Міністерство освіти й науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
«КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

Створення ML системи генерування стилізованих описів до зображень у Swift Package Manager

Текстова частина до кваліфікаційної роботи

за спеціальністю 122 «Комп'ютерні науки»

Керівник кваліфікаційної роботи

ст.в. Смиш Олег Русланович

_____ (підпис

Виконала студентка 2 курсу

Сидорова Єлізавєта Олександрівна

«____»_____2026 р.

Київ 2026

АНОТАЦІЯ

Проблема автоматичного генерування стилізованих підписів до зображень залишається актуальною через питання стилістичної, контекстної й семантичної гнучкості, оскільки важливо відокремити змістову та жанрову компоненти для узагальненого застосування стилів через інструменти пам’яті. Важливою тенденцією розробляння є придатний і відкритий фреймворк через питання інтеграції комплексів у мобільне середовище та дочірні обмеження через обчислювальні затрати пристроїв. Актуальність дослідження зумовлена стрімким розвитком соціальних мереж, цифрового контенту та потребою в автоматизації створення текстових описів для фотографій, що здатні адаптуватися до різних стилів подання інформації.

Кваліфікаційну працю присвячено дослідженню завдання автоматизованого генерування та стилізації текстових підписів до зображень із застосуванням сучасних методів комп’ютерного зору та оброблення природної мови.

Уперше реалізовано інтегровану систему автоматичного генерування стилізованих текстових підписів до зображень, яка поєднує мультимодальну трансформерну модель створення описів із послідовною структурою стилізації тексту, розроблення комбінованої збірки синтетичних знань й інтегрування програмної архітектури з підтриманням API та SPM-модуля.

У тексті кваліфікаційного дослідження описано реалізацію комплексного інструмента з користувацьким інтерфейсом, який дає змогу отримувати стилізовані підписи до вхідних фото за застосування SPM та розробленого конвеєра. Окрім цього, розглянуто докладну реалізацію мультимодальної програми автоматичного генерування підписів до зображень на підставі попередньо навченої трансформерної архітектури BLIP. Додатково для завдання стилізації тексту застосовано модель T5 типу «послідовність-у-послідовність». Для навчання програмної структури стилізації сформовано власний синтетичний набір даних, що докладно описано під час дослідження. Так само

реалізовано API підтримку для доступу до серверної підтримки конвеєра та Swift Package для інтеграції в iOS-проєкти.

Проведено та описано докладний огляд сучасних підходів до завдання створення підписів до зображень та стилізації, обґрунтовано вибір архітектур програмних засобів, сформовано власний набір жанрових підписів, реалізовано програмний концепт та виконано його експериментальне дослідження. Наголошено на значущості створення інструментарію для автоматизованого одержання стилізованих підписів до фото.

Під час експериментальних досліджень проведено тестування комплексів із послугуванням метрик, які підтвердили здатність інструменту формувати семантично коректні, синтаксично узгоджені та стилістично адаптовані текстові підписи.

Практичне значення роботи полягає в створенні готового програмного рішення, що охоплює серверну архітектуру для оброблення запитів, API для взаємодії з клієнтськими застосунками та Swift Package модуля для інтеграції реалізації в мобільні застосунки на платформі iOS. Розроблена структура впроваджується в галузі соціальних мережах, пов’язаних із генеруванням текстового супроводу до зображень.

Далі описано структуру кваліфікаційної роботи. У розділі 1 проведено експертизу релевантних інструментів, методів й архітектур, що охоплює поточний стан реалізованих відкритих рішень, які наявні до моменту написання кваліфікаційної праці. Огляд та висвітлення переваг дали змогу реалізувати створену комбіновану систему. У розділі 2 описано поступову реалізацію компонентів конвеєра, а саме модель завдання створення описів до фото, модель стилізації та поєднання їхнього виконання в один комбінований інструмент. Додатково описано створення власного набору синтетичних даних, налаштування ґостинг-рішення та архітектуру Swift модуля. У розділі 3 описано результати проведених досліджень та протестовано створений механізм за обраними метриками. Додатково порівняно результати тренування підсистем на різних параметрах.

Ключові слова: генерування підписів до зображень, стилізація тексту, мультимодальні моделі, трансформери, оброблення природної мови, мобільні застосунки.

Зміст

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

Міністерство освіти й науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ

Зав.кафедри мультимедійних систем,

к.ф.-м.н., доц. Жежерун О. П.

_____

(підпис)

«___» _____ 2025 р

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на кваліфікаційну роботу

студентці 2-го курсу, факультету інформатики Сидоровій Єлізавєті

Тема: Створення ML системи генерування стилізованих описів до зображень у Swift Package Manager

Зміст ТЧ до кваліфікаційної роботи:

Зміст

Вступ

1.Огляд сучасних рішень та літератури

2.Проєктування та розроблення системи

3.Тестування і апробація розробленої системи

Висновок

Список використаної літератури

Дата видачі «____»_____ 2025 р.

Завдання отримала _____

(підпис)

Календарний план виконання дипломної роботи

Тема: Створення ML системи генерування стилізованих описів до зображень у Swift Package Manager:

№	Назва етапу курсового проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на дипломну роботу	від 1 вересня до 1 жовтня	
2.	Аналіз матеріалів за темою дипломної роботи	від 5 вересня до 30 січня	
3.	Розробка та реалізація алгоритму створення системи	від 5 вересня до 10 березня	
4.	Написання текстової частини дипломної роботи	від 5 вересня до 10 травня	
5.	Надання дипломної роботи на перевірку керівником	від 15 квітня до 10 травня	
6.	Коригування на основі результатів перевірки	від 15 квітня до 20 травня	
7.	Остаточне оформлення роботи та презентації	від 30 квітня до 15 травня	
8.	Захист дипломної роботи	від 8 до 9 червня	

Сидорова Є.О. _____

Смиш О.Р. _____

«____» _____ 2025 р.

Вступ

Обґрунтування вибору теми дослідження

Сучасний цифровий світ наповнений медіаконтентом: соціальні мережі, електронна комерція, мобільні застосунки, реклама та навчальні процеси — усе це цілком залежить від візуального подання інформації. Підтримання текстового супроводу (caption) виконує описову функцію з можливістю кращого розуміння комунікативного наміру автора, його настрою, стилю й змісту загалом. З огляду на це автоматичне генерування заголовків і підписів до зображень є важливим напрямом розвитку мультимодальних систем оброблення інформації.

Розвиток нейронних мереж, мовно-візуальних засобів і глибинного навчання спрощує процес автоматичного анотування до зображень. Проте теперішні моделі з відкритим кодом здебільшого адаптовані саме на створення тексту без персоналізації необхідної стилістики, що може слугувати некоректним або не доцільним інструментом для окремих прикладних завдань. Через такі обмеження знижується їхня придатність у завданнях із жанровим контекстом. Отже, актуальним викликом для процесу еволюції моделей наразі залишається персоналізовані анотування (Personalized Captioning) [1], які є прикладом застосування мультимодальних нейронних концептів для створення інтерпретованих адаптацій до заданого цільового сегмента.

Важливим аспектом є інтеграція комплексів такого змісту в мобільні застосунки. Середовище для об’єднання споживачів із цифровим контентом має брати адаптивні механізми для впровадження доступу до ML-продуктів. Віддалені серверні рішення не завжди дають змогу гарантувати конфіденційність і швидкість оброблення вхідних запитів, тоді як користування фреймворків представляє набір інструментів, які допомагають уникнути залежності з хмарними технологіями та зменшують обмеження впровадження подібної архітектури. [2]

У підсумку, актуальність теми визначається потребою в створенні програмного рішення, яке забезпечує автоматичне генерування стилізованих

текстових підписів до зображень і гарантує інтегроване в мобільні застосунки без сторонньої хмарного опрацювання наданої інформації.

Мета й завдання дослідження

Основною метою цієї діяльності є автоматизація процесу генерування стилізованих описів до зображень згідно з наданим визначеним жанром для практичного застосування в мобільних пристроях у різних предметних галузях.

Для досягнення поставленої мети виконано такі завдання:

1. Проведено аналіз сучасних підходів для генерування підписів із застосуванням мультимодальних моделей.
2. Досліджено архітектури комплексів із відкритим кодом і окреслено методи їхнього налаштування (fine-tuning) для створення анотацій.
3. Створено власний набір синтетичних даних для навчання моделі з урахуванням стилів (формальний, емоційний, смішний, інстаграм-стиль тощо).
4. Реалізовано, натреновано та проведено тестування мультимодальної системи автоматичного генерування підписів до зображень із параметром стилю.
5. Забезпечено автономне власне гостинґ-рішення для структури.
6. Розроблено Swift-модуль (Swift Package), що забезпечує локальну інтеграцію моделі в мобільні застосунки.
7. Проведено експериментальне тестування моделі за метриками та оцінено якість стилізації.

Об’єктом дослідження кваліфікаційної роботи є процес автоматичного генерування стилізованих підписів до візуального медіаконтенту. Предметами експерименту є методи машинного навчання ій архітектурні рішення для реалізації в середовищі Swift Package Manager.

Методи дослідження

Методами дослідження є експертиза і синтез, моделювання, машинне навчання, процес налаштування (fine-tuning) для моделей, експериментальне тестування, математичні методи обчислення метрик якості та їхній розбір.

Практичне значення отриманих результатів

У результаті наукової діяльності запропоновано фреймворк, написаний мовою програмування Swift, для можливості інтегрування зв’язку між моделлю та iOS-застосунками в адаптивний спосіб. Отримані результати підтверджують доцільність через впровадження в мобільні застосунки, які потребують автоматичного генерування описів, стилізованих під потреби клієнта з метою підвищення рівня персоналізації для контентних галузей застосування, освітніх процесах і маркетингу.

Уперше реалізовано інтегровану систему автоматичного генерування стилізованих текстових підписів до зображень, яка поєднує мультимодальну трансформерну модель створення описів із послідовною моделлю стилізації тексту, розроблення комбінованої збірки синтетичних даних для жанрування, а також інтегрування програмної архітектури з підтриманням API та SPM-модуля. Це створює передумови для подальшого розвитку модульних ML-рішень у мобільних екосистемах та розширює практичні можливості застосування машинного навчання в iOS-застосунках.

Особистий внесок здобувача

Усі результати дослідження отримано самостійно авторкою. Авторкою розроблено архітектуру системи, виконано fine-tuning мультимодельного конвеєра, реалізовано інтеграцію через Swift Package, проведено експериментальне тестування та підготовлено аналітичну частину. Науковий керівник здійснював наукове консультування та перевірку методологічних рішень.

Структура та обсяг кваліфікаційної роботи

Кваліфікаційна робота складається зі вступу, трьох основних розділів, висновків до розділів, загальних висновків роботи, вступу і використаних джерел. Загальний обсяг кваліфікаційної становить 68 сторінок.

Робота налічує 29 рисунків. Список використаних джерел містить 21 найменування.

1. Огляд сучасних рішень і літератури

1.1. Проблематика та актуальність дослідження

Проблема автоматичного генерування стилізованих підписів до зображень залишається актуальною через питання стилістичної, контекстної і семантичної гнучкості. Вона є одним зі значущих мультимодальних завдань штучного інтелекту через візуальний канал опрацювання, мовне моделювання і семантичне узгодження. Багато сучасних систем із трансформерами стикаються з викликами семантичного розриву між медіа та текстовими форматами для абстрактного або емоційного аспекту; обмеженнями узагальненості для нових або рідкісних об'єктів; недостатньою інтерпретованістю і контролем стилістики; бракуванням метрик для оцінювання на рівні сприйняття людиною і надмірною залежністю для роботи з неповними й розрізненими даними. [1]

Для генерування стилізованих підписів важливо відокремити змістову та стилістичну компоненти з використанням інструментів пам'яті для узагальненого застосування стилів. Наприклад, модель MemCap долає обмеження, використовуючи не парні дані корпусу, алгоритм для автоматичного розподілу на стилістичну й контентну частини, має модуль пам'яті для зберігання ключових фраз позначення певного стилю і використовує підхід self-critical reinforcement learning для вираховування стильової точності та зв'язності тексту. [3]

Одним із найбільш актуальних питань щодо інтеграції моделей глибинного навчання в мобільне середовище, наприклад, iOS розроблення є обмеження через великі обчислювальні затрати, тобто GPU-кластери або хмарні платформи. Великі моделі потребуватимуть ресурсів пам'яті й енергоспоживання на мобільних пристроях, а процес стискання моделей не завжди є доступним або придатним для кінцевих вимог. Згідно з високою обчислювальною вартістю тренування і використання, складнощами оптимізації і проблемою розповсюдження проблема стає дотичною до вибору локальної та мережної інтеграції глибинних моделей, тобто проєктування і розробка власного

фреймворку має бути адаптованою до технічної сумісності з високим рівнем персоналізації та енергозаощадження. [4]

Отже, важливою тенденцією розробки є ресурсоефективний, компактний, відкритий і енергоефективний фреймворк на основі моделі з адаптивним підходом для навчання і чітким розподілом на певні етапи.

1.2. Огляд наявних підходів до генерування підписів до зображень

Генерування підписів до зображень є мультимодальним завданням для комп'ютерного зору й обробки природної мови. Розвиток і досягнення в цих галузях сприяють удосконаленню отриманих результатів і зменшенню складнощів власного впровадження нових архітектур. Далі розглянуто сучасні наявні підходи для генерування, їхні переваги й недоліки.

Класичною моделлю без трансформерів є *кодувальник-декодувальник*, де першим є глибока згорткова нейронна мережа (CNN), яка виділяє візуальні ознаки, а другим — рекурентна нейронна мережа (RNN) часто в поєднанні з LSTM для послідовного генерування підписів. Роль CNN полягає у виділенні локальних і глобальних просторових ознак, а також у формуванні feature map — описі змісту зображення на високому рівні абстракції. Для архітектури частим представником є моделі ResNet з механізмом residual learning. Схему роботи кодувальника зображено на рисунку 1.2.1, де основною стратегією є поступові кроки: видалення останнього повнозв'язного шару, вихід CNN через лінійний шар для модифікацій відповідно до вимог вбудованого простору ознак і застосування векторизації даних до отриманого результату. [5]

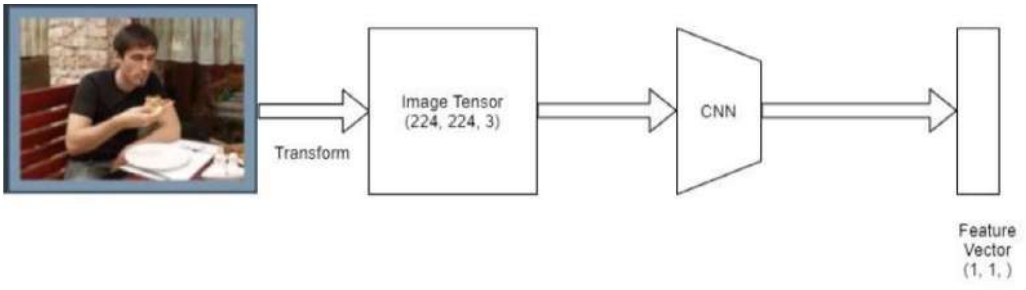

Рисунок 1.2.1 Схема роботи кодувальника CNN [5]

Після цього декодувальник RNN оперує наданою йому послідовністю, зберігаючи стан попередніх кроків. Це забезпечує накопиченню контексту в часі за допомогою маркування залежностей між словами під час опису зображення, чого не можна досягти в CNN. Проте при великих послідовностях градієнт прямує до нуля, що унеможливлює зберігання довготривалого контексту. Це породжує проблему vanishing gradients, і для її вирішення пропонується використання механізму LSTM. Він допомагає додавати керовані елементи Forget Gate, Learn Gate, Remember Gate та Use Gate, які в сукупності дають змогу мережі селективно працювати з інформацією для управління з метою видалення непотрібних даних і збереженням важливих, а також для формування граматично правильних послідовностей. Схему роботи механізму LSTM зображено на рисунку 1.2.2. [5]

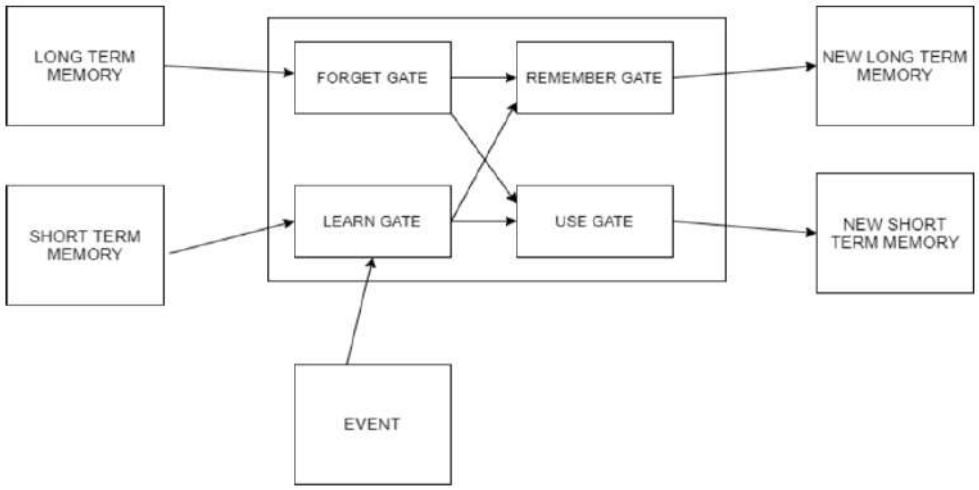

Рисунок 1.2.2 Схема роботи кодувальника CNN [5]

Отже, процес генерування описів для зображень в архітектурі кодувальник-декодувальник складається з кількох послідовних етапів. Спочатку вхідне зображення проходить CNN-оброблення для отримання вихідного вектору ознак за допомогою послідовного застосування фільтрів. Далі результат його роботи передається до RNN-LSTM архітектури, де поступово формується текстовий опис із прогнозуванням наступного слова на основі контексту

попередніх станів. Вибір найкращого токена здійснюється після застосування softmax-активації (перетворення вихідного вектора моделі в імовірнісний розподіл) на вихідному шарі.

Надалі формується словник із набору підписів до зображень і здійснюється оптимізація параметрів. На завершальному етапі модель генерує послідовність індексів словника, яка перетворюється в повноцінне речення природною мовою. [5]

Перевагами цього підходу є прозорість, модульність, стійкість до проблеми згасання градієнта й ефективне використання просторових та послідовних властивостей даних. Серед недоліків можна виокремити відсутність механізму уваги, обмежену здатність для моделювання складних залежностей і високу залежність від словника й попередньої токенізації.

Наступним підходом для автоматичного генерування підписів до зображень є *механізм уваги (attention-based models)*, який розв'язує проблему обмеженого використання одного глобального вектора ознак у CNN–RNN моделей. На відміну від попередньої архітектури, цей метод дає змогу динамічно фокусуватися на релевантних сегментованих даних зображення під час формування кожного слова. Тобто метод забезпечує можливість спеціалізованого оброблення та відбирання інформації на низькорівневих і високорівневих ознаках для підвищення контекстуальної точності та ступеня змістового охоплення вихідного опису до фото. [6]

Принцип роботи полягає в отриманні багатовимірного вектора ознак від кодувальника, а на етапі декодування впроваджуються ваги для кожної ділянки зображення згідно з поточним станом LSTM або GRU (аналогічний підхід для довгострокового збереження контексту). Механізм уваги отримує набір параметрів від CNN і порівнює їх з поточним прихованим станом декодера. На основі цього аналізу обчислюється коефіцієнт важливості (attention weights), тобто відбувається відбір найсуттєвіших фрагментів вхідної інформації для поточного кроку генерування. Далі формується контекстний вектор, що враховує їхню важливість, і він подається на вхід наступного етапу декодера,

забезпечуючи точніше та змістовне формулювання результату. Схему роботи моделі типу кодувальника-декодувальника з використанням механізму уваги подано на рисунку 1.2.3. [6]

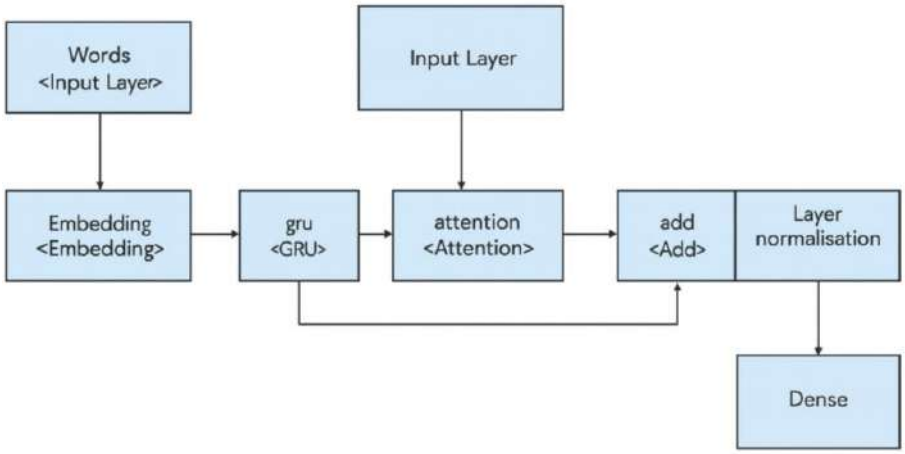

Рисунок 1.2.3 Схема роботи моделі типу кодувальника-декодувальника з використанням механізму уваги [6]

Перевагами цього підходу є динамічне оброблення найінформативніших частин зображення під час формування кожного слова, гнучкіша та універсальна архітектура в порівнянні з минулим способом і покращена здатність до моделювання довгих текстових залежностей. Недоліками є необхідність високоякісних ознак кодувальника, підвищена чутливість до динаміки навчання і ризик помилок фокусування.

Провідним методом для генерування описів до фото в сучасних системах є *Vision Transformer (ViT)*. На відміну від попередників, він використовує глибший механізм уваги — multi-head self-attention. Цей підхід дає змогу моделі аналізувати глобальні відношення між частинами зображення на ранніх етапах опрацювання та досліджувати локальні та глобальні закономірності паралельно. [7]

Vision Transformer обробляє зображення, поділяючи його на фіксовані фрагменти для конвертації в лінійні вектори, і додає до них позиційні набори ознак. Відповідно до отриманих структур, трансформер застосовує

багатошаровий блок механізму самоуваги, який моделює взаємозв’язки між усіма візуальними компонентами. Використання проєкційного шару для перенесення візуального представлення в текстовий простір допомагає отримати візуальний префікс для можливого мультимодального узгодження без необхідності складних механізмів. Отже, такий метод дає змогу зменшити кількість параметрів, покращити стабільність навчання та уникнути перевантаження архітектури. Недоліками таких систем можуть стати: обмеження глибини зв’язків між модальностями, менша точність у складних сценаріях і значна залежність від проєкційного шару. [7]

Мультимодальні великі моделі належать до класу нейронних систем, призначених для оброблення, інтегрування та узгодження різнорідних типів даних. З архітектурного погляду, вони побудовані таким чином, щоб єдиний трансформерний стек або узгоджений набір модулів міг навчатися одночасно на кількох модальностях, забезпечуючи спільний простір подання для текстової та візуальної інформації. Тобто завдання генерування текстових підписів до зображень є одним з основних для цього підходу. [8]

Головними компонентами є візуальний кодувальник й універсальний декодувальник. Першим елементом є ViT або масштабовані варіанти базових архітектур, які аналогічні з попередніми дослідженими методами. Для другого компонента використовують декодувальники великих мовних моделей (LLM-decoders), двонапрямні текстові кодувальники (BERT-подібні) та уніфіковані трансформери (один стек для спільних представлень). Для узгодження модальностей застосовують проєкційні шари, механізм перехресної уваги та архітектуру уніфікованих мультимодальних трансформерів. Проєкційні шари забезпечують перехід між просторами ознак різних модальностей, механізм cross-attention дає змогу кожному токену знаходити релевантні ділянки у візуальних ознаках, а уніфіковані трансформери оперують спільною послідовністю візуальних і текстових елементів, забезпечуючи їх інтегроване представлення. [8]

У сучасних дослідженнях виокремлюють три основні підходи до

генерування підписів у межах мультимодальних великих мовних моделей: пряме автоматичне генеративне моделювання (модель отримує візуальні токени та генерує текст, використовуючи LLM-компонент як декодер), інструкція для створення підписів (модель формує підпис на основі явно заданих інструкцій) і складні когнітивні операції (пояснення причинно-наслідкових зв'язків на зображенні). [8]

Перевагами дослідженого підходу є універсальність, бо одна модель здатна вирішувати багато мультимодальних задач, масштабованість і здатність до інструкційного виконання. Недоліками є потреба у спеціалізованих наборах даних, високій обчислювальній складності і труднощі з інтерпретацією.

Підсумовуючи, аналіз сучасних методів генерування підписів до зображень демонструє різноманіття підходів та концепцій, що відрізняються масштабом, складністю та принципами узгодження модулів у межах систем. Рівень адаптивності, універсальності та семантичної узгодженості таких рішень свідчить про помітну еволюцію — від базових архітектур до інтелектуальних систем нового покоління, здатних забезпечувати глибоке розуміння та інтерпретацію візуального контенту.

1.3. Огляд сучасних моделей і підходів до текстової стилізації

Імпліцитно є змога розділити термін емоційна оцінка (sentiment), стильовий тон (tone) та тон (emotional tone). Поняття емоційного забарвлення тексту можливо трактувати ширше за полярність емоційного тону (sentiment polarity) та звичайних емоцій через загальний неявний або контекстний настрій. Отже, стилістику розглядають як семантико-прагматичну властивість інформації, яка доповнює лексичний і синтаксичний розбір. [9]

Трансформація текстового стилю (Text Style Transfer) — це завдання оброблення природної мови для зміни стилістичних властивостей інформації за умови збереження семантичного змісту. Емоційне перетворення (Emotion transfer) — процес зміни тексту з однією емоційною домінантою на іншу зі збереженням семантичного ядра інформаційної одиниці. Складність завдання

модифікації стилістичних характеристик тексту зумовлена браком паралельних корпусів, тобто наявністю пар речень «вхід-вихід» з однаковим змістом, але різним стилем. Внаслідок цього переважна частина сучасних систем орієнтовані на підхід навчання без учителя (unsupervised learning). Тобто завдання адаптування тексту до стильових вимог наразі є одними з актуальних напрямів розвитку інтелектуальних структур. [9]

Базові методи текстової стилізації систематизуються на рівні речень (sentence-level methods) та слів (word-level methods). Перший тип застосовує декомпозицію латентних представлень для повної генерації нового тексту цільового стилю. У межах цього алгоритму модель намагається відокремити стилістичне забарвлення від змісту. Після цього контекстний вектор комбінується з новим стилістичним атрибутом для генерування переформовуваного речення. Зазвичай підхід на підставі речень часто супроводжується інтеграцією варіаційних автоенкодерів (variational autoencoders), трансформерів або латентних змінних із регуляризацією через дискримінатори стилю (style discriminators). Основним недоліком є проблема збереження змісту (content preservation problem), коли генератор має надлишкову гнучкість або нестачу обмежень у побудові тексту, що нерідко призводить до втрати семантичної точності або появи несумісних слів. [10]

Підхід на рівні слів і фрагментів має подолати проблему втрати змісту через маркування стилістично та структурно важливих компонентів. Стильова основа визначається евристиками або статистичними критеріями, після чого дочірні слова видаляються, маскуються або змінюються для трансформації. Такий метод дає змогу значно краще зберігати семантичний зміст тексту, оскільки більша частина вихідного речення залишається незмінною. У поєднанні з не авторегресивними моделями (non-autoregressive models) помітне підвищення ефективності генерації. Основним недоліком є залежність від евристичного визначення стилістичних слів, що може спричинити нестабільність у різних доменах і, відповідно, зменшувати різноманітність згенерованих текстів. [10]

Класичним представником напряму стилізації на рівні слів є модель Delete–Retrieve–Generate (DRG), структура якої представлена на рисунку 1.3.1. Алгоритм передбачає, що стилістично релевантні слова визначаються за допомогою статистики їхніх частот (word frequency statistics) та класифікаторів. Після етапу видалення провідних елементів на їхнє місце вставляються слова цільового стилю, отримані з корпусу прикладів. Подальші дослідження удосконалили цей підхід через інтеграцію механізму самоуваги (self-attention) та замаскованих мовних моделей (masked language models) для природнішого відновлення тексту. [11]

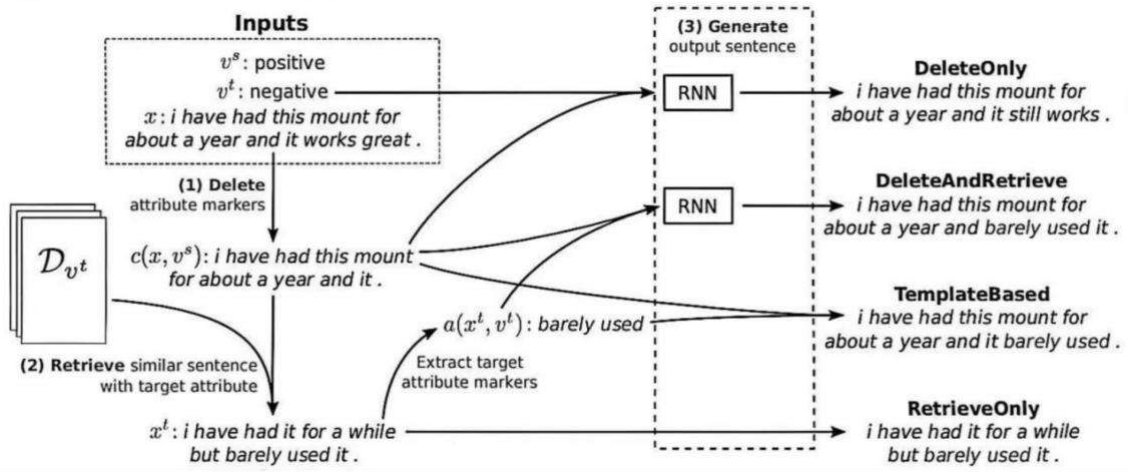

Рисунок 1.3.1 Схема DRG (The Delete-Retrieve-Generate) моделі [11]

Альтернативним напрямом є підходи, які не розділяють стиль зі змістом у явний спосіб. Стилістичний перехід досягається через редагування латентного представлення або через зовнішні трансформації. Одним із таких методів є перехресно узгоджені автоенкодери (cross-aligned autoencoders) та зворотний переклад (back-translation). На їхньому принципі будуються методи ітеративного зіставлення та перекладу (iterative matching and translation), які поступово формують корпуси для навчання моделей стилізації. Методи за допомогою навчання з підкріпленням (reinforcement learning) допомагають у розвитку процесу оптимізації моделі одночасно за двома винагородами: відповідність цільовому стилю та збереження змісту. Використання навчання з підкріпленням

дає змогу моделі водночас збалансовувати стильову точність і змістову узгодженість, оптимізуючи процес генерації через подвійний сигнал винагороди. Основним недоліком цих методів є інтерпретація, оскільки стиль і зміст залишаються змішаними в латентному просторі, й так само ресурсна затратність. [11]

Підходи на основі абзацу або довгого тексту розглядають стиль як глобальну характеристику тексту, а не як суму локальних змін. Стилістику є змога розглядати як багаторівневий компонент, який визначають носії стилю, що змінюються залежно від масштабу тексту: від емоційної полярності та словникового вибору — на рівні слів, до синтаксичних і композиційних особливостей — на рівні речень і абзаців. Головними обмеженнями такого підходу стилізації є суттєва складність моделей, значні обчислювальні витрати та контроль збереження змісту на довгих відрізках тексту. Окрім того, таке теоретичне розмежування ускладнює побудову універсальних моделей і вимагає окремих архітектур для кожного рівня текстової гранулярности. [12]

Підсумовуючи, слід зазначити, що текстова стилізація є актуальним напрямом досліджень у галузі NLP, що поєднує прагнення до гнучкої персоналізації мовлення та збереження семантичної цілісності тексту. Аналіз наукових підходів свідчить, що ефективне розв’язання цього завдання можливе лише за умови глибокого моделювання взаємодії між рівнями текстової структури.

1.4. Використання мультимодальних моделей у завданнях оброблення зображень та тексту

Мультимодальні великі мовні моделі призначені для спільного оброблення візуальної та текстової інформації з метою виконання завдань з аналізування, інтерпретації та генерування. Звичний метод розділяють на такі етапи: зображення трансформуються у візуальні представлення (visual representations), а потім поєднуються з текстовими токенами в спільному просторі ознак, що дає змогу моделі застосувати перехресно-модальне логічне

міркування (cross-modal reasoning). Таким способом, типова архітектура охоплює візуальний енкодер (vision encoder), модуль з'єднання та велику мовну модель. Попри можливість повторного застосування високопродуктивних моделей однієї модальності, цей підхід обмежує глибину взаємодії між компонентами через наявні архітектурні обмеження. Таксономію візуального розуміння в мультимодальних великих мовних моделях представлено на рисунку 1.4.1 [13]

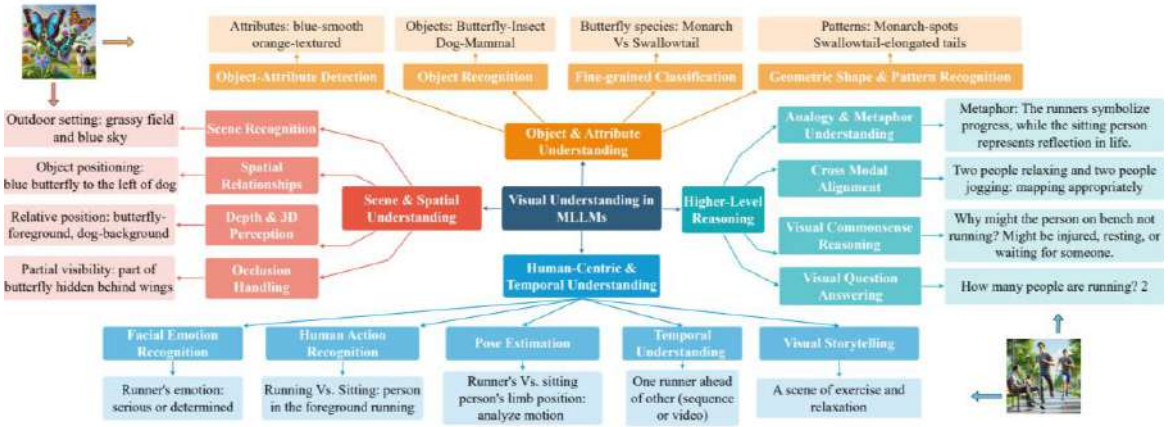

Рисунок 1.4.1 Концептуальне візуальне представлення в мультимодальних великих мовних моделях [13]

Візуальний кодувальник виконує роль перетворення сирих піксельних даних зображення у векторне представлення завдяки трансформерам зору (Vision Transformers, ViT), натренованих у рамках контрастивного навчання зображення-текст (CLIP). Візуальний кодувальник перетворює піксельні показники зображення у векторні ознаки, застосовуючи архітектуру трансформера зору (Vision Transformer ViT), навчену в процесі контрастивного попереднього навчання CLIP, що узгоджує простори зображення та тексту. Для подолання невідповідності форматів просторів ознак використовується модуль трансформації візуальних ознак у простір LLM. Після цього візуальні токени об'єднуються з текстовими та подаються на вхід мовній моделі для виконання кінцевого завдання. [13]

Поєднання візуального та мовного представлень є вагомим етапом у

завданнях, що передбачають спільне оброблення зображень і тексту. Зазвичай у сучасних системах використовується так зване пізнє злиття (late fusion), коли візуальні токени додаються до текстових уже після первинного кодування обох модальностей. Внаслідок цього комплексу мовна модель трактує візуальні токени як текстові елементи, що позбавляє її можливості враховувати просторові, геометричні та піксельні особливості зображення. Отже, візуальна інформація не впливає на внутрішні мовні міркування в достатньому обсязі, а використовується поверхово або ігнорується. Це є однією з головних причин низької якості візуального розуміння в мультимодальних системах. [13]

Підсумовуючи результати аналізу, можна стверджувати, що мультимодальні системи мають значний потенціал для подальшого розвитку, однак залишаються структурно нерівноважними через відмінності в способах представлення та узгодження модальностей. Вони демонструють істотну ефективність у завданнях семантичного міркування, генерації текстових відповідей та контекстного розуміння, проте суттєво поступаються у випадках, коли необхідне докладне візуальне сприйняття.

1.5. Використання ML-систем на платформі iOS

Розрізняють два основні підходи для інтеграції ML-систем у мобільні застосунки: хмарно-орієнтований (cloud-based) та орієнтований на пристрій (on-device). У першому оброблення інформації здійснюється на віддалених серверах, тоді як у другому — модель виконує обчислення на мобільному пристрої користувача. Наразі через розвиток технічних характеристик пристроїв і зростанням їхніх обчислювальних можливостей, підхід орієнтований на пристрій є найпоширенішим. Зменшення затримки є ще одним аргументом на користь локального виконання моделей. Оскільки виконання обчислень відбуваються на пристрої, усувається мережева залежність і забезпечується час відповіді на рівні від 120 до 150 мс. Підхід орієнтований на пристрій також дає безпечнішу форму оброблення персональних і чутливих даних. Оптимізовані моделі досягають вищих показників задоволеності споживачів, довіри до

конфіденційності та точності рекомендацій (89 %, 92 % і 91 % відповідно), що підкреслює важливість комплексної оптимізації технічних і користувацьких параметрів. [14]

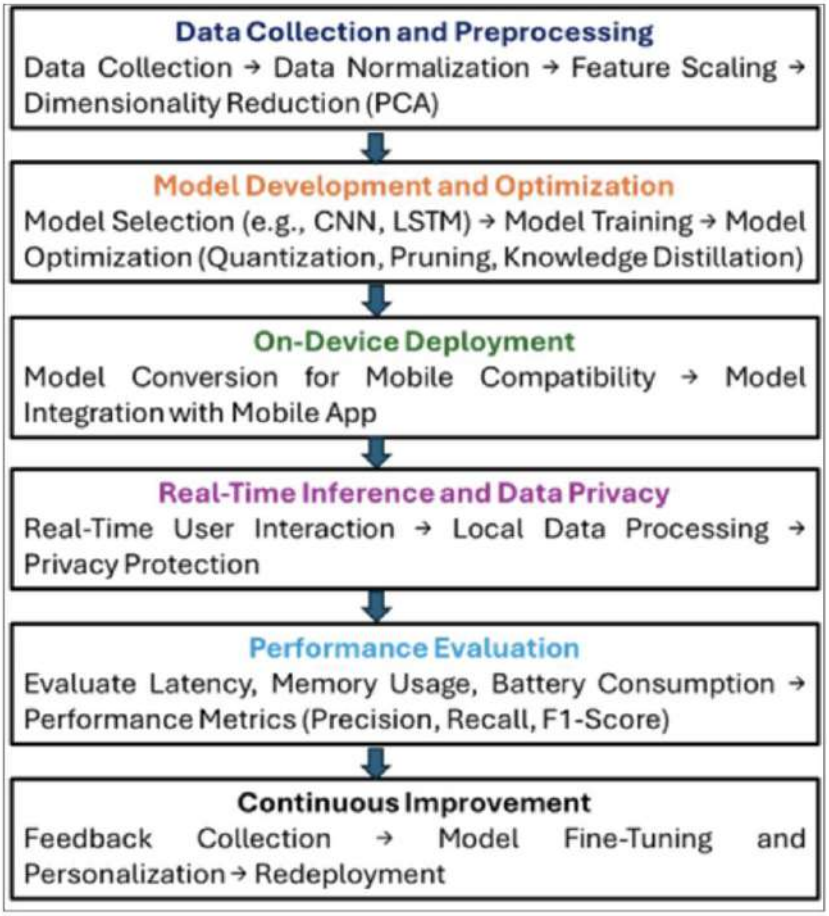

Рисунок 1.5.1 Концептуальне візуальне представлення в мультимодальних великих мовних моделях[14]

На рисунку 1.5.1 продемонстровано поетапну архітектуру мобільної ML-системи, яка охоплює весь життєвий цикл моделі — від збору даних до вдосконалення й оновлення. Першим етапом є збір і підготовка даних, яка охоплює нормалізацію, масштабування ознак і зменшення розмірності. На етапі розроблення моделі здійснюється вибір архітектури відповідно до типу даних. Перед інтеграцією в мобільний застосунок модель проходить етап оптимізації, що передбачає квантизацію, проріджування та дистиляцію знань (knowledge distillation) з метою підвищення ефективності та зменшення ресурсоспоживання

під час виконання на пристрої. Оптимізована модель інтегрується в мобільний застосунок і використовується локально, передбачаючи оцінювання затримки, використання пам’яті, енергоспоживання та точності роботи моделі. Останнім етапом архітектури є збір зворотного зв’язку та подальше вдосконалення моделі. Такий цикл дає змогу підтримувати актуальність системи та її адаптивність до змін відповідно до поведінки користувачів. [14]

Персоналізований підхід є адаптивною системою, що функціонує протягом усього життєвого циклу використання застосунку, і принципово відрізняється від класичних ML-рішень, у яких користувач взаємодіє з фіксованою, наперед натренованою моделлю. Локальне додаткове навчання моделей на iOS пристроях розширює традиційні підходи за допомогою використання 8-бітного трансферного навчання (8-bit transfer learning). Воно передбачає квантування параметрів, часткове донавчання моделі та скорочення обчислювальної складності до рівня, сумісного з можливостями мобільних процесорів. [16]

Використання ML-систем у мобільних застосунках на платформі iOS потребує глибокої архітектурної інтеграції з життєвим циклом застосунку та системними механізмами операційної системи. Локальні моделі, на відміну від серверних, функціонують у середовищі з обмеженими ресурсами, де енергоспоживання, фонові процеси та управління пам’яттю мають критичне значення. Використання on-device ML у системі iOS дає змогу створювати повністю автономні від хмари рішення, що змінює традиційний підхід до архітектури застосунків та зміщує акцент у бік локального оброблення даних. Отже, ML-система в iOS розглядається як повноцінний архітектурний компонент, що взаємодіє з шаром даних, бізнес-логікою, механізмами збереження стану, системними API. [16]

Core ML є вбудованим фреймворком Apple, призначеним для виконання моделей машинного навчання на пристрої користувача. Він є як основний засіб розгортання моделей орієнтованих на пристрої в iOS-застосунках. Його архітектура реалізує модель типу «сірої скриньки» (gray-box), що забезпечує

часткову доступність процесу виконання, водночас приховуючи внутрішні параметри моделі від розробника чи сторонніх осіб. Core ML має низку архітектурних обмежень: обмежене підтримання донавчання, обмежений доступ до параметрів моделі, а також пріоритет стабільності та безпеки над гнучкістю. Як висновок, рішення найчастіше представлено у форматі компактних спеціалізованих моделей, інференсних компонентів і системно інтегрованих модулів. [16]

У підсумку, розвиток on-device підходу для впровадження ML-систем у iOS орієнтований на модульність і багаторазове використання компонентів ML-систем. Реалізація цього підходу здійснюється завдяки уніфікованій архітектурі інтеграції, побудованій на основі Swift Package Manager (SPM).

1.6. Swift Package Manager як засіб побудови модульних рішень

Swift Package Manager (SPM) є стандартним інструментом екосистеми Apple для розповсюдження та інтегрування модулів. Він є самостійною одиницею, що дає змогу інкапсуляції внутрішньої логіки реалізації, має власну архітектуру та чітко визначений прикладний програмний інтерфейс. Для впровадження ML-систем він слугує інструментом для виокремлення логіки, що дає змогу ізолювати залежності, уникати дублювання коду та централізовано оновлювати моделі без змін у програмі клієнта. Під’єднання до інтеграційних моделей гарантується через окремі незалежні функції-зв’язки, що дає змогу уникати складнощів конфігурацій і ручних сценаріїв, а також залежностей від конкретних кінцевих предметних галузей. [17]

Отже, можна узагальнити, що використання SPM є доцільним у проєктах, що передбачають інтеграцію моделей машинного навчання, оскільки воно забезпечує слабку зв’язність між моделлю, та інтерфейсом користувача, та підвищення керованості підтримуваної системи.

1.7. Аналіз наявних рішень та формування вимог до системи

Емпіричний аналіз, проведений Journal of Information Systems Engineering

and Management у 2025 році, демонструє на вибірці з 2907 iOS-застосунків такі результати:

- приблизно 11,5 % застосунків містили принаймні одну локальну модель машинного навчання,
- загальна кількість ідентифікованих моделей становила 1883 одиниці,
- переважну кількість завдань займає комп'ютерний зір (computer vision),
- в середньому один застосунок містить 5 моделей, що свідчить про модульний підхід до побудови ML-функціональності, коли різні моделі відповідають за окремі підзавдання,
- провідним інструментом реалізації таких рішень є Core ML.

Порівняльний аналіз iOS та Android застосунків показав, що приблизно 17,6 % моделей є спільними для обох платформ. Дослідження безпеки виявило, що iOS-орієнтовані моделі демонструють високу вразливість до змагальних атак (adversarial attacks): середній рівень успішності атак із перенесенням змагальних прикладів (transferability) з повністю доступних моделей (white-box models). Ці відомості свідчать, що широке використання on-device ML у iOS-застосунках супроводжується не лише перевагами продуктивності та приватності, а й суттєвими викликами у галузі безпеки, які мають враховуватись під час проєктування та впровадження ML-систем на мобільній платформі iOS. [15]

Однією з істотних особливостей впровадження систем машинного навчання в мобільні застосунки на платформі iOS є вплив платформних обмежень, які визначають допустимі архітектурні та функціональні рішення. Головна політика компанії Apple орієнтована на енергоефективність, стабільність роботи та захист персональних даних користувача. На рисунку 1.7.1 показано розподіл причин відмови публікацій і розповсюдження застосунків у App Store, пов'язаних із використанням ML-функціональності. Згідно з показниками діаграми, вагома частина відповідає проблемам оброблення у фоновому режимі, надмірному енергоспоживанні та недотриманні вимог

приватності. [16]

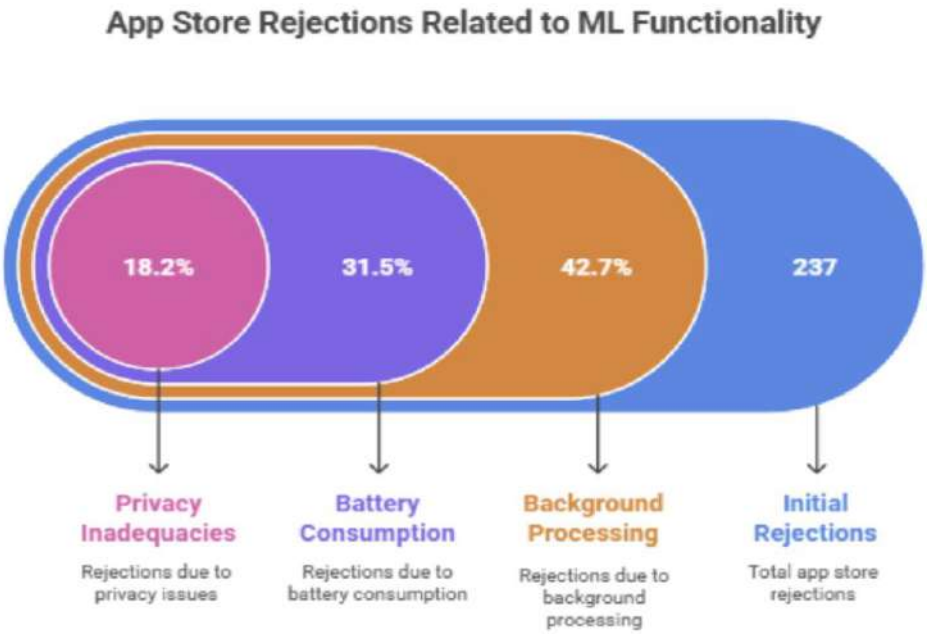

Рисунок 1.7.1 Розподіл причин відмови публікацій і розповсюдження застосунків в App Store [16]

Ще одним визначальним фактором є гетерогенність апаратного забезпечення в екосистемі iOS, що зумовлює потребу в адаптації ML-рішень до різних апаратних конфігурацій. Мобільні застосунки мають забезпечувати стабільну роботу як на нових пристроях, так і на старіших моделях, що мають обмежені ресурси пам’яті та процесора. Платформа iOS, крім технічних обмежень, формує комплекс регуляторних принципів, які визначають умови використання ML-компонентів у мобільних застосунках. [16]

Отже, загальні вимоги охоплюють прозорість процесу оброблення інформації, відповідність політикам конфіденційності та контроль впливу ML-алгоритмів на енергоспоживання й стабільність системи.

Проведений аналіз дає змогу сформулювати загальний висновок: пропонується створення універсального програмного пакета на основі Swift Package Manager, який виступатиме як проміжний інтеграційний шар між iOS-застосунком та власним ML-конвеєром. Розроблений модуль створює уніфікований механізм взаємодії з ML-моделями, які розгортаються на

приватному сервері, що своєю чергою уможливлює масштабувати систему та поступово розширювати її функціональні можливості.

Застосування серверного доступу до моделей дає змогу уникнути прямої інтеграції великих моделей у мобільний застосунок, що, своєю чергою, зменшує обсяг займаної пам’яті та знижує ризики, пов’язані з обмеженнями продуктивності й енергоспоживання.

Архітектура системи передбачає використання захищених протоколів зв’язку, повне шифрування даних у процесі передавання й оброблення, а також розмежування функціональної відповідальності між клієнтським застосунком і серверною частиною ML-конвеєра.

Розроблена система має універсальний характер, що забезпечує можливість її інтеграції в різні предметні галузі без необхідності модифікації основної архітектури. Пакет, крім основної ML-функціональності, підтримує гнучку персоналізацію користувацького інтерфейсу, що охоплює зміну кольорових схем, налаштування сценаріїв відображення та параметризацію інтерфейсних елементів. Завдяки такій архітектурі система може використовуватись не лише з дослідницькою метою, але й у промислових рішеннях різних типів iOS-застосунків.

2. Проєктування та розроблення системи

2.1. Структура системи та обґрунтування технічних рішень

Сучасні мобільні застосунки дедалі частіше інтегрують методи машинного навчання для підвищення рівня персоналізації взаємодії з клієнтом. Водночас наявні впровадження нейронних мереж переважно орієнтовані або на локальне впровадження таких систем, або на експлуатацію відкритих прив’язаних серверних API. Це значно обмежує масштабованість, повторне застосування та гнучкість механізму. У такому контексті актуальним є створення уніфікованої архітектури, яка забезпечує результативну інтеграцію ML-систем у мобільні застосунки без надмірного використання клієнтського середовища третіх осіб.

Запропонована в дослідженні реалізація призначена для автоматичного генерування стилізованих текстових описів до зображень із застосуванням мультимодальних моделей машинного навчання та подальшої інтеграції цієї функціональності в iOS-застосунки у форматі універсального програмного пакета. Особливістю підходу є поєднання серверного конвеєра моделей із клієнтською частиною через проміжний інтеграційний шар, реалізований з використанням Swift Package Manager. Такий формат побудови архітектури дає змогу відокремити комплексну логіку машинного навчання від мобільного застосунку, забезпечуючи прозору та стандартизовану взаємодію між компонентами.

Архітектура програми ґрунтується на клієнт-серверній структурі із чітким розподілом відповідальностей між її компонентами. Клієнтська частина відповідає за створення запитів, оброблення результатів та інтеграцію отриманої інформації у користувацький інтерфейс. Серверна частина реалізує конвеєр моделей, охопно з обробленням вхідних зображень, генеруванням базових підписів та їхньою подальшою стилізацією. Обмін даними між клієнтом і сервером здійснюється із застосуванням захищених протоколів зв’язку та повного шифрування інформації на всіх етапах передавання й модифікації.

Важливою особливістю розробленої системи є її універсальний характер.

Застосування SPM дає змогу інтегрувати створений модуль у різних Swift-проєктах без необхідності зміни базової архітектури або внутрішньої логіки конвеєра. Окрім основної функціональності автоматичного генерування текстових описів, пакет підтримує можливості параметризації та персоналізації користувацького інтерфейсу (налаштування колірних схем, сценаріїв відображення і поведінки інтерактивних елементів).

Реалізація запропонованої структури ґрунтується на застосуванні сучасного набору технологій, програмних і обчислювальних ресурсів. Вони орієнтовані на розроблення, навчання та розгортання моделей із подальшою інтеграцією результатів у мобільні iOS-застосунки.

Для їхньої побудови, тренування та апробації реалізовано програму, написаною мовою програмування Python із користуванням бібліотеки PyTorch. Це забезпечує гнучкий інструментарій для модифікації нейронних мереж, автоматичного диференціювання та відстежування виконання обчислень.

Для завдання генерування базових підписів до зображень застосовано архітектуру трансформерної реалізації Encoder-Decoder, Навчання моделі здійснювалося із експлуатації функції втрат Cross Entropy Loss. У якості базового набору даних для проєктування, налаштування та реалізації системи застосовано набір Flickr8k.

Другий етап ML-конвеєра спрямований на стилізування згенерованих описів та формування публікацій, орієнтованих на різні вхідні тематичні сценарії. Для цього взято модель T5 For Conditional Generation, яка підтримує авторегресивне генерування тексту згідно з заданим контекстом. У межах наукової діяльності обрано конфігурацію t5-small, що містить приблизно 60 млн параметрів. Навчання здійснювалось із застосуванням стандартного механізму Trainer, який автоматизував виконання прямих і зворотних проходів, окрім цього обчислення функції втрат за допомогою механізму крос-ентропії між результатами та цільовими мітками. Під час тренування враховувалися спеціальні токени, що давало змогу уникнути спотворення виходів оптимізації.

Для завдання стилізації сформовано власний набір даних з пар ключів і

значень, побудований на підставі роботи першої структури. Зображення з початкових наборів даних використовувалися як нейтральні підписи для демонстрації частини вхідного контексту разом із параметром стилю.

З метою уніфікації розгортання та забезпечення відтворюваності обчислювального середовища обидві моделі запаковано в конвеєр завдяки Docker. Серверна частина реалізована у вигляді HTTP API з послуговуванням фреймворку FastAPI, який забезпечує асинхронне оброблення запитів і чітку типізацію вхідних та вихідних відомостей. API надає окремі кінцеві точки для генерування базових описів, стилізованих текстів і повного використання двоетапної системи конвеєра. Для організації захищеного шлюзу застосовано інструментарій Tailscale Serve, що забезпечує приватний мережевий канал без необхідності публічного відкриття сервісу.

Інтеграція в мобільні застосунки реалізована через Swift Package Manager. Створений Swift-пакет інкапсулює логіку мережевої взаємодії, оброблення відповідей і передавання параметрів стилізації. Застосування отриманого пакету дає змогу інтегрувати його як незалежний програмний компонент, що може мати повторне застосування у різних проєктах без зміни серверної інфраструктури.

2.2. Проєктування моделі генерування підписів до зображень

Відповідно до проведеного аналізу сучасних підходів для автоматичного генерування описів до зображень обрано архітектуру трансформерної реалізації Encoder-Decoder, яка докладно описана в розділі 1.2. Початковою моделлю є BLIP, тому візуальний кодувальник здійснює виокремлення ознак із зображення, у той час як текстовий декодувальник створює послідовність слів для формування опису до зображення.

Імплементація виконана на основі поточного набору технологій:

- Мова програмування Python, яка послуговується для створення подібних програм, у завданнях машинного навчання та ком’ютерного зору, і має розгалужений набір програмних бібліотек [19].
- Фреймворк PyTorch обрано для реалізації нейронної мережі через можливість параметризації процесу навчання та під’єднання оптимізаторів.

- Для роботи з мультимодальною моделлю BLIP (Bootstrapping Language-Image Pretraining) використано бібліотеку Hugging Face Transformers. ЇЇ представником є Salesforce/blip-image-captioning-base, яка є попередньо навченою на великих збірках даних на завданнях зі створення описів до фото (image captioning).
- Бібліотеки TorchVision, NLTK, PIL і Torch DataLoader використані у формі інструментарію для навчання моделі, завантаження й оброблення зображень, токенізації слів і трансформації інформаційних одиниць.
- Навчальною вибіркою даних слугував набір Flicktr8k, що містить 8 000 зображень з 5 текстовими описами до кожного. Фото сцени охоплюють різні об'єкти включно з людьми, тваринами, природою тощо.

У дослідженнях завдань автоматичного генерування підписів до зображень широко застосовується такий або наближених вибір технологічних ресурсів, зокрема для архітектурних рішень підходу CNN+LSTM. [18]

У запропонованій реалізації архітектура системи сформована з кількох незалежних компонентів, прив'язаних до власних етапів оброблення, навчання й оцінювання. Завантаження даних відбувається через клас FlickrDataLoader, який формує структуру даних у форматі списку словників. Клас поділу вибірки на навчальну та валідаційну забезпечує достатню кількість фото для співвідношення 85 % та 15 % відповідно. Етап підготовки також охоплює нормалізацію інформації збірки, яка містить процеси перетворення у RGB формат, токенізацію тексту й формування тензорів.

Головним компонентом системи є модуль BlipCaptionModel, який координує процес генерування підписів до фото. Візуальний кодувальник виконує виокремлення ознак зображення, а текстовий декодувальник генерує опис на основі векторів попередника. Під час ініціалізації, окрім базових завантажень та передач на обчислювальний пристрій, відбувається заморожування параметру кодувальника та частини декодувальника задля зменшення кількості налаштувань та стабільного навчання на невеликій збірці даних. Оскільки Flickr8k містить близько 8 000 фото загалом, то існує ризик

перенавчання та швидкого запам’ятовування тренувальної збірки, особливо з потужностями BLIP. На рисунку 2.2.1 представлений фрагмент програмної імплементації моделі генерування підписів у класі BlipCaptionModel.

```python
class BlipCaptionModel:
    def __init__(self, model_name, device):
        self.device = torch.device(device)
        self.processor = BlipProcessor.from_pretrained(model_name, use_fast=True)
        self.model = BlipForConditionalGeneration.from_pretrained(model_name, use_safetensors=True)
        self.model.to(self.device)

        for param in self.model.vision_model.parameters():
            param.requires_grad = False
        for param in self.model.text_decoder.bert.encoder.parameters():
            param.requires_grad = False


    def generate_caption(self, image):
        inputs = self.processor(images=image, return_tensors="pt").to(self.device)
        out = self.model.generate(**inputs, num_beams=5, max_length=30, early_stopping=True)
        caption = self.processor.decode(out[0], skip_special_tokens=True)
        return caption
```

Рисунок 2.2.1 Реалізація архітектури моделі генерування підписів

Навчання моделей реалізовано за використанням класу Trainer, який застосовує оптимізатор AdamW, який працює на основі обчислених градієнтів. Такий вибір обгрунтовано адаптивністю коефіцієнтів навчання кожного параметра на основі градієнтів, завдяки чому корелюється швидкість навчання вагових коефіцієнтів мережі й стабільність процесу тренування [21]. Етап навчання на кожній епосі забезпечує послідовне оброблення пакетів даних, створює прямий прохід моделі з обчисленням функції втрат, зворотним поширенням помилки та оновленням параметрів. Для оптимізації моделі використовується функція втрат CrossEntropyLoss, оскільки завдання генерування формулюється як послідовна класифікація токенів. Для архітектур типу «послідовність-у-послідовність» з багатокласовим групуванням наведена математична функція є стандартною, бо дає змогу поступово покращувати здатність моделі генерувати правильні слова в упорядкованих наборах токенів

[20]. Також використовується механізм ранньої зупинки для запобігання перенавчанню й обмеження градієнтів для стабілізації. На рисунку 2.2.2 представлений фрагмент програмної імплементації.

```
for epoch in range(self.config.EPOCHS):
    self.model.train()
    total_loss = 0
    for step, batch in enumerate(tqdm(self.dataloader, desc=f"Epoch {epoch}")):
      batch = {k: v.to(self.device)
               for k, v in batch.items()}
      outputs = self.model(**batch)
      loss = outputs.loss
      loss.backward()
      torch.nn.utils.clip_grad_norm_(self.model.parameters(), 1.0)
      self.optimizer.step()
      self.optimizer.zero_grad()
      total_loss += loss.item()
      if step % 20 == 0:
        print(f"epoch {epoch} step {step}/{len(self.dataloader)} loss {loss.item():.4f}")
        avg_loss = total_loss / len(self.dataloader)
        self.save_checkpoint(epoch)
        val_bleu = self.evaluate(val_loader)
        if val_bleu > self.best_bleu:
          self.best_bleu = val_bleu
          self.no_improve = 0
          torch.save(self.model.state_dict(),"/models/best_model.pth")
          else:
            self.no_improve += 1
            if self.no_improve >= self.patience:
              break
```

Рисунок 2.2.2 Реалізація архітектури моделі генерування підписів

Конфігурація системи визначає основні шляхи, назви, залежність обчислювального пристрою та параметри процесу тренування. Далі розглянуто обрані гіпераметри. Розмір пакету (batch size) становив 8 через обмеження GPU пристрою тренування. Використання малого розміру пакета призводить до підвищеної варіативності градієнтів, однак покращує узагальнювальну здатність моделі під час навчання на невеликих збірках. Це є компромісним рішенням для створеної архітектури. Темп навчання становив 5e-5, оскільки трансформери є

чутливими до цього параметру, а занадто великі значення можуть зруйнувати попередньо навчені ваги. Кількість епох обрано 3 через невеликий набір даних, попередньо навчену модель з можливістю ранньої зупинки та обмеження пристрою. Променевий пошук (beam search) описано показником 5 для обрання найкращого кандидата серед створених. Такий вибір значно покращує якість згенерованих підписів, водночас не спричиняючи суттєвого уповільнення процесу виконання моделі (inference). Максимальною довжиною обрано 30 символів, оскільки збірка даних містить підписи довжиною 8-30 символів в цілому. Окремо реалізовано модуль для оцінювання якості генерування підписів за метриками BLEU. Детальне обґрунтування та огляд результатів апробації представлені в Розділі 3.

Отже, головний сценарій роботи охоплює завантаження конфігурації, підготовку збірки даних зі створенням нормалізованого представника PyTorch Dataset, ініціалізацію моделі та запуску тренування. Модуль train.py виконує роль центрального компонента, який координує роботу всіх частин системи. На рисунку 2.2.3 представлений фрагмент програмної імплементації.

```
config = Config()
train_images = SplitLoader.load_split(config.TRAIN_SPLIT)
val_images = SplitLoader.load_split(config.VAL_SPLIT)

model = BlipCaptionModel(config.MODEL_NAME, config.DEVICE)

train_data = FlickrDataLoader.load_captions(config.CAPTION_FILE, train_images)
val_data = FlickrDataLoader.load_captions(config.CAPTION_FILE, val_images)

train_dataset = FlickrCaptionDataset(train_data, config.IMAGE_DIR, model.processor)
val_dataset = FlickrCaptionDataset(val_data, config.IMAGE_DIR, model.processor)

train_loader = DataLoader(
  train_dataset,
  batch_size=config.BATCH_SIZE,
  shuffle=True,
  num_workers=4,
  pin_memory=True
)

val_loader = DataLoader(
  val_dataset,
  batch_size=config.BATCH_SIZE
)

trainer = Trainer(model, train_loader, config)

trainer.train(val_loader)
```

Рисунок 2.2.3 Реалізація головного модуля системи генерування підписів до зображень

Отже, реалізований модуль навчання забезпечує повний цикл тренування моделі генерування підписів. Докладні результати апробації системи генерування підписів до зображень описані в Розділі 3.

2.3. Розроблення та підготовка власного набору даних для стилізування текстових підписів

Аналіз відкритих наборів даних демонструє, що на сьогодні відсутні збірки, які містять різножанрові стилізовані пари базового опису до зображень.

Відповідно до цього в межах роботи запропоновано та підготовлено власний набір для завдання стилізації підписів до зображень, який поєднує результати моделі генерації описів до фото та додаткові текстові джерела. Основною метою розроблення цього корпусу є впровадження в навчання моделі для стилізування опису за вхідним жанром.

Процес формування датасету сформовано з декількох етапів. На першому кроці інтегрована навчена модель генерування підписів до зображень створює базові текстові описи до наданих фото. Медіа отримані з декількох відкритих наборів даних, а отримані тексти виконують роль початкового опису сцени на зображенні. Наступний етап охоплює використання отриманих конструкцій як вхідних даних для формування стилізованих текстових варіантів.

Для розширення та урізноманітнення збірки використано додаткові відкриті набори, які містять приклади підписів у різних стилях. Зокрема, Instagram-зображень із підписами, Memotion Dataset, який містить гумористичні фото та текстові підписи з відповідним змістом.

У межах роботи визначено кілька штучно сформованих категорій стилю, які відображають різні типи комунікації в соціальних мережах та інформаційних платформах. Кінцевим переліком є:

- *Casual* — базовий опис зображення, згенерований першою моделлю конвеєра з нейтральним рівнем стилізації.
- *Instagram* — підпис, адаптований до формату соціальних мереж, з використанням хештегів, емодзі та неформальних мовних структур.
- *Formal* — формалізований текстовий опис, який охоплює нейтральну лексику та граматично коректні конструкції.
- *Funny* — гумористичний стиль, який формується через додавання жартівливих елементів за ідентифікацією тригерних слів у тексті.

Для зберігання отриманих даних використовується формат CSV, де кожен запис містить такі поля:

- *id* — унікальний ідентифікатор запису;
- *style* — стиль текстового підпису;

- *image_caption* — базовий опис зображення, отриманий із моделі генерації підписів;
- *post* — стилізований текстовий підпис.

Отже, результат демонструє пари з базового опису зображення та стилізованого тексту для навчання моделі трансформації між різними жанрами. Такий підхід дає змогу отримати відповідний корпус текстів, у якому один опис забезпечує представлення у різних мовних формах.

2.4. Проєктування моделі стилізації текстових підписів

Після одержання можливості генерування базового опису зображення та наповнення стилістичного набору даних для навчання, наступним етапом є жанрова модифікація текстового підпису. Метою розроблення моделі є перетворення нейтрального опису фото сцени в такий, що відповідає заданому стилю. Завдання формується як умовне генерування (conditional text generation). На рисунку 2.4.1 показано процес оброблення, де модель отримує стиль та базовий опис зображення, після чого трансформує їх у результат за використання мережі T5. Вона належить до трансформерних архітектур типу «послідовність-у-послідовність».

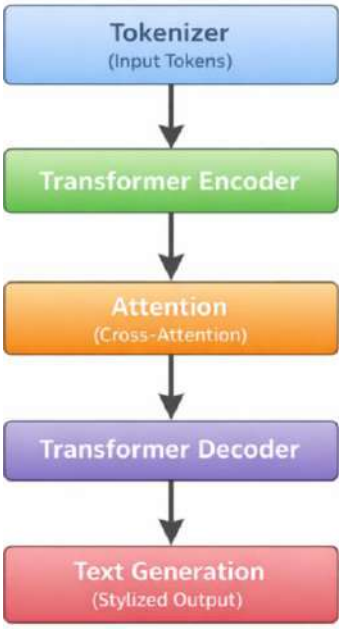

Рисунок 2.4.1 Процес стилізації підписів

Для імплементації процесу навчання й налаштування моделі використано такий набір технологій для оброблення природної мови:

- Мова програмування Python.
- Фреймворк машинного навчання PyTorch.
- Бібліотека трансформерів HuggingFace Transformers, яка надає готові реалізації моделей оброблення природної мови.
- Архітектура T5 (Text-to-Text Transfer Transformer) FLAN-T5-small (містить приблизно 60 мільйонів параметрів).
- Клас AutoModelForSeq2SeqLM, який реалізує sequence-to-sequence генерування тексту.
- AutoTokenizer використовується для перетворення тексту в числове представлення.
- Для оброблення набору даних використовується бібліотека datasets.

Для реалізації поставленого завдання використовується трансформерна модель типу «послідовність-у-послідовність», що забезпечує можливість умовного створення тексту. Вона отримує на вхід текстовий опис і бажаний стиль, після чого генерує кінцевий підпис. У роботі застосовано попередньо натреновану модель FLAN-T5-small, яка сформована з двох компонентів. Першим є кодувальник, який обробляє вхідний запит зі стилем та базовим описом, а другим є декодувальник, що генерує стилізований текст по одному токену, використовуючи контекст попередника. Вхідний текст формується у вигляді єдиного рядка, що дає змогу моделі одночасно враховувати семантичний зміст і жанрову приналежність. Тренування системи здійснюється в режимі керованого навчання (supervised learning) на основі підготовленого набору стилізованих підписів. З використанням його записів формується навчальна пара вхідного та вихідного значення.

Під час підготовки виконується фіксація значення випадкового зерна для відтворюваності результатів експериментів. За можливості використання графічного процесора додатково запам’ятовується CUDA генератор й

вимикаються деякі оптимізації відповідної бібліотеки для детермінованої поведінки алгоритмів. Також перед навчанням інформація перетворюється у формат HuggingFace Dataset. Через попередньо навчену модель та відповідний токенізатор відбувається завантаження. Параметризація дає змогу визначити базову трансформерну модель як FLAN-T5-small та обрати можливість прискореного обробляння. Вирівнювання довжин та попередня підготовка даних дає змогу нормалізувати вхідний набір та отримати токенізовану збірку з числовими представленнями тексту. Для формування пакетів застосовано колектор, який об’єднує приклади, вирівнює довжини й перетворює на тензори. Параметризація вирівнювання за типом longest дає змогу нормалізувати набори за найдовшою послідовністю пакета. На рисунку 2.4.2 надано уривок програмної реалізації цього етапу.

```
tok = AutoTokenizer.from_pretrained(args.base_model, use_fast=True)
model = AutoModelForSeq2SeqLM.from_pretrained(args.base_model)

if tok.pad_token is None:
    tok.pad_token = tok.eos_token or tok.sep_token

preprocess = build_preprocess(
    tok,
    args.input_max_len,
    args.target_max_len
)

tokd = dataset.map(preprocess, batched=True, remove_columns=dataset["train"].column_names)

collator = DataCollatorForSeq2Seq(
    tokenizer=tok,
    model=model,
    padding="longest",
    label_pad_token_id=-100,
    return_tensors="pt"
)
```

Рисунок 2.4.2 Налаштування архітектури моделі стилізації підписів

Процес навчання виконується за допомогою класу Seq2SeqTrainer, який

автоматизує прямий прохід моделі (forward pass), зворотне поширення помилки (backpropagation) та оптимізацію параметрів. Протягом цього етапу за допомогою параметризації зберігаються результати навчання. Розмір пакетів навчальних прикладів за один крок оптимізації становить 8 для зменшення обсягів ресурсів відеокарти пристрою та стабілізації тренування. Конфігурація епох визначена як 50 із темпом навчання в обсязі 5×10^{-4}, що надало можливість моделі відносно швидко адаптуватись до специфіки набору. Регуляризація додає штраф 0.01 для обмеження надмірного збільшення ваг і перенавчання. Кількість альтернативних варіантів генерації становила 4 для отримання задовільніших результатів. На основі цих даних створено тренера, який є об'єктом Seq2SeqTrainer і відповідає за повний цикл навчання моделі. Параметри навчання моделі стилізації підписів продемонстровано на рисунку 2.4.3.

```
training_args = Seq2SeqTrainingArguments(
        output_dir=args.out_dir,
        overwrite_output_dir=True,
        per_device_train_batch_size=args.bsz,
        per_device_eval_batch_size=args.bsz,
        num_train_epochs=args.epochs,
        learning_rate=args.lr,
        warmup_steps=0,
        weight_decay=0.01,
        logging_steps=50,
        logging_dir=os.path.join(args.out_dir, "logs"),
        save_strategy="epoch",
        predict_with_generate=True,
        generation_num_beams=args.num_beams,
        fp16=False,
        report_to="none",
        load_best_model_at_end=True,
        metric_for_best_model="eval_loss",
        greater_is_better=False,
        **extra_kwargs
    )
```

Рисунок 2.4.3 Параметри навчання моделі стилізації підписів

Параметри навчання тренера моделі стилізації підписів продемонстровано

на рисунку 2.4.4.

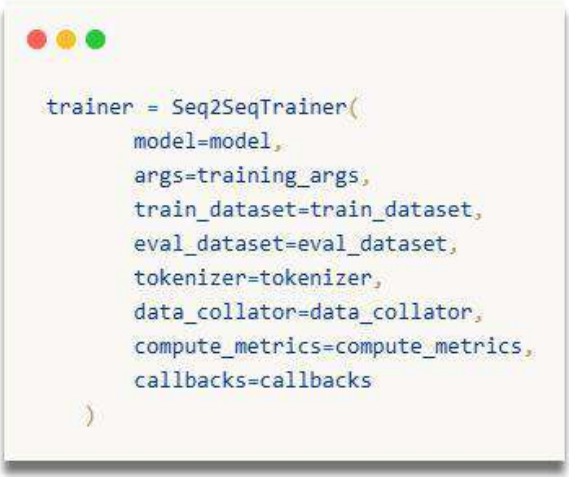

Рисунок 2.4.4 Параметри навчання тренера моделі стилізації підписів

Отже, результатом імплементації стала натренова модель генерування стилізованих описів, результати апробації якої детально розглянуті в Розділі 3.

2.5. Побудова конвеєра взаємодії моделей

Для імплементації автоматичного генерування стилізованих підписів до зображень застосовується багатокроковий конвеєр оброблення інформації. Він об’єднує два комплекси машинного навчання, реалізовані й оглянуті в попередніх пунктах. Загальна схема конвеєра системи представлена на рисунку 2.5.1. Вхідними даними є зображення, яке передається до першої моделі генерації опису. Після цього отриманий текст використовується як вхід для другої архітектури генерування стилізованого підпису.

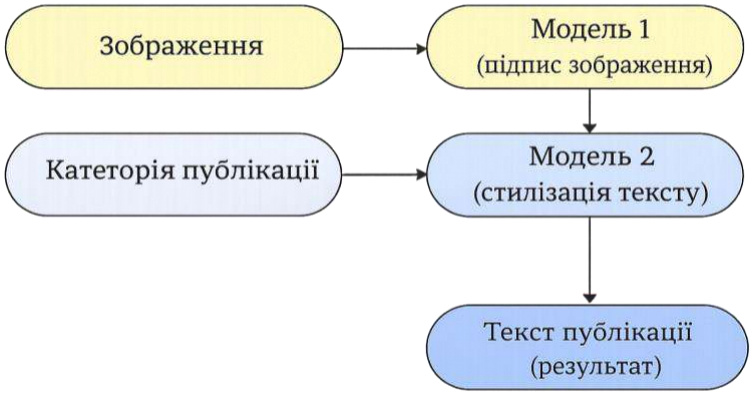

Рисунок 2.5.1 Конвеєр моделей генерування стилізованих підписів до зображень

Отже, побудований конвеєр дає змогу комбінувати можливості комп'ютерного зору та сучасних моделей генерування тексту, забезпечуючи модифікації за заданими стилістиками.

2.6. Налаштування серверного гостинґу та API

Для забезпечення взаємодії клієнтського застосунку (iOS платформа) з моделями машинного навчання реалізовано серверний інструмент. Він надає змогу використовувати API для оброблення запитів генерування підписів до фото. Сервер ініціалізує завдання моделей машинного навчання, об'єднує їх у єдиний конвеєр та повертає результати у вигляді JSON-відповідей. Для імплементації серверної частини обрано такий технологічний набір:

- Основною мовою реалізації серверної частини є Python.
- Фреймворк FastAPI застосовано для реалізації REST API. Він використовується для створення кінцевих точок API, через які клієнтський застосунок може надсилати запит та отримувати результати роботи моделей.
- Для запуску сервера використовується Uvicorn — ASGI-сервер. Він виконує роль вебсервера, який обробляє HTTP-запити та передає їх до FastAPI-застосунку.
- Бібліотека PyTorch для реалізації та запуску моделей машинного навчання.

- Бібліотека Pillow для роботи із зображеннями.
- Для спрощення майбутнього розгортання сервісу використовується Docker-контейнеризація, а для керування ними використовується Docker Compose.

Основною функцією сервера є отримання зображення від клієнта, виконання завдання оброблення моделей та повернення результату у форматі JSON. Система побудована за багаторівневою архітектурою з чітким розділенням відповідальностей між API-рівнем, сервісним рівнем та рівнем моделей машинного навчання. Бізнес-логіка оркестратора реалізовано в класі PipelineService, який забезпечує розмежування на окремі завдання та цілий конвеєрний запит. Програмну реалізацію продемонстровано на рисунку 2.6.1.

```python
class PipelineService:
    def __init__(self):
        self.caption_model = CaptionModel()
        self.stylizer_model = StylizerModel()

    def warmup(self):
        self.caption_model.load()
        self.stylizer_model.load()

    def generate_caption(self, image: Image.Image) -> str:
        return self.caption_model.generate(image)

    def stylize(self, style: str, caption: str) -> str:
        return self.stylizer_model.generate(style, caption)

    def run_pipeline(self, image: Image.Image, style: str) -> dict:
        caption = self.generate_caption(image)
        stylized = self.stylize(style, caption)

        return {
            "model_caption": caption,
            "stylized_caption": stylized
        }
```

Рисунок 2.6.1 Сервісний прошарок API

Логіка виконання запитів до розроблених моделей є однаковою зі вже дослідженими реалізаціями. Головний серверний модуль ініціалізує FastAPI-застосунок, слугує центральною точкою конфігурації сервера та використовується для реєстрації кінцевих точок за допомогою модуля routes.py.

Сервер реалізує декілька HTTP-ендпоінтів, які забезпечують різні функції системи. Список представників продемонстровано на рисунку 2.6.2.

Рисунок 2.6.2 Список наявних кінцевих точок API

Система надає доступ до точки /v1/caption, що виконує генерування базового текстового опису зображення. Алгоритм роботи охоплює одержання, зчитування, модифікацію зображення, передавання до першої системи й отримання результату в JSON-відповіді. Точка /v1/stylize експлуатується для генерування стилізованого підпису. Клієнт передає стиль та текстовий опис зображення. Модель стилізації надає результат на основі цих даних. Для повного використання системи реалізовано endpoint /v1/pipeline, який об’єднує обидві моделі в єдиний конвеєр, архітектуру якого описано в підрозділі 2.5. Отже, клієнт може виконати повний процес генерування підпису за один запит, або ж розділити завдання за окремими частинами конвеєра.

Для обмеження доступу до сервісу використовується API-ключ. Для розгортання системи додатково реалізовано Docker-контейнер, конфігурація

якого описана у файлі docker-compose.

Отже, реалізована серверна частина забезпечує повний цикл оброблення запитів стилізованого генерування підписів до зображень.

2.7. Проєктування та реалізація Swift Package Manager модуля

Для створення модульного програмного компонента для зв'язки мобільних застосунків iOS з конвеєром моделей, розміщеного на сервері, використано інструмент Swift Package Manager. Архітектура модуля побудована з урахуванням принципів відповідальностей, інверсії залежностей та реактивного керування станами. Для цього використано нативні фреймворки Foundation та SwiftUI.

Структура модуля організована на основі логічних підсистем:

- Папка BusinessLogic відповідальна за оброблення даних, мережевих запитів, інформаційних моделей об'єктів та їхньої взаємодії.
- UI папка є представником компонентів інтерфейсу користувача для роботи з фабрикою (абстракція усіх UI компонентів) і системи стилізації.
- Точка входу є View та ViewModel для прямої інтеграції в будь-які мобільні рішення.

Компонент APIManager відповідає за взаємодію з віддаленим сервером. Його основні функції – це формування HTTP-запитів, побудова формату передачі даних із кількома частинами, обробляння відповіді й десереалізація JSON. Основними етапами опрацювання є завантаження зображення, формування кінцевої точки, формування запиту з кількома частинами, його відправлення та декодування відповіді. Для систематизування створено бізнес моделі Post, PostResponse й PostStyle. Фрагмент реалізації наведено на рисунку 2.7.1.

```swift
public final class APIManager: APIManaging {
    private let configuration: APIConfig

    public init(configuration: APIConfig) {
        self.configuration = configuration
    }

    public func getPost(imageURL: URL, style: PostStyle) async throws -> Post {
        guard let imageData = try? Data(contentsOf: imageURL) else { throw
PipelineError.fileNotFound }

        let url = try buildEnpoint(style: style)
        let boundary = "Boundary-\(UUID().uuidString)"
        var req = buildRequestBody(url: url, boundary: boundary)
        let body = try buildBody(imageURL: imageURL, imageData: imageData, boundary: boundary)
        req.httpBody = body

        let postText = try await sendRequest(req)
        return Post(imageURL: imageURL, styleName: style, postText: postText)
    }

    ...

    private func sendRequest(_ req: URLRequest) async throws -> String {
        let (data, resp) = try await URLSession.shared.data(for: req)
        guard let http = resp as? HTTPURLResponse else { throw PipelineError.invalidResponse }
        guard (200...299).contains(http.statusCode) else {
            let msg = String(data: data, encoding: .utf8)
            throw PipelineError.serverError(statusCode: http.statusCode, message: msg)
        }
        let decoded = try JSONDecoder().decode(PostResponse.self, from: data)
        return decoded.stylized_caption ?? ""
    }
}
```

Рисунок 2.7.1 Головний компонент бізнес логіки для взаємодії API

Інтерфейс UIFactory визначає абстрактний компонент для створення UI-засобів. Фрагмент реалізації наведено на рисунку 2.7.2. Конкретною його реалізацією є DefaultUIFactory, яка створює компоненти SwiftUI, враховуючи дизайн тем. Система візуальної стилізації охоплює кольорові палітри, типографіки й відступи для централізованого керування й консистентності UI.

```
public protocol UIFactory {
    var theme: Theme { get }
    var imageURL: URL? { get }

    func makeBackground(custom builder: AnyView?) -> AnyView
    @MainActor
    func makeImagePicker() -> AnyView
    @MainActor
    func makeStylesPicker(selectedStyle: Binding<String>, styles:
[PostStyle]) -> AnyView
    @MainActor
    func makeGenerateButton(action: @escaping (() -> Void)) ->
AnyView
    @MainActor
    func makeLoading<Style: ProgressViewStyle>(generating:
Binding<Bool>, style: Style) -> AnyView
    @MainActor
    func makeLabel(_ title: String) -> AnyView
}
```

Рисунок 2.7.2 Інтерфейс для реалізації фабрики управління UI

Головним компонентом UI частини є PostGeneratorView, який можна використовувати як точку входу функціонального модуля. Його відповідальністю є композиція побудови інтерфейсу з використанням абстракції UIFactory.

Патерн MVVM дотримано за імплементації PostGeneratorViewModel, який інкапсулює стан, обробляє події користувача та координує управління компонентів бізнес-логіки. Реактивна взаємодія досягнута через зміну станів за принципами реактивного програмування. Фрагмент реалізації наведено на рисунку 2.7.3.

```swift
@MainActor
@Observable
class PostGeneratorViewModel {
    struct State {
        var currentImageURL: URL? = nil
        var selectedStyle: PostStyle = PostStyle.casual
        var generating: Bool = false
        var generatedText: String? = nil
    }

    private(set) var state: State = .init()

    func setCurrentImageURL(_ url: URL?) {
        state.currentImageURL = url
    }

    func setSelectedStyle(_ style: PostStyle) {
        state.selectedStyle = style
    }

    func setGenerating(_ value: Bool) {
        state.generating = value
    }

    func api() async {
        do {
            state.generating = true
            let config = try APIConfig()
            guard let imageURL = state.currentImageURL else {
                print("Missing current image URL")
                state.generating = false
                return
            }
            let manager = APIManager(configuration: config)
            let post = try await manager.getPost(imageURL: imageURL, style:
state.selectedStyle)
            state.generatedText = post.postText
            state.generating = false
        } catch {
            print("Config error:", error.localizedDescription)
            state.generating = false
        }
    }
}
```

Рисунок 2.7.3 Імплементація PostGeneratorViewModel

У підсумку, запропонована архітектура поєднує шаблони MVVM, Abstract Factory та принципи Clean Architecture, тому розроблений модуль можна інтегровувати в більші системи або використати як незалежний компонент.

2.8. Висновки до розділу

Отже, в другому розділі розглянуто комплексне проєктування та реалізацію системи генерування та стилізації підписів до зображень. Під час дослідження обґрунтовано вибір архітектури системи, яка створена на основі модульного підходу з поділом на клієнтську та серверну логіки. Це забезпечує гнучкість розгортання та надає можливості інтеграції в мобільні застосунки.

Розроблено модель генерування підписів до зображень, яка виконує завдання створення базового семантичного опису вхідних даних. Для стилізування результатів створено окрему модель, що дає змогу трансформувати отримані підписи відповідно до заданого жанру. Значним аспектом роботи є імплементація власної збірки для завдання стилізації, що охоплює структурування даних за жанрами та попереднє оброблення текстів. Такий підхід розділяє завдання генерування змісту та стилістичного обробляння для більшого контролю результатів.

Побудовано конвеєр взаємодії моделей, який забезпечує послідовне опрацювання даних, що дає змогу масштабувати систему та модифікувати окремі компоненти без впливу на загальну архітектуру. Налаштування серверної інфраструктури та API забезпечує взаємодію між клієнтською частиною та моделями. Реалізовано механізми обробляння запитів, передавання зображень та повернення результатів, що відповідає вимогам продуктивності та безпеки.

Завершальним етапом є проєктування та реалізація універсального модуля на основі Swift Package Manager, який інкапсулює логіку взаємодії з сервером і забезпечує інтеграцію функціональності в iOS-застосунки.

Отже, у розділі описане поєднання модульного підходу до побудови системи створення підписів з двоетапною архітектурою обробляння даних, де процеси є розділеними, але інтегрованими в єдиний конвеєр. Запропоновано підхід до інтеграції моделей машинного навчання в мобільні застосунки за допомогою SPM-модуля, що забезпечує повторне використання, масштабованість та незалежність від конкретної реалізації моделей.

У наступному розділі розглянуто результати тестування та апробації

розробленої системи, зокрема оцінювання якості згенерованих підписів, продуктивності системи та її ефективності в умовах реального використання.

3. Тестування й апробація розробленої системи

3.1. Методологія та критерії оцінювання ефективності системи

Для оцінювання якості генерування стилізованих описів до зображень застосовано комбінований підхід із кількісних та якісних методів для експертизи.

Оцінювання моделей охоплює автоматичні метрики BLEU-1 – BLEU-4, що дають змогу визначити відповідність згенерованих підписів до еталонних показників на рівні n-грам різних довжин.

Додаткову динаміку функції втрат у процесі навчання наочно розглянуто завдяки графікам. Це дає змогу оцінити стабільність та здатність програмних засобів до узагальнення. Характер збіжностей, наявність флуктацій та момент досягнення плато проаналізовано згідно з цими показниками.

Для експертизи повного конвеєра моделей застосовано якісний аналіз результатів генерування, оснований на експертній оцінці виокремлених критеріїв. Вибір цих показників охоплює семантичну відповідність отриманого тексту до вмісту зображення, збереження початкового змісту підпису, відповідність заданому стилю та граматичну коректність.

3.2. Тестування моделі генерування підписів

Для базового підходу завдання генерування текстових описів до зображень спочатку застосовано класичну архітектуру типу Encoder–Decoder. Вона поєднує згорткову нейронну мережу (CNN) та рекурентну (RNN).

Кодувальником слугувала модель ResNet-50, яка експлуатується як фіксований екстрактор ознак. Ваги попередньо натренованої мережі заморожено, а додатковий лінійний шар проєктував отримані ознаки в простір ембедінгів. Декодувальник реалізовано за використання LSTM, яка генерувала текстовий опис токен за токеном. На вхід моделі надавався вектор ознак зображення як початковий контекст, після чого використано словники

попередніх слів для передбачення наступних мовних одиниць. Навчання здійснювалося із застосування керованого навчання з підстановкою правильних відповідей.

Основними параметри навчання послуговувалися функція втрат CrossEntropyLoss, оптимізатор: Adam, кількість епох становила 25, а час навчання становив 12 годин. Динаміка функції втрат під час навчання представлена на рисунку 3.2.1.

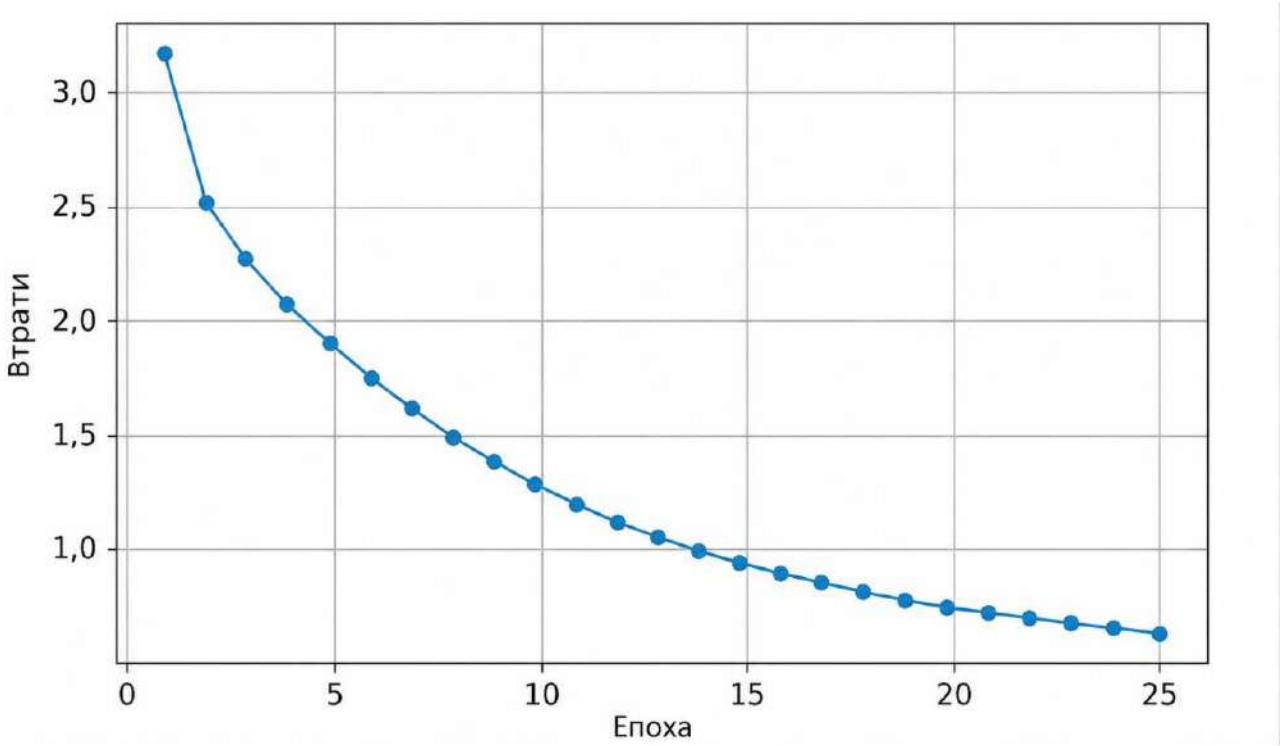

Рисунок 3.2.1 Динаміка функції втрат моделі генерування текстових описів початкової архітектури

Аналіз графіку демонструє, що значення функції втрат монотонно зменшується від значення 3,2 на початковій епосі до 0,63 на 25-й епосі. Такі показники свідчать про успішну оптимізацію моделі та відсутність явних проблем зі збіжністю.

Для оцінювання якості застосовано метрики BLEU-1–4, отримані результати продемонстровано на рисунку 3.2.2. Вони свідчать, що модель належно відтворює окремі слова (BLEU-1 медіана ≈ 0,37; середнє ≈ 0,42),

помітне зниження якості на рівні пар слів (BLEU-2 медіана ≈ 0,19; середнє ≈ 0,28) і рідко генерує довгі коректні фрагменти тексту (BLEU-3 медіана ≈ 0,08; середнє ≈ 0,20 та BLEU-4 медіана ≈ 0,04; середнє ≈ 0,17).

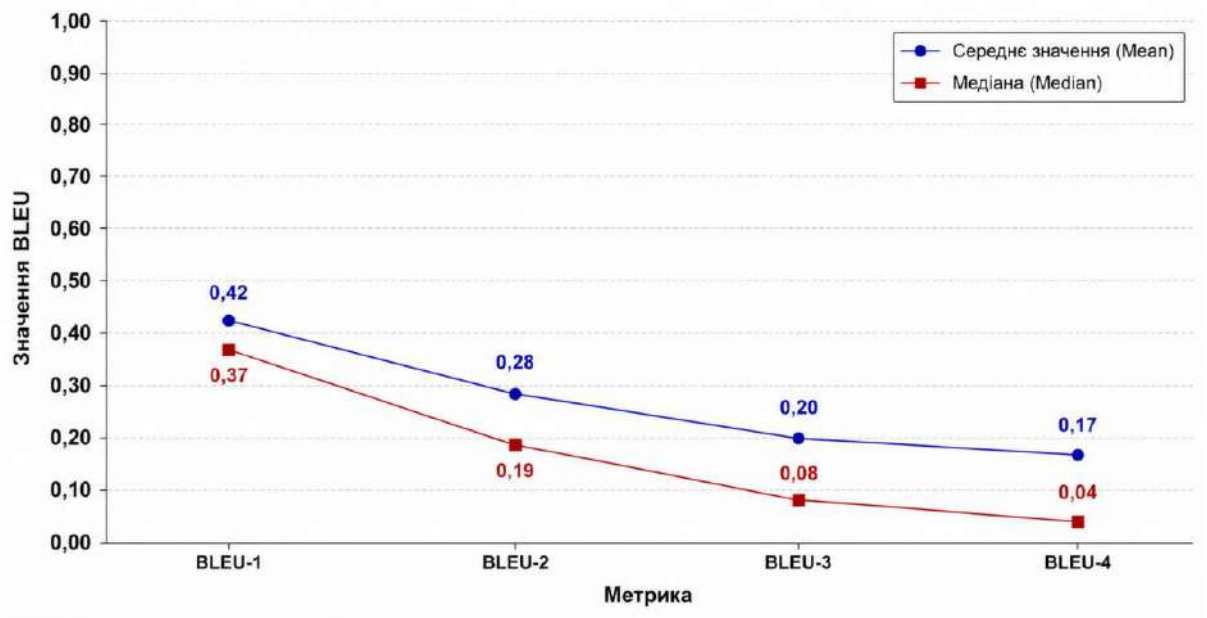

Рисунок 3.2.2 Метрики оцінки текстових описів до фото початкової архітектури моделі

Попри належну збіжність функції втрат, результати за BLEU-метриками виявилися обмеженими. Під час навчання модель отримує правильні попередні токени, але під час інференсу користується власними передбаченнями. Це призводить до накопичення помилок у послідовності. LSTM має обмеження під час роботи з довгими текстами, що проявляється в різкому падінні BLEU-3 та BLEU-4. ResNet-50 видає глобальний вектор ознак, що призводить до втрати локальних деталей зображення. Застосування жадібного декодування призводить до менш різноманітних підписів і застрягання в шаблонних фразах.

Збільшення кількості епох до 100 призвело до подальшого зниження функції втрат (до 0,4), що свідчило про покращення апроксимації тренувальних показників. Однак це не дало суттєвого покращення BLEU-метрик, що вказує на обмежену здатність моделі до узагальнення та можливе перенавчання. Результати продемонстровано на рисунку 3.2.3.

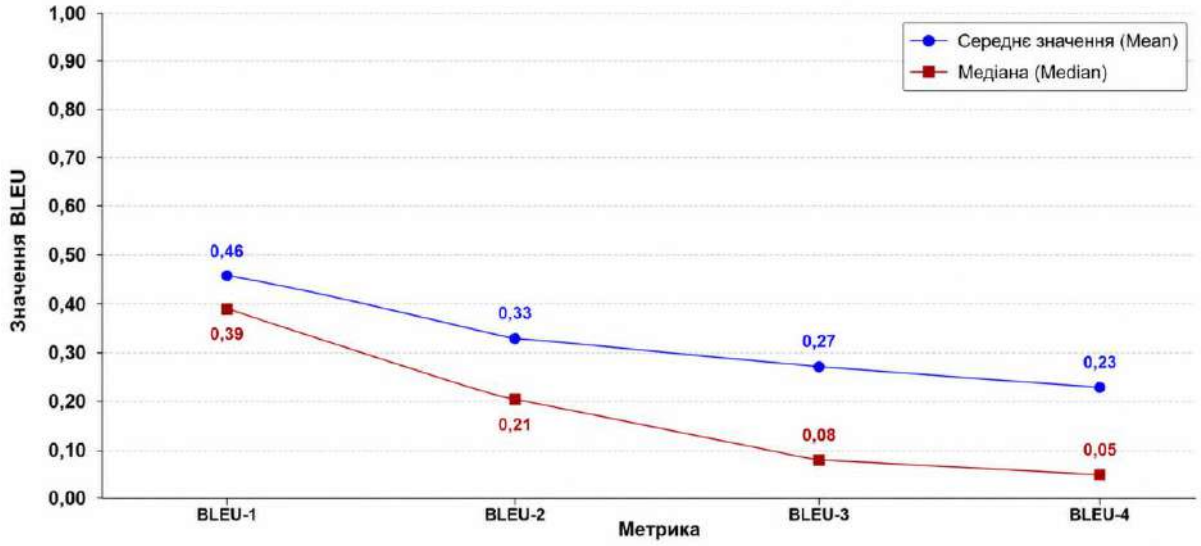

Рисунок 3.2.3 Метрики оцінки описів до фото початкової архітектури зі збільшенням епох

Для зображення з квітами модель згенерувала підпис «a man in a hat is lying in a restaurant». Отже, вона не розпізнає пріоритетні об’єкти сцени й, загалом, генерує семантично нерелевантні описи.

Через описані обмеження змінено архітектуру на попередньо навчений мультимодальний трансформерний підхід. Докладні параметри моделі, налаштування та процес інтеграції наведені в Розділі 2. Графік функції втрат демонструє відмінну поведінку від попередньої структури з CNN-LSTM. На початкових епохах відбувається різке падіння loss (з ~0,68 до ~0,32) з браком коливання або нестабільності навчання. Результати кривої переходять на плато. Графік навчання продемонстровано на рисунку 3.2.4. Характеристика демонструє швидку збіжність моделі й брак необхідності в тривалому навчанні.

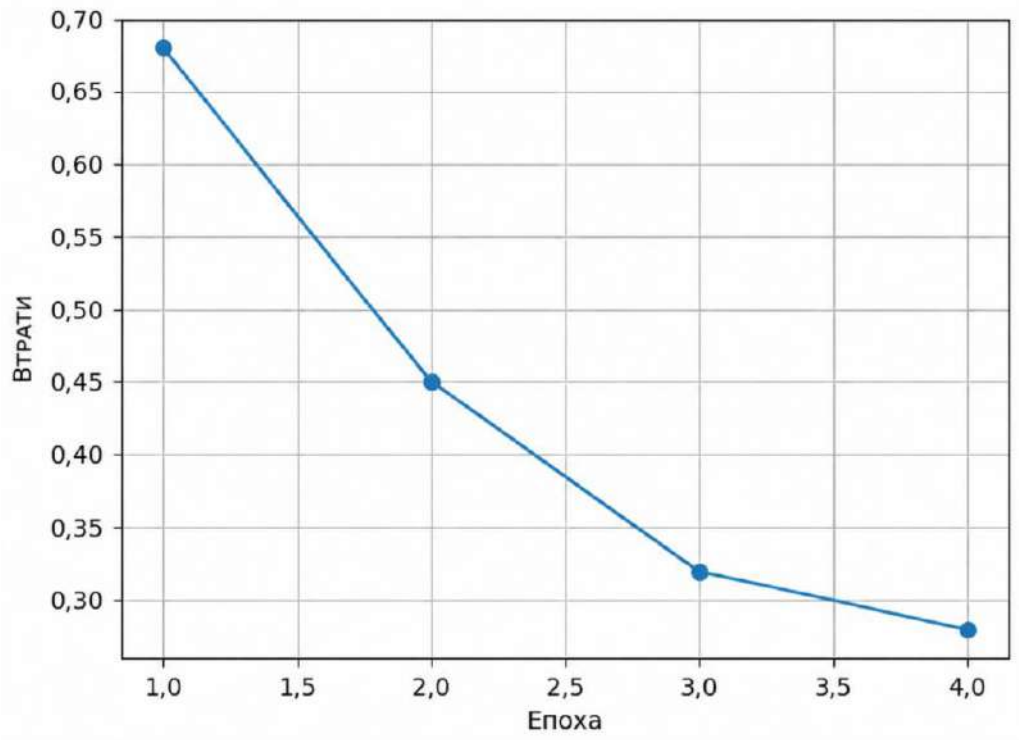

Рисунок 3.2.4 Функція втрат мультимодальної моделі генерування описів до фото

Розподіл BLEU-метрик має значно кращі характеристики в порівнянні з попередньою архітектурою:

- BLEU-1: медіана ≈ 0,45.
- BLEU-2: медіана ≈ 0,30.
- BLEU-3: медіана ≈ 0,19.
- BLEU-4: медіана ≈ 0,12.

Результати представлені на рисунку 3.2.5.

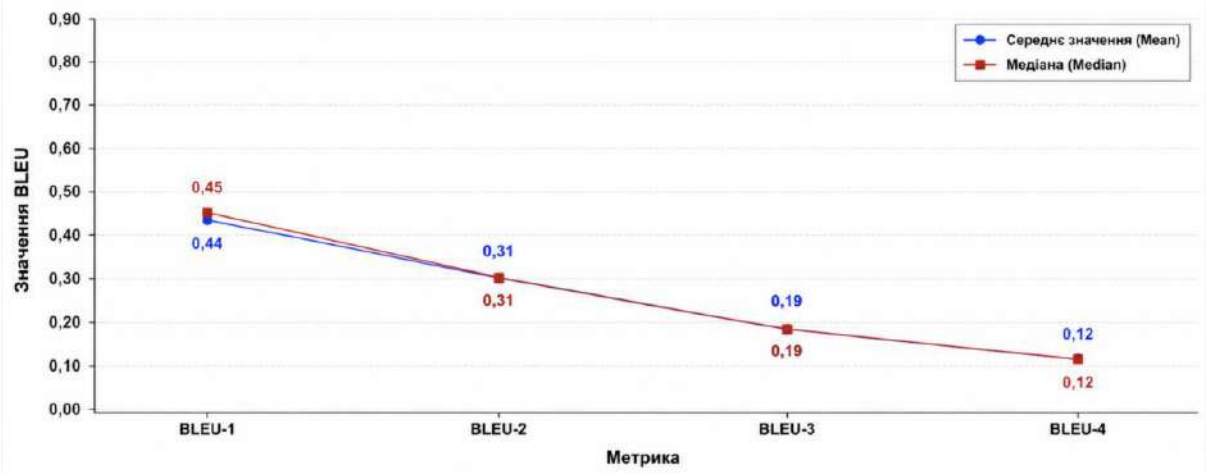

Рисунок 3.2.5 Метрики оцінки генерування описів до фото мультимодальної моделі

Отже, розподіли мають менше шуму, а результати демонструють значно більше генерувань із істотними значеннями. За загальним процесом створення підписів BLIP демонструє коректне розпізнавання об’єктів, природну граматику і браком галюцинацій для нерелевантних зображень.

3.3. Тестування моделі стилізації тексту

На першому етапі формування власного набору даних для завдання стилізації застосовано результати генерування підписів від першої моделі. Зображення подавалися їй на вхід для отримання базових описів, які стали основою для створення нової вибірки.

Для реалізації застосовано модель T5ForConditionalGeneration, архітектуру якої докладно описано в Розділі 2. Результати першого підходу до навчання за 50 епох, часового проміжку в 15 годин показали, що фінальна функція втрат становила 0,568. Графік динаміки функції втрат продемонстровано на рисунку 3.3.1.

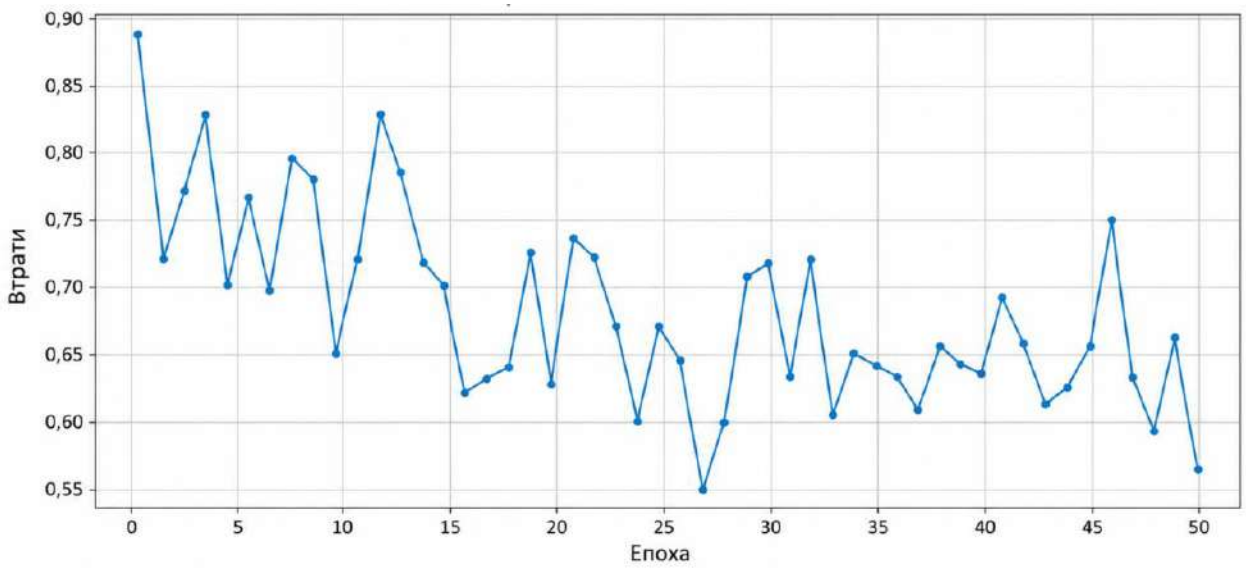

Рисунок 3.3.1 Графік динаміки функції втрат початкового етапу навчання моделі стилізації тексту

Спостережувана динаміка функції втрат характеризується нерівномірним та нестабільним зниженням її значень. Брак монотонної спадної тенденції та плавної конвергенції свідчить про те, що процес оптимізації не досягає стійкого мінімуму. Зокрема, це вказує на недостатню узгодженість і однорідність навчальної збірки, що ускладнює формування стабільних залежностей між вхідними та цільовими послідовностями. У результаті отримано нестабільність процесу навчання та зниження здатності моделі до задовільного узагальнення.

З метою усунення виявлених недоліків створено новий набір синтетичних даних із різноманітнішою контрольованою стилізацією. Після переходу спостерігається суттєве покращення як процесу навчання, так і якості отриманих результатів. Графік на рисунку 3.3.2 демонструє стабільне та монотонне зниження значення функції втрат протягом усього процесу навчання. На початкових етапах відбувається різке зниження збитку (loss), що свідчить про швидке засвоєння базових закономірностей. Надалі спостерігається плавна та стабільна конвергенція, відсутні значні коливання, характерні для попереднього підходу, а фінальні значення наближаються до нуля.

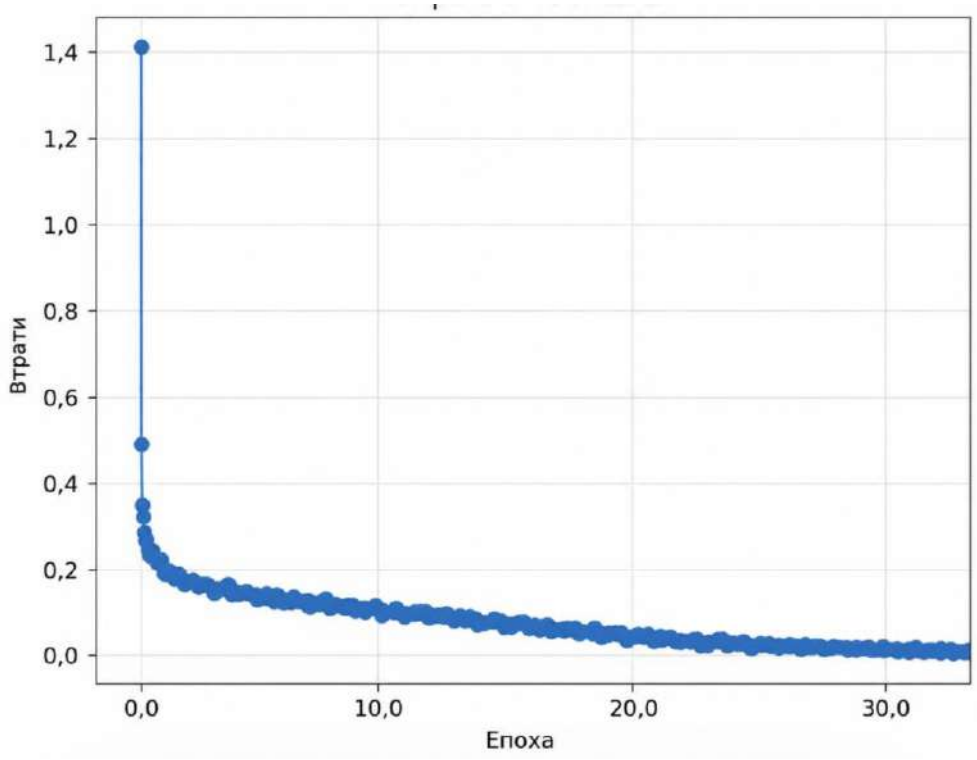

Рисунок 3.3.2 Графік функції втрат кінцевої моделі стилізації тексту

За результатами експертизи залежності функції втрат від кількості епох вдається зробити висновок, що після 25–30 епох спостерігається суттєве уповільнення зменшення значення втрат та вихід кривої на плато. Такі показники свідчать про те, що модель уже засвоїла основні закономірності навчальних відомостей. З метою запобігання перенавчанню (overfitting) та забезпечення кращої узагальнювальної спроможності моделі, для фінального послуговування обрано ваги, отримані на одній із епох у діапазоні стабілізації.

Розподіл BLEU-метрик демонструє загалом істотний рівень якості генерування тексту на рисунку 3.3.3. BLEU-1 та BLEU-2 показують, що середні значення перебувають на рівні ≈ 0,91. BLEU-3 та BLEU-4 так само демонструють суттєві значення (≈ 0,9).

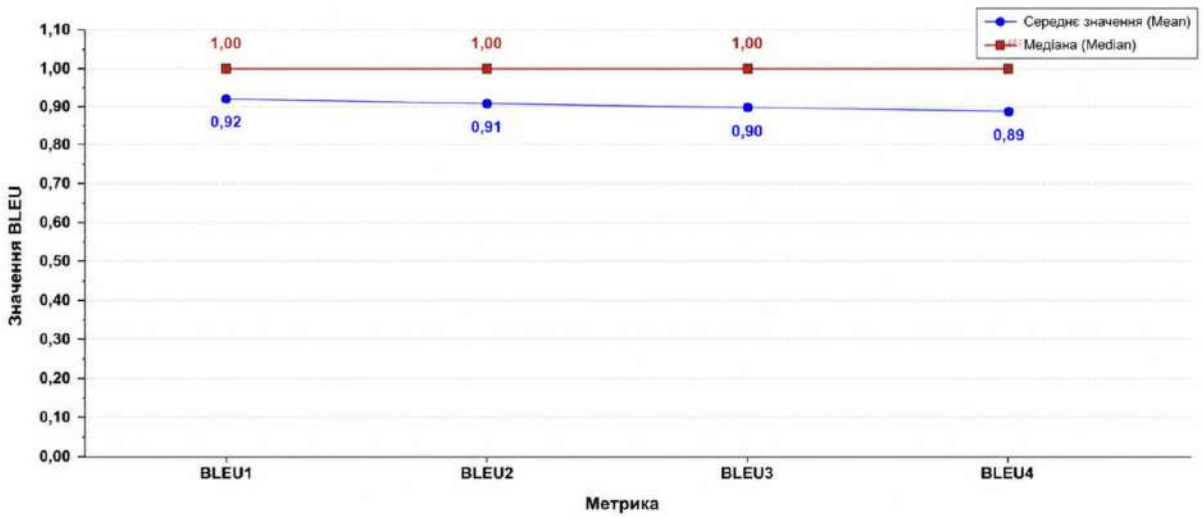

Рисунок 3.3.3 Аналіз метрик якості кінцевої моделі стилізації тексту

Отже, структура здатна коректно відтворювати довші залежності в тексті, хоча присутні поодинокі викиди з несуттєвими значеннями, їхня кількість є суттєво меншою, аніж у попередньому підході. Отримані результати підтверджують, що застосування оновленого синтетичного набору в поєднанні з інструкційно-налаштованою моделлю FLAN-T5 забезпечує суттєве підвищення якості отриманих результатів завдання стилізації.

3.4. Тестування конвеєра моделей

У межах заданої діяльності проведено тестування розробленого конвеєра, що поєднує модель генерування підписів до зображень та модель стилізації тексту. Архітектуру конвеєра обробляння докладно описано в Розділі 2.

Особливістю поставленого завдання є брак еталонного набору даних, що містить стилізовані підписи для невикористаних зображень. Це унеможливлює пряме застосування автоматичних метрик для оцінення повного конвеєра.

Згідно з цим ускладненням експертизу виконано через поєднання кількісного аналізу окремих компонентів системи, проведеного на попередніх етапах і якісного аналізу результатів діяльності інтегрованого конвеєра. Загалом, модель генерування підписів продемонструвала здатність формувати семантично правильні описи зображень, а модель стилізації показала суттєві

значення BLEU-метрик на підготовленій збірці.

Оцінювання кінцевих результатів виконання конвеєра здійснювалося з використанням комплексного розгляду, що враховує семантичну відповідність тексту до вмісту зображення, збереження початкового змісту без його спотворення, коректність стилістичної трансформації відповідно до заданого стилю, та додатково граматичну правильність. Проведений аналіз показав, що базові описи, сформовані першою програмою, адекватно відображають зміст зображень, тоді як друга структура належно виконує їхню стилізацію, забезпечуючи узгоджене перетворення тексту.

Для ілюстрації використання запропонованого конвеєра розглянемо приклад генерування стилізованих підписів до вхідного зображення. Базовий опис, отриманий першою моделлю має вигляд: «A blond girl standing in front of a band», що відображає нейтральний опис сцени. На подальшому етапі друга модель виконує стилістичну трансформацію цього підпису. Зокрема, для стилю Instagram текст доповнюється емоційними маркерами та хештегами («#goodtimes, #cozy, #smile»). У стилі Formal відбувається ускладнення синтаксичної структури та формалізація викладу («highlighting scene activity recorded clearly observed»), що наближає текст до описового наукового стилю. Водночас стиль Funny передбачає доповнення гумористичними інтерпретаціями («Breaking science news: girls discovered caffeine works even without coffee»). Отже, наведений приклад демонструє здатність інструменти отримувати кінцевий підпис до заданого фото та жанру.

3.5. Апробація API та SPM модуля

Проведено апробацію розробленого програмного засобу, що охоплює серверний API для оброблення зображень та клієнтський SPM-модуль для інтеграції в мобільний застосунок.

Для перевірки результативності виконано серію тестових запитів із різними вхідними даними (зображення відмінного змісту; стилі підписів).

Отже, результати тестування підтвердили коректність роботи API,

оскільки запити обробляються без помилок; час відповіді є прийнятним для практичного послуговування (в межах 10–30 секунд); результати є узгодженими та відповідають заданим параметрам.

У процесі апробації перевірено коректність формування запитів до API через модуль; оброблення відповідей та відображення результатів у застосунку; стабільність діяльності модуля в різних сценаріях використання.

На рис. 3.5.1 наведено приклад роботи системи, що охоплює вхідне зображення, згенерований базовий опис і стилізований результат відповідно до заданого стилю.

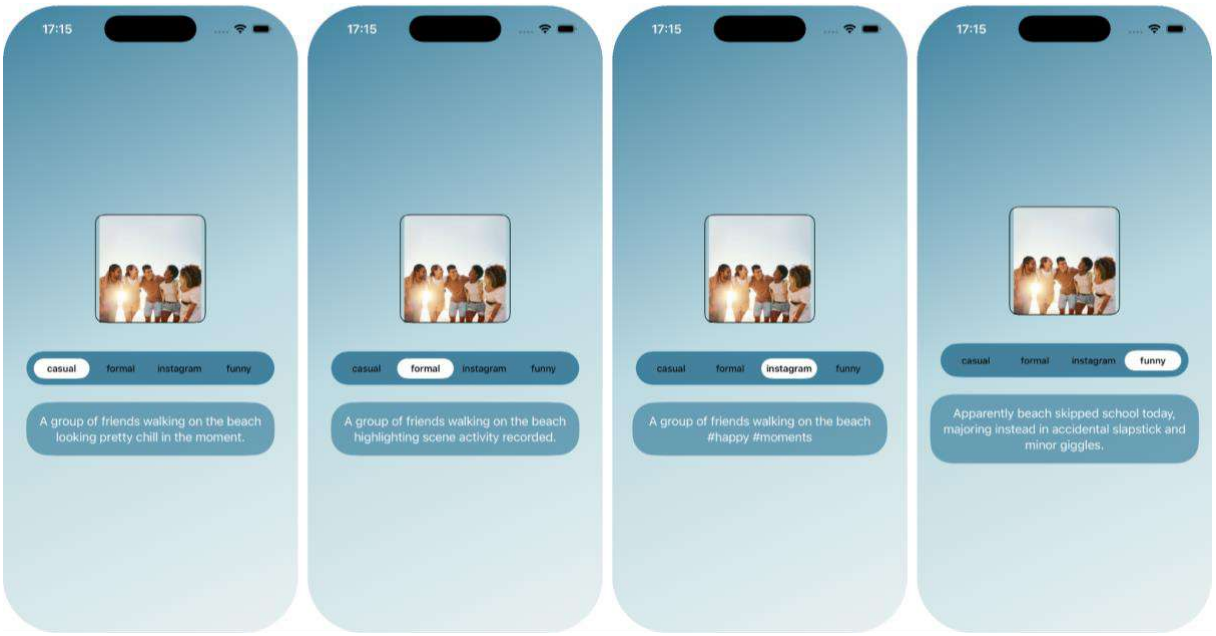

Рисунок 3.5.1 Приклад роботи системи модуля в застосунку

Отримані результати апробації підтверджують працездатність реалізованого рішення та можливість його практичного застосування в мобільних застосунках.

3.6. Висновок

Результатом проведеного тестування та апробації розробленого комплексу є цілісна робоча система підходу до генерування та стилізування текстових підписів до зображень. Кількісна експертиза окремих моделей

засвідчує здатність першого компоненту до генерування підписів із семантично коректним відображенням вмісту зображень. Друга компонента стилізації демонструє вагомі значення BLEU-метрик та здатність до відтворення стилістичних особливостей кінцевого тексту.

Дослідження динаміки навчання засвідчило стабільність процесу оптимізації, дало змогу окреслити оптимальний момент зупинки з метою запобігання перенавчанню.

Для повного конвеєра оцінювання структури здійснювалося за використання якісного аналізу результатів. Підсумок демонстрував узгодженість роботи компонентів: збереження семантичної цілісності опису за одночасної коректної жанрової трансформації.

Апробація серверного API та SPM-модуля засвідчила працездатність реалізованого програмного рішення, коректність оброблення запитів та стабільність взаємодії між клієнтською та серверною частинами.

Отже, запропонована система демонструє гнучкість і задовільні результати виконання поставленого завдання.

Висновок

Мету роботи, тобто автоматизацію процесу генерування стилізованих описів до зображень згідно з наданим визначеним жанром для застосування в мобільних пристроях у різних предметних галузях, досягнуто. Оскільки виконано поставлені завдання: проведено розгляд сучасних підходів для генерування підписів через мультимодальні моделі; досліджено архітектури програмних комплексів із відкритим кодом і окреслено методи їхнього налаштування; створено збірку синтетичних даних для навчання програми з урахуванням стилів; реалізовано, натреновано та протестовано мультимодальну систему автоматичного генерування підписів до зображень із параметром жанру; забезпечено автономне гостинґ-рішення; розроблено Swift-модуль для локальної інтеграції засобу в мобільні застосунки; проведено експериментальне тестування комплексу за метриками та оцінено якість стилізації.

Отже, уперше реалізовано інтегровану систему автоматичного генерування стилізованих текстових підписів до зображень, яка поєднує мультимодальну трансформерну модель створення описів із послідовною структурою стилізації тексту, розроблення комбінованої збірки синтетичних знань для жанрування та додаткове інтегрування програмної архітектури з підтриманням API та SPM-модуля.

Проведене тестування та апробація підтвердили результативність розробленої структури та її можливості генерувати змістовні та стилістично різноманітні підписи до зображень. Отримані висновки демонструють доцільність експлуатації трансформерних моделей у завданнях мультимодальної генерації тексту. Це підтверджується актуальністю персоналізованих стилістично забарвлених описів серед галузей соціальних мереж, месенджерів, навчальних платформ тощо. Практичне значення отриманих результатів підтверджено впровадженням одержаного механізму для генерування, яке відбувається в межах соціальних мереж.

Використані джерела

1. Deep Learning Approaches for Image Captioning: Opportunities, Challenges and Future Potential / A. Jamil et al. *IEEE Access*. 2024. P. 1. URL: https://doi.org/10.1109/access.2024.3365528 (date of access: 16.10.2025).

2. Packiaraj Kasi Rajan. On-Device ML Personalization Workflow: From User Input to Adapted Models. *Journal of Information Systems Engineering and Management*. 2025. Vol. 10, no. 59s. P. 654–661. URL: https://doi.org/10.52783/jisem.v10i59s.12928 (date of access: 26.10.2025).

3. Zhao W., Wu X., Zhang X. MemCap: Memorizing Style Knowledge for Image Captioning. *Proceedings of the AAAI Conference on Artificial Intelligence*. 2020. Vol. 34, no. 07. P. 12984–12992. URL: https://doi.org/10.1609/aaai.v34i07.6998 (date of access: 26.10.2025).

4. Tveit A., Morland T., Røst T. B. DeepLearningKit – an GPU Optimized Deep Learning Framework for Apple's iOS, OS X and tvOS developed in Metal and Swift. 2016. URL: https://arxiv.org/pdf/1605.04614. (date of access: 26.10.2025).

5. Hitoishi D. Image Captioning using Convolutional Neural Networks and Long Short Term Memory Cells. *International Journal of Recent Technology and Engineering (IJRTE)*. 2022. Vol. 11, no. 1. P. 91–95. URL: https://doi.org/10.35940/ijrte.E6741.0511122 (date of access: 29.11.2025).

6. Vikash Kumar Singh E. a. Review Paper on Enhanced Image Captioning with Deep Learning: Encoder-Decoder and Attention Mechanism. *International Journal on Recent and Innovation Trends in Computing and Communication*. 2023. Vol. 11, no. 9. P. 733–738. URL: https://doi.org/10.17762/ijritcc.v11i9.8866 (date of access: 29.11.2025).

7. Nadeem Q., Dewaji I., Khan N. *Vision Transformer-Based Image Captioning for the Visually Impaired. AHFE International*. 2025. URL: https://doi.org/10.54941/ahfe10059725964 (date of access: 29.11.2025).

8. Malik H., Shamshad F., Naseer M., Nandakumar K., Khan F., Khan S. *Robust-LLaVA: On the Effectiveness of Large-Scale Robust Image Encoders for*

Multi-modal Large Language Models. arXiv. 2025. URL: https://doi.org/10.48550/arxiv.2502.01576 (date of access: 29.11.2025).

9. Kumar M. Emotion Recognition in Natural Language Processing: Understanding How AI Interprets the Emotional Tone of Text. *Journal of Artificial Intelligence & Cloud Computing*. 2024. P. 1–5. URL: https://doi.org/10.47363/jaicc/2024(3)e238 (date of access: 02.01.2026).

10. Cheng P., Li R. *Replacing Language Model for Style Transfer. arXiv.* 2022. URL: https://doi.org/10.48550/arXiv.2211.07343 (date of access: 02.01.2026).

11. Guo Z., Rao Y. *A review of unsupervised text style transfer based on deep learning. Proceedings Volume 12709, Seventh International Conference on Computer Science and Application Engineering.* 2023. URL: https://doi.org/10.1117/12.2684814 (date of access: 02.01.2026).

12. Rashno E., Eskandari A., Anand A., Zulkernine F. *Survey: Transformer-based Models in Data Modality Conversion. arXiv.* 2024. URL: https://doi.org/10.48550/arxiv.2408.04723 (date of access: 02.01.2026).

13. Jain A., Vatsa M., Singh R. *Words Over Pixels? Rethinking Vision in Multimodal Large Language Models. Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence.* 2025. P. 10481–10489. URL: https://doi.org/10.24963/ijcai.2025/1164 (date of access: 02.01.2026).

14. Sridhar Rao Muthineni. Advancements in mobile AI: A machine learning-driven approach to enhance user experience and functionality. *World Journal of Advanced Research and Reviews*. 2024. Vol. 24, no. 3. P. 2536–2546. URL: https://doi.org/10.30574/wjarr.2024.24.3.3998 (date of access: 05.01.2026).

15. Hu H., Huang Y., Chen Q., Zhuo T. Y., Chen C. *A First Look at On-device Models in iOS Apps. arXiv.* 2023. URL: https://doi.org/10.48550/arxiv.2307.12328 (date of access: 05.01.2026).

16. Packiaraj Kasi Rajan. On-Device ML Personalization Workflow: From User Input to Adapted Models. *Journal of Information Systems Engineering and Management*. 2025. Vol. 10, no. 59s. P. 654–661. URL: https://doi.org/10.52783/jisem.v10i59s.12928 (date of access: 05.01.2026).

17. Levchuk V. Y. The Universal Module for Integration of an Intelligent Assistant into iOS Applications. *Control Systems and Computers*. 2024. No. 3 (307). P. 53–59. URL: https://doi.org/10.15407/csc.2024.03.053 (date of access: 08.01.2026).

18. Kapuriya M., Lakkad Z., Shah S. Image Caption Generator Using CNN and LSTM. *International Journal of Innovative Science and Research Technology (IJISRT)*. 2024. P. 1375–1382. URL: https://doi.org/10.38124/ijisrt/ijisrt24aug851 (date of access: 16.03.2026).

19. Top Python-Based Deep Learning Packages: A Comprehensive Review / Y. M. Mohialden et al. *International Journal Papier Advance and Scientific Review*. 2024. Vol. 5, no. 1. P. 1–9. URL: https://doi.org/10.47667/ijpasr.v5i1.283 (date of access: 16.03.2026).

20. Mallinson J., Adámek J., Malmi E., Severyn *A. EdiT5: Semi-Autoregressive Text Editing with T5 Warm-Start. Findings of the Association for Computational Linguistics: EMNLP* 2022. 2022. URL: https://doi.org/10.18653/v1/2022.findings-emnlp.156 (date of access: 20.03.2026).

21. Kingma D. P., Ba J. *Adam: A Method for Stochastic Optimization. International Conference on Learning Representations.* 2015. URL: https://arxiv.org/pdf/1412.6980 (date of access: 11.03.2026).