

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Навчально-науковий інститут прикладного системного аналізу
Кафедра системного проектування

До захисту допущено:

Завідувач кафедри

_____ Вадим МУХІН

«__» _____ 2022 р.

Магістерська дисертація
на здобуття ступеня магістра
за освітньо-професійною програмою
“Інтелектуальні сервіс-орієнтовані розподілені обчислювання”
зі спеціальності 122 "Комп'ютерні науки"

на тему: «Нейромережева система синтезу українського мовлення»

Виконав (-ла):

студент II курсу, групи ДА-11мп

Самвелян Артур Рафаєлович _____

Науковий керівник:

доцент, к.т.н.

Кисельов Геннадій Дмитрович _____

Консультант з Стартап проекту:

доцент, к.т.н.

Кисельов Геннадій Дмитрович _____

Рецензент:

професор каф. ММСА

Бідюк Петро Іванович _____

Засвідчую, що у цій магістерській
дисертації немає запозичень з праць
інших авторів без відповідних
посилань.

Студент _____

Київ – 2022

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу
Кафедра системного проектування

Рівень вищої освіти – другий (магістерський)

Спеціальність – 122 "Комп'ютерні науки"

Освітньо-наукова програма – "Інтелектуальні сервіс-орієнтовані розподілені обчислення"

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Вадим МУХІН

«___» _____ 2022 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Самвелюну Артуру Рафаельовичу

1. Тема дисертації «Нейромережева система синтезу українського мовлення», науковий керівник дисертації Кисельов Геннадій Дмитрович, к.т.н, доцент,

затверджені наказом по університету від «___» _____ 20__ р. № _____

2. Термін подання студентом дисертації – 10 червня 2022 р.

3. Об'єкт дослідження

4. Предмет дослідження

5. Перелік завдань, які потрібно розробити

6. Орієнтовний перелік графічного (ілюстративного) матеріалу

7. Орієнтовний перелік публікацій

Розділ		Підпис, дата
--------	--	--------------

	Прізвище, ініціали та посада консультанта	завдання видав	завдання прийняв

9.Дата видачі завдання 27.06.2022

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Звіт з проходження практики	25 жовтня 2022	
2	Дослідження проблематики у дисертації	10 листопада 2022	
3	Аналіз даних та написання дисертації	30 листопада 2022	
4	Документування та редагування результатів роботи	10 грудня 2022	

Студент

А.Р. Самвелян

Науковий керівник

Г.Д. Кисельов

РЕФЕРАТ

магістерської дисертації Самвеліяна Артура Рафаельовича на тему
“Нейромережева система синтезу українського мовлення”

Робота виконана на 98 сторінках, містить 14 ілюстрацій, 25 таблиць.
При підготовці використовувалась література з 11 джерел.

Актуальність теми дослідження полягає в тому, що система синтезу мовлення дає компаніям можливість використовувати штучний інтелект для автоматизації і вирішення певних потреб і проблем у своєму бізнесі. Це дозволяє автоматизувати . Йдеться про розширення можливостей людей за допомогою штучного інтелекту та більш цілісного, доступного перегляду всіх бізнес-даних, щоб вони могли приймати найкращі рішення. Це дає змогу компаніям обробляти та прогнозувати використання даних для прийняття більш обґрунтованих рішень на всіх рівнях компанії та отримувати кращу видимість своїх операцій, водночас змінюючи комерційні результати.

Мета та задачі дослідження. Метою даної роботи є розробка системи синтезу українського мовлення.

Об’єкт досліджень - система синтезу українського мовлення.

Предмет досліджень - випадки використання системи синтезу мовлення, методи та архітектури побудови даної системи.

Методи досліджень. Для вирішення проблеми в даній роботі використовуються методи аналізу і синтезу, системного аналізу, порівняння, логічного узагальнення результатів, проектування логічних структур даних.

Наукова новизна. Наукова новизна дипломної роботи полягає в тому, що розроблена система є однією з перших систем синтезу українського мовлення використовуючи нові архітектури. Потенційні застосування та практична цінність результатів дипломної роботи:

1) Робота містить опис популярних архітектур для розробки систем синтезу мовлення, які можна використовувати при виборі фреймворку для створення додатку;

2) Дану систему можна запровадити як додатковий плагін для інтерактивної взаємодії з комп'ютером.

Ключові слова: Pytorch, Text-to-Speech, FastSpeech2, Tacotron2.

ABSTRACT

Artur Rafaelovych Savmelian's master's thesis on the topic

on “Text-To-Speech system for ukrainian language”

The work is completed on 98 pages, contains 14 illustrations, 25 tables. Literature from 11 sources was used in the preparation.

The relevance of the research topic is that the speech synthesis system gives companies the opportunity to use artificial intelligence to automate and solve certain needs and problems in their business. It's about empowering people with artificial intelligence and a more holistic, accessible view of all business data so they can make better decisions. It enables companies to process and predict the use of data to make more informed decisions at all levels of the company and gain better visibility into their operations, while transforming commercial results.

The purpose and objectives of the study. The purpose of this work is to develop a system for synthesis of Ukrainian speech.

Object of research - system of synthesis of Ukrainian speech.

Subject of research - use cases of speech synthesis system, methods and architectures for building this system.

Research methods. To solve the problem in this work are used methods of analysis and synthesis, system analysis, comparison, logical generalization of results, design of logical data structures.

Scientific novelty. The scientific novelty of the thesis is that the developed system is one of the first systems of synthesis of Ukrainian speech using new architectures. Potential applications and practical value of the results of the thesis:

- 1) The work contains a description of popular architectures for the development of speech synthesis systems that can be used when choosing a framework for creating an application;
- 2) This system can be implemented as an additional plug-in for interactive interaction with the computer.

Keywords: Pytorch, Text-to-Speech, FastSpeech2, Tacotron2.

3MICT

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

ВСТУП	10
1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ	13
1.1 Поняття процесу синтезу мовлення	13
1.2 Застосування синтезу мовлення	13
1.2.1 Роботи відповідачі в колл-центрах	13
1.2.2 Застосунки для людей з вадами зору	14
1.2.3 Голосові помічники	14
1.3 Загальний алгоритм синтезу мовлення	15
1.3.1 Текстовий аналізатор	16
1.3.2 Акустичний аналізатор	19
1.3.3 Вокодер (генератор мовленнєвого сигналу)	22
1.4 Математична модель системи синтезу мовлення	26
1.5 Метрики точності моделей синтезу мовлення	27
1.6 Висновки до розділу	28
2. АНАЛІЗ РІШЕНЬ ЗАДАЧІ СИНТЕЗУ МОВЛЕННЯ	29
2.1 Статистична модель синтезу мовлення	29
2.2 DCTTS	30
2.3 Tacotron 1/2	33
2.6 FastSpeech	37
2.7 FastSpeech2	40
2.6 Висновки до розділу	44
3. СИСТЕМА СИНТЕЗУ УКРАЇНСЬКОГО МОВЛЕННЯ	45
3.1 Вибір фреймворку розробки моделі синтезу мовлення	45
3.2 Опис фреймворку Pytorch	45
3.3 Розробка програми диктора для збору даних	46
3.4 Вибрана модель для тренування системи синтезу мовлення	47
3.7 Вибір існуючих моделей для порівняння та вимірювання їх якості	48
4. РОЗРОБКА СТАРТАП-ПРОЕКТУ «SaaS система синтезу мовлення»	49
4.2 ТЕХНОЛОГІЧНИЙ АУДИТ ІДЕЇ ПРОЕКТУ	52
4.3 АНАЛІЗ РИНКОВИХ МОЖЛИВОСТЕЙ	53
4.4 РОЗРОБКА РИНКОВОЇ СТРАТЕГІЇ ПРОЕКТУ	62
4.5 РОЗРОБКА МАРКЕТИНГОВОЇ ПРОГРАМИ	65

	9
4.6 ВИСНОВКИ ДО РОЗДІЛУ 4	69
ВИСНОВКИ	71
ПЕРЕЛІК ПОСИЛАНЬ	72
ДОДАТОК А	74
ДОДАТОК Б	78
ДОДАТОК В	90

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

RNN - рекурентна нейронна мережа

TDNN - нейронна мережа з часовою затримкою

CNN - згорткова нейронна мережа

DTW - алгоритм динамічної трансформації часової шкали

PLP - коефіцієнти перцептивного лінійного передбачення

LPC - лінійне передбачуване кодування

LPCC - кепстральні коефіцієнти лінійного передбачення

MFCC - мел-кепстральні коефіцієнти

CRF - умовне випадкове поле

MLP - мульти шаровий перцептрон

LSTM - нейронна мережа короткотривалої-довготривалої пам'яті

ВСТУП

В наш час взаємодія із технікою стає більш інтерактивною. З року в рік людство шукає шляхи для того щоб зробити цю взаємодію більш зручною для себе. Під цими діями виникають такі способи взаємодії як

голосове управління, в якому невід'ємною складовою є синтез мовлення наряду з автоматичним розпізнаванням мовлення та розумінням тексту.

Оскільки людям набагато зручніше вести інтерактивність та робити пошуки за допомогою голосу, методи рішення даної задачі здійснили довгий шлях: від статистичних методів, до нейромережевих, і до сих пір цей розвиток продовжується.

Одні з перших спроб синтезу голосового сигналу були основані на акустично-механічних приладах. Ще в далекому 1779 році, Німецько - Дацький вчений Христіан Готліб Кратценштейн побудував прототип людського голосового тракту, який був спроможний відтворювати п'ять довгих голосних звуків 'а', 'е', 'і', 'о' та 'у'. Згодом його модель стала основою для акустично-механічної машини Вольфганга фон Кемпелена, яку вчений удосконалив та описав у своїх працях, датованих 1791 роком. Удосконалена версія вже мала язик та губи, що дозволяло їй відтворювати приголосні звуки.

Вже сьогодні на ринку рішень ми маємо системи, які можуть генерувати людську мову досить високої якості і складаються вони не з акустичних приладів, а реалізовані повністю програмно з використанням методологій зі сфери обробки сигналів, та знань в областях штучного інтелекту і машинного навчання.

Не зважаючи на різкий прогрес у сфері обробки та синтезу мовлення, до сих пір є проблеми таких систем до вирішення яких підходять з різних боків, тобто для яких не використовують один метод вирішення. Однією з таких проблем є проблема довгих фраз. Дана проблема виникає коли довга фраза подається до система синтезу мовлення та отримується голосовий сигнал, в якому кінець фрази, починаючи з визначеного моменту, не озвучується, а затихає.

Також не відкритим, але і до кінця невирішеним залишається питання оцінки якості генерованого голосового сигналу. Оскільки людина, як отримувач інформації, є ключовим елементом взаємодії з системою, то в

даному випадку приходить логічний висновок, що саме вона має вирішувати на скільки задовільною є якість, якої досягає система. Але якщо взяти до уваги варіацію голосів, а найголовніше різноманітність мов – вже стає неабияким завданням. Наприклад, нам, українцям, доволі просто оцінити характер звучання української, чи російської мови, адже вони дуже схожі. А якщо питання постане в оцінці якості синтезу мови, не притаманної нашим регіонам, наприклад, китайської – цю оцінку можна вважати не компетентною. Адже ми просто не зможемо правильно охарактеризувати результат системи по таким важливим параметрам як натуральність та розбірливість звучання.

Дані наявні проблеми і недоліки в сфері синтезу мовлення роблять її цікавою і перспективною для користувачів та розробників зі всього світу. Тому на даний момент аналіз та розробка моделі для синтезу мовлення є доцільною, як ніколи.

Результуюча модель зможе знайти своє практичне застосування в різних сферах і закріпити результат який люди змогли досягти на сьогодні.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття процесу синтезу мовлення

Синтез мовлення - це процес перетворення текстової інформації в мовленнєвий сигнал.

Девайс, який здійснює дане перетворення називається синтезатором мовлення і може бути реалізований за допомогою різних підходів та архітектур.

1.2 Застосування синтезу мовлення

Процес синтезу мовлення широко застосовуються у нашому повсякденному житті. Починаючи з років цифрової революції розумні девайси та смартфони стали невід'ємною частиною нашого життя, тому людина постійно придумує запроваджує нові технології які можуть спростити їй життя.

1.2.1 Роботи відповідачі в колл-центрах

Існує багато розгортань систем перетворення тексту в мову в рамках IVR. IVR розшифровується як інтерактивна голосова відповідь. Це автоматизована, інтерактивна система, яка забезпечує обслуговування клієнтів, доповнюючи або замінюючи людського агента.

TTS створює систему IVR, яка не обмежується словами і фразами, з якими вона була запрограмована.

Створення або оновлення IVR зазвичай означає роботу з голосовим актором. Все, що вам коли-небудь може знадобитися сказати, ви повинні заздалегідь продумати і записати. Само по собі це дорого та забирає багато часу, і це ускладнює розвиток вашого IVR. Тому системи синтезу мовлення в даному випадку можуть допомогти зекономити багато часу та сил тим, що беруть на себе відповідальність по генеруванню аудіосигналу, а

відповідачам в колл-центрах залишається лише зберігати необхідні фрази у вигляді тексту.

1.2.2 Застосунки для людей з вадами зору

Порушення зору є однією з найскладніших інвалідностей, які може мати людина, оскільки воно пов'язане з втратою одного з найважливіших людських відчуттів - зору. До довгого переліку речей, в яких людина з вадами зору може мати обмеження, додається доступ до Інтернету. Деякі частини незрячої спільноти були відрізані від решти суспільства, особливо в онлайн-спільноті, через відсутність інструментів, що надають. На щастя, розвиток технологій зробив неможливе можливим для незрячих завдяки використанню програмного забезпечення Text-To-Speech.

1.2.3 Голосові помічники

Синтез мовлення займає ключову роль й у технології голосового помічника.

Голосові помічники, або як їх ще називають, віртуальні помічники - це програмні продукти які за допомогою технологій автоматичного розпізнавання мовлення, аналізу та розуміння тексту та синтезу мовлення проводять пряме спілкування з людиною і виконують її команди.

Дані програмні продукти складаються із трьох функціональних частин:

- 1) Розпізнавання мовлення
- 2) Розуміння та аналіз тексту
- 3) Синтез мовлення

При надходженні звукового сигналу до помічника, розпізнавач мовлення проводить перетворення цього сигналу в текст, який потім йде до

модулю аналізу та розуміння тексту. Модуль розуміння тексту аналізує вхідний текст і видає якусь відповідь також у вигляді тексту. Вихідний текст йде до модулю синтезу мовлення який вже генерує аудіо сигнал з цього тексту і програвляє його. Дана технологія з роками набуває все більше популярності.

1.3 Загальний алгоритм синтезу мовлення

Як і в будь-якій задачі машинного навчання, для вирішення задачі синтезу мовлення використовуються як нейромережеві методи так і класичні методи машинного навчання (без вчителя).

Типова система синтезу мовлення складається з компонентів, які включають текстовий аналіз, лінгвістичний аналіз (акустичну модель) та генератор мовленнєвого сигналу. Текстовий аналізатор дбає про приведення тексту у відповідний вигляд (нормалізація, фонетизація, проставлення наголосів). Приведено до належного вигляду текст прямує до акустичної моделі. Акустична модель (лінгвістичний аналізатор) бере на вхід ці дані, та перетворює їх на спектрограму. Далі починається черга генератора мовленнєвого сигналу. Його головна задача - перетворити спектрограму у мовленнєвий сигнал максимально якісно.

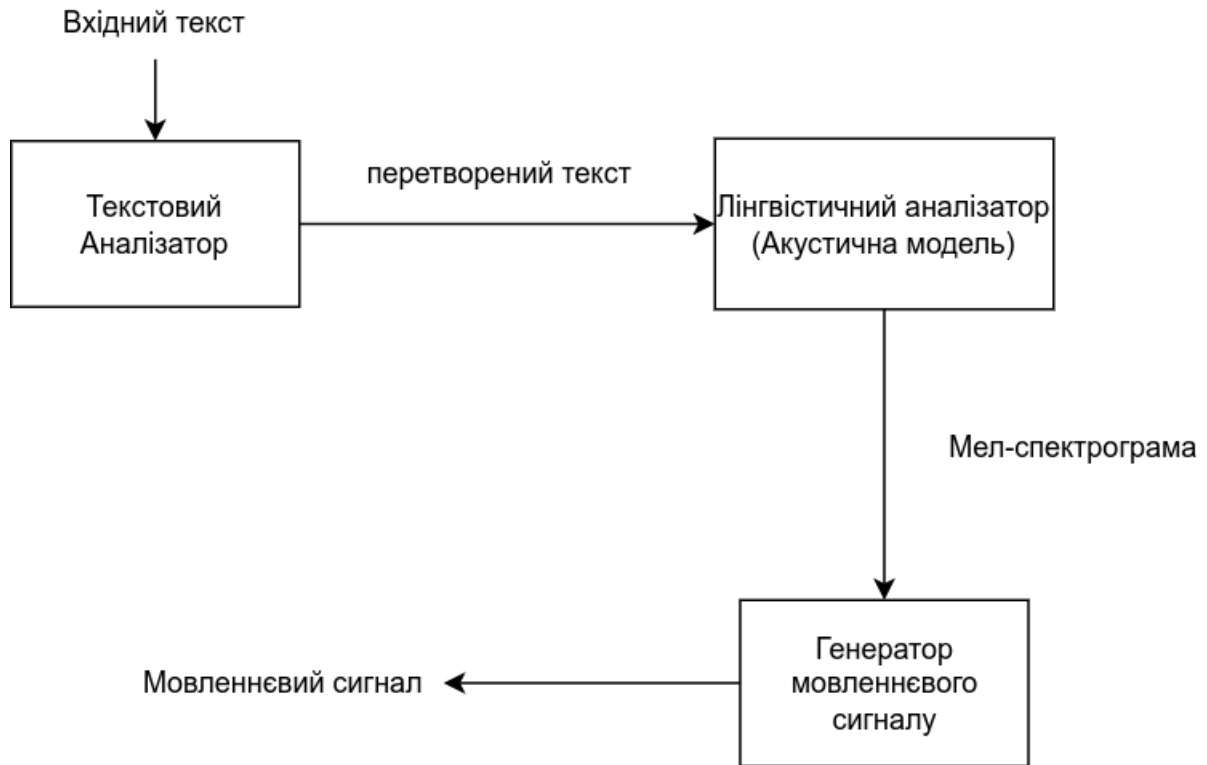


Рисунок 1.1 - Загальна архітектура системи синтезу мовлення

Однак, процес синтезу мовлення в різних підходах (фреймворках) може містити різні частини з вищенаведеної схеми, або не містити однієї чи двох частин.

1.3.1 Текстовий аналізатор

Аналіз тексту, який в TTS також називається інтерфейсом, перетворює вхідний текст в лінгвістичні ознаки, які містять багато інформацію про вимову та просодію для полегшення синтезу мовлення.

У статистичному параметричному синтезі аналіз тексту використовується для вилучення послідовності векторів лінгвістичних ознак, і містить кілька функціональних можливостей, таких як нормалізація тексту, сегментація слів, маркування частин мови (POS) тегів, прогнозування просодії та перетворення графем у фонемі. У наскрізних нейронних TTS, завдяки великій моделюючій здатності нейронних моделей, символ або послідовності символів або фонем безпосередньо беруться як

вхідні дані для синтезу, і таким чином модуль аналізу тексту значно спрощується.

У цьому сценарії нормалізація тексту все ще необхідна для отримання стандартного формату слів з символьного вводу, а для отримання фонем із стандартного формату слова додатково потрібне графемно-фонемне перетворення. зі стандартного формату слів.

Хоча деякі моделі TTS заявляють про повністю наскрізний синтез, який безпосередньо генерує форму сигналу з тексту, нормалізація тексту все ще необхідна для обробки необробленого тексту з будь-якими можливими нестандартними форматами для практичного використання. Крім того, деякі наскрізні моделі TTS включають в себе звичайні функції аналізу тексту.

Наприклад, Char2Wav та DeepVoice 1/2 реалізують перетворення символів у лінгвістичні ознаки у своєму конвеєрі виключно на основі нейронних мережі, а деякі роботи явно передбачають особливості просодії за допомогою кодера тексту. У решті цього підрозділу ми спочатку представимо типові задачі аналізу тексту при статистичному параметричному синтезі, а потім обговорюється розвиток аналізу тексту в наскрізних моделях TTS.

Узагальнимо деякі типові задачі аналізу текстів, а також наведемо деякі репрезентативні роботи для кожної задачі наступним чином.

Нормалізація тексту

Вихідний письмовий текст (нестандартні слова) повинен бути перетворений в усну форму за допомогою нормалізації тексту. слова в розмовну форму шляхом нормалізації тексту, що може зробити слова легкими для вимови для TTS-моделей. моделей TTS. Наприклад, рік "1989" нормалізується в "дев'ятнадцять вісімдесят дев'ять", "24 січня" нормалізовано до "двадцять четвертого січня". Ранні роботи з нормалізації тексту базуються на правилах, а потім нейронні мережі використовуються

для моделювання нормалізації тексту від послідовності до послідовності де вихідною та цільовою послідовностями є нестандартні слова та слова у розмовній формі відповідно. Останнім часом в деяких роботах пропонується поєднати переваги моделей на основі правил та нейронних моделей для подальшого покращення продуктивності нормалізації тексту.

Сегментація слів

Для ієрогліфічних мов, таких як китайська, сегментація слів необхідна для виявлення межі слова з вихідного тексту, що важливо для забезпечення точності для подальшого POS-тегування, прогнозування просодії та процесу перетворення графем у фонemi.

Розпізнавання частин мови

Частина мови (POS) кожного слова, такого як іменник, дієслово, прийменник, також важлива для перетворення графем у фонemi та прогнозування просодії в TTS.

Прогнозування просодії.

Просодична інформація, така як ритм, наголос та інтонація мовлення, відповідає варіаціям тривалості, гучності та висоти складів, що відіграє важливу перцептивну роль у мовленнєвому спілкуванні людини. Прогнозування просодії спирається на системи тегів для позначення кожного виду просодії. У різних мовах існують різні системи та інструменти маркування просодії. Для англійської мови популярною системою тегування є ToBI (tones and break in TTSces), яка описує теги для тонів (наприклад, висотні акценти, фразові акценти та граничні тони) та пауз (наскільки сильна пауза між словами). Наприклад, у цьому реченні "Mary went to the store ?", "Mary" і "store" можуть бути підкреслені, і це речення має підвищений тон. У багатьох роботах досліджуються різні моделі та особливості прогнозування просодичних тегів на основі ToBI. Для синтезу китайського мовлення типові просодичні граничні мітки складаються з

просодичного слова (PW), просодичної фрази (PPH) та інтонаційної фрази (IPH), які можуть побудувати тришарове ієрархічне просодичне дерево . Деякі роботи досліджують різні модельні структури, такі як CRF, RNN та self-attention для прогнозування просодії в китайській мові.

Перетворення графеми в фонему (G2P)

Перетворення символу (графеми) у вимову (фонему) може значно полегшити синтез мови. Наприклад, слово "speech" перетворюється на "s p iu ch". Для перетворення зазвичай використовується зібраний вручну лексикон графем у фонemi. Однак для алфавітних мов, таких як англійська, лексика не може охопити вимову всіх слів. Таким чином, G2P-перетворення для англійської мови в основному відповідає за генерування вимови слів, що не входять до словника. Для таких мов, як китайська, хоча лексика може охоплювати майже всі ієрогліфи, існує багато поліфонів, які можуть бути визначені тільки відповідно до контексту ієрогліфа. Таким чином, G2P-перетворення в таких мовах в основному відповідає за поліфонічне розпізнавання, яке вирішує відповідну вимову на основі поточного контексту слова. Після вищезазначеного аналізу тексту ми можемо далі побудувати лінгвістичні ознаки та взяти їх як вхідні дані для подальшої частини конвеєра TTS, наприклад, акустичні моделі в SPSS або вокодери. Зазвичай лінгвістичні ознаки можна побудувати шляхом агрегування результатів аналізу тексту з різних рівнів, включаючи рівні фонем, складів, слів, словосполучень та речень.

1.3.2 Акустичний аналізатор

Акустичні моделі в нейронних наскрізних TTS мають ряд переваг у порівнянні з моделями в SPSS:

1) Звичайні акустичні моделі вимагають узгодження між лінгвістичними та акустичними ознаками, в той час як нейронні моделі, що базуються на послідовності, неявно вивчають узгодження через увагу або прогнозують тривалість разом, що є більш ефективним. тривалість, які є більш наскрізними і вимагають меншої попередньої обробки.

2) Зі збільшенням моделюючої здатності нейронних мереж, лінгвістичні особливості спрощуються лише до послідовності символів або послідовності фонем, а акустичні ознаки змінилися з низькорозмірних і конденсованих цепстрами (наприклад, MGC) до високорозмірних мелоспектрограм або ще більш високорозмірних лінійних спектрограм. спектрограми.

Моделі на основі ШНМ (наприклад, серія Tacotron) Tacotron використовує структуру encoder-attention-decoder, приймає символи як вхідні дані і виводить лінійні спектрограми, а також використовує алгоритм Гріффіна - Ліма для генерації форми сигналу. Tacotron 2 пропонується для генерації mel-спектрограм та перетворення mel-спектрограм в осцилограму з використанням додаткової моделі WaveNet. Tacotron 2 значно покращує якість голосу в порівнянні з попередніми методами, включаючи конкатенативні TTS, параметричні TTS, параметричні TTS, нейронні TTS, такі як Tacotron. Пізніше з'явилося багато робіт, що покращують Tacotron з різних аспектів:

1) Використання еталонного кодера та токенів стилю для підвищення виразності синтезу мовлення, такі як GST-Tacotron та Ref-Tacotron.

2) Видалення механізму уваги в Tacotron, а замість нього використання тривалості предиктор тривалості для авторегресійного прогнозування, такі як DurIAN Non-attentative Tacotron.

3) Зміна авторегресійної генерації в Tacotron на не авторегресійну генерацію, наприклад, Parallel Tacotron 1/2. 4) Побудова наскрізних моделей перетворення тексту в хвильову форму на основі Tacotron, таких як Wave-Tacotron.

Моделі на базі CNN (наприклад, DeepVoice Series) DeepVoice фактично є системою SPSS, доповненою згортковими нейронними мережами. Після отримання лінгвістичних ознак за допомогою нейронних мереж, DeepVoice використовує вокодер на основі WaveNet для генерації форми сигналу. DeepVoice 2 слідує базовому потоку перетворення даних DeepVoice та покращує DeepVoice за допомогою покращених мережевими структурами та моделюванням з декількома динаміками. Крім того, DeepVoice 2 також використовує конвеєр моделей Tacotron + WaveNet, який спочатку генерує лінійні спектрограми за допомогою Tacotron, а потім генерує форму сигналу за допомогою WaveNet. DeepVoice 3 використовує повністю згорткову мережеву структуру для синтезу мовлення, яка генерує mel-спектрограми з символів і може масштабуватися до реального слова набори даних з декількома дикторами. DeepVoice 3 покращує попередні системи DeepVoice 1/2 за рахунок використання більш компактною моделлю "послідовність-послідовність" і прямим прогнозуванням мелоспектрограм, а не складних лінгвістичних особливостей. Пізніше було запропоновано ClariNet для повного наскрізного генерування форми сигналу з тексту. ParaNet - повністю згорнута не авторегресійна модель, яка може прискорити генерацію мелоспектрограм та отримати спектрограми та отримати досить хорошу якість мовлення. DCTTS має схожий конвеєр перетворення даних з Tacotron, і використовує повністю згорткову мережу кодерів-уваги-декодерів для генерації мелоспектрограм. мережу кодера-уваги-декодера для генерації mel-спектрограм з послідовності символів. Потім вона використовує спектрограму мережу надвисокої роздільної здатності для отримання лінійних спектрограм і синтезує форму сигналу за допомогою Гріффіна-Ліма.

Трансформаторні моделі (наприклад, FastSpeech Series) TransformerTTS використовує архітектуру енкодерів, засновану на нейромережеві

архітектурі transformer на основі архітектури кодера-уваги-декодера для генерації мелоспектрограм з фоном. Вони стверджують, що моделі кодера-уваги-декодера на основі RNN, такі як Tacotron 2, страждають від наступними двома проблемами:

1) Через повторювану природу, як кодер, так і декодер на основі ШНМ не можуть навчатися паралельно, а кодер на основі ШНМ не може бути паралельним у виведенні, що впливає на ефективність як при навчанні, так і при виведенні.

2) Оскільки текстові та мовні послідовності зазвичай дуже довгі, то ШНМ погано моделює довгі залежності в цих послідовностях. Трансформатор TTS приймає базову структуру моделі Transformer і поглинає деякі конструкції з Tacotron 2, такі як декодер перед-мережевого/пост-мережевого передбачення та передбачення стоп-токенів. Він досягає аналогічної якості голосу з Tacotron 2, але має більш швидкий час навчання.

Однак, у порівнянні з моделями на основі ШНМ, такими як Tacotron які використовують стабільні механізми уваги, такі як увага, чутлива до місця розташування, кодер-декодер уваги в Transformer не є стійким через паралельні обчислення. Таким чином, в деяких роботах пропонується підвищити робастність акустичних моделей на основі Transformer. Наприклад, в роботі MultiSpeech показано, що доводить робастність механізму уваги через нормалізацію кодера, вузьке місце декодера та діагональне обмеження уваги, а RobuTrans використовує прогнозування тривалості для підвищення стабільності в авторегресійній генерації.

1.3.3 Вокодер (генератор мовленнєвого сигналу)

Грубо кажучи, розвиток вокодерів можна розділити на два етапи: вокодери що використовуються в статистичному параметричному синтезі мови (SPSS) , та вокодери на основі нейронних мереж вокодери. Деякі популярні

вокодери в SPSS включають STRAIGHT та WORLD. Для прикладу візьмемо вокодер WORLD, який складається з вокодерного аналізу та синтезу вокодера. На етапі аналізу вокодер аналізує мову та отримує акустичні ознаки такі як мел-кепстральні коефіцієнти, аперіодичність смуги та F0. При синтезі вокодера він генерує форму мовного сигналу з цих акустичних характеристик. У цьому розділі ми в основному розглянемо роботи на основі нейронних вокодерів через їх високу якість передачі голосу. Ранні нейронні вокодери, такі як WaveNet, Char2Wav, WaveRNN, безпосередньо беруть лінгвістичні особливості як вхідні дані та генерують форму сигналу. Пізніше, Prenger та ін., Kim та ін., Kumar та ін., Yamamoto та ін. використовують mel-спектрограми як вхідні дані та генерують форму сигналу. Оскільки мовленнєвий сигнал є дуже довгим, авторегресійна генерація сигналу займає багато часу на виведення. Таким чином, генеративні моделі, такі як Flow, GAN, VAE та DDPM (Denoising TTSffusion Probabilistic Model, скорочено TTSffusion) використовуються при генерації сигналів. Відповідно, розділимо нейронні вокодери на різні категорії:

- 1) Авторегресійні вокодери,
- 2) Вокодери на основі потоку,
- 3) Вокодери на основі GAN,
- 4) Вокодери на основі VAE та
- 5) Вокодери на основі дифузії вокодери.

Авторегресійні вокодери WaveNet - це перший нейронний вокодер, який використовує розширену згортку для авторегресійної генерації точок форми сигналу. На відміну від аналізу та синтезу вокодерів в SPSS, WaveNet майже не використовує попередніх знань про аудіосигналів, і покладається виключно на наскрізне навчання. Оригінальний WaveNet, а також деякі наступні роботи, які використовують WaveNet як вокодер, генерують форму мовного сигналу, обумовлену лінгвістичних особливостей, в той час як WaveNet може бути легко адаптований до умов лінійних спектрограм і мел-

спектрограм . та mel-спектрограмами [336, 270, 303]. Хоча WaveNet досягає хорошої якості голосу, він страждає повільною швидкістю виведення. Тому багато робіт присвячено дослідженню легких та швидких вокодерів. SampleRNN використовує ієрархічну рекурентну нейронну мережу для безумовного для безумовного генерування форми сигналу, і далі інтегрована в Char2Wav для генерування форми сигналу, залежної від акустичних особливостей. залежних від акустичних особливостей. Крім того, розроблено WaveRNN для ефективного синтезу звуку, використовуючи рекурентну нейронну мережу та використовуючи кілька конструкцій, включаючи подвійний шар softmax, вагову обрізку та методи суб масштабів для зменшення обчислень. Lorenzo-Trueba та ін. , Paul та ін. , Jiao та ін. ще більше покращують надійність та універсальність вокодерів. LPCNet вводить звичайну цифрову обробку сигналів у нейронні мережі та використовує лінійні коефіцієнти прогнозування для обчислення наступної точки форми сигналу, одночасно використовуючи легкі ШНМ для обчислення залишку. LPCNet генерує форму мовного сигналу, обумовлену BFCC (частота кори cepstral coefficients), і може бути легко адаптована до умов на mel-спектрограмах. Деякі наступні роботи вдосконалюють LPCNet з різних точок зору, таких як зменшення складності, прискорення для прискорення та покращення стабільності для покращення якості . Вокодери на основі потоку Нормалізуючий потік є різновидом генеративної моделі. Вона перетворює щільність ймовірності за допомогою послідовності інверсних відображень . Оскільки ми можемо отримати стандартний/нормований розподіл ймовірностей (наприклад, гауссівський) через послідовність інверсійних відображень відображень на основі правил зміни змінних, такий вид потокової генеративної моделі називається нормалізуючим потоком. Під час вибірки вона генерує дані зі стандартного розподілу ймовірностей за допомогою зворотних перетворень. Поточкові моделі, що використовуються в нейронних TTS, можна розділити на дві категорії відповідно до двох різних методів: 1) авторегресійні перетворення

(наприклад, зворотний авторегресійний потік, що використовується в Parallel WaveNet), та 2) двосторонні перетворення (наприклад, Glow , що використовується в WaveGlow, та RealNVP, що використовується в FloWaveNet), як показано в Таблиця 4.

- Авторегресійні перетворення, наприклад, обернений авторегресійний потік (ОАП). IAF можна розглядати як подвійне формулювання авторегресійного потоку (АП) . Навчання АП відбувається паралельно, а вибірка - послідовно. вибірка є послідовною. На відміну від цього, вибірка в МАФ є паралельною, тоді як висновок для оцінки ймовірності є послідовним. оцінки ймовірності є послідовним. Parallel WaveNet використовує дистиляцію щільності ймовірності для поєднання ефективну вибірку IAF з ефективним навчанням AR-моделювання. Він використовує авторегресійну WaveNet як мережу вчителя, щоб керувати навчанням мережі учнів (Parallel WaveNet) для апроксимації ймовірності даних. Аналогічно, ClariNet використовує IAF та дистиляцію вчителя, і використовує дивергенцію KL закритої форми для спрощення та стабілізації процесу дистиляції. Хоча Parallel Wavenet та ClariNet можуть генерувати мову паралельно, вони покладаються на складну підготовку викладача і студента і все ще вимагає великих обчислень. - Двосторонні перетворення, наприклад, Glow або RealNVP. Для того, щоб перетворення були оберненими, двосторонні перетворення використовують афінні шари зв'язку, які гарантують, що вихідні дані можуть бути обчислені за вхідними і навпаки. Деякі вокодери, засновані на двосторонніх перетвореннях, включають WaveGlow та FloWaveNet , які досягають високої якості голосу та швидкої швидкості виведення. Як авторегресійні, так і двосторонні перетворення мають свої переваги та недоліки:

1) Авторегресійні перетворення є більш виразними, ніж двосторонні перетворення, за рахунок моделювання залежності між розподілом даних x

та стандартним розподілом ймовірностей z , але потребують дистиляції викладача, що є складним у навчанні.

2) Двосторонні перетворення мають набагато простіший конвеєр навчання, але, як правило, вимагають більшої кількості параметрів (наприклад, глибші шари, більший прихований розмір) для досягнення порівнянної з авторегресійними моделями, але зазвичай вимагають більшої кількості параметрів для досягнення порівнянної з ними потужності.

Для поєднання переваг як авторегресійних, так і двосторонніх перетворень, WaveFlow надає уніфіковане уявлення про моделі, засновані на ймовірності, для аудіоданих, щоб явно обміняти паралелізм виведення на пропускну здатність моделі. Таким чином, WaveNet, WaveGlow та FloWaveNet можна розглядати як окремі випадки WaveFlow.

Вокодери на основі GAN Генеративні змагальні мережі (GAN) широко використовуються в задачах генерації даних, таких як генерація даних, таких як генерація зображень, генерація тексту та генерація звуку.

GAN складається з генератора для генерації даних і дискримінатора для оцінки автентичності згенерованих даних. згенерованих даних. Багато вокодерів використовують GAN для забезпечення якості генерації звуку, в тому числі WaveGAN, GAN-TTS, MelGAN, Parallel WaveGAN, HiFi-GAN та інші вокодери на основі GAN.

1.4 Математична модель системи синтезу мовлення

Перш ніж ми почнемо аналізувати різні архітектури, давайте розглянемо, як ми можемо математично сформулювати TTS. Нехай ми

маємо вхідну текстову послідовність Y , та хочемо знайти цільову послідовність X , тоді:

$$W = \operatorname{argmax} P(X | Y, \theta)$$

В даному випадку θ - це параметри нашої нейромережевої моделі.

1.5 Метрики точності моделей синтезу мовлення

Оцінка якості синтезованого мовлення є досить цікавим та актуальним питанням на сьогоднішній день. Велика кількість мов та варіацій звучання диктора створює проблему в виденні суто математичного методу однозначної оцінки правдоподібності сигналу. Через це найбільш популярним підходом до оцінки є проведення опитувань серед людей.

Mean opinion score (MOS) – це числовий показник загальної якості події або досвіду, що оцінюється людиною. У синтезі мовлення середня оцінка - це рейтинг якості синтезованих мовленнєвих сигналів.

Найчастіше оцінюється за шкалою від 1 (погано) до 5 (відмінно), середня оцінка є середнім значенням ряду інших індивідуальних параметрів, що оцінюються людиною. Хоча спочатку середні оцінки були отримані з опитувань експертів-спостерігачів, сьогодні MOS часто виробляється за допомогою методу об'єктивного вимірювання, що наближається до людського рейтингу.

Також часто можна зустріти оцінку MOS з відповідною словесною анотацією, яка визначається за ACR (Absolute Category Rating).

Рейтинг	Підпис
5	Чудово (Excellent)
4	Гарно (Good)
3	Допустимо (Fair)
2	Погано (Poor)
1	Дуже погано (Bad)

Таблиця 1.1 – Шкала MOS оцінки з підписами відповідно до ACR

MOS обчислюється як середнє арифметичне всіх оцінок даних експертами:

$$MOS = \frac{\sum_{n=1}^N R_n}{N}$$

1.6 Висновки до розділу

В даному розділі було надано визначення процесу синтезу мовлення, виділено загальну функціональну схему синтезу мовлення і описано кожну її частину та досліджено і порівняно сучасні архітектури для даних систем.

Також було надано визначення математичної моделі системи автоматичного розпізнавання мовлення та було введено такі метрики точності як MOS.

2. АНАЛІЗ РІШЕНЬ ЗАДАЧІ СИНТЕЗУ МОВЛЕННЯ

Задача синтезу мовлення може бути вирішена як неймережевими методами так і статистичними методами. Навчання без вчителя, сховані Марковські моделі, алгоритми перетворення

2.1 Статистична модель синтезу мовлення

У статистичному параметричному синтезі ми, як правило, маємо дві частини. Навчання та синтез. Під час навчання ми виділяємо набір параметрів, які характеризують звуковий зразок, таких як частотний спектр (голосовий тракт), основна частота (джерело голосу) і тривалість (просодія) мови. Потім ми намагаємося оцінити ці параметри за допомогою статистичної моделі. Історично доведено, що найкращі результати дає прихована марковська модель (HMM).

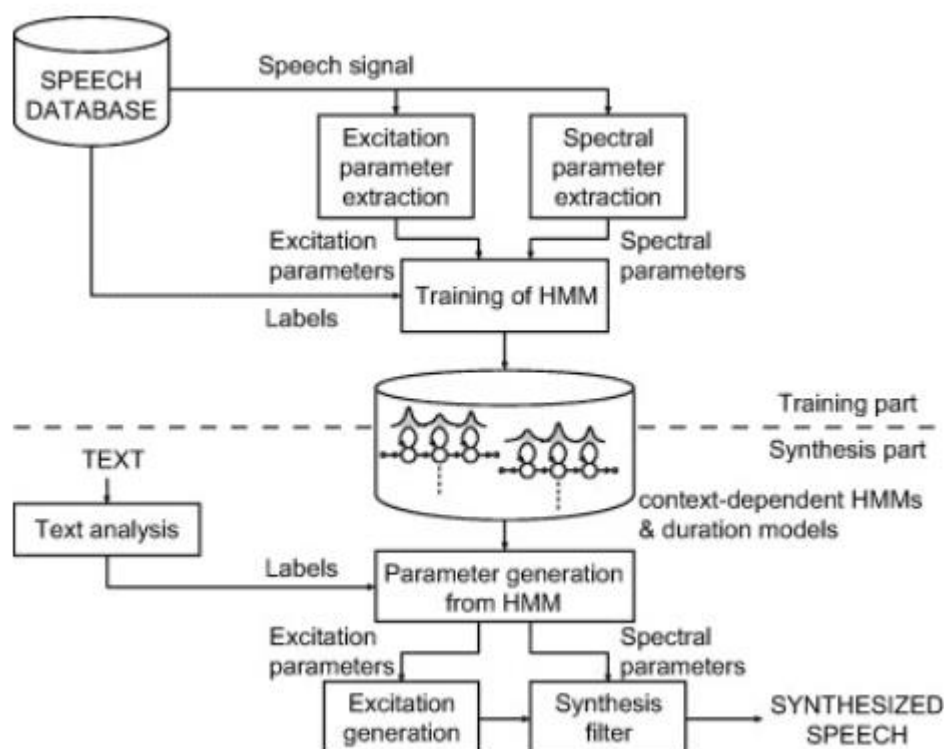


Рисунок 2.1 Схеми статистичної моделі синтезу мовлення

Під час синтезу НММ генерує набір параметрів з нашої цільової текстової послідовності. Ці параметри використовуються для синтезу кінцевих мовних сигналів.

Переваги статистичного параметричного синтезу:

- Немає необхідності зберігати аудіо-зразки в базі даних
- Мовна незалежність
- Гнучкість в характеристиках голосу

Однак у більшості випадків якість синтезованого мовлення не є ідеальною. Саме тут у гру вступають методи, засновані на глибокому навчанні.

2.2 DCTTS

Рекурентні нейронні мережі (RNN) зовсім нещодавно були стандартним підходом до моделювання послідовних даних. Не дивлячись на те, що навчання рекурентних нейронних мереж процес доволі складний і потребує великих ресурсів комп'ютера чи доволі тривалий період часу (зазвичай декілька днів чи тижнів), цей метод був використаний у деяких передових розробках в області синтезу голосових сигналів. Хоча недавні дослідження показали, що використання згорткових графових моделей можна розглядати як більш перспективний підхід, адже процес навчання відбувається швидше, через можливість зробити обчислення паралельно.

DCTTS (Deep Convolutional Text-To-Speech) – нейромережева архітектура, яка повністю побудована на згорткових нейронних мережах, без застосування рекурентних нейронних мереж. Головною ідеєю розробників було спрощення процесу навчання без втрати якості синтезованого сигналу.

Основну частину системи складають дві нейронні мережі:

- 1) Text2Mel - котра генерує мел-спектрограми з тексту;
- 2) SSRN – конвертує мел-спектрограми в повні STFT спектрограми.

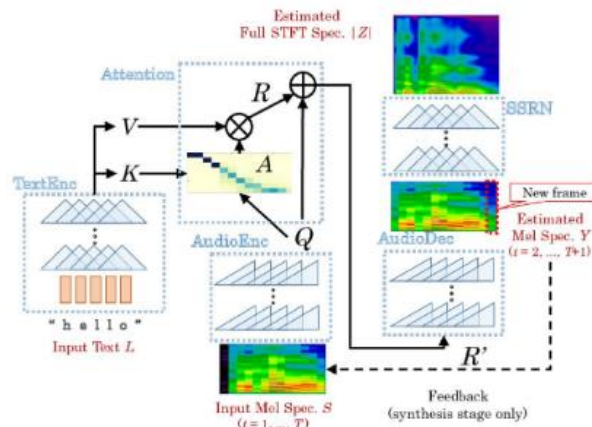


Рисунок 2.2 – Загальна архітектура системи DCTTS

Text2Mel мережа складається 4 компонент:

- 1) Text Encoder;
- 2) AuTTSO Encoder;
- 3) Attention;
- 4) AuTTSO Decoder.

Мережа Text Encoder спочатку кодує вхідне речення, яке складається з N символів в 2 матриці K,V:

$$\text{Речення: } L = [l_1, \dots, l_N] \in \text{Char}^N$$

$$\text{Матриці: } K, V \in R^{d \times N}$$

Мережа AuTTSO Encoder кодує мел-спектрограми $S (= S_{1:F, 1:T}) \in R^{F \times T}$, попередньо сказаної мови, довжина якої T, в матрицю $Q \in R^{d \times T}$.

$$(K, V) = \text{TextEnc}(L)$$

$$Q = \text{AudioEnc}(S_{1:F, 1:T})$$

Матриця уваги $A \in R^{N \times T}$ визначає як сильно зв'язані n -ий символ та t -ий часовий інтервал $S_{1:F, t}$:

$$A = \text{softmax}_{n\text{-axis}}(K^T Q / \sqrt{d})$$

$A_{nt} \sim 1$ означає, що модуль слідкує на n -тим символом l_n з інтервалу часу t і збільшує коефіцієнт уваги для l_n чи l_{n+1} чи символів біля них в наступному часовому фреймі $t + 1$.

Таким чином основа $R \in R^{d \times T}$, декодована в наступні фрейми $S_{1:F, 2:T+1}$ отримується як

$$R = \text{Att}(Q, K, V) := V A$$

Отримана R об'єднується з кодованим звуковим сигналом Q , як $R' = [R, Q]$, бо це виявилось більш вигідним у пілотних дослідженнях.

Потім ця об'єднана матриця $R' \in R^{2d \times T}$ декодується AuTTSO Decoder модулем для синтезу мел-спектрограми.

$$Y_{1:F, 2:T+1} = \text{AudioDec}(R')$$

Результат $Y_{1:F, 2:T+1}$ порівнюється з тимчасово зсунутим правдивим основним екземпляром $S_{1:F, 2:T+1}$ функцією втрат $L_{\text{spec}}(Y_{1:F, 2:T+1} \mid S_{1:F, 2:T+1})$ і отримана похибка враховується в параметри мережі за алгоритмом зворотного розповсюдження помилки (error back-propagation algorithm).

Функцією втрат є сума втрати L1 та бінарна дивергенція D_{bin} .

$$D_{\text{bin}}(Y|S) := E_{ft}[-S_{ft} \log Y_{ft} - (1 - S_{ft}) \log(1 - Y_{ft})] = E_{ft}[-S_{ft} \widehat{Y}_{ft} + \log(1 + \exp \widehat{Y}_{ft})]$$

$$\text{де } \widehat{Y}_{ft} = \text{logit}(Y_{ft}).$$

Оскільки бінарна дивергенція забезпечує ненульове значення градієнта нейронної мережі, це дозволяє застосовувати навчання, за

градієнтом (graTTSent-based training). Також легко впевнитись, що похибка спектрограми є додатною:

$$L_{spec}(Y|S) = D_{bin}(Y|S) + E[|Y_{ft} - S_{ft}|] \geq 0$$

і нерівність виконується, якщо $Y = S$.

Spectrogram Super-Resolution Network – генерує повну спектрограму з отриманої мел-спектрограми. Структура SSRN може бути знайдена на рисунку 2.7.

Архітектура DCTTS є досить цікавою для дослідження і є не поганим варіантом для побудови розроблюваної системи.

Якість її синтезу порівняна з Tacotron, але при цьому швидкість її навчання є набагато вищою, через використання згорткових нейронних мереж замість рекурентних, що і було головною ідеєю вчених.

2.3 Tacotron 1/2

Tacotron – end-to-end генеративна модель для синтезу голосового сигналу з тексту. Вона може бути навчена на наборах даних (датасетах), які являють собою пари з текстовим і голосовим представленням фраз. Початкові коефіцієнти системи при цьому можна задати повністю випадковими значеннями. Також архітектура генерує звуковий сигнал з досить високою швидкістю, хоча і оперує даними на рівні фреймів.

Системи, які навчаються на наборах (текст, аудіо), як правило, мають низку переваг. По-перше, процес створення такої системи спрощується на початку, як і проектування в цілому. По-друге, спрощується налаштування різних аспектів майбутньої системи, таких як інтонація та мова диктора. І причиною цього є те, що ці значення встановлюються на початку навчання та розробки моделі, а не на конкретних проміжних компонентах. Аналогічно, більш простий сценарій також може призвести до адаптації до нових фактів. Така модель, ймовірно, буде більш надійною, ніж

багатоступенева форма, де помилки в кожному з її компонентів можуть додаватися і негативно впливати на результат. Це також справедливо і в міру того, як спрощується дизайн.

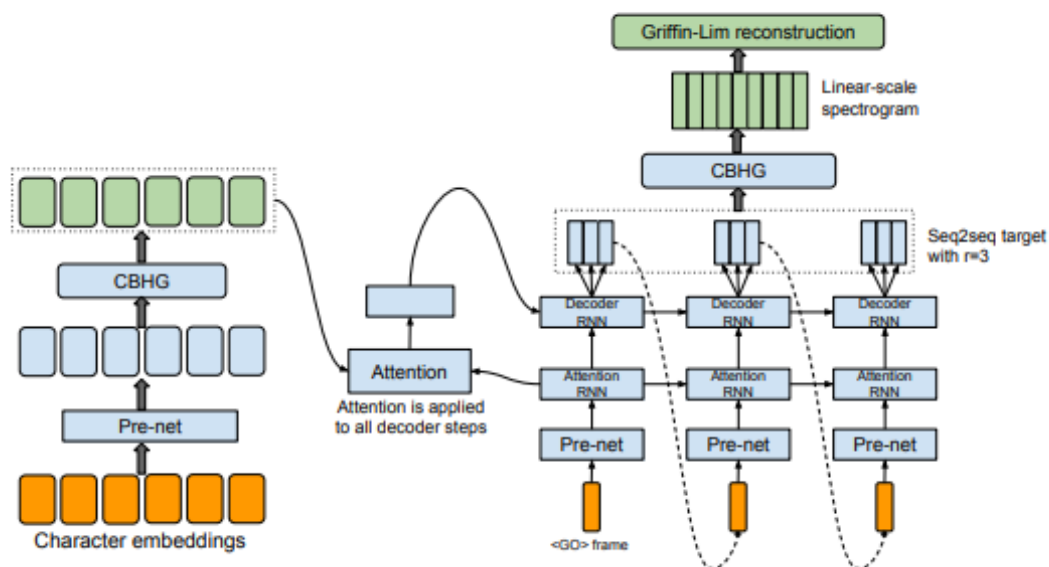


Рисунок 2.3 Загальна архітектура Tacotron

На рисунку 2.3 зображена архітектура системи. Вона включає в себе енкодер, керований увагою декодер та мережу для подальшої обробки.

Загальний алгоритм роботи можна описати наступним чином. Система приймає на вхід послідовність символів, а на виході продукує спектрограму, з якої за допомогою алгоритму Гріффіна-Ліма синтезується мовленнєвий сигнал.

Основою архітектури Tacotron є модель послідовність-в-послідовність (seq2seq) з механізмом уваги.

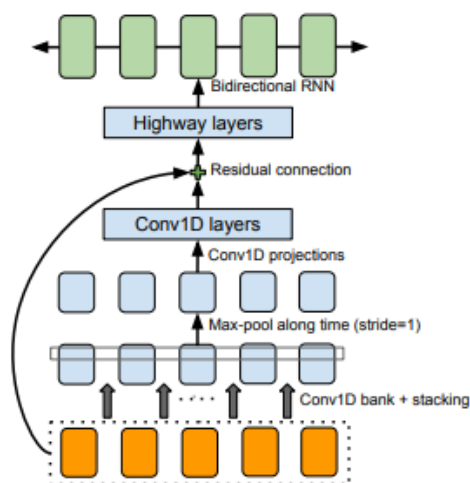


Рисунок 2.4 Seq2Seq модель Tacotron

CBHG компонента системи складається з 1-D згорткового фільтру, після якого слідує highway нейронна мережа з двонаправленим рекурентним юнітом. Модуль слугує для отримання представлень з послідовностей символів і створений на базі моделі, яка використовувалася для машинного перекладу.

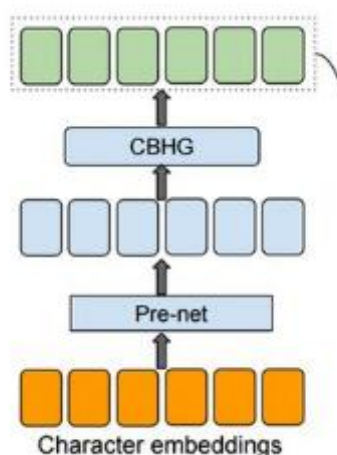


Рисунок 2.5 – Архітектура енкодеру

Енкодер слугує для отримання послідовних представлень тексту.

Структура:

- вхід: one-hot вектор символів вбудованих в подальшу послідовність

- pre-net: мережа для проведення нелінійних перетворень
- CBHG модуль

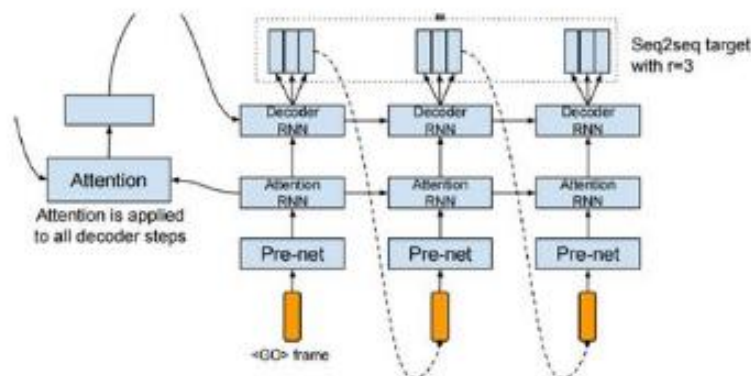


Рисунок 2.6 – Декодер системи

Декодер системи потрібен для отримання 80-розмірної мел-масштабної спектрограми. Розробники відмовились від прямого використання вейвформи сигналу, адже таке представлення було б занадто надлишковим.

Декодер складається з:

- 1) рекурентної нейронної мережі уваги (1 шар з 256 GRU юнітів);
- 2) рекурентної мережі декодери (2 шари по 256 GRU юнітів);
- 3) вихідних шар зі з'єднанням нейронів «кожен-з-кожним».

Мережа постобробки є останнім компонентом конструкції Tacotron. Вона використовується для отримання вихідного сигналу системи. До цього компоненту входять алгоритм Гріффіна-Ліма та модуль CBHG. Останній використовує ітераційну техніку для зниження середньоквадратичної похибки та оцінює форму сигналу за спектрограмою даних. Вчені стверджують, що цей алгоритм використовувався лише короткочасно і з метою оптимізації архітектури системи. У майбутньому його планувалося замінити нейронною мережею, яка повинна забезпечити більш точну форму сигналу і вирішити проблеми алгоритму.

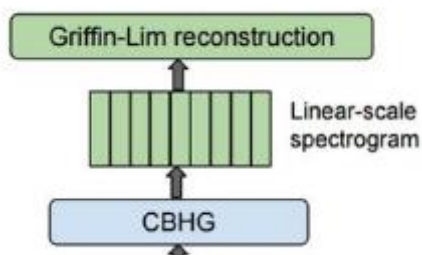


Рисунок 2.7 – Post-processing мережа

Архітектуру Tacotron в перспективі можна використати для побудови майбутнього застосунку.

З переваг системи можна виділити непогану, порівняно з іншими реалізаціями, якість синтезованого сигналу. А з недоліків - помилки, у вихідних вейвформах, які може допускати алгоритм її побудови та складність навчання, адже мережа здебільшого побудована на рекурентних юнітах.

2.6 FastSpeech

Архітектура FastSpeech - це структура прямого перетворення, заснована на самоуважності в трансформаторі та 1D згортки. Ми називаємо цю структуру Прямим Трансформатором (Feed-Forward Transformer, FFT), як показано на рисунку 1a. Трансформатор прямого перетворення складається з декількох блоків ШПФ для перетворення фонемі в мелоспектрограму, з N блоками на стороні фонемі і N блоками на стороні мелоспектрограми, з регулятором довжини (який буде описано в наступному підрозділі) між ними для подолання розриву в довжині між послідовністю фонемі і мелоспектрограми. Кожен блок ШПФ складається з мережі самоуваги та 1D згортки, як показано на рисунку 1b. Мережа самоуваги складається з декількох головок для вилучення інформації про перехресну позицію. На відміну від 2-шарової щільної мережі в Transformer, ми використовуємо 2-шарову 1D згорткову мережу з активацією ReLU. Мотивація полягає в тому,

що сусідні приховані стани більш тісно пов'язані в послідовності символ/фонема та мело-спектрограма в мовленнєвих задачах. Ми оцінюємо ефективність 1D-згорткової мережі в експериментальній частині. Наслідуючи Transformer, залишкові зв'язки, нормалізація шарів та відсів додаються після самонавчання. відсіву додаються після мережі самоуваги та 1D згорткової мережі відповідно

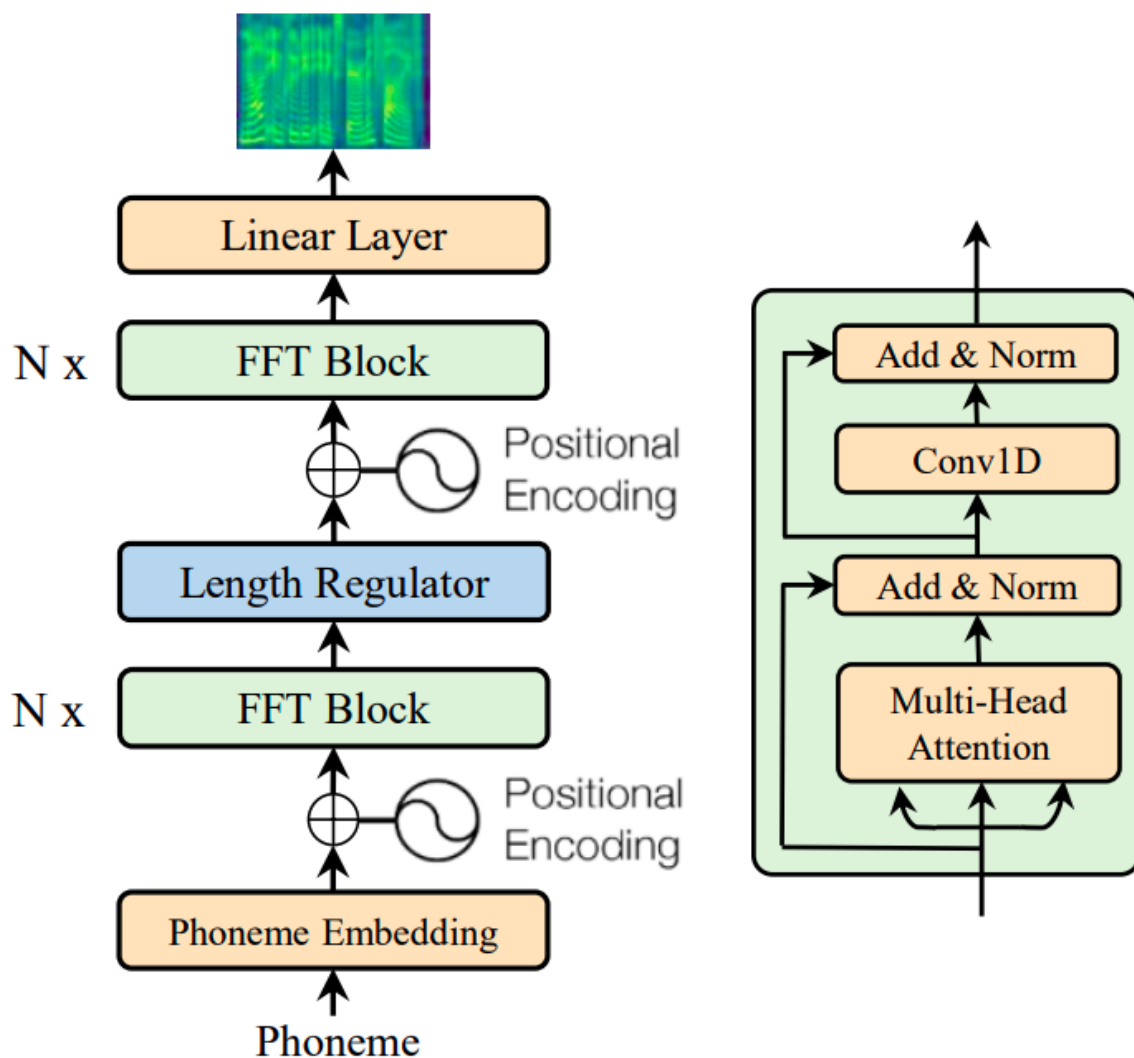


Рисунок 2.8 Архітектура трансформера та блок швидкого перетворення Фур'є

Регулятор довжини використовується для вирішення проблеми невідповідності довжини між послідовністю фонем і послідовністю

спектрограми в ФНЧ, а також для управління швидкістю голосу і частиною просодії. Довжина послідовності фонем, як правило, менша, ніж довжина її мелос-спектрограмної послідовності, і кожна фонема послідовності, і кожній фонемі відповідає декілька мелос-спектрограм. Ми маємо на увазі довжину мел-спектрограм, яка відповідає фонемі, називаємо тривалістю фонемі (ми опишемо як визначати тривалість фонемі в наступному підрозділі). На основі тривалості фонемі d регулятор довжини регулятор довжини розширює приховані стани послідовності фонем у d разів, і тоді сумарна довжина прихованих станів дорівнює довжині мелоспектрограм

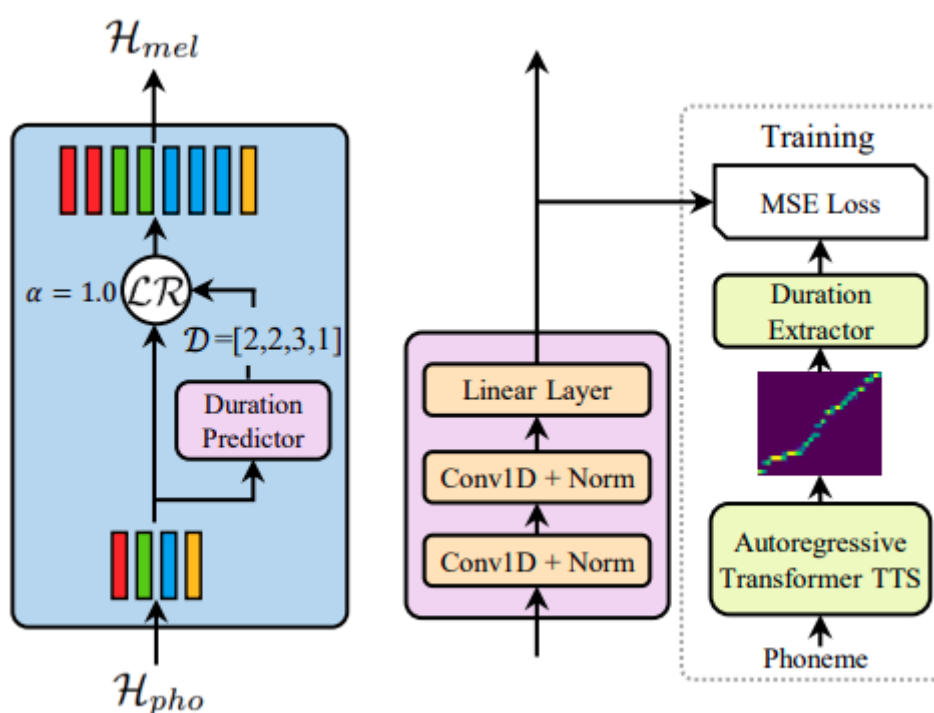


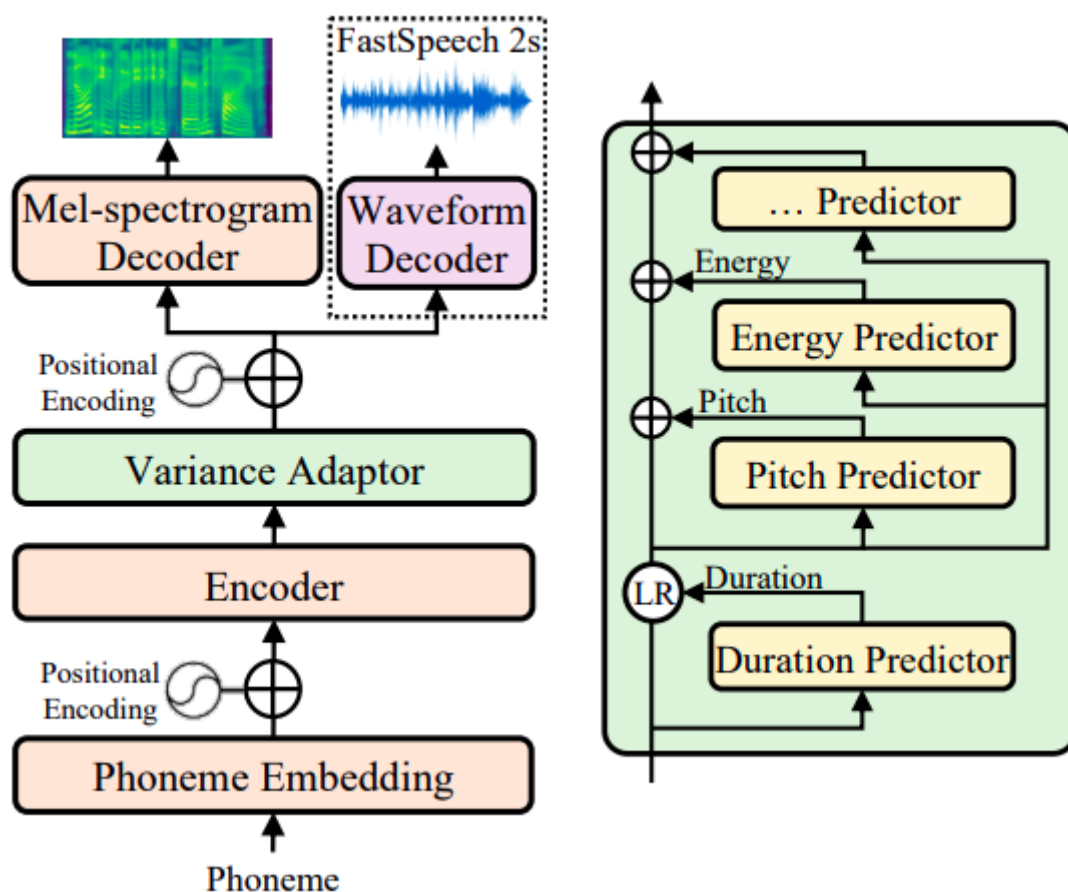
Рисунок 2.9 Регулятор довжини та предіктор

Прогнозування тривалості фонемі є важливим для регулятора довжини. Як показано на рисунку 1d, предиктор тривалості складається з 2-шарової 1D згорткової мережі з активацією ReLU, за кожним з яких слідує шар нормалізації та шар відсіву, а також додатковий лінійний шар для виведення скаляра, який є точно прогнозованою тривалістю фонемі. Зауважте, що цей

модуль накладається поверх БПФ на стороні фонем і спільно навчається з моделлю FastSpeech для прогнозування довжини mel-спектрограм для кожної фонемі з середньоквадратичною похибкою (MSE) втрат. Ми прогнозуємо довжину в логарифмічній області, що робить їх більш гаусівськими та легшими для навчання. Зауважте, що навчений предиктор тривалості використовується лише на етапі виведення TTS, оскільки ми можемо безпосередньо використовувати тривалість фонемі тривалість фонемі, витягнуту з авторегресійної моделі вчителя в навчанні (див. наступні обговорення).

2.7 FastSpeech2

Загальна архітектура моделі FastSpeech 2 показана на рисунку 1a. Кодер перетворює послідовність вбудовування фонем у приховану послідовність фонем, а потім адаптер дисперсії додає до прихованої послідовності різну дисперсійну інформацію, таку як тривалість, висота та енергія, нарешті, декодер мелоспектрограми перетворює адаптовану приховану послідовність в мелоспектрограму послідовність паралельно. Ми використовуємо блок feed-forward Transformer, який є стеком шару самоуваги (Vaswani et al., 2017) та 1D-згортки, як у FastSpeech (Ren et al., 2019), в якості базовою структурою для кодера та декодера мелоспектрограм. На відміну від FastSpeech, який покладається на конвеєр дистиляції між вчителем та учнем та тривалість фонемі з моделі вчителя, FastSpeech 2 робить кілька вдосконалень.



2.10 Архітектура FastSpeech2

По-перше, прибрано конвеєр дистилляції між вчителем та учнем, і безпосередньо використовуємо реальні мелоспектрограми як ціль для навчання моделі, що дозволяє уникнути втрати інформації в дистильованих мелоспектрограмах і підвищити верхню межу якості голосу. По-друге, наш дисперсійний адаптер складається не тільки з предиктора тривалості, але й з предикторів висоти тону та енергії де

1) предиктор тривалості використовує тривалість фонем, отриману шляхом примусового вирівнювання (McAuliffe et al., 2017), як навчальну ціль, яка є більш точною, ніж та, що витягується з карти уваги авторегресійного вчителя. уваги авторегресійної моделі вчителя, що було експериментально перевірено в розділі 3.2.2; та

2) додаткові предиктори висоти тону та енергії можуть надавати більше інформації про дисперсію, що є важливим для полегшення проблеми відображення "один-до-багатьох" в TTS. По-третє, для подальшого спрощення процесу підготовки і підштовхнути його до повністю наскрізної системи, ми пропонуємо FastSpeech 2s, який безпосередньо генерує хвильову форму з тексту, без каскадної генерації мелоспектрограми (акустична модель) та генерації хвильової форми (вокодер). (вокодер), без каскадної генерації мелоспектрограми (акустична модель). У наступних підрозділах ми детально описуємо конструкції дисперсійного адаптера та безпосередньої генерації сигналу в нашому методі.

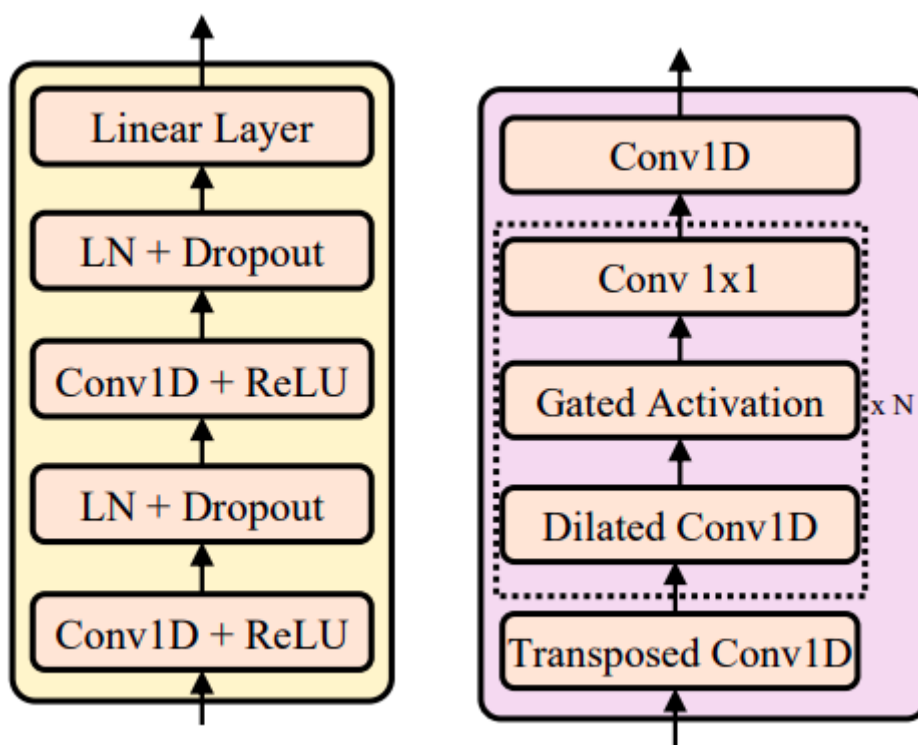


Рисунок 2.11 Декодер хвилі

Адаптер дисперсії має на меті додати інформацію про дисперсію (наприклад, тривалість, висоту, енергію і т.д.) до прихованої послідовності фонем. прихованої послідовності фонем, що може надати достатньо інформації для прогнозування варіантів мови для проблеми відображення

"один-до-багатьох" в TTS. Коротко представимо інформацію про дисперсію наступним чином:

- 1) тривалість фонем, яка представляє, як довго звучить голос;
 - 2) висота тону, яка є ключовою є ключовою характеристикою для передачі емоцій і сильно впливає на просодію мови;
 - 3) енергія, яка вказує на величину фреймового рівня мелоспектрограми і безпосередньо впливає на гучність і просодію мови.
- Більше дисперсійної інформації можна додати в дисперсійний адаптер, такої як емоції, стиль та диктор, а залишимо це для подальшої роботи. Відповідно, дисперсійний адаптер складається з

- 1) предиктора тривалості (тобто регулятора тривалості, як у FastSpeech),
- 2) предиктора висоти тону та
- 3) предиктора енергії, як показано на рисунку 1b.

Під час навчання ми беремо істинні значення тривалості, висоти тону та енергії витягнуті із записів, як вхідні дані для прихованої послідовності, щоб передбачити цільову мову. У той же час, ми використовуємо Водночас, ми використовуємо істинні значення тривалості, висоти тону та енергії як цілі для тренування тривалості, висоти тону та енергії, які використовуються у висновку для синтезу цільового мовлення. Як показано на рисунку 1с, предиктори тривалості, висоти тону та енергії мають схожу структуру моделі (але різні параметри моделі), яка складається з 2-шарової 1D-згорткової мережі з активацією ReLU, кожен з яких супроводжується шаром нормалізації та шаром відсіву, а також додатковим лінійним шаром для проектування прихованих станів у вихідну послідовність прихованих станів у вихідну послідовність. У наступних параграфах ми детально описуємо трьох предикторів відповідно. Предиктор тривалості Предиктор тривалості приймає приховану послідовність фонем як вхідну і прогнозує тривалість кожної фонем, яка представляє, скільки кадрів мела відповідають цій фонемі, і перетворюється в логарифмічну область для зручності

прогнозування. Предиктор тривалості оптимізується з втратою середньоквадратичної помилки (MSE), приймаючи витягнуту тривалість як навчальну ціль. Замість того, щоб витягувати тривалість фонем за допомогою попередньо вивченої авторегресійної моделі TTS у FastSpeech, ми використовуємо інструмент Монреальського примусового вирівнювання (MFA) (McAuliffe et al., 2017)³ для вилучення тривалості фонем, щоб підвищити точність вирівнювання і, таким чином, зменшити інформаційний розрив між входом та виходом моделі.

2.6 Висновки до розділу

В даному розділі було проведено аналіз сучасних архітектур систем синтезу мовлення мовлення, визначено їх переваги та недоліки та порівняно між собою.

3. СИСТЕМА СИНТЕЗУ УКРАЇНСЬКОГО МОВЛЕННЯ

3.1 Вибір фреймворку розробки моделі синтезу мовлення

Для розробки власної моделі автоматичного розпізнавання мовлення було обрано фреймворк Pytorch через його легкість у використанні для побудови нейронних мереж.

На даному фреймворку буде розроблено як кожен частину системи синтезу так і допоміжні нейромереві моделі.

3.2 Опис фреймворку Pytorch

Заснована на бібліотеці Torch, PyTorch - це бібліотека машинного навчання з відкритим вихідним кодом, яка використовується для таких завдань, як комп'ютерний зір та обробка природної мови. За її розробку в першу чергу відповідає дослідницька група зі штучного інтелекту Meta. Це програмне забезпечення з відкритим вихідним кодом, яке доступне безкоштовно за модифікованою ліцензією BSD. Крім того, PyTorch включає в себе зовнішній інтерфейс C++, незважаючи на те, що інтерфейс Python є більш досконалим і є основним напрямком розвитку. PyTorch пропонує дві високорівневі функції.:

- 1) Тензорні обчислення (як NumPy) із сильним прискоренням через графічні процесори
- 2) Глибинні нейронні мережі, побудовані на системі автоматичного диференціювання на основі плівки

3.3 Розробка програми диктора для збору даних

Перед тим як почати тренувати модель, треба дістати дані для тренування. Загалом для тренування системи синтезу мовлення потрібно зібрати дані у вигляді аудіо-запис - текстова транскрибація.

Для даних цілей було вирішено знайти диктора на сайті фрілансу, та купити в нього озвучування. Далі задача стояла в тому щоб зібрати необхідний набір текстових даних для озвучування, розбити їх на окремі речення та розробити зручну систему для диктора для виводу речень та запису голосу. Було завантажено більше 300 книг з відкритих джерел інтернету, серед яких переважно була художня українська література. Після перетворення тексту книжок в речення, була сформована база даних з 45984 реченнями.

	id	sentence	audio_path	duration	updated_at
	Filter	Filter	Filter	Filter	Filter
1	1	Саме про таку міру запобіжного заходу клопотала ...	/usr/src/app/tts_database/wavs/1.wav	3.72	2022-06-12 11:32:39.764135
2	2	Павлусь мало очей не видивив, так шукав межі юрбою ...	/usr/src/app/tts_database/wavs/2.wav	4.2	2022-06-14 18:58:16.873236
3	3	Бо я її бачив у сірих надвечірніх сутінках торговельного ...	/usr/src/app/tts_database/wavs/3.wav	4.98	2022-06-16 13:24:44.409713
4	4	Наприклад, наполегливо працювати, читати, тренуватися.	/usr/src/app/tts_database/wavs/4.wav	3.78	2022-06-07 16:35:41.224580
5	5	Перед верховним катом Ордену зловісним шоломом ...	NULL	NULL	NULL
6	6	Зупинилось і більше не рухається	NULL	NULL	NULL
7	7	Нагадаємо, раніше сини принцеси Діани — принци Вільям...	NULL	NULL	NULL
8	8	Тільки не тіш себе марними ілюзіями	/usr/src/app/tts_database/wavs/8.wav	2.76	2022-06-18 14:13:46.640228
9	9	пояснійте дітям про небезпеку водяної стихії;водяної стих...	NULL	NULL	NULL

Рисунок 3.1 База даних речень

Наступним кроком була реалізація веб-додатку для диктора. Для її реалізації була вибрана архітектура SPA (Single Page Application) с таким стеком технологій:

- бекенд: python (FastAPI)
- фронтенд: js (Vue.js)

Зовнішній вигляд додатку зображений на рисунку 3.2

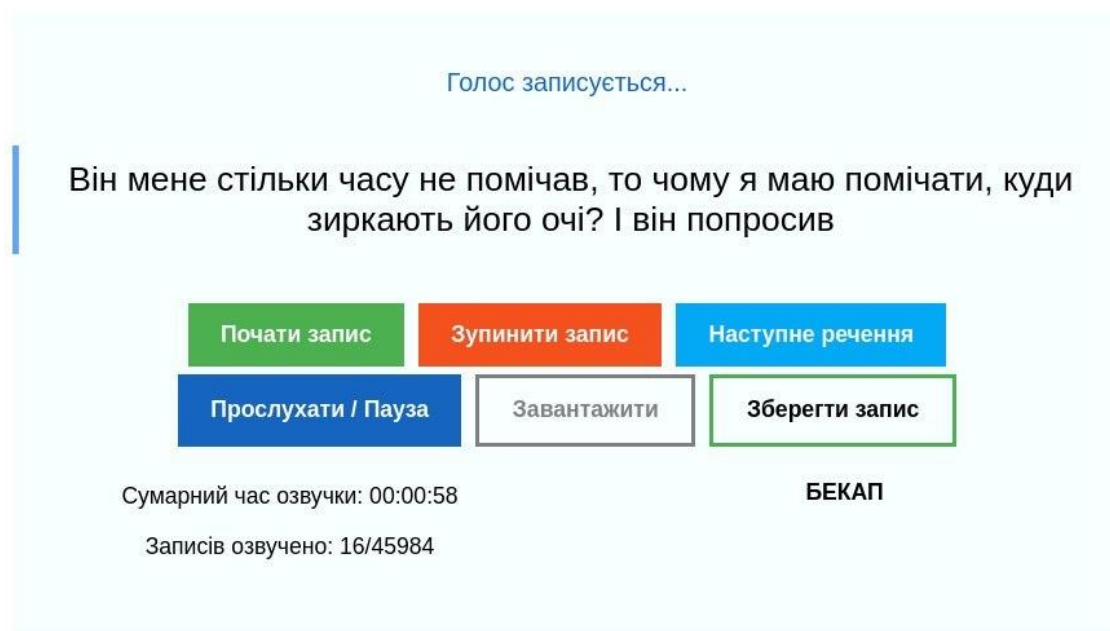


Рисунок 3.2 Зовнішній вигляд системи диктора

Таким чином було зібрано та збережено близько 11 годин аудіоданих диктора.

3.4 Вибрана модель для тренування системи синтезу мовлення

Для тренування було вирішено натренувати такі варіанти архітектур:

- 1) FastSpeech2 та HiFi-GAN вокодер
- 2) FastSpeech + HiFi-GAN
- 3) Tacotron2 + HiFi-GAN

Результати тренування відображено в таблиці 3.1.

Архітектура	MOS	Час однієї епохи, хв
FastSpeech2 + HiFi-GAN	4.0	9
FastSpeech + HiFi-GAN	3.8	10
Tacotron2 + HiFi-GAN	4.0	18

Таблиця 3.1 Результат тренування

Не авторегресивність FastSpeech1/2 моделей надає можливість розпаралелювати тренування на декілька графічних ядер - що пришвидшує процес тренування та тестування в рази.

3.7 Вибір існуючих моделей для порівняння та вимірювання їх якості

Серед простору існуючих моделей для автоматичного розпізнавання мовлення для порівняння було обрано дві моделі: Silero Speech-to-Text uk v3 model та Google Cloud Speech-to-Text API. Обидві моделі є безкоштовними

у власному використанні і доступні без кількісного обмеження на використання.

Після проведення тестування трьох моделей на тестових даних отримали такі результати:

Модель	MOS		
	Художні тексти	Технічні терміни	Англiцизми та молодіжні терміни
Власна модель	4.0	3.9	4.0
Silero TTS	4.0	3.8	4.1
Google Cloud Speech-to-Text API	3.9	4.1	3.9

Таблиця 3.2 - Результати порівняння моделей

В порівнянні з моделями Google та Silero власна модель показала гарні показники, випередивши ці моделі в домені аудіокниг та відео з Youtube. На домені телефонії власна модель показала гірший результат, зайнявши друге місце. Це обумовлено тим що в тренувальних даних, на яких було натреновано власну модель, дані з домену телефонії займали дуже маленький відсоток від всіх.

4. РОЗРОБКА СТАРТАП-ПРОЕКТУ «SaaS система синтезу мовлення»

Описана технологія може бути реалізована в якості веб-додатку або API. Цей розділ ставить перед собою мету реалізації технології

використання нейромережевої системи синтезу українського мовлення у вигляді Software as a Service додатку: система авторизації/реєстрації клієнтів додатку, адмін панель з трекінгом використання синтезу.

Процес включає в себе:

- імплементацію веб-сервісу з використанням підходів та технологій описаних у попередніх розділах роботи;
- розробку стратегії виходу конкурентоспроможного продукту на ринок та подальший розвиток стартапу.

Для отримання цілісного уявлення про зміст ідеї та можливі базові потенційні ринки, в межах яких потрібно шукати групи потенційних клієнтів важливо побудувати таблицю з описом ідеї можливими напрямками застосування та основними вигодами, що може отримати користувач товару.

Таблиця 4.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Створення SaaS системи синтезу українського мовлення	1. Надання компаніям, які займаються створення голосових помічників можливість генерувати мовленнєвий сигнал по тексту	
	2. Надання компаніям, які займаються телефонією, можливість генерувати фрази, які будуть відправлені клієнту по телефону	

Отже, ідея проекту заключається в тому, що компанії, які займаються наданням послуг ШІ голосового помічника, або компанії які займаються телефонією зможуть за вказаним тарифом використовувати систему синтезу українського мовлення через відкритий API SaaS

системи.

Визначений перелік слабких, сильних та нейтральних характеристик та властивостей ідеї потенційного товару є підґрунтям для формування його конкурентоспроможності. Тому у таблиці 8.2 показано чим товар відрізняється від існуючих аналогів чи замінників на основі техніко-економічних властивостей та характеристик.

№ п/п	Техніко- економічні характеристик и ідеї	(потенційні) товари/концепції конкурентів				W (слабка сторона)	N (нейтра льна сторон а)	S (сильна сторона)
		Мій проект	Конкуре нт 1	Конкуре нт 2	Конкуре нт 3			
1.	Форма виконання	Веб- додат ок	Дескто пний додаток	Дескто пний додаток	Веб- додаток			+
2.	Собівартість	Низька	Низька	Низька	Висока		+	
3.	Емоційність	Контр ольова на	Випадк ова	Випадк ова	Випадко ва			+
4.	Крос- платформенні сть	Так	Ні	Ні	Так			+
5.	Потреба в інтернеті	Так	Так	Ні	Так		+	

Таблиця 4.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

Сильними сторонами даного проекту є те, що форма виконання ідеї - веб-додаток, що є більш гнучким способом взаємодії з програмою, оскільки не вимагає її встановлення на машину, а також крос-платформенність, адже здатність підтримувати велику кількість платформ напряду збільшує кількість потенційних клієнтів. Всі інші характеристики є нейтральними. Тому даний проект можна вважати конкурентоспроможним.

4.2 ТЕХНОЛОГІЧНИЙ АУДИТ ІДЕЇ ПРОЕКТУ

В межах даного підрозділу необхідно провести аудит технології, за допомогою якої можна реалізувати ідею проекту (технології створення товару).

Таблиця 4.3 – Технологічна здійсненність ідеї проекту

п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1.	Створення SaaS системи синтезу українського мовлення	Pytorch	Наявна	Безкоштовна, доступна
		TensorFlow	Наявна	Платна, доступна
		Theano	Наявна	Безкоштовна, доступна
Обрана технологія реалізації ідеї проекту: Pytorch				

Отже, проект буде реалізовано за допомогою платформи ..., адже вона є найбільш зрілою з-поміж трьох, постійно оновлюється та має добре побудовану документацію. Тому даний проект можна вважати технологічно здійсненним.

4.3 АНАЛІЗ РИНКОВИХ МОЖЛИВОСТЕЙ

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє планувати напрями розвитку проекту з урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

Спочатку проводимо аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку.

Таблиця 4.4 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	100
2	Загальний обсяг продаж, грн/ум.од	1000
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу	Висока вартість початкового капіталу
5	Специфічні вимоги до стандартизації та сертифікації	-
6	Середня норма рентабельності в галузі (або по ринку), %	$R = (3000000 * 100) / (1000000 * 12) = 25\%$

Отже, середня норма рентабельності в галузі менша, ніж банківський відсоток на вкладення. Тому має сенс вкласти кошти в саме цей проект, адже проект не має специфічних вимог до стандартизації та сертифікації, бо він буде виконаний у вигляді веб-додатку.

Далі визначаються потенційні групи клієнтів, їх характеристики, та

формується орієнтовний перелік вимог до товару для кожної групи (табл. 4.5).

Таблиця 4.5 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Оптимізація роботи магазинів	а) Магазины, яким потрібно оптимізувати об'єми товарів на складах; б) магазини, яким потрібно оптимізувати стратегії ціноутворення	б) націлені на постійне поповнення різноманіття послуг, середнє спілкування з клієнтом б) націлені на постійне поповнення різноманіття послуг, середнє спілкування з клієнтом	Веб-додаток: Надійний, Функціонально розвинений, зручний дизайн; Постачальник веб-сервісу: Надає підтримку, Постійно покращує рішення

Визначено характеристики стартап-проекту: основну потребу, що формує ринок - оптимізацію роботи магазинів; наведено основні цільові сегменти ринку – компанії з великими відділами підтримки, продажів, замовлень тощо; відмінності у поведінці різних потенційних цільових груп клієнтів - отримання інформації та різноманітне спілкування;

затверджено основні вимоги до споживачів

– надійні та високофункціональні чат-боти з яскраво вираженою емоційною складовою.

Далі складаються таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають (табл. 4.6-4.7).

Таблиця 4.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Зростаюча вимогливість покупців	Споживачі вимагають більш точних прогнозів та рекомендацій від алгоритмів	Створення дослідницької лабораторії, присвяченої покращенню роботи алгоритмів
2	Зміна потреб і смаків покупців	Окрім служб підтримки/продажу клієнти можуть придумати інші застосування, потрібні на тому чи іншому етапі розвитку клієнтського продукту	Додавати нові проекти та функції до списку послуг
3.	Збільшення витрат на технічну підтримку	Невчасне реагування на сучасний ринок потреб користувачів	Вчасно оновлювати програмне забезпечення
4.	Конкуренція	Вихід на ринок великої компанії	вихід з ринку;
5.	Зменшення кількості	Зменшення	Постійні оновлення

	замовників	зацікавленості замовників у наданих сервісах	алгоритму
--	------------	--	-----------

Було наведено основні фактори загроз стартап-проекту. Найбільшою загрозою для проекту є зміна потреб користувачів (користувачам потрібен новий функціонал) та їх вимогливість (користувачам потрібні більш точні прогнози та рекомендації від алгоритмів). Для зменшення цих загроз потрібне створення дослідницької лабораторії, присвяченої покращенню роботи алгоритмів, а також потрібне своєчасного оновлення програмного забезпечення і користувацького інтерфейсу. Найменшими загрозами є збільшення витрат на технічну підтримку та зменшення кількості користувачів замовників.

Таблиця 4.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Підвищились технологічні бар'єри входу на ринок: розробка TTS - важкий дослідницький проект	потрібні найкращі дослідники	Наймати та навчати кадри, які могли б революціонувати область
2	Можливість швидкого розвитку у зв'язку з різким зростанням попиту на ринку	Майже всі власники магазинів хотіли б оптимізувати роботу закладів та збільшити прибутки, тому користувачів багато	Максимальне захоплення ринку за рахунок впливових клієнтів

3	Зростання можливостей потенційних покупців	Зростання фінансування досліджень у галузі машинного навчання	Запропонувати свої послуги зацікавленим підприємствам
4	Зниження довіри до конкурента 1	У додатку конкурента 1 нещодавно було знайдено витік інформації, яка збиралася для аналітики	При виході на ринок звертати увагу покупців на безпеку нашого додатку
5	Зменшення витрат на технічну підтримку	Збільшення продуктивності роботи штату компанії за рахунок підвищення їхнього професійного рівня	Підвищувати рівень кваліфікації своїх співробітників

Було наведено основні фактори сприяння ринковому впровадженню проекту: підвищення технологічних бар'єрів входу на ринок: розробка додатків Decision Intelligence - важкий дослідницький проект, можливість швидкого розвитку у зв'язку з різким зростанням попиту на ринку, зростання можливостей потенційних покупців. Основними реакціями компанії є: надання своїх послуг зацікавленим підприємствам, безпечність програмного забезпечення, найм та навчання кадрів, які могли б революціонувати область та максимальне захоплення ринку за рахунок впливових клієнтів.

Надалі визначаються загальні риси конкуренції на ринку (табл. 4.8).

Таблиця 4.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Вказати тип конкуренції - олігополія	Існує мала кількість фірм-конкурентів на ринку	Створення більш технологічно досконалих рішень, ніж конкуренти
2. За рівнем конкурентної - інтернаціональна	Майже всі конкуренти - закордонні	Можна знайти дешевших дослідників
3. За галузевою ознакою - внутрішньогалузева	Конкуренти мають сервіси, які використовуються лише всередині даної галузі	Добре описані рамки галузі та способи впливу на неї
4. Конкуренція за видами товарів: - товарно-видова	Види товарів є однаковими, а саме - програмне забезпечення	Конкуренція між різними TTS продуктами
5. За характером конкурентних переваг - нецінова	Удосконалення технології створення додатку, щоб собівартість була нижчою	Поліпшення якості продукції
6. За інтенсивністю - не марочна	Бренди відсутні	Легше вийти на ринок молодій компанії

Було наведено проведено аналіз конкуренції на ринку, а саме визначено: тип конкуренції - олігополія; конкуренція за рівнем конкурентної боротьби - міжнародна; конкуренція за галузевою ознакою - внутрішньогалузева; конкуренція за видами товарів - товарно-видова; конкуренція а характером конкурентних переваг - нецінова; конкуренція за інтенсивністю - не марочна. Також було наведено можливі дії компанії, щоб бути конкурентоспроможною: створення більш технологічно досконалих рішень, ніж конкуренти; пошук дешевших дослідників; використання способів впливу на галузь; поліпшення якості продукції.

Далі розробляється перелік факторів конкурентоспроможності для ринку на основі аналізу складових моделі 5 сил М. Портера (табл. 4.9).

Таблиця 4.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Google	Бар'єр - потужний дослідницький відділ	Amazon Google Сильні сторони: стабільні, мають підтримку	Rozetka eBay	-
Висновки:	Дуже інтенсивна боротьба за першість в TTS технологіях	Потенційно може вийти будь-яка компанія з необхідними ресурсами	Нічого не диктують.	Диктують умови	-

Отже, з огляду на конкурентну ситуацію можна з упевненістю сказати, що проект має можливість роботи на ринку, тому що серед наведених конкурентів немає тих, які б могли його потіснити, адже розроблене рішення спрощує та пришвидшує роботу спеціаліста.

На основі аналізу висновків, наведених вище, визначається та обґрунтовується перелік факторів конкурентоспроможності (табл. 4.10).

Таблиця 4.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Науково-технічний потенціал	Кращі дослідження=вища якість послуг
2	Кадровий потенціал	Кращі кадри=кращі дослідження

Було наведено основні фактори конкурентоспроможності, які будуть представлені на ринку, а саме: науково-технічний потенціал, який дозволяє надавати вищу якість послуг, а також кадровий потенціал, який гарантує більш швидкі та більш якісні дослідження у галузі TTS.

За визначеними факторами конкурентоспроможності проводиться аналіз сильних та слабких сторін стартап-проекту (табл. 4.11).

Таблиця 4.11 – Порівняльний аналіз сильних та слабких сторін «Емоційна діалогова система для рекомендації онлайн курсів»

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні зі стартап-проектом						
			-3	-2	-1	0	1	2	3

1	Науково-технічний потенціал	19			+				
2	Кадровий потенціал	19			+				

Було наведено порівняльний аналіз сильних сторін проекту товарів-конкурентів і нашого підприємства. Найбільше балів набрано для таких факторів конкурентоспроможності – науково-технічний потенціал та кадровий потенціал.

Далі проводимо SWOT-аналіз (табл. 4.12).

Таблиця 4.12 – SWOT- аналіз стартап-проекту

Сильні сторони: найкращі фахівці в області	Слабкі сторони: слабкий маркетинговий відділ, відсутність репутації, молода компанія
Можливості: стрімкий ріст ринку та технологічності рішень у сфері TTS	Загрози: конкуренти значно більш відомі і потужніші, мають більше ресурсів, з часом здатні придумати кращий алгоритм або покращити наш

Отже, внутрішні можливості компанії і спроможності щодо виведення продукту на ринок характеризуються такими сильними і слабкими сторонами: сильні - найкращі фахівці в області; слабкі - слабкий маркетинговий відділ, відсутність репутації, молода компанія. Ринкові та можливості компанії щодо зовнішнього оточення характеризуються можливостями і загрозами: можливості - стрімкий ріст ринку та технологічності рішень у сфері чат- ботів; загрози - конкуренти значно більш відомі і потужніші, мають більше ресурсів, з часом здатні придумати кращий алгоритм або покращити наш.

На основі SWOT-аналізу складаються альтернативи ринкового впровадження стартап-проекту (табл. 4.13).

Таблиця 4.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Покращуємо алгоритм настільки, щоб конкурентів залишити далеко позаду	50%	2 роки
2	Виходимо на ринок з обмеженим функціоналом, граємо на швидкому рості ринку	50%	1 рік

Отже, було визначено альтернативу ринкового впровадження стартап-проекту - вихід на ринок з обмеженим функціоналом. Оскільки час на покращення алгоритмів є значно більшим та не пропонує більшу ймовірність отримання ресурсів.

4.4 РОЗРОБКА РИНКОВОЇ СТРАТЕГІЇ ПРОЕКТУ

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів (табл. 4.14).

Таблиця 4.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних Клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Компанії, яким потрібно автоматизація відділу підтримки	Готові, якнайшвидше	Попит - високий	Не дуже інтенсивна	Середня

2	Колл-центри, яким потрібна автоматизація дзвінків	Готові, якнайшвидше	Попит - середній, через недовіру технологіям	Не дуже інтенсивна	Середня
Які цільові групи обрано: обрано обидві групи, так як вони майже не відрізняються, а проект їх може задовольнити					

Отже, було обрано основні цільові групи: компанії, які потребують автоматизацію відділу підтримки та колл-центри, які потребують автоматизацію відділу дзвінків, адже обидві групи мають схожі вимоги та готові сприйняти продукт.

Далі визначається базова стратегія розвитку (табл. 4.15).

Таблиця 4.15 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
1	Створення TTS застосунку що є новою і унікальною властивістю	Створення продуктових інновацій та їх маркетинг	Основна позиція - новизна, адже є ключовою технологією і перевагою нашого проекту	Стратегія диференціації

Наступним кроком є вибір стратегії конкурентної поведінки (табл. 4.16).

Таблиця 4.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
1	Ні	Шукати нових і забирати існуючих	Стабільність/надійність сервісів, підтримка, інтерфейс	Оборонна

Отже, було визначено базову стратегію конкурентної поведінки - оборонну, також матиме технічну підтримку та зручний інтерфейс, що формуватиме довіру і прихильність споживачів.

Далі визначається стратегія позиціонування проекту, яка допоможе користувачам ідентифікувати програмний продукт (табл. 4.17).

Таблиця 4.17 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару	Базова стратегія	Ключові конкурентоспромож	Вибір асоціацій, які мають
	цільової аудиторії	розвитку	ні позиції власного стартап-проекту	сформувати комплексну позицію власного проекту (три ключових)

1	Коректна робота, надійність, підтримка	Стратегія диференціації	Основна позиція - емоційність, адже є ключовою технологією і перевагою нашого проекту	Швидкодія, безпека, простота
---	---	----------------------------	--	---------------------------------

Отже, було визначено стратегію позиціонування, а саме визначено основні вимоги до товару цільової аудиторії: коректна робота, надійність, підтримка; базову стратегію розвитку: диференціація; ключові конкурентоспроможні позиції стартап-проекту: емоційність, адже є ключовою технологією і перевагою нашого проекту. Також сформовано комплексну позицію проекту: швидкодія, безпека, простота.

4.5 РОЗРОБКА МАРКЕТИНГОВОЇ ПРОГРАМИ

Першим кроком є формування маркетингової концепції товару, який отримує споживач. Для цього у табл. 4.18 потрібно підсумувати результати попереднього аналізу конкурентоспроможності товару.

Таблиця 4.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Автоматизація певного бізнес-процесу: користувацької підтримки та відділу продажів	Не потрібно наймати додаткових фахівців для консультацій користувачів	Ширший спектр аналітики обраних рішень

Надалі розробляється трирівнева маркетингова модель товару (табл. 4.19).

Таблиця 4.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Автоматизація певного бізнес-процесу: озвучування необхідної інформації клієнту. Внаслідок цього не потрібно наймати додаткових фахівців для консультацій користувачів. Також це допомагає швидше та ефективніше працювати з клієнтами.		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Надійність	М	Тл
	2. Емоційність	М	Тл
	3. Найдовший час відповіді	М	Тл
	Оптимізує роботу магазинів, дає рекомендації щодо управління складами та стратегій ціноутворення		
	Веб-додаток, на який заходять клієнти		
Google TTS			
III. Товар із підкріпленням	Акції, знижки		
	Підтримка		
За рахунок чого потенційний товар буде захищено від копіювання: патент.			

Наступним кроком є визначення цінових меж (табл. 4.20). Аналіз проводиться експертним методом.

Таблиця 4.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	125 грн. / 1000 викликів	250 грн. / 1000 викликів	>100000 грн. / місяць	275 грн. – 500 грн. / 1000 викликів

Визначено межі встановлення ціни на мобільний додаток, а саме рівень цін на товари-замінники – 125 грн. за 1000 викликів програмного інтерфейсу, рівень цін на товари-аналоги 250 грн., рівень доходів цільової групи споживачів - 100000 грн, верхня та нижня межі встановлення ціни на товар – 275-500 грн. Аналіз був проведений експертним методом.

Наступним кроком є визначення оптимальної системи збуту, в межах якого приймається рішення (табл. 4.21).

Таблиця 4.21 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Клієнти зазвичай оцінюють можливості платформи через безкоштовну	Фінансування витрат на функціонування	Канал нульового рівня	Проводити збут власними силами

	пробну версію, а потім почнуть платити за кожні 1000 викликів сервісу протягом довгого часу	каналу збуту, фінансування збутових операцій. Обслуговування проданих товарів.		
--	---	--	--	--

Отже, було сформовано систему збуту у вигляді щомісячної оплати за певну кількість викликів сервісу, та буде включати користувацьку підтримку. Збут буде проводитися власними силами.

Далі розробляється концепція маркетингових комунікацій (табл. 4.22).

Таблиця 4.22 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Це інтернет-компанії, які надають перевагу новітнім технологіям	Е-mail, письмові звернення до компаній, конференції	Унікальна особливість чат-ботів - емоційність, покращує задоволення клієнтів	Привернути максимум уваги до продукту, змусити компанію спробувати продукт	Користувачі хочуть спілкуватись з живими людьми, компанії хочуть наймати менше людей - ми компроміс.

Отже, було розроблено концепцію маркетингових комунікацій: ключові позиції для позиціонування – емоційність розроблених чат-ботів при спілкуванні, основне завдання рекламного повідомлення - привернути максимум уваги до продукту, змусити компанію спробувати продукт, з фокусуванням на концепцію рекламного звернення – компроміс між потребами компаній та їх користувачів.

4.6 ВИСНОВКИ ДО РОЗДІЛУ 4

У даному розділі були досліджені основні аспекти виходу на ринок веб-додатку “SaaS система синтезу мовлення”. Описаний продукт є доцільний для користувачів, які бажають оптимізувати роботу своїх магазинів/колл-центрів: управління складами та вибір стратегії ціноутворення.

В рамках розділу було визначено перелік слабких, сильних та нейтральних характеристик та властивостей ідеї потенційного товару, що є підґрунтям для формування його конкурентоспроможності; обрана технологія реалізації ідеї проекту: для створення SaaS системи синтезу мовлення, яка є зручною та гнучкою для експлуатації; проведений ступеневий аналіз конкуренції на ринку, SWOT аналіз та обґрунтовані фактори конкурентоспроможності. Також було обрано основні цільові групи: компанії, які потребують оптимізувати роботу складів та компанії, які потребують оптимізацію стратегій ціноутворення, адже обидві групи мають схожі вимоги та готові сприйняти продукт. Його основні характеристики: новизна, надійність та широкий спектр послуг аналітики обраних рішень. Технологія буде захищена від копіювання за рахунок патенту.

Отже відповідно до проведених досліджень:

- існує можливість ринкової комерціалізації проекту;
- існують перспективи впровадження з огляду на потенційні групи клієнтів, бар'єри входження не є високими, проект має дві значні

переваги перед конкурентами: кроссплатформність, широкий спектр послуг аналітики обраних рішень;

- необхідно реалізувати систему синтезу мовлення на базі фреймворку Pytorch;
- подальша імплементація є доцільною.

ВИСНОВКИ

У ході виконання дипломної роботи, було проведено детальний аналіз історії та еволюції методів області синтезу голосових сигналів, описані варіанти застосувань, де ця область може бути максимально корисною та наведені приклади її найбільш вдалого використання.

Оскільки, передовим підходом до вирішення задачі синтезу мовлення в наш час є нейронні мережі, було вирішено використати саме їх для розробки системи синтезу мовлення.

В ході проведення дослідження моделей систем для синтезу мовлення було виконано структурний аналіз, описано логіку роботи та перевірено їх відповідність до ряду вимог, які були встановлені для кандидатів порівняння існуючих на сьогодні реалізацій. За результатами порівняння, найкращі результати продемонструвала моделі FastSpeech 2 та Tacotron 2.

Для реалізації системи синтезу було використано фреймворк Pytorch через його:

- зручність у побудові нейронних мереж;
- легкість в освоєнні.

Після обґрунтування вибору засобів для створення системи та опису остаточного формату застосунку була розроблена його програмна реалізація. Описана структура розробленої системи, ключові її компоненти та логіка їх взаємодії між собою.

Було також протестовано натреновану систему і визначено переваги і недоліки даної системи. Серед переваг можна виділити швидкість роботи системи, через її не авторегресивність надається можливість максимально використати паралельні можливості комп'ютеру (відеокарти та процесору) для роботи даної нейронної мережі та її малий розмір в пам'яті (близько 30 МБ) в порівнянні з іншими архітектурами.

З недоліків виділяється складність синтезу довгих речень: близько до кінця речення система може чи повністю перестати синтезувати, чи додати до результуючого сигналу непотрібний шум.

ПЕРЕЛІК ПОСИЛАНЬ

1. Karagiannakos S. Speech synthesis: A review of the best text to speech architectures with Deep Learning [Електронний ресурс] / Sergios

- Karagiannakos. – 2021. – Режим доступу до ресурсу:
<https://theaisummer.com/text-to-speech/>
2. A Survey on Neural Speech Synthesis [Електронний ресурс] / X.Tan, T. Qin, F. Soong, T. Liu. – 2021. – Режим доступу до ресурсу:
<https://arxiv.org/pdf/2106.15561.pdf>
 3. FastSpeech: Fast, Robust and Controllable Text to Speech [Електронний ресурс] / [Y. Ren, Y. Ruan, X. Tan та ін.]. – 2019. – Режим доступу до ресурсу: <https://arxiv.org/pdf/1905.09263.pdf>
 4. FASTSPEECH 2: FAST AND HIGH-QUALITY END-TO-END TEXT TO SPEECH [Електронний ресурс] / Y. Ren, C. Hu, X. Tan. – 2022. – Режим доступу до ресурсу: <https://arxiv.org/pdf/2006.04558.pdf>
 5. Taylor P. Text-to-Speech Synthesis / Paul Taylor., 2011.
 6. Wang Y. TACOTRON: TOWARDS END-TO-END SPEECH SYNTHESIS [Електронний ресурс] / Y. Wang, R. Skerry-Ryan, D. Stanton. – 2017. – Режим доступу до ресурсу:
<https://arxiv.org/pdf/1703.10135.pdf>
 7. Kong J. HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis [Електронний ресурс] / J. Kong, J. Kim, J. Bae. – 2020. – Режим доступу до ресурсу:
<https://arxiv.org/pdf/2010.05646.pdf>
 8. Prenger R. WAVEGLOW: A FLOW-BASED GENERATIVE NETWORK FOR SPEECH SYNTHESIS [Електронний ресурс] / R. Prenger, R. Valle, B. Catanzaro. – 2018. – Режим доступу до ресурсу:
<https://arxiv.org/pdf/1811.00002.pdf>
 9. Elias I. Parallel Tacotron 2: A Non-Autoregressive Neural TTS Model with Differentiable Duration Modeling [Електронний ресурс] / I. Elias, H. Zen, J. Shen. – 2021. – Режим доступу до ресурсу:
<https://arxiv.org/pdf/2103.14574.pdf>
 10. Jurafsky D. Speech and Language Processing / D. Jurafsky, J. Martin.

11. Parallel Tacotron 2: A Non-Autoregressive Neural TTS Model with Differentiable Duration Modeling [Електронний ресурс] / [I. Elias, H. Zen, J. Shen та ін.]. – 2021. – Режим доступу до ресурсу: <https://arxiv.org/pdf/2103.14574.pdf>

ДОДАТОК А

Програмний код бекенду додатку для диктора

```
from fastapi import FastAPI
```

```
from fastapi.middleware.cors import CORSMiddleware
```

```
from api import samples
from db.base import database
```

```
app = FastAPI()
```

```
origins = ["*"]
```

```
app.add_middleware(
    CORSMiddleware,
    allow_origins=origins,
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"]
)
```

```
app.include_router(router=samples.router, prefix="/samples", tags=["samples"])
```

```
@app.on_event("startup")
async def startup():
    await database.connect()
```

```
@app.on_event("shutdown")
async def shutdown():
    await database.disconnect()
```

```

@app.get("/")
async def ping():
    return {"ping": "pong"}

```

```

router = APIRouter()

```

```

@router.post("/backup")
async def backup():
    cwd = os.getcwd()
    db_dir = os.path.join(cwd, "tts_database")

```

```

    backup_name = str(datetime.now())
    backup_path = os.path.join(cwd,
                               "backups",
                               backup_name)

```

```

    shutil.make_archive(backup_path, "zip", db_dir)

```

```

    return {"status": "success"}

```

```

@router.get("/progress", response_model=ProgressInfo)
async def get_progress_info(samples: SampleRepository = Depends(get_sample_repository)):
    """

    full_amount = await samples.get_full_amount()
    filled_amount = await samples.get_amount_of_filled()

```

```
whole_time = await samples.calculate_whole_time()
```

```
progress = ProgressInfo(
    amount=full_amount,
    whole_time=whole_time,
    filled_amount=filled_amount
)
```

```
return progress
```

```
@router.post("/upload_audio")
```

```
async def receive_audio(sample_id: int,
```

```
    file: UploadFile,
```

```
    samples: SampleRepository = Depends(get_sample_repository)):
```

```
    file_name = f"{sample_id}.wav"
```

```
    file_path = os.path.join(WAVS_PATH, file_name)
```

```
    audio_bytes = file.file.read()
```

```
# Saving file to the drive
```

```
    async with aiofiles.open(file_path, mode="wb") as f:
```

```
        await f.write(audio_bytes)
```

```
    duration = await get_duration(file_path)
```

```
    su = SampleUpdate(duration=duration)
```

```
    await samples.update(sample_id, su)
```

```
    return {"file_path": file_path}
```

ДОДАТОК Б

Програмний код фронтенду додатку для диктора

```

<template>
  <div id="app">

    <div class="container">
      <div class="content">
        <div class="details"
          >{{ detail }}</div>
        <TextToPronounce :text="sample.sentence" />
        <div class="buttons">
          <div class="up-part">
            <button
              @click="startRecording"
              ref="startButton"
              class="start-button"
            >Почати запис</button>
            <button
              @click="stopRecording"
              ref="stopButton"
              class="stop-button"
            >Зупинити запис</button>

```

```

<button
  class="next-button"
  @click="nextSample"
>Наступне речення</button>
</div>
<div class="down-part">

  <button
    @click="playAudio"
    class="play-button"
  >Прослухати / Пауза</button>

  <a :class="{ 'inactive-link': !isDownloadActive}"
    :href="getAudioSrc()"
    :download="getAudioName()">
    <button
      class="download-button"
    >Завантажити</button>
  </a>

  <button
    class="save-button"
    @click="sendFileAndUpdate"
  >
    Зберегти запис
  </button>

</div>
</div>
<div class="whole-info">

```

```

<div class="whole-time">
  <p>Сумарний час озвучки: {{ wholeTimeFormatted }} </p>
  <p>Записів озвучено: {{ filledAmount }}/{{ fullAmount }}</p>
</div>

<div class="backup">
  <button
    class="backup-button"
    @click="makeBackup"
  >БЕКАП</button>
</div>
</div>
</div>
</div>

<footer></footer>
</div>
</template>

<script>
import axios from "@api/axios"
import TextToPronounce from '@components/TextToPronounce'
export default {
  name: 'App',
  components: {
    TextToPronounce,
  },
  data(){
    return {
      sample: {
        id: null,

```

```

    sentence: "",
    audio_path: ""
  },
  wholeTime: 0,
  filledAmount: 0,
  fullAmount: 0,
  audioChunks: [],
  detail: "",
  isDownloadActive: false,
  blob: null,
  audio: null,
  error: null,
}
},
methods: {
  startRecording() {
    this.blob = null
    this.detail = "Голос записується..."
    console.log("Recording are started...")
    this.$refs["startButton"].disabled = true
    this.$refs["stopButton"].disabled = false
    this.audioChunks = []
    this.rec.start()
    if (this.audio) {
      this.audio.pause()
    }
  },
  stopRecording() {
    console.log("Recording are stopped.")
    this.detail = ""
  }
}

```

```

    this.$refs["startButton"].disabled = false
    this.$refs["stopButton"].disabled = true
    this.rec.stop()
  },
  handleStream(stream){
    this.rec = new MediaRecorder(stream)
    this.rec.ondataavailable = e => {
      this.audioChunks.push(e.data)
      if (this.rec.state == "inactive"){
        this.blob = new Blob(this.audioChunks, {"type": "audio/wav; codecs=opus"})
        console.log("BLOB:", this.blob)
        let audioURL = window.URL.createObjectURL(this.blob)
        this.audio = document.createElement("audio")
        this.audio.src = audioURL
        this.isDownloadActive = true;
      }
    }
  },
  playAudio(){
    let isPlaying = !this.audio.paused;
    if(isPlaying){
      this.audio.pause();
    }else{
      this.audio.play()
    }
  },
  getAudioSrc(){
    if(this.audio){
      return this.audio.src
    }
  }
}

```

```

    }else{
        return "
    }
},
getAudioName(){
    if(this.sample){
        return `${this.sample.id}.wav`
    }
},
downloadAudio(){
    if(this.audio){
    }else{
        this.detail = "Помилка: не можливо завантажити аудіо, яке не було
записано."
    }
},
fetchRandom(){
    axios.get("/samples/random")
        .then(response => response.data)
        .then(sample => {
            if(sample.error){
                this.sample.sentence = "Всі аудіозаписи були озвучені!"
            }
            else{
                this.sample = sample
            }
        })
},
nextSample(){
    if(this.audio){

```

```

    this.audio.pause()
  }
  this.blob = null
  this.audio = null
  this.isDownloadActive = false
  this.fetchRandom()
},
updateSample(sampleId, audioPath){
  let requestData = {audio_path: audioPath}
  axios.put(`/samples/${String(sampleId)}`, requestData)
  .then(response => {
    console.log("Sample has been updated.")
  })
  .catch(e => {
    this.error = e
    console.log("Error: cannot update sample.\nDetails: ", e)
  })
},
sendFileAndUpdate(){
  if(this.blob == null){
    alert("Помилка: не можу відправити аудіофайл так як ви його не  
записали!.")
    return
  }
  let formData = new FormData()
  let params = new URLSearchParams({
    sample_id: this.sample.id
  })

  formData.append('file', this.blob)

```

```

axios.post(`/samples/upload_audio?${params}`, formData)
.then(response => {
  let data = response.data
  this.sample.audio_path = data.file_path
  console.log("Audio file has been sent: ", data)
})
.then(() => {
  this.updateSample(this.sample.id, this.sample.audio_path)
  this.fetchProgress()
})
.then(() => {
  alert("Файл збережено!")
})
.catch(e => {
  console.log("Error while sending file to the backend.\nDetails: ", e)
  this.error = e
})
},
fetchProgress(){
  axios.get("/samples/progress")
  .then(response => {
    let data = response.data
    this.filledAmount = data.filled_amount
    this.wholeTime = data.whole_time
    this.fullAmount = data.amount
  })
  .catch(e => {
    console.log("Error during getting whole info.\nDetails: ", e)
    this.error = e
  })
}

```

```

},
makeBackup(){
  axios.post("/samples/backup")
    .then(response => {
      let data = response.data
      alert("Бекап створено!")
    })
    .catch(e => {
      this.error = e
      console.log("Error during backup.\nDetails: ", e)
    })
  }
},
computed: {
  wholeTimeFormatted(){
    let date = new Date(this.wholeTime * 1000).toISOString()
    let result = date.slice(11, 19)
    return result
  }
},
mounted(){
  navigator
    .mediaDevices
    .getUserMedia({audio: true})
    .then(stream => { this.handleStream(stream) });

  this.fetchRandom()
  this.fetchProgress()
},
}

```

```
</script>
```

```
<style>
```

```
body{
  box-sizing: border-box;
  padding: 0;
  margin: 0;
}
button{
  background: none;
  border: none;
  font-size: 16px;
  font-weight: 600;
  cursor: pointer;
  padding: calc(.875rem - 1px) calc(1.5rem - 1px);
}
.inactive-link{
  pointer-events: none;
  cursor: default;
}
.inactive-link button{
  color: gray;
  border: 3px solid gray;
}
button:hover,
button:focus {
  box-shadow: rgba(0, 0, 0, 0.1) 0 4px 12px;
}
.details{
  color: #1565C0;
```

```
    font-size: 18px;
}
.start-button{
    color: #fff;
    background: #4CAF50;
}
.stop-button{
    color: #fff;
    background: #F4511E;
}
.play-button{
    color: #fff;
    background: #1565C0;
}
.next-button{
    color: #fff;
    background: #03A9F4;
}
.download-button{
    border: 3px solid #1565C0;
}
.save-button{
    border: 3px solid #4CAF50;
}
#app {
    font-family: Avenir, Helvetica, Arial, sans-serif;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
    text-align: center;
    /* color: #2c3e50; */
}
```

```
/* margin-top: 60px; */  
max-width: 780px;  
margin: 0 auto;  
display: grid;  
height: 100vh;  
align-items: center;  
grid-row-gap: 10px;  
}  
.content{  
  display: grid;  
  grid-row-gap: 10px;  
  background: #f8ffff;  
  padding: 40px 0;  
}  
.buttons{  
  display: grid;  
  grid-row-gap: 5px;  
}  
.buttons .up-part, .buttons .down-part{  
  display: flex;  
  justify-content: center;  
  flex-wrap: wrap;  
}  
.buttons button{  
  margin: 0 5px;  
}  
.whole-info{  
  display: grid;  
  grid-template-columns: 1fr 1fr;  
/* text-align: left; */
```

```
}  
</style>
```

ДОДАТОК В

Програмний код тренування нейронної мережі

```
import argparse  
import os  
  
import torch  
import yaml  
import torch.nn as nn  
from torch.utils.data import DataLoader  
from torch.utils.tensorboard import SummaryWriter  
from tqdm import tqdm  
  
from utils.model import get_model, get_vocoder, get_param_num  
from utils.tools import to_device, log, synth_one_sample  
from model import FastSpeech2Loss  
from dataset import Dataset  
  
from evaluate import evaluate  
  
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")  
  
def main(args, configs):
```

```

print("Prepare training ...")

preprocess_config, model_config, train_config = configs

# Get dataset
dataset = Dataset(
    "train.txt", preprocess_config, train_config, sort=True, drop_last=True
)
batch_size = train_config["optimizer"]["batch_size"]
group_size = 4 # Set this larger than 1 to enable sorting in Dataset
assert batch_size * group_size < len(dataset)
loader = DataLoader(
    dataset,
    batch_size=batch_size * group_size,
    shuffle=True,
    collate_fn=dataset.collate_fn,
)

# Prepare model
model, optimizer = get_model(args, configs, device, train=True)
model = nn.DataParallel(model)
num_param = get_param_num(model)
Loss = FastSpeech2Loss(preprocess_config, model_config).to(device)
print("Number of FastSpeech2 Parameters:", num_param)

# Load vocoder
vocoder = get_vocoder(model_config, device)

# Init logger
for p in train_config["path"].values():

```

```

    os.makedirs(p, exist_ok=True)
train_log_path = os.path.join(train_config["path"]["log_path"], "train")
val_log_path = os.path.join(train_config["path"]["log_path"], "val")
os.makedirs(train_log_path, exist_ok=True)
os.makedirs(val_log_path, exist_ok=True)
train_logger = SummaryWriter(train_log_path)
val_logger = SummaryWriter(val_log_path)

```

Training

```

step = args.restore_step + 1
epoch = 1
grad_acc_step = train_config["optimizer"]["grad_acc_step"]
grad_clip_thresh = train_config["optimizer"]["grad_clip_thresh"]
total_step = train_config["step"]["total_step"]
log_step = train_config["step"]["log_step"]
save_step = train_config["step"]["save_step"]
synth_step = train_config["step"]["synth_step"]
val_step = train_config["step"]["val_step"]

outer_bar = tqdm(total=total_step, desc="Training", position=0)
outer_bar.n = args.restore_step
outer_bar.update()

```

while True:

```

    inner_bar = tqdm(total=len(loader), desc="Epoch {}".format(epoch),
position=1)
    for batchs in loader:
        for batch in batchs:
            batch = to_device(batch, device)

```

Forward

```
output = model(*(batch[2:]))
```

Cal Loss

```
losses = Loss(batch, output)
```

```
total_loss = losses[0]
```

Backward

```
total_loss = total_loss / grad_acc_step
```

```
total_loss.backward()
```

```
if step % grad_acc_step == 0:
```

```
    # Clipping gradients to avoid gradient explosion
```

```
    nn.utils.clip_grad_norm_(model.parameters(), grad_clip_thresh)
```

Update weights

```
optimizer.step_and_update_lr()
```

```
optimizer.zero_grad()
```

```
if step % log_step == 0:
```

```
    losses = [l.item() for l in losses]
```

```
    message1 = "Step {}/{}", ".format(step, total_step)
```

```
    message2 = "Total Loss: {:.4f}, Mel Loss: {:.4f}, Mel PostNet Loss: {:.4f}, Pitch Loss: {:.4f}, Energy Loss: {:.4f}, Duration Loss: {:.4f}".format(*losses)
```

```
with open(os.path.join(train_log_path, "log.txt"), "a") as f:
```

```
    f.write(message1 + message2 + "\n")
```

```
outer_bar.write(message1 + message2)
```

```
log(train_logger, step, losses=losses)
```

```
if step % synth_step == 0:
```

```
    fig, wav_reconstruction, wav_prediction, tag = synth_one_sample(
        batch,
        output,
        vocoder,
        model_config,
        preprocess_config,
    )
```

```
    log(
        train_logger,
        fig=fig,
        tag="Training/step_{}_{}_".format(step, tag),
    )
```

```
    sampling_rate = preprocess_config["preprocessing"]["audio"][
        "sampling_rate"
    ]
```

```
    log(
        train_logger,
        audio=wav_reconstruction,
        sampling_rate=sampling_rate,
        tag="Training/step_{}_{}_reconstructed".format(step, tag),
    )
```

```
    log(
        train_logger,
        audio=wav_prediction,
        sampling_rate=sampling_rate,
        tag="Training/step_{}_{}_synthesized".format(step, tag),
    )
```

```
)
```

```
if step % val_step == 0:
    model.eval()
    message = evaluate(model, step, configs, val_logger, vocoder)
    with open(os.path.join(val_log_path, "log.txt"), "a") as f:
        f.write(message + "\n")
    outer_bar.write(message)
```

```
model.train()
```

```
if step % save_step == 0:
    torch.save(
        {
            "model": model.module.state_dict(),
            "optimizer": optimizer._optimizer.state_dict(),
        },
        os.path.join(
            train_config["path"]["ckpt_path"],
            "{}.pth.tar".format(step),
        ),
    )
```

```
if step == total_step:
    quit()
    step += 1
    outer_bar.update(1)
```

```
inner_bar.update(1)
```

```
epoch += 1
```

```

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("--restore_step", type=int, default=0)
    parser.add_argument(
        "-p",
        "--preprocess_config",
        type=str,
        required=True,
        help="path to preprocess.yaml",
    )
    parser.add_argument(
        "-m", "--model_config", type=str, required=True, help="path to model.yaml"
    )
    parser.add_argument(
        "-t", "--train_config", type=str, required=True, help="path to train.yaml"
    )
    args = parser.parse_args()

    # Read Config
    preprocess_config = yaml.load(
        open(args.preprocess_config, "r"), Loader=yaml.FullLoader
    )
    model_config = yaml.load(open(args.model_config, "r"), Loader=yaml.FullLoader)
    train_config = yaml.load(open(args.train_config, "r"), Loader=yaml.FullLoader)
    configs = (preprocess_config, model_config, train_config)

```

```
main(args, configs)
```