація даних. За допомогою GAN можна створювати віртуальні світи, які відображають різні параметри даних, такі як рівень продажів, географічні розподіли.

1.3.2 Convolutional neural network (CNN)

Згорткова нейронна мережа (Convolutional Neural Network або CNN) є підходом до глибокого навчання у сфері комп'ютерного зору та обробки зображень. Це багатошарова нейронна мережа, яка здатна ефективно розпізнавати та аналізувати зображення завдяки використанню спеціальних шарів, таких як шари згортки, та повнозв'язані шари. CNN

використовується для багатьох завдань обробки зображень, включаючи класифікацію зображень, виявлення об'єктів, трекінгу об'єктів, розпізнавання облич, сегментацію зображень та багато інших.

Основна ідея полягає в тому, щоб розпізнавати зображення на основі його візуальних ознак, таких як кольори, форми, текстури та інші. У CNN використовуються спеціальні шари, які виконують певні операції над зображеннями. Наприклад, перший шар може виявляти грані та контури на зображенні, другий шар - форми, а третій шар - вищі рівні абстракції, такі як геометричні форми або обличчя. Після проходження через ці шари зображення зменшується у розмірі, але стає більш абстрактним та зручним для подальшої обробки.

На відміну від звичайної нейронної мережі з передачею вперед, в якій нейрони розташовані в плоских, повністю зв'язаних шарах, шари у ConvNet розташовані у трьох вимірах (ширина × висота × глибина). Згортки виконуються шляхом переміщення одного або кількох фільтрів по вхідному шару. Кожен фільтр має відносно мале рецептивне поле (ширина × висота), але завжди простягається через всю глибину вхідного тензору. На кожному кроці, коли фільтр переміщується по входу, кожен фільтр виводить одне значення активації: добуток між значеннями входу та елементами фільтру. Цей процес призводить до отримання двовимірної карти активації для кожного фільтру. Карти активації, отримані кожним фільтром, потім накладаються одна на одну, щоб утворити 3-вимірний вихідний шар; глибина виходу дорівнює кількості використаних фільтрів.

Для кращого розуміння наводимо приклад (рис 2), де 3 × 3 згортковий фільтр, “ковзає” по 5 × 5 вхідному зображенню — зліва направо, зверху вниз. На кожному кроці фільтр переміщується на два стрибки; відповідно, він робить загалом чотири кроки, що призводить до створення карти активації розміром 2 × 2. Зверніть увагу на те, як на кожному кроці весь фільтр породжує одне значення активації.

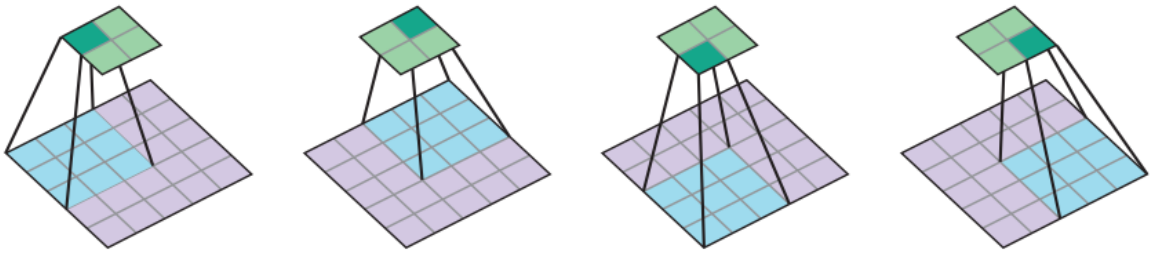

Рисунок 2 — Згортка 3 × 3 ядра на 5 × 5 вхідному зображенні з використанням кроків 2 × 2 [12]

Принцип роботи згорткових нейронних мереж можна зрозуміти на прикладі читання книги за допомогою збільшувального скла. Людина може прочитати всю сторінку, але для розгляду дрібних деталей вона дивиться на менші ділянки зображення.

Використання CNNs дозволяє досягати вражаючих результатів в завданнях розпізнавання образів, класифікації та визначенні об'єктів на зображеннях. Одним з основних способів використання CNNs є розпізнавання образів. Наприклад, в обробці зображень медичних знімків, CNN може бути використаний для автоматичного визначення різних патологічних змін на зображенні, таких як рак або захворювання серця.

1.3.3 Graph Convolutional Network (GCN)

GCN (Graph Convolutional Network) - це тип нейромережі, призначений для обробки графів. Граф є математичною структурою, що складається з вершин (вузлів) та зв'язків (ребер) між ними. GCN використовує згортання для обробки графа, аналогічно до того, як згортка використовується для обробки зображення.

Зазвичай існує два сценарії: структурний сценарій і неструктурний сценарій. У структурних сценаріях, структура графа є явною, такі як програми на молекулах, фізичних системах, графах знань тощо. У неструктурних сценаріях графи є неявними, тому ми спочатку повинні побудувати граф зі завдання, такі як побудова повністю зв'язаного графа слів для тексту або побудова сценарію для зображення. Після отримання графа, наступний процес проєктування спрямований на пошук оптимальної моделі GNN на цьому конкретному графі.

Для завдань навчання на графах, зазвичай існує три типи завдань:

1. Завдання на рівні вузлів зосереджені на вузлах і включають класифікацію вузлів, регресію вузлів, кластеризацію вузлів тощо. Класифікація вузлів намагається категоризувати вузли у декілька класів, а регресія вузлів передбачає неперервне значення для кожного вузла. Кластеризація вузлів має на меті розбити вузли на декілька непересічних груп, де схожі вузли мають бути в одній групі. Класичним прикладом проблеми передбачення на рівні вузлів є клуб карате Зака (Zach's karate club). Модель Закарі досліджує, як інформація поширюється в малих групах та як це впливає на взаємодію між окремими членами групи, зокрема на виникнення конфліктів і поділів.
2. Завдання на рівні ребер включають класифікацію ребер та прогнозування зв'язків, які вимагають від моделі класифікувати типи ребер або передбачити, чи існує ребро між двома заданими вузлами. Наприклад, можна застосувати GCN до білок-білкових взаємодій і передбачити, які пари білків взаємодіятимуть.
3. Завдання на рівні графів включають класифікацію графів, регресію графів та відповідність графів, для кожного з яких потрібна модель, щоб вивчати представлення графів. Наприклад, можна застосувати

GCN до дорожньої мережі і визначити, які ділянки доріг належать до одного місцевого району.

GCN використовує метод графової згортки для вирішення задач на графах. Графова згортка полягає у поелементному перемноженні матриці ознак вузлів на матрицю суміжності графа, що дозволяє врахувати інформацію про сусідні вузли. Згортка виконується кілька разів, кожен раз з оновленням матриці ознак вузлів. При кожному оновленні, кожен вузол залежить від своїх сусідніх вузлів, що дозволяє передавати інформацію по графу.

У GCN також використовується пулінг, який дозволяє скорочувати розмір графа та зменшувати кількість параметрів в мережі. Операція пулінгу зводиться до обчислення агрегованих властивостей вузлів та їх сусідів та обчислення нових векторів ознак для нового графа.

Класифікація зображень необхідна для багатьох реальних застосувань. Застосовуючи метод побудови графів, структуровані дані зображень можна перетворити на структуровані графи, які можна передати GCN. Зображення зазвичай складні, оскільки містять більше одного об'єкту, і розуміння зв'язку між цими об'єктами стає необхідним, щоб охарактеризувати взаємодію між ними. Narashimen та ін. запропонували модель на основі GCN для відповіді на це запитання, використовуючи кілька фактів зображення. [13] Cui та інші запропонували модель на основі GCN, яка бере до уваги як графи слів, так і сцен [14]. Yang та інші запропонували модель GCN з урахуванням надійних зв'язків, ігноруючи вплив ненадійних зв'язків у графі [15]. Модель Graph R-CNN ефективна у виявленні об'єктів на зображенні та їх взаємозв'язку. Результати дослідження показали, що запропонована модель перевершує існуючі методи генерації сценаріїв графів. На відміну від цієї моделі, Johnson та інші запропонували модель на основі GCN, яка приймає на вхід граф і генерує зображення з нього [16]. Таким чином, GCN допомагає в

комп'ютерному баченні для видобування інформації з зображення, а також для генерації зображень з графічних даних.

GCN може бути використаний для генерації GUI шляхом перетворення інформації про компоненти і їх взаємодії на графову структуру, яку можна подати на вхід до моделі GCN. Кожен компонент інтерфейсу може бути представлений вузлом графа, а його взаємодія з іншими компонентами може бути відображена за допомогою зв'язків між вузлами графа. Модель GCN може використовуватись для виконання різних задач на цьому графі, наприклад, прогнозування властивостей компонентів, класифікації інтерфейсів, визначення зв'язків між компонентами і т.д. Крім того, GCN може бути використаний для генерації нових інтерфейсів шляхом використання графічних моделей як шаблонів для створення нових варіантів інтерфейсів.

Graph Convolutional Networks (GCN) застосовуються в багатьох галузях, включаючи комп'ютерний зір, обробку природних мов, рекомендації та розпізнавання мови. Деякі приклади застосування GCN включають:

1. Аналіз соціальних мереж: GCN може використовуватися для виявлення впливових вузлів в соціальних мережах, виявлення спільнот або для передбачення зв'язків між користувачами.
2. Рекомендації: GCN може використовуватися для рекомендацій товарів або послуг на основі поведінки користувачів або захоплення інформації про товари.
3. Комп'ютерний зір: GCN може використовуватися для розпізнавання об'єктів на зображеннях, виявлення об'єктів на зображеннях або для генерації зображень.
4. Обробка природних мов: GCN може використовуватися для виявлення семантичних залежностей між словами, аналізу текстів або для класифікації текстів.

5. Прогнозування часових рядів: GCN може використовуватися для прогнозування часових рядів, наприклад, для прогнозування трафіку на дорогах або для прогнозування цін на фінансових ринках.
6. Біоінформатика: GCN можуть бути використані для аналізу біологічних мереж, таких як мережі білків та генів, для класифікації біологічних послідовностей та прогнозування біологічних властивостей.

РОЗДІЛ 2

ОГЛЯД АРХІТЕКТУР ГЛИБОКОГО НАВЧАННЯ ДЛЯ АВТОМАТИЧНОЇ ГЕНЕРАЦІЇ КОДУ ІНТЕРФЕЙСУ КОРИСТУВАЧА (GUI)

2.1 Генеративні архітектури з глибоким навчанням

2.1.1 Pix2code

Pix2Code - це фреймворк або підхід, який використовує глибоке навчання для автоматичного перетворення дизайнів графічного інтерфейсу (GUI) в робочий код. Він передбачає вивчення моделлю відповідності між візуальними представленнями дизайнів GUI (у вигляді статичних зображень або ескізів) та відповідним вихідним кодом.

Проблема генерації GUI поділяється на три підзадачі. По-перше, це проблема комп'ютерного зору, яка полягає у розумінні заданої сцени (у цьому випадку зображення GUI) та виведенні об'єктів, їх ідентифікації та позицій. Друга підзадача - це проблема моделювання мови, яка полягає у розумінні тексту (у цьому випадку комп'ютерного коду) та генерації синтаксично та семантично правильних зразків. І останнє нам потрібно використати інформацію про сцену та об'єкти, яку ми отримали з розуміння зображення, щоб згенерувати відповідний текстовий опис комп'ютерного коду для цих об'єктів.

Архітектура моделі складається з двох основних компонентів: енкодера та декодера. Енкодер використовує CNN для аналізу вхідних зображень GUI і отримання їх важливих особливостей. Це допомагає

розуміти об'єкти, їх положення та взаємозв'язки в графічному інтерфейсі. Далі, отримані вектори ознак передаються до декодера.

Декодер використовує рекурентну нейронну мережу (RNN) , зокрема LSTM, для генерації послідовності комп'ютерного коду. Він приймає вектори ознак від енкодера і поступово генерує токени коду один за одним. Кожен токен передається як вхід до наступного кроку генерації, а модель вчиться прогнозувати наступний токен з урахуванням попереднього контексту.

Набір даних pix2code відображає вебсайти, побудовані на основі Bootstrap, у власну мову опису із 18 лексичними токенами. Він складається з 3500 пар зображень і розмітки, який був розподілений на такі частини: 80% - навчальний набір, 10% - валідаційний набір і 10% - тестовий набір. Для відповідності вимогам моделі ResNet, набір даних був масштабований з розміру 2,400 × 1,380 пікселів до 224 × 224 пікселів. Модель ResNet використовується для вилучення ознак зображення (розмір 1 × 1 × 2,048 пікселів на кожний знімок), які передаються до моделі декодування.[6]

Результати показали, що модель pix2code демонструє високу точність і здатність генерувати коректний код відповідно до графічних інтерфейсів. Зокрема, результати підтверджують, що модель може правильно ідентифікувати та генерувати код для різних компонентів GUI, таких як кнопки, мітки, поля вводу тощо. Модель pix2code вдалося згенерувати правильний HTML код або код, який вимагає незначаних правок для 77.3% зображень на тестовому наборі.

2.1.2 Sketch2Code

Sketch2Code є інноваційною архітектурою глибокого навчання, яка перетворює скетчі (або малюнки) інтерфейсу на функціональний код. Вона дозволяє користувачам візуалізувати свої ідеї щодо інтерфейсу шляхом простих малюнків, а потім автоматично генерувати код, що відповідає цим малюнкам.

Основні компоненти архітектури Sketch2Code включають:

- Модуль передобробки: цей модуль відповідає за передобробку вхідних скетчів. Він може включати розмиття, вирівнювання, нормалізацію та інші операції, які покращують якість вхідних даних перед подальшою обробкою.
- Візуальний парсер: цей модуль використовує глибокі нейронні мережі для виявлення різних візуальних елементів на скетчі, таких як кнопки, поля введення, списки тощо. Він розпізнає та локалізує ці елементи на скетчі, що визначає їх положення та тип.
- Модуль генерації коду: Після розпізнавання візуальних елементів, цей модуль перетворює їх на функціональний код. Він використовує шаблони коду та правила, які вивчені в процесі тренуваання, для генерації відповідного коду, який визначає розміщення, стилізацію та поведінку інтерфейсу. Цей модуль може використовувати різні методи, включаючи послідовний підхід (наприклад, за допомогою Seq2Seq моделі) або генеративні змагальні мережі (GAN).
- Модуль виводу: Після генерації коду, цей модуль відповідає за візуалізацію та представлення згенерованого коду у зручному для користувача форматі. Це може включати показ інтерфейсу користувача, перегляд HTML/CSS коду або інші способи виводу результату.

Sketch2Code може перетворювати ескізи вебсайту на функціональний HTML-код. Цей інструмент був навчений з використанням зображень різних рукописних дизайнів, для яких були відповідні HTML-елементи, такі як текстові поля, кнопки та зображення. Він може зберігати інформацію, пов'язану з кожним кроком процесу генерації HTML, включаючи початкове зображення, результати прогнозування, інформацію про розташування та групування елементів. Незважаючи на цікавість цього прикладу машинно-підтриманого дизайну, неясно, наскільки повністю ця модель була навчена від початку до кінця та наскільки вона залежить від ручного створення функцій для роботи з зображеннями.[6]

2.2 Генеративно-змагальні архітектури з глибоким навчанням

2.2.1 DeepGUI

DeepGUI (Глибокий графічний інтерфейс) - це концепція використання технік та моделей глибокого навчання для розробки графічних користувацьких інтерфейсів (GUI). Головною метою DeepGUI є покращення процесу взаємодії з графічним інтерфейсом шляхом фільтрації неважливих з точки зору взаємодії, частин екрана та збільшення ймовірності успішної взаємодії з програмою.

DeepGUI використовує глибокі нейронні мережі для аналізу зображення екрана та створення теплової карти (heatmap). Теплова карта показує ймовірність того, що кожен піксель екрана належить до інтерактивного елемента GUI, такого як кнопка чи поле вводу. За допомогою отриманої теплової карти, DeepGUI може згенерувати вхідні дані для взаємодії з GUI. Це можуть бути координати дотику на екрані або інші дії, які можуть сприяти взаємодії з програмою.

DeepGUI використовує підхід глибокого навчання з підсиленням, який базується на використанні генеративно-змагальних мереж (GANs). Цей підхід дозволяє вдосконалювати ітераційний процес генерації вводу шляхом взаємодії між генератором (який генерує введення) та дискримінатором (який оцінює, наскільки введення є правдоподібним). У випадку DeepGUI, генератор створює вхідні дані, такі як координати дотику на екрані або інші дії, які сприятимуть правильній взаємодії з програмою. Генератор навчається покращувати якість генерованого вводу з кожною ітерацією. В DeepGUI дискримінатор використовується для оцінки, наскільки теплова карта (heatmap), створена генератором, відповідає реальним дотиковим елементам інтерфейсу.

Використання генеративно-змагальних мереж в DeepGUI дозволяє створювати більш точні та ефективні моделі генерації вводу для GUI, що поліпшує процес взаємодії з програмою і забезпечує кращі результати у випадку складних інтерфейсів.

2.2.2 LayoutGAN

LayoutGAN (Generative Adversarial Network for Layout Generation) є архітектурою глибокого навчання, розробленою для генерації користувацького інтерфейсу (UI). Вона використовує підхід генеративних змагальних мереж (GAN), який заснований на змагальному процесі між генератором і дискримінатором.

Архітектура LayoutGAN складається з двох ключових компонентів: генератора і дискримінатора.

Генератор приймає на вхід випадковий вектор, який представляє випадковий об’єкт в латентному просторі ознак. Цей вектор перетворюється в графічний макет шляхом послідовного розміщення графічних елементів. Графічні елементи включають класи (наприклад,

"заголовок", "кнопка", "зображення" тощо) та геометричні параметри (наприклад, розташування, розмір, колір тощо). Генератор навчається створювати реалістичні макети, які подібні до реальних макетів, заснованих на навчальних даних.

Дискримінатор виконує завдання розрізнення між синтезованими макетами, створеними генератором, і реальними макетами з навчального набору даних. Він приймає на вхід макети і намагається визначити, чи є вони реальними, чи синтезованими. Дискримінатор навчається розрізняти неправдоподібні макети, створені генератором, від реальних макетів.

LayoutGAN використовує архітектуру, яка приділяє особливу увагу контексту інтерфейсу. Вона дозволяє забезпечити зв'язок між різними елементами інтерфейсу, щоб забезпечити правдоподібне і логічне розміщення.

LayoutGAN використовує комбінацію різних функцій втрат, таких як функція втрати контексту та функція втрати дискримінатора. Це допомагає підвищити якість згенерованих зображень і забезпечити більш реалістичний результат.

РОЗДІЛ 3

ТЕСТУВАННЯ НАЯВНИХ МОДЕЛЕЙ ДЛЯ ГЕНЕРАЦІЇ ЛЕНДІНГ СТОРІНОК

3.1 Підбір зображень для датасету

Лендинг сторінка (англ. landing page) є веб-сторінкою, спеціально розробленою з метою захоплення уваги та переконання відвідувачів виконати певну дію, таку як придбання продукту, заповнення форми, підписка на розсилку тощо. Вона зазвичай містить уривок відповідної інформації, яка привертає увагу та викликає інтерес у відвідувачів, а також використовує ефективні методи дизайну та впливу, щоб стимулювати їх до дії.

У цьому розділі описано процес вибору зображень для створення датасету, який буде використано для генерації лендінг сторінок. Мета полягає в знаходженні якісних та відповідних зображень, що підкреслять естетику та смислове навантаження створюваних лендінг-сторінок.

Визначення критеріїв підбору зображень містить кілька аспектів. Перш за все, зображення повинні бути релевантними тематиці лендінг сторінки, щоб передати її цільовий контекст. Крім того, вони повинні мати високу якість з високою роздільною здатністю та чіткістю. Не менш важливо, щоб зображення були ліцензованими та доступними для використання в датасеті, дотримуючись правових обмежень та ліцензійних умов.

Для підбору зображень для датасету лендінг сторінок, було використано ресурс Bootstrap теми, а саме категорію лендинг і корпоративні сторінки. Цей ресурс має кілька переваг, які заслуговують на увагу.

- Доступний HTML код: на вказаному ресурсі можна знайти лендінг сторінки, для яких надається HTML код. Це дозволяє перевірити реальний вигляд та функціональність лендінг-сторінок перед їх використанням у моделях генерації. Можна адаптувати цей код для потреб дослідження та перевірити, наскільки моделі відтворюють вихідний дизайн.
- Тренування на Bootstrap: враховуючи, що деякі моделі генерації лендінг сторінок можуть бути натреновані на Bootstrap, використання ресурсу, який спеціалізується на Bootstrap-темах, має свою вагу. Моделі, навчені на даних, які відповідають особливостям Bootstrap-структури та елементів дизайну, можуть більш ефективно працювати зі згенерованими лендінгами.
- Різноманітність лендінг сторінок: ресурс містить велику кількість різних лендінг-сторінок, що демонструють різні теми, стилі та варіації дизайну. Це надає можливість вибору зображень, що найкраще відповідають конкретному контексту вашої задачі створення лендінг сторінок.

Після огляду доступних лендінгів сторінок, було обрано 100 зображень, які відповідають вимогам критеріїв підбору зображень, зазначених раніше. Приклад зображення з датасету представлений на рис 3.

Рисунок 3 — Приклад лендінг сторінки, відібраної для набору даних

Для подальшої роботи з цими зображеннями я розділила їх на тренувальні та тестувальні дані. З 100 відібраних зображень я резервувала певну частину, 20% (20 зображень) для тестувальних даних, а решту 80% (80 зображень) використовувала як тренувальні дані. Таке розподілення дозволяє провести об'єктивну оцінку продуктивності моделей та їх здатність узагальнювати на нові зображення, які вони не бачили раніше.

3.2 Тестування на моделі Pix2Code

Перш ніж розпочати тестування моделі Pix2Code на наборі лендінг сторінок, було підготовлено модель зі стандартними параметрами. Далі, ми встановили необхідні залежності, включаючи бібліотеки для машинного навчання та обробки зображень. Перед експериментом модель була натренерована на датасеті, який представлений у репозиторії і досягнули максимально можливої точності - 77.3%.

Набір даних, що складався з 100 лендінг сторінок, було підготовлено для використання в тестуванні. Кожне зображення лендінг сторінки було підготовлено до вхідного формату, який може бути оброблений моделлю Pix2Code. Це включало зменшення розміру зображення та нормалізацію значень пікселів.

Для тренування моделі було встановлено 50 епох. Початкове значення функції втрат (loss) становило 3.4 (рис 4). Протягом перших 30 епох спостерігалося швидке зменшення втрати, після чого процес поступово стабілізувався, досягнувши значення 0.6. Це характерний сценарій зменшення втрати під час тренування моделі.

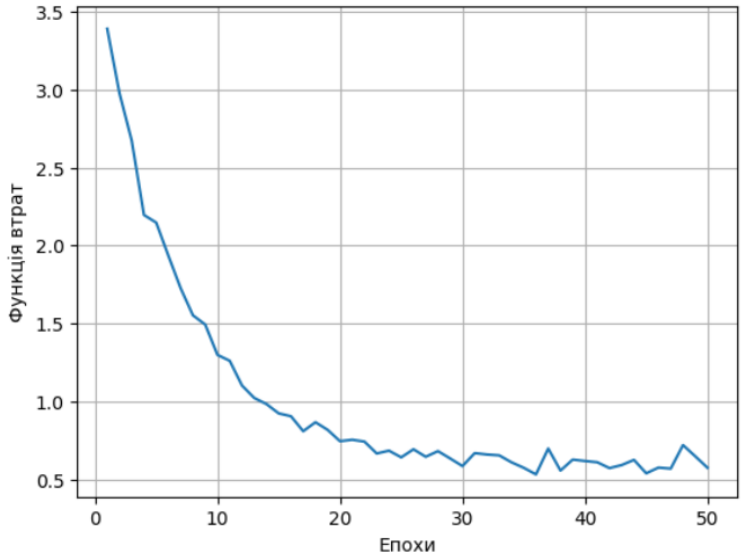

Рисунок 4 – Графік зменшення функції втрат протягом тренування

На рисунку 5 наведено вхідне зображення разом із результатом генерації моделлю. Модель зберегла структуру розташування елементів інтерфейсу. Проте, є деякі невідповідності, які можна помітити. Розташування кнопки у полі заголовка було модифіковане, логотип був замінений на сірий прямокутник, а деякі дрібні піктограми були замінені на символ "X". Незважаючи на ці невідповідності, загальний зовнішній

вигляд та кольорова гамма були відтворені моделлю згідно з вхідним зображенням.

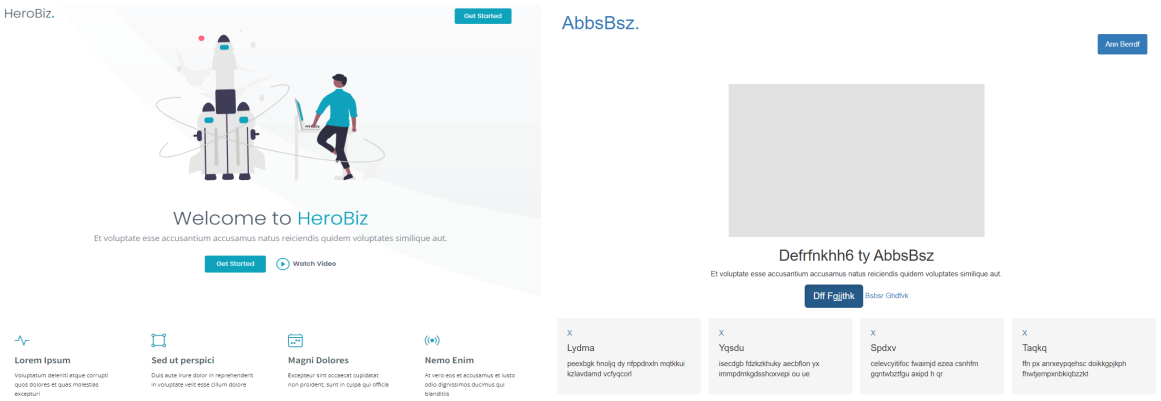

Рисунок 5 – Скриншот лендінг сторінки (ліворуч), результат генерації моделі (праворуч)

Пояснити таку точність можна недостатньою кількістю тренувальних даних, які були доступні для моделі. З наданою кількістю тренувальних екземплярів (80) модель може бути обмежена у своїй здатності до повного вивчення широкого спектра варіацій інтерфейсів та відповідного коду.

Крім того, модель має обмежений словник, тобто обмежений набір слів або символів, з яких вона може формувати сторінки. Це означає, що модель може бути обмежена у своїй здатності виражати різноманітність можливих структур та елементів інтерфейсу, оскільки вона обмежена вибором слів або символів для генерації вихідного коду. Враховуючи ці фактори, можна зрозуміти, чому точність моделі може бути нижчою.

Для покращення моделі pix2code можна замінити LSTM на вентильні рекурентні вузли (gated recurrent unit - GRU). У цій модифікації вектор ознак зображення передається на вхід GRU як вбудований вхід. Це дозволяє моделі захоплювати всю інформацію з вектора ознак під час генерації мовних токенів DSL.

Крім того, для підвищення якості моделі, можна використовувати різні енкодери, такі як VGG16, VGG19 та ResNet34, які допомагають отримати кращу якість векторів ознак зображення. Комбінація заміни LSTM на GRU та використання різних енкодерів може сприяти покращенню загальної продуктивності моделі pix2code.

3.3 Тестування на моделі HTML-Net

Архітектура моделі HTML-net, базується на комбінації згорткових нейронних мереж (CNN) та рекурентних нейронних мереж (RNN). Ця модель приймає вхідне зображення макета дизайну вебсторінки і використовує згорткові шари для розпізнавання візуальних ознак різних елементів інтерфейсу, таких як кнопки, текстові поля, заголовки тощо. Після цього вона використовує рекурентні шари для послідовного генерування HTML та CSS коду, який відповідає цим елементам.

Процес генерації відбувається шар за шаром, де кожен шар моделі відповідає певному аспекту структури вебсторінки. Наприклад, шар може генерувати заголовки, інший - текстові блоки, інший - фонові зображення. Шари RNN використовують контекст попередніх шарів для кращого моделювання послідовностей.

Модель була натренована на датасеті протягом 500 епох, а також окремі частини, наприклад CNN, передтреновані на датасеті ImageNet.[18] Точність такої моделі є 88 відсотків.

Для цілей власного тренування було підготовлено 80 екземплярів зображень вебсторінок та відповідного HTML-коду з дотриманням однакового порядку. Процес тренування проводився протягом 100 епох. На основі цього тренування була досягнута точність на рівні 75 відсотків.

На рисунку 6 представлено вхідне зображення поряд із результатом, згенерованим моделлю. Головною перевагою моделі є збереження тексту та правильне розташування верхніх кнопок. Проте, знову виникла проблема з генерацією логотипа, при цьому модель пропустила його. Невеликі піктограми були замінені елементами, на яких навчалася модель, а саме зображеннями чорного кольору розміром 32x32 пікселя.

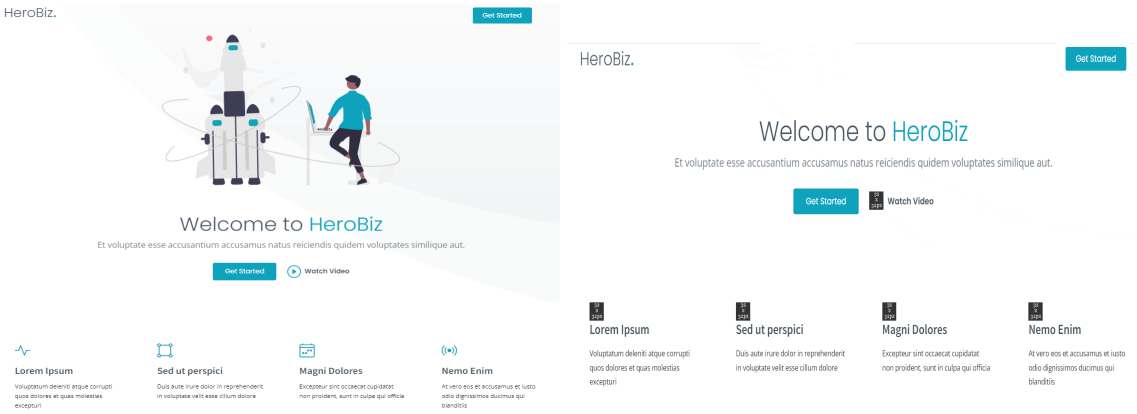

Рисунок 6. Скриншот лендінг сторінки (ліворуч), результат генерації моделі (праворуч)

Такий результат свідчить, що під час тестування, зображення мають незвичайну або складну структуру, яку модель не бачила під час тренування, тому вона може зіткнулася з проблемою відтворення цих елементів.

3.4 Тестування на моделі Sketch2Code

Для тестування було взято модель розроблену Робінсоном у 2019 році.[18] Модель отримує на вхід скан або фотографію паперового макету веб-сторінки. Вхідне зображення попередньо обробляється для видалення шуму, зміни масштабу, нормалізації тощо. За допомогою попередньо

натренерованої CNN, виявляє окремі блоки на зображенні, такі як заголовки, текстові блоки, кнопки, зображення тощо. На основі визначених блоків та їх взаємозв'язків система генерує вихідний код HTML та CSS, що описують структуру та вигляд веб-сторінки.

Модель Sketch2Code була попередньо натренерована на створеному датасеті, який складався з 1250 фотографій ескізів інтерфейсу та відповідного спрощеного HTML коду. Після тренування модель досягла 60% точності в класифікації елементів, 70% - генерації коду інтерфейсу.

Для забезпечення точності експерименту було здійснено перетворення датасету лендінг сторінок. Усі зображення було конвертовано в ескізи згідно з визначеними правилами. Під час тренування було використано 50 епох. Незважаючи на це, точність класифікації залишилася незмінною, але точність генерації знизилася до 63 відсотків. З цих результатів можна зробити припущення про різницю в характеристиках вхідного HTML-коду, що спричинило такий вплив, оскільки самі ескізи мали структуру, ідентичну до описаної в науковій статті.

На рисунку 7 зображений початковий скріншот (зображення у формі ескізу) поруч з результатом роботи моделі. Хоча дизайн не був збережений, оскільки модель працювала з чорно-білим прикладом, всі елементи присутні на отриманому результаті. Проте, виявлено помилку у розташуванні тексти "subscribe" та іконки поруч, які перекривають частину вище розташованого результату. Крім того, варто зазначити, що пропорції деяких елементів не були збережені.

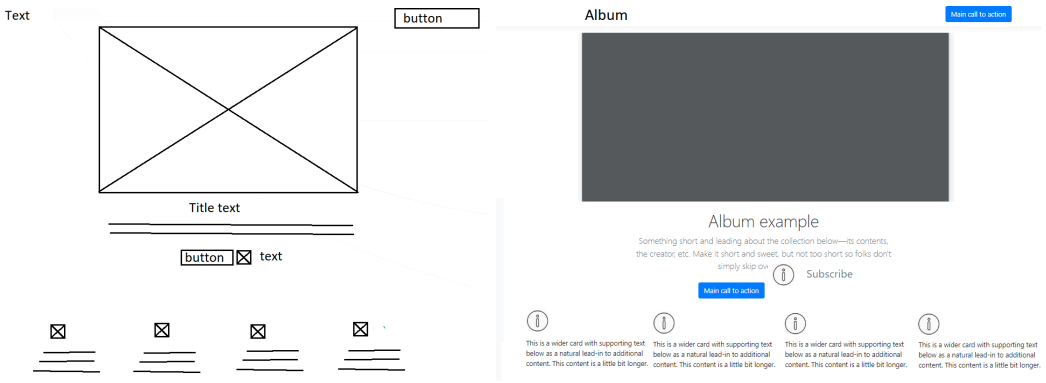

Рисунок 7 — Нарис лендінг сторінки (ліворуч), результат генерації моделі (праворуч).

Причиною таких результатів є те, що макети веб-сторінок можуть мати неоднозначності або недостатньо деталей, які ускладнюють правильне перетворення їх на код.

ВИСНОВКИ

В ході розробки проекту були розглянуті алгоритми на основі нейронних мереж для генерації інтерфейсів користувачів з високою якістю та ефективністю.

У процесі дослідження було покращено навички роботи з мовою програмування Python та бібліотеками для машинного навчання Numpy та TensorFlow, було поглиблено знання нейронних мереж.

Було проведено аналіз різних архітектур для генерації інтерфейсів та описано технології, що вони використовують, їх переваги та недоліки.

У результаті проведених досліджень на підготовленому датасеті лендінг сторінок і відповідних їм html сторінок були отримані наступні результати. Модель Pix2Code показала точність на рівні 60%, модель HTML-net показала точність на рівні 75%, модель Sketch2Code показала точність на рівні 63%. Хоча різні моделі для генерації коду з макетів сторінок мають різну точність, кожна з них має свої переваги. Результати були ретельно проаналізовані, і були запропоновані можливі шляхи вирішення проблем. Один з них - збільшення розміру датасету шляхом додавання додаткових прикладів.

Після виконання даної роботи була досягнута фінальна мета – проаналізовано моделі різної архітектури для задачі генерації лендінг сторінок.

Генерація інтерфейсів користувача за допомогою нейронних мереж є важливою задачею, яка відкриває перспективи для автоматизації та прискорення процесу розробки інтерфейсів. Дана робота надає базу для подальших досліджень та розробок у цій області. Розробники можуть використовувати цей огляд для розширення архітектур моделей, удосконалення алгоритмів та збільшення точності генерації лендинг сторінок за допомогою нейронних мереж.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Aparna Halbea, Dr. Abhijit R. Joshib, "A Novel Approach to HTML Page Creation Using Neural Network," *International Conference on Advanced Computing Technologies and Applications (ICACTA2015),* 2015 DOI:10.1016/j.procs.2015.03.122
2. Tony Beltramelli. “pix2code: Generating code from a graphical user interface screenshot” *In Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, pages 1–6, 2018.
3. Chunyang Chen, Ting Su1, Guozhu Meng, Zhenchang Xing, and Yang Liu, “From UI design image to GUI skeleton: a neural machine translator to bootstrap mobile GUI implementation” 2018, *ICSE '18: Proceedings of the 40th International Conference on Software Engineering* DOI:10.1145/3180155.3180240.
4. Daniel De Souza Baulé, Christiane Gresse von Wangenheim “Recent Progress in Automated Code Generation from GUI Images Using Machine Learning Techniques”, November 2020, *Journal Of Universal Computer Science*, DOI:10.3897/jucs.2020.058.
5. Mohamed Elgendy, *Deep Learning for Vision Systems [6 ed.]*. Reading, MA: Addison Wesley, 2012. [E-book] Available: Safari e-book.
6. Yao, X. (2020). Automatic GUI Code Generation with Deep Learning (Дисертація на здобуття наукового ступеня доктора філософії). Manchester Metropolitan University.
7. Zou, Z., Chen, K., Shi, Z., Guo, Y., Ye, J., "Object Detection in 20 Years: A Survey", IEEE Access, (2021), vol. 9, pp. 58684-58703.
8. Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S., “Feature Pyramid Networks for Object Detection”, *Proceedings of the*

IEEE Conference on Computer Vision and Pattern Recognition, 2117-2125. DOI: 10.1109/CVPR.2017.106

9. LeCun, Y., Bengio, Y., & Hinton, G., "Deep learning", *Nature*, 521(7553), 436-444. https://doi.org/10.1038/nature14539
10. McCulloch, W.S. & Pitts, W. (1943), "A logical calculus of the ideas immanent in nervous activity" *Bulletin of Mathematical Biophysics*, 5, 115-133.
11. Hinton, G. E. (2007). "Learning multiple layers of representation", *Trends in Cognitive Sciences*, 11, 428-434.
12. Langr, J. and Bok, V. "GANs in Action; Deep Learning With Generative Adversarial Networks", Manning Publications Co., 2018. ISBN: 978-1-61729-554-9.
13. Narasimhan, M., Lazebnik, S., & Schwing, A. G., "Out of the box: Reasoning with graph convolution nets for factual visual question answering", arXiv preprint arXiv:1811.00538.
14. Cui, Z., Henrickson, K., Ke, R., & Wang, Y, "Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting", IEEE Transactions on Intelligent Transportation Systems, 21(11), 4883-4894.
15. Yang, J. Lu, J. Lee, S. Batra, D. Parikh, "Graph R-CNN for Scene Graph Generation" *In Proceedings of the European Conference on Computer Vision (ECCV)*, pages 670-685.
16. J. Johnson, A. Gupta, Li Fei-Fei. "Image generation from scene graphs", *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1219–1228, 2018.
17. A Robinson, "Sketch2code: Generating a website from a paper mockup". Dissertation, University of Bristol, 2019.
18. ImageNet: a Large-Scale Hierarchical Image Database. June 2009 DOI: 10.1109/CVPR.2009.5206848.