Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

магістра
(назва освітньо-кваліфікаційного рівня)

студента	*Снісара Андрія Владиславовича*
	(ПІБ)
академічної групи	*122М-22-1*
	(шифр)
спеціальності	*122 Комп'ютерні науки*
	(код і назва спеціальності)
освітньої програми	*«122 Комп'ютерні науки»*
	(назва освітньої програми)
на тему:	*Дослідження ефективності застосування генеративних методів штучного інтелекту до синтезу звуку*

_____*А. В. Снісар*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
розділів кваліфікаційної роботи				
спеціальний	*Проф. Алексєєв М.О.*			

Рецензент				

Нормоконтролер	*Проф. Лактіонов І.С.*			

Дніпро
2023

Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
Завідувач кафедри
Програмного забезпечення комп'ютерних систем
(повна назва)

_____ М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » _____ 20 ____ Року

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

студенту *122м-22-1* *Снісару Андрію Владиславовичу*
(група) (прізвище та ініціали)

Тема кваліфікаційної роботи *Дослідження ефективності застосування генеративних методів штучного інтелекту до синтезу звуку*

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від 09.10.2023 р. № 1227-с

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень – процес синтезу мовлення за допомогою генеративних методів штучного інтелекту.

Предмет досліджень – моделі та методи синтезу мовлення.

Мета НДР – дослідження моделі синтезу мовлення з використанням нейронної мережі.

Вихідні дані для проведення роботи – теоретичні та експериментальні дослідження, аудиодані для навчання мовному синтезу.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Новизна запропонованих рішень полягає у тому, що удосконалено процедуру навчання моделі синтезу мовлення VALL-E, що дозволило підвищити ефективність синтезу мовлення.

Практична цінність результатів полягає у тому, що навчена модель синтезу мовлення дозволяє накопичувати і використовувати знання для вирішення задач синтезу мовлення та подальшого дослідження предметної галузі.

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Результати досліджень мають бути подані у вигляді, що дозволяє побачити та оцінити якість генерації мовлення за допомогою навченої моделі VALL-E.

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок – кінець**
Аналіз предметної галузі та постановка завдання	01.09.2023-30.09.2023
Опис моделей та методів розв’язання задачі	01.10.2023-31.10.2023
Програмна реалізація та тренування моделі синтезу мовлення, проведення оцінювання якості синтезу та аналіз результатів	01.11.2023-01.12.2023

6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Соціальний ефект від реалізації результатів роботи очікується позитивним завдяки покращенню синтезу мовлення в результаті впровадження ефективної моделі у застосунки з інтегрованими синтезаторами мовлення.

Завдання видав _____ *Алексєєв М.О.*
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ *Снісар А.В.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *01.09.2023 р.*

Термін подання кваліфікаційної роботи до ЕК *22.12.2023*

РЕФЕРАТ

Пояснювальна записка: 82 стор., 14 рис., 3 табл., 4 додатка, 93 джерела.

Об'єкт дослідження: процес синтезу мовлення за допомогою генеративних методів штучного інтелекту.

Предмет дослідження: моделі та методи синтезу мовлення.

Мета роботи: дослідження моделі синтезу мовлення з використанням нейронної мережі.

Методи дослідження. Задля вирішення поставлених задач використовувались механізми генеративного штучного інтелекту, такі, як самоувага та позиційне кодування, і математичні методи для квантування часових рядів.

Новизна отриманих результатів полягає у тому, що удосконалено процедуру навчання моделі синтезу мовлення VALL-E, що дозволило підвищити ефективність синтезу мовлення.

Практична цінність результатів полягає у тому, що навчена модель синтезу мовлення дозволяє накопичувати і використовувати знання для вирішення задач синтезу мовлення та подальшого дослідження предметної галузі.

Область застосування. Удосконалена модель може буде застосована як основа для створення синтезаторів мовлення.

Значення роботи та висновки. В ході роботи були досліджені методи та засоби розв’язання задач TTS з метою адаптивного перетворення тексту в мову. Була обрана модель на основі трансформера як найбільш раціональний спосіб синтезувати високоякісне мовлення для невидимих мовців без тонкого налаштування.

Прогнози щодо розвитку досліджень. Необхідно дослідити ефективність прийнятого рішення протягом довшого проміжку часу, ніж час тестування. Потрібно розширити навчальні дані, щоб покращити продуктивність моделі з точки зору просодії, стилю мовлення та схожості мовців.

Список ключових слів: синтез мовлення, трансформер, генеративний штучний інтелект, нульовий удар, TTS, VALL-E, NLP.

ABSTRACT

Explanatory note: 82 pages, 14 figures, 3 tables, 4 appendices, 93 sources.

Object of research: the process of speech synthesis using generative methods of artificial intelligence.

Subject of research: models and methods of speech synthesis.

Purpose of Master's thesis: to study the model of speech synthesis using a neural network.

Research methods. To solve the problems, mechanisms of generative artificial intelligence, such as self-attention and positional coding, and mathematical methods for quantization of time series were used.

Originality of obtained results is that the training procedure of the VALL-E speech synthesis model has been improved, which made it possible to increase the efficiency of speech synthesis.

Practical value of the results is that the learned model of speech synthesis allows accumulating and using knowledge to solve the problems of speech synthesis and further research in the subject area.

Scope of application. The improved model can be used as a basis for creating speech synthesizers.

The value of the work and conclusions. In the course of the work, methods and tools for solving TTS problems were investigated for the purpose of adaptive transformation of text into speech. A transformer-based model was chosen as the most rational way to synthesize high-quality speech for invisible speakers without fine-tuning.

Research forecast and development. It is necessary to investigate the effectiveness of the adopted decision during a longer period of time than the time of testing. The training data should be expanded to improve the performance of the model in terms of prosody, speaking style, and speaker similarity.

Keyword: speech synthesis, transformer, generative artificial intelligence, Zero-Shot, TTS, VALL-E, NLP.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ASR - Automatic Speech Recognition

AR – Autoregressive

CNN - Convolutional Neural Network

DL – Deep Learning

DNN - Deep Neural Network

GAN - Generative Competitive Network

GMM - Gaussian Mixing Models

GPT - Generative Pre-Trained

HMM - Hidden Markov Model

LLM - Large Language Model

LSTM - Long Short-Term Memory

MHSA - Multi-Headed Self-Attention

ML - Machine Learning

NAR – Non-autoregressive

NLP - Natural Language Processing

NMF - Non-negative Matrix Factorization

RNN - Recurrent Neural Network

SA - Self-Attention

SP - Speech Processing

SPSS – Statistical Parametric Speech Synthesis

TTS - Text to Speech

VAE - Variational Auto-Encoder

VC - Voice Conversion

ІШІ – штучний інтелект

ЗМІСТ

ВСТУП

Інноваційні інформаційні та цифрові технології все більше входять у життя сучасної людини – наприклад, системи глибокого навчання, такі, як розпізнавання голосу, зображень, розпізнавання та синтез мовлення. Мовленнєві технології широко використовуються в комунікаціях, робототехніці та інших сферах діяльності. Синтез мовлення — це завдання генерування мовлення з іншої модальності, наприклад тексту, рухів губ тощо. У більшості програм, зокрема в системах Text To Speech (TTS), саме текст вибирається як матеріал для перетворення через швидкий розвиток систем природної мови. Однак, синтезоване аудіо часто має збої або нестабільність просодії та вимови порівняно з людською мовою, і тому звучить неприродно.

У той час як більшість досліджень синтезу мовлення зосереджено на синтезі високоякісного мовлення для мовців із набором даних, не менш важливою проблемою є синтез мовлення для невидимих мовців, які знаходяться поза набором даних із обмеженими довідковими даними, тобто адаптивний синтез мовлення для мовлення. Багато досліджень запропонували підходи до адаптивного перетворення тексту в мову та перетворення голосу, спрямовані на вирішення цього завдання. Одним з потужних інструментів є генеративний штучний інтелект, який може використовувати обробку природної мови та розпізнавання мовлення, щоб зрозуміти значення та контекст введення, а також створити мову, яка відповідає бажаному тону, емоціям, акцентам і стилю.

Метою кваліфікаційної роботи є дослідження моделі синтезу мовлення з використанням нейронної мережі. Досліджувана модель має відповідати наступним критеріям: генерація зразків аудіо, якість яких дуже близька до записаних людиною, і забезпечення паралельного навчання та навчання на великій відстані (long-distance dependency), для виявлення оптимальних показників швидкості навчання та плавності звукової просодії.

Кваліфікаційна робота складається з трьох розділів. У першому розділі було розглянуто визначення поняття та характеристику процесу синтезу мови, проаналізовані методи розв’язання подібних задач, проведено порівняльний аналіз програмних інструментів задля виявлення їх слабких та сильних сторін. У другому розділі розглянуто види методи генеративного штучного інтелекту та обґрунтовано вибір способу розв’язання задач TTS з метою адаптивного перетворення тексту в мову для синтезу високоякісного мовлення для невидимих мовців без тонкого налаштування. Третій розділ містить опис та реалізацію навчання моделі синтезу мовлення на основі трансформера VALL-E.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальні відомості з предметної галузі

Метою системи синтезу мовлення є генерування штучного мовлення, схожого на людське. Найперші (цифрові) системи синтезу мовлення намагалися імітувати людську артикуляційну систему, створюючи моделі для руху губ, язика, голосової щілини та голосового тракту – це стало відомо як артикуляційний синтез [1]. Ця парадигма зіткнулася з серйозними проблемами при моделюванні артикуляційної поведінки, і від неї відмовилися заради простішої моделі джерело-фільтра, яка краще піддається контролю параметрів: формантний синтез [1]. Цей тип синтезу ґрунтується на модифікації формантних амплітуд і частот сигналу збудження на основі правил для створення необхідного висловлювання. Ці правила виводяться шляхом лінгвістичного аналізу. Хоча ця система має більшу модульність, ніж артикуляційний синтез, труднощі з визначенням відповідного набору правил призвели до відмови від неї на користь парадигм, керованих даними.

Щоб подолати труднощі, пов’язані зі створенням правильної артикуляційної моделі чи складанням повного списку формантних правил, спільнота потім звернулася до конкатенативного синтезу мовлення, де цільове висловлювання будується з набору попередньо записаних будівельних блоків: слів, складів, пів -склади, фонеми, дифони або трифони [1]. Ці попередньо записані одиниці об’єднуються, щоб створити цікаве висловлювання. Однак конкатенативний синтез страждає від ефекту розриву (оскільки просодія кожного запису може відрізнятися) і різкого збільшення даних, необхідних для охоплення всіх комбінацій одиниць.

Усі ці недоліки призвели до прийняття парадигми навчання під назвою статистичного параметричного синтезу мовлення (SPSS) [2, 3]. SPSS використовує триетапну модель, а саме: використання аналізу тексту для

відповідних лінгвістичних представлень цільового висловлювання, передбачення мовних параметрів за допомогою акустичної моделі та остаточний синтез сигналу (вокодування).

Зокрема, модуль аналізу тексту включає необхідні етапи попередньої обробки (нормалізація тексту, перетворення графем у фонеми тощо) з подальшим виділенням усіх відповідних функцій, таких як фонеми, тривалість або теги частини мови. Ці функції разом із супровідними мовними параметрами передаються в модель статистичного машинного навчання (ML), яка вивчає відображення лінгвістичних характеристик на акустичні (наприклад, основна частота, спектр або кепстр). Завдяки послідовному характеру цих даних, приховані моделі Маркова (HMM) досягли успіху в цьому типі моделювання [2]. Нарешті, акустичні характеристики поширюються на відповідний вокодер для етапу синтезу. Деякі відомі вокодери WORLD [4] і STRAIGHT [5]. Важливо підкреслити, що деякі (навіть усі) з цих кроків можна дізнатися з даних – саме це і дало назву цьому сімейству методів. Зокрема, щоб дізнатися будь-яке відображення від графем до фонем до акустики та хвилі, необхідні відповідні дані (тобто відповідні пари тексту та мовлення, часто отримані від кількох мовців і у великій кількості). Ця фундаментальна властивість SPSS робить його родоначальником сучасних методів глибокого синтезу мовлення.

1.2. Глибокий синтез мовлення

Синтез на основі глибокої нейронної мережі об’єднує нейронні мережі як моделі вибору для заміни одного або кількох компонентів традиційного конвеєра SPSS. Перші спроби зазвичай зосереджувалися на заміні HMM послідовними моделями (RNN [6] або довгострокові мережі короткочасної пам'яті (LSTM) [7]) для акустичного моделювання, наприклад ранні системи DeepVoice [8]. WaveNet була першою нейронною моделлю, яка безпосередньо генерувала хвилю з лінгвістичних особливостей [9]. Пізніше за цим послідували моделі, які намагалися перейти безпосередньо від послідовності символів/фонем до аудіо,

наприклад, Tacotron [10]. Визначальні характеристики глибокого синтезу мовлення, таким чином, є наступними: а) методи дотримуються формулювання SPSS, б) усі методи використовують глибокі нейронні мережі на деяких або всіх кроках свого конвеєра, і в) деякі методи включають частину або всі проміжні елементи.

Методи на основі DL можна класифікувати за кількома категоріями:

• авторегресійні (AR) [9] і неавторегресійні (NAR) структури [12];

• тип мережевої структури: згорточні нейронні мережі (CNN) [8], послідовні моделі (RNN, мережі зі стробованим рекурентним блоком (GRU), LSTM) [10], та моделі самоуваги (трансформери) [13, 14];

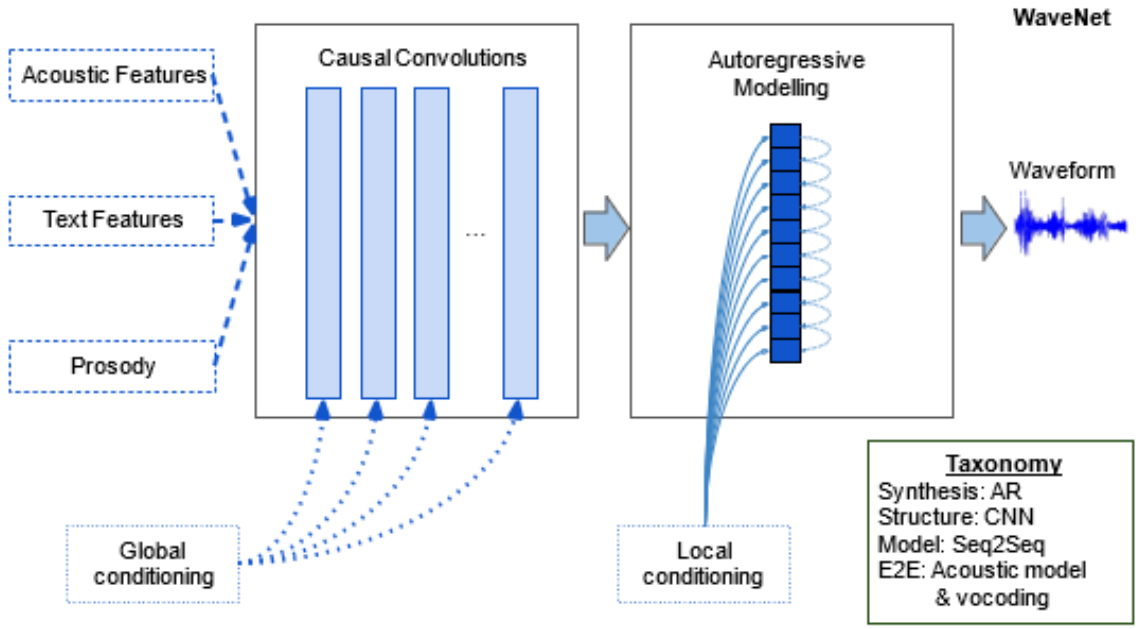
• тип генеративної моделі (наприклад, варіаційний автокодер (VAE) [15], GAN [16]);

• ступінь наскрізної поведінки, яка характеризується етапами традиційного конвеєра SPSS, які включають одну або більше (спільно навчених) моделей.

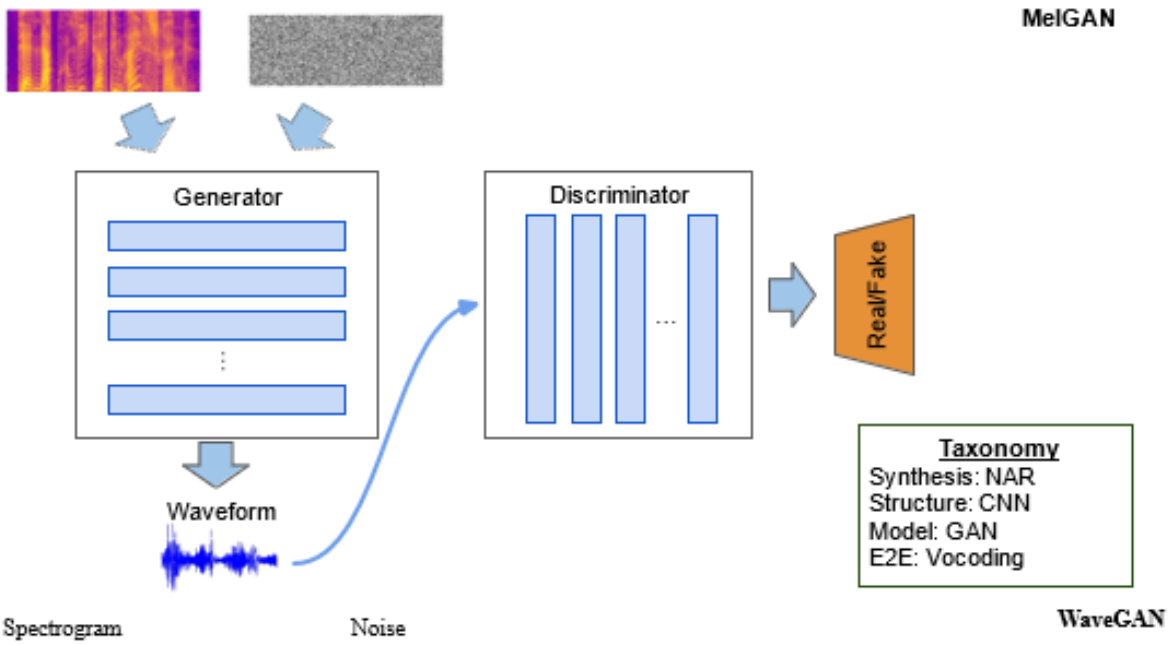
Хоча така категоризація корисна для розрізнення різних підходів TTS – важливо підкреслити, що межі між цими категоріями є рухливими та постійно змінюються. Наприклад, у той час як WaveNet спочатку було представлено як модель авторегресії, яка генерує хвилю безпосередньо з лінгвістичних особливостей [9], таким чином об’єднуючи акустичну модель і аспекти вокодування конвеєра SPSS, пізніше її було розширено до неавторегресійного синтезу [17] і змінено на виробляти мовлення, обумовлене Mel-спектрограмами, а не лінгвістичними особливостями [10].

Огляд останніх ключових наробок TTS зі сфери глибокого навчання показано на рис.1.1. Як згадувалося раніше, WaveNet [9] була першою нейронною моделлю, запропонованою для синтезу мовлення. У своєму першому вступі вона була концептуалізована як відображення текстових і просодичних особливостей у необроблену форму сигналу – таким чином інтегруючи останні два кроки конвеєра SPSS. WaveNet також представила дві ключові інновації в галузі аудіомоделювання: а) використання розширених згорток, що дозволило

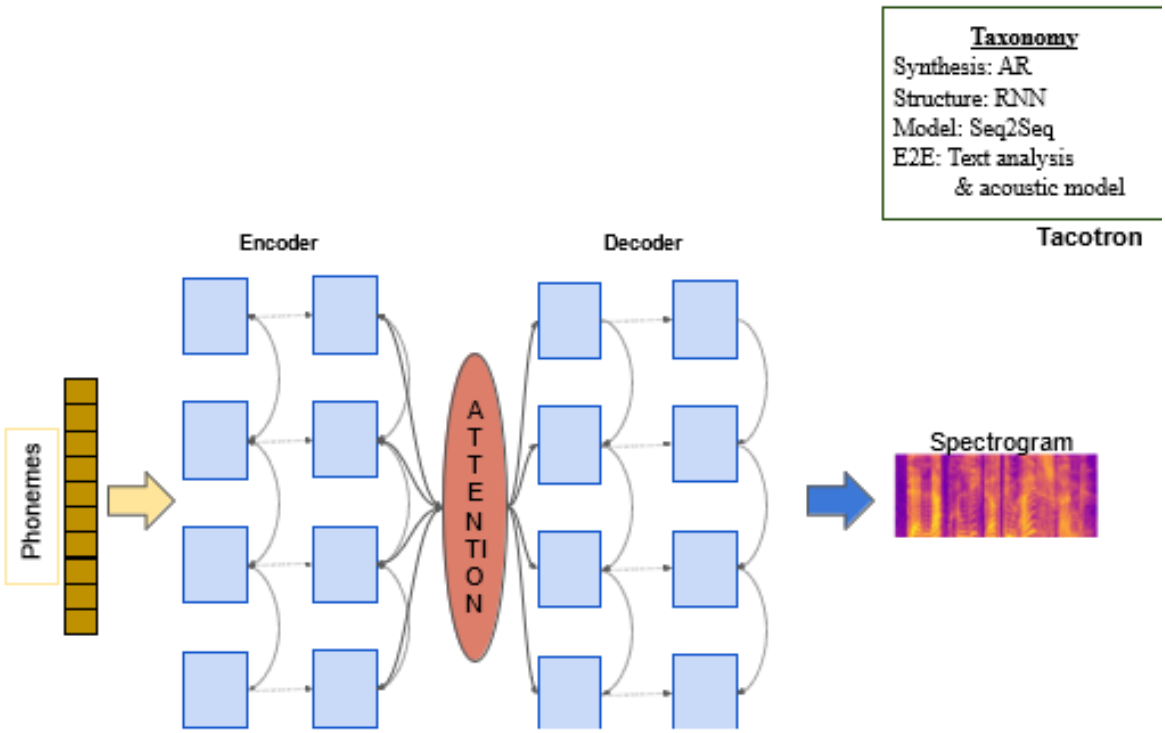
збільшити сприйнятливе поле та змоделювати взаємодії на великій відстані; б) здатність глобально та локально обумовлювати процес генерації. Подальші ітерації адаптували модель для прийняття Mel-спектрограм як вхідних даних [10], таким чином ефективно перетворюючи її на більш традиційний вокодер.



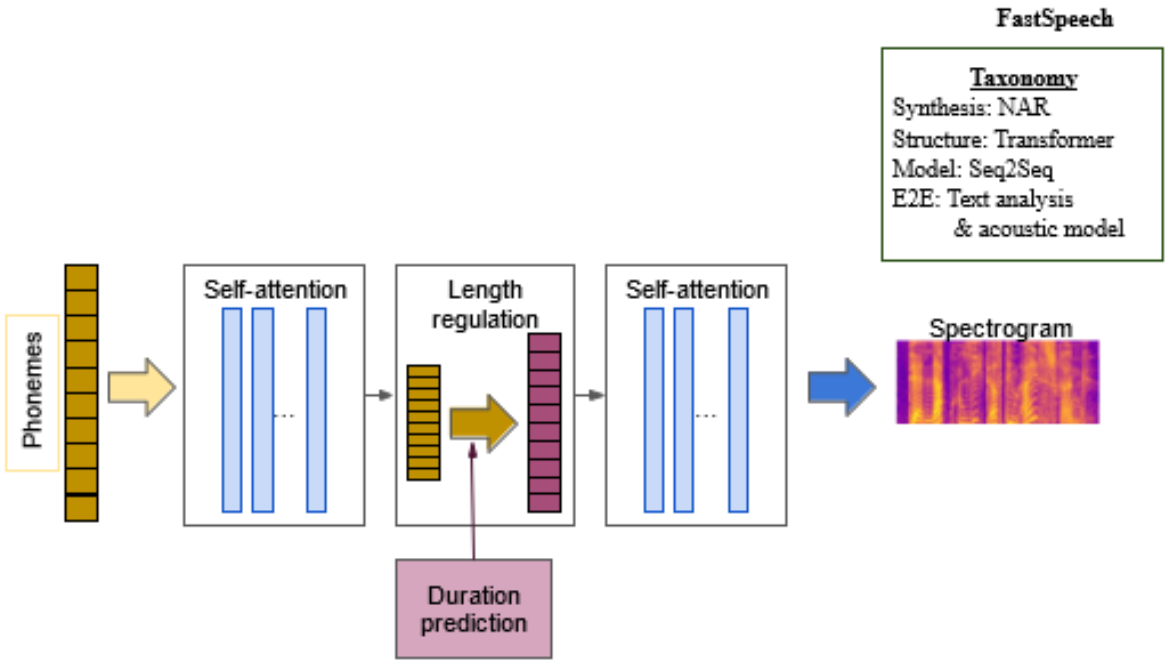
а



б



в



г

Рис. 1.1. а – WaveNet; б – GAN; в – Tacotron; г- FastSpeech.

Tacotron [11] підійшов до нейронної TTS, поєднавши два інтерфейси конвеєра SPSS, аналіз тексту та акустичне моделювання, використовуючи

структуру кодувальник-увага-декодер. Покладаючись на моделі seq2seq, додатково доповнені увагою, Tacotron вивчає відображення фонем/символів у спектрограми. Потім ці спектрограми подаються у відповідний вокодер; для цієї мети Tacotron1 використовував Griffin-Lim [1], тоді як Tacotron2 використовував WaveNet [10]. Через послідовну природу кодера та декодера Tacotron страждає від повільнішої обробки та труднощів у вирішенні довгострокових залежностей.

Після недавнього успіху архітектур самоуваги в моделюванні таких залежностей [18] і їхньої здатності генерувати вихідні дані неавторегресійним способом шляхом паралельної обробки вхідних даних, трансформери були представлені як альтернатива RNN у моделі FastSpeech [14]. FastSpeech покладається на серію блоків трансформера для кодування вхідної текстової послідовності; інша серія блоків декодує його до вихідних акустичних характеристик, які потім служать вхідними даними для відповідного вокодера. Хоча трансформери мають перевагу паралельної обробки всієї послідовності, таким чином зменшуючи час виконання під час логічного висновку, вони потребують деяких адаптацій для вирішення проблеми невідповідності довжини послідовності, оскільки цільові акустичні характеристики зазвичай мають набагато більшу тривалість, ніж вхідний текст. Це обробляється мережею передбачення тривалості, яка навчена передбачати цю невідповідність і підвищує дискретизацію вивчених представлень кодера до необхідної довжини перед тим, як поширювати їх на декодер. FastSpeech2 також навчений спільно прогнозувати висоту та енергію цільової мови, яка потім використовується для подальшої модуляції вивчених представлень кодера під час логічного висновку та покращення виразності.

Ще одним ключовим внеском у сукупність нейронних підходів TTS є впровадження генеративних змагальних мереж (GAN). Після основоположної роботи Гудфеллоу та ін. [16], GAN стали основою для створення зображень, відео та аудіо. Для TTS існують дві основні категорії GAN. Перший — це вокодери GAN, генератори яких приймають як вхідні спектрограми та виводять необроблену форму хвилі, яка згодом перевіряється за допомогою

дискримінатора на її «реалістичність». Ключовими прикладами цієї категорії є MelGAN [19], Parallel WaveGAN [20], HiFi-GAN [21] та інші.

Наближаючись до оригінальної формули Гудфеллоу та ін. [16], друга категорія включає такі моделі, як WaveGAN [22], які намагаються створити реалістичну мову з випадкових вхідних даних. Таким чином, вони фактично замінюють всю модель генерації мовлення, включаючи вибір відповідного тексту для виведення, однією моделлю.

1.3. Глибоке перетворення голосу

Перетворення голосу (VC) — це завдання, щоб мовне висловлювання вихідного мовця звучало так, ніби воно надійшло від цільового мовця, при цьому мовний зміст залишається незмінним. Необхідно маніпулювати декількома атрибутами, щоб зробити мову однієї особи схожою на мову іншої. Перший – це вибір самих слів. Різні люди використовують різні словники та стилі мовлення [23, 24]; тому, щоб ефективно трансформувати «ідентичність» мовленнєвого висловлювання, слід починати зі слів, які його утворюють. Однак, оскільки пізніше ми також будемо ігнорувати зміни словникового запасу, викликані змінами в емоціях, ми також ігноруємо цей важливий аспект перетворення голосу. Замість цього ми зосереджуємося на двох інших атрибутах: надсегментних особливостях, таких як просодія, і сегментарних, таких як спектр і форманти. Короткочасні спектральні характеристики є корелятами тембру, який фіксує «тон» висловлювання та пов’язаний із фізіологічними характеристиками мовця [24]. Просодія фіксує як фізіологічні характеристики, так і стиль мовлення [24]. З цієї причини деякі роботи VC розглядають лише тембр; це, однак, обмежує успіх цих методів, оскільки виявлено, що імітатори людей також адаптують свою просодію [24].

Історія досліджень VC налічує понад 30 років. Ранні підходи використовували артикуляційний синтез, але синтезували мову з використанням параметрів цільового мовця [1]. Більш пізні спроби використовували моделі

змішування Гауса (GMMs) [1], рамки на основі зразків, засновані на факторизації невід'ємної матриці (NMF) [1], і HMMs [1]. Однак, незважаючи на кілька спроб, помітного прогресу не було досягнуто до останніх років, коли з'явилися глибокі нейронні мережі (DNN). Нещодавно запропоновані методи використовують можливості представлення DNN за допомогою моделей VAE [25, 26], GAN [27, 28] і seq2seq [29].

Ключова відмінність підходів VC полягає між тими, які використовують паралельні та непаралельні навчальні дані. Паралельні дані означають, що висловлювання ідентичного мовного змісту доступні як від джерела, так і від цільового мовця. Незважаючи на те, що цей тип даних полегшує вивчення відображення функцій, які фіксують ідентичність мовця, зберігаючи вміст незмінним, їх важче отримати в достатній кількості, особливо для методів DL, які потребують більше даних. З цієї причини алгоритми, що спираються на більш непаралельні дані, стали більш помітними в останні роки.

Загалом, потужність DL полягає в його здатності вивчати складні функції відображення з даних. Під час перетворення голосу ця здатність використовується для вивчення перетворення вхідного мовного сигналу в цільовий, зазвичай шляхом перетворення характеристик вихідного динаміка на цільовий динамік перед вокодуванням. У глибокому перетворенні голосу це відображення може бути досягнуто за допомогою механізму кондиціонування, подібного до того, який запровадив WaveNet [9]. WaveNet підтримує як глобальне, так і локальне кондиціонування – ці інтерфейси кондиціонування можуть бути кооптовані алгоритмами перетворення голосу для зміни надсегментних і сегментних атрибутів відповідно. Таким чином, представлення особистості мовця стає важливим аспектом ВК. Це можна зробити або за допомогою кодування фіксованого набору гучномовців [26], дверних пристроїв [25], або за допомогою функцій вузького місця, як представлення гучномовців з DNN [30].

Як і у випадку з TTS, GAN також відіграють важливу роль у глибоких перетвореннях голосу. Структура GAN-VC сформульована за допомогою

генератора для відображення висловлювання від джерела до цільового мовця, з дискримінатором, який використовується для керівництва навчанням, класифікуючи, чи цільовий оратор справді правильний. Оскільки це відображення може бути важко вивчити з непаралельних даних, пропонується додаткова форма регуляризації шляхом введення втрати узгодженості циклу [31], що призводить до CycleGAN [28, 32]. CycleGAN має два генератори, один для перетворення мовлення вихідного мовця в цільовий, а інший для зворотного перетворення. Це використовується для зіставлення мовлення вихідного/цільового мовця з цільовим/джерелом і назад і забезпечення узгодженості (через втрату L_1) з оригінальним вихідним/цільовим висловлюванням. У процесі це гарантує, що шуканий генератор (від джерела до цілі) належним чином навчений. У StarGAN [27, 33] знайдено розширення узгодженої з циклом генеративної змагальної мережі (CycleGAN) для кількох мовців, яка поширює принцип узгодженості на кілька доменів джерело-ціль. Нарешті, моделі seq2seq, які використовують архітектуру кодера-декодера, також були широко вивчені в області VC [29]. Ці моделі мають додаткову перевагу обробки змін у довжині послідовності, спричинених змінами в функціях. Наприклад, зміна просодії може зробити висловлювання цільового мовця коротшим або довшим, ніж вихідний мовець, що нелегко впоратися за допомогою методів покадрового відображення.

1.4. Постановка задачі

Ранні підходи глибокого навчання в області обробки мовлення (SP) зазвичай використовували варіанти згорткових нейронних мереж (CNN) [34]. Однак недоліком цих підходів, заснованих на CNN, є їх нездатність охопити послідовний характер мовних даних. Це обмеження CNN призвело до розробки архітектур послідовності до послідовності (seq2seq), таких як RNN і мережі довготривалої короткочасної пам’яті (LSTM), які спеціально розроблені для послідовних даних. RNN добре підходять для послідовних даних, оскільки вони

можуть обробляти довгі послідовності крок за кроком з обмеженою пам’яттю попередніх елементів послідовності. Нещодавно дослідники також поєднали сильні сторони CNN і RNN, використовуючи CNN для вилучення звукових функцій і використовуючи ці функції як вхідні дані для навчання RNN. Проте було показано, що RNN мають проблеми з проблемою зникнення або вибуху градієнта. Щоб вирішити цю проблему, LSTM використовують механізм стробування та комірки пам’яті для контролю потоку інформації та пом’якшення проблем градієнта. Багато варіацій LSTM, таких як Frequency-LSTM, Time-Frequency LSTM, Bi-directional LSTM, ConvLSTM і Stacked LSTM, були запропоновані для завдань SP [35]. Незважаючи на свою ефективність, моделі seq2seq обмежені в важливих аспектах: вони не можуть скористатися перевагами апаратного забезпечення паралельних обчислень і мають труднощі з моделюванням довгострокового контексту.

Трансформери мають перевагу в розумінні мовлення, оскільки вони аналізують все речення одночасно, тоді як RNN можуть обробляти лише менші частини за раз. Це стало можливим завдяки унікальній архітектурі трансформерів на основі самоуваги [36], яка дозволяє їм вивчати довготривалі залежності, що є критичним для завдань обробки мовлення. Крім того, механізм уваги з кількома головками [37] — спеціальна функція в трансформерах — дозволяє більш ефективно розпаралелювати під час навчання, що робить їх ідеальними для обробки великих наборів даних, що є загальною проблемою в задачах обробки мовлення. Ця унікальна комбінація самоуваги та уваги кількох головок дає змогу трансформерам досягати виняткової продуктивності в послідовному моделюванні, що робить їх незамінним інструментом для обробки мови.

Задача синтезу мовлення, яка полягає у навчанні синтезатора вимовляти інші слова заданим голосом, має проблему на етапі навчання моделі мовленевого синтезу. Оскільки нейронна мережа не сприймає різнорідні вхідні дані, лише «прямокутні», для навчання моделі мовленевого синтезу неможливо використовувати «живу» мову без попередньої обробки, бо дуже складно

автоматично поділити її на слова. Для вирішення проблеми в даній кваліфікаційній роботі пропонується використовувати трансформер, який перетворюватиме дані перед їх подачею на вхід нейронної мережі. Таким чином, модель синтезу мовлення для навчання повинна складатися з варіаційного авто-інкодера, який створює з навчальної вибірки стислий простір з ознаками, і декодера. Декодер буде використовуватися окремо, для отримання інших слів, бо декодер повторює стилістику (використовує буквосполучення).

1.5. Висновки

У першому розділі були розглянуті та порівняні між собою засоби синтезу мовлення. Наведений аналіз демонструє, що найчастіше для розв’язання задач синтезу мовлення з тексту та перетворення мовлення застосовуються методи на основі нейронних мереж.

Було наведено опис та класифікацію сучасних методів на основі глибокого навчання за критеріями регресійної структури, типу мережевої структури, типу генеративної моделі, та ступеню наскрізної поведінки.

Для вирішення проблеми, окресленій в даній кваліфікаційній роботі, були обрані методи на основі трансформера.

РОЗДІЛ 2

МОДЕЛІ ТА МЕТОДИ РОЗВ'ЯЗАННЯ ЗАДАЧІ

2.1. Генеративні методи штучного інтелекту

Генеративний штучний інтелект (ШІ) - це технологія, що використовує моделі машинного навчання, зокрема неконтрольовані та напівконтрольовані алгоритми, для створення нового вмісту на основі наявних даних.

Генеративний ШІ «розуміє» основні закономірності у вхідних даних, що дозволяє створювати нові результати, які нагадують вихідні дані. Цьому процесу сприяють нейронні мережі, які можуть створювати широкий спектр контенту, від зображень і відео до тексту та аудіо.

Генеративні змагальні мережі (GAN) і моделі на основі трансформерів, такі, як мовні моделі Generative Pre-Trained (GPT), є двома найбільш широко використовуваними моделями генеративного ШІ. GAN можуть створювати візуальні та мультимедійні артефакти із зображень і текстових вхідних даних, тоді як моделі на основі трансформерів можуть генерувати текстовий вміст, використовуючи інформацію з Інтернету.

Генеративний ШІ має широкий спектр застосувань, включаючи створення мистецтва, покращення та доповнення даних у комп'ютерному зорі, генерацію синтетичних даних для навчання інших моделей машинного навчання та обробку природної мови, де він використовується для LLM (Large Language Model, великі мовні моделі), наприклад, GPT.

Ключові генеративні методи ШІ:

- генеративні змагальні мережі (GAN);
- трансформери;
- варіаційні автокодери.

Генеративні змагальні мережі (GAN).

GAN [16]— це клас алгоритмів штучного інтелекту, які використовуються в неконтрольованому машинному навчанні, представлені

Яном Гудфеллоу та його колегами в 2014 році. Вони складаються з двох частин: генератора, який створює зображення, і дискримінатора, який намагається відрізнити реальні зображення від згенерованих. Конкуренція між цими двома мережами породжує нові, синтетичні зображення, які майже не відрізняються від реальних. GAN знайшли застосування в різних сферах, включаючи мистецтво, рекламу та охорону здоров'я. GAN використовуються для синтезу зображеннь, семантичного редагування зображень, передачі стилю, суперроздільності зображення та класифікації.

Генеративні моделі ШІ поєднують різні алгоритми ШІ для представлення та обробки вмісту. Такі методи, як GAN і варіаційні автокодери (VAE), підходять для генерації реалістичних людських облич, синтетичних даних для навчання ШІ або навіть факсиміле конкретних людей.

Трансформери.

Модель трансфермеру [36] реалізує кулька інноваційних механізмів, найважливішим з яких є механізм самоуваги [37], який дозволяє моделі зважувати релевантність слова в реченні іншим словам під час генерування результату. Цей механізм дозволяє моделі обробляти довгострокові залежності в тексті ефективніше, ніж попередні моделі. Трансформер також представиляє концепцію позиційного кодування, яка дозволяє моделі враховувати положення слів у реченні.

GPT-3 і GPT-4, дві найпотужніші генеративні моделі ШІ, засновані на архітектурі трансформера. Ці моделі використовувалися для генерування тексту, перекладу мов, допомоги у виконанні завдань кодування та відповідей на запитання у корисний та інформативний спосіб.

Здатність трансформера обробляти довгострокові залежності в тексті та його масштабованість зробили його кращою моделлю для розробки найсучасніших генеративних моделей ШІ.

Варіаційні автокодери (VAE)

Варіаційні автокодери (VAE) [15]— це генеративні моделі, представлені Кінгмою та Веллінгом у фундаментальній статті 2013 року «Автоматичне

кодування варіаційного Байєса», які привнесли новий підхід до генеративного моделювання шляхом поєднання глибокого навчання та імовірнісного графічного моделювання.

Ключовою ідеєю VAE є впровадження моделі латентної змінної з безперервним латентним простором і механізмом наближеного висновку на основі нейронних мереж. Модель навчена максимізувати нижню межу логарифмічної правдоподібності даних, яку можна ефективно обчислити за допомогою методів стохастичного градієнтного спуску. Цей підхід забезпечує ефективне навчання та висновок у складних багатовимірних наборах даних.

VAE мають унікальну архітектуру, що складається з двох основних компонентів: кодера та декодера. Кодер бере вхідні дані та кодує їх у прихований простір, фіксуючи базовий розподіл даних. Потім декодер бере ці приховані змінні та реконструює вихідні вхідні дані. Ця архітектура дозволяє VAE генерувати нові дані, які нагадують навчальні дані, що робить їх потужним інструментом для генеративних завдань.

У контексті генеративного ШІ VAE відіграють вирішальну роль. Вони забезпечують надійну та гнучку структуру для вивчення складних розподілів даних, що важливо для створення реалістичних та різноманітних даних. Крім того, імовірнісний характер VAE дозволяє визначити певну невизначеність у створених даних, що може бути вирішальним у багатьох програмах.

Генеративний штучний інтелект має багато переваг, але також і певні недоліки. Як і всі моделі машинного навчання, генеративні моделі штучного інтелекту сприйнятливі до упередженості в своїх навчальних даних. Якщо навчальні дані містять упередження, генеративна модель ШІ, ймовірно, відтворить і посилить ці упередження. Це може призвести до несправедливих або дискримінаційних результатів.

Також генеративні моделі ШІ зазвичай вимагають великого обсягу навчальних даних для отримання високоякісних результатів. Збір і зберігання цих даних може бути ресурсомістким.

2.2 Трансформери

Трансформери вперше були представлені в основоположній роботі Васвані та ін. [36] для завдань машинного перекладу в обробці природної мови і показали вражаючу продуктивність в інших областях, включаючи комп'ютерне зір, медичне зображення та дистанційне зондування. Моделі трансформерів використовують шари самоуваги для ефективного захоплення довгострокових залежностей між вхідною послідовністю, на відміну від традиційних рекурентних нейронних мереж, яким важко вловити таку взаємодію. Модель (рис. 2.1) містить шари кодера та декодера, кожен з яких містить підрівні самоуваги та прямої передачі. Кодувальник виробляє приховані стани з послідовності вхідних маркерів, тоді як декодер генерує прогнози з послідовності вихідних маркерів і звертає увагу на стани кодувальника для отримання вхідної інформації.

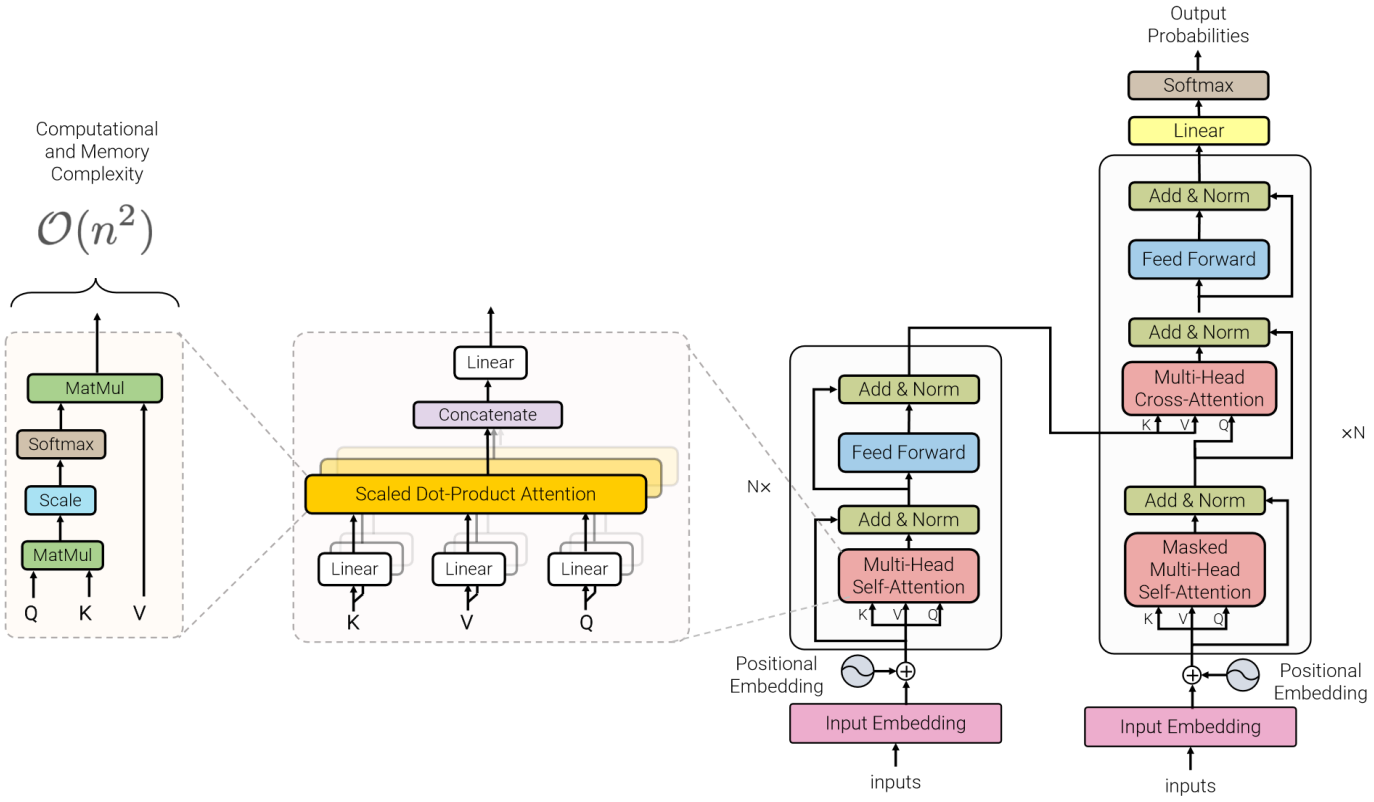


Рис. 2.1. Архітектура стандартного трансформера (адаптовано з [36, 38])

Крім того, самоувага дозволяє більш розпаралелювати порівняно з рекурентними нейронними мережами, оскільки вони можуть обробляти послідовність мовлення в цілому, не покладаючись на минулі стани для захоплення залежностей. Більш конкретно, Васвані та ін. представили два типи уваги, включаючи (1) масштабовану увагу за множинним продуктом і (2) багатоголовну увагу. Крім того, позиційне кодування також використовується для введення інформації про відносну або абсолютну позицію токенів у послідовності. Завдяки цим властивостям трансформери викликали величезний інтерес у мовленнєвій спільноті, і було запропоновано декілька підходів, які базуються на трансформерах. Далі подано короткий огляд основних компонентів трансформерів:

1) Рівень самоуваги: рівень самоуваги (SA) має на меті охопити внутрішню кореляцію послідовності або ознак шляхом агрегування глобальної інформації з усієї вхідної послідовності, завдання, яке є складним для звичайних повторюваних моделей. Зокрема, вхід $\mathbf{X} \in R^{N\times D}$, що складається з N сутностей кожна має розмірність D, перетворюється на матриці запиту $\boldsymbol{Q} \in R^{N\times D_k}$, ключа $\boldsymbol{K} \in R^{N\times D_k}$ і значення $\boldsymbol{V} \in R^{N\times D_k}$ за допомогою вагових матриць $\boldsymbol{W}^Q \in R^{N\times D_k}$, $\boldsymbol{W}^K \in R^{N\times D_k}$, і $\boldsymbol{W}^V \in R^{N\times D_k}$ відповідно. Математично

$$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}^Q, \quad \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}^K, \quad \boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}^V. \tag{2.1}$$

Потім обчислюється скалярний добуток матриці запиту Q з усіма ключами Q у заданій послідовності, а результуюча матриця нормалізується за допомогою оператора softmax, щоб отримати матрицю уваги $\boldsymbol{A} \in \mathrm{R}^{N\times N}$ як

$$A = softmax\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{D_k}}\right)\boldsymbol{V} \tag{2.2}$$

Результатом шару SA $\boldsymbol{Z}$ є матриця уваги $\boldsymbol{A}$, помножена на матрицю значень $\boldsymbol{V}$:

$$\boldsymbol{Z} = \boldsymbol{A}\boldsymbol{V} \tag{2.3}$$

2) Замаскована самоувага: в оригінальному документі [36] блоки SA, що використовуються в декодері, маскуються, щоб запобігти зверненню уваги на

наступні майбутні сутності шляхом поелементного множення маски $\boldsymbol{M} \in \mathrm{R}^{N\times N}$ як:

$$\boldsymbol{Z} = softmax\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}} \circ \boldsymbol{M}\right)\boldsymbol{V}, \tag{2.4}$$

де $\boldsymbol{M}$ – верхня трикутна матриця, а ◦ позначає добуток Адамара. Це називається прихованою самоувагою.

3) Багатоголова увага: замість того, щоб обчислювати увагу один раз, багатоголова самоувага (MHSA) складається з кількох блоків SA. Ці блоки SA об’єднані разом по каналах для моделювання залежностей між різними елементами у вхідній послідовності. Кожна голова в MHSA має власні вагові матриці, які можна вивчати, позначаються $\{\boldsymbol{W}^{Q_i}, \boldsymbol{W}^{K_i}, \boldsymbol{W}^{V_i}\}$, де $i = 0\cdots(h-1)$, а h означає загальну кількість голів у блоці MHSA. Зокрема,

$$\mathrm{MHSA}(\mathbf{Q},\mathbf{K},\mathbf{V}) = [\mathbf{Z}_0,\mathbf{Z}_1,...,\mathbf{Z}_{\mathrm{h}-1}]\mathbf{W}^{\mathrm{O}}, \tag{2.5}$$

тоді як $\boldsymbol{W}^{O} \in R^{h.D_k\times N}$ обчислює лінійне перетворення головок, а $\mathbf{Z}_i$ можна записати як

$$\mathbf{Z}_i = softmax\left(\frac{\mathbf{Q}\mathbf{W}^{Q_i}(\mathbf{K}\mathbf{W}^{K_i})^T}{\sqrt{D_k/h}}\right)\mathbf{V}\mathbf{W}^{V_i} \tag{2.6}$$

4) Позиційне кодування: механізм самоконтролю в трансформерних моделях дозволяє обробляти вхідні мовні кадри без певного порядку чи позиції. Для врахування цього використовується позиційне кодування, щоб надати моделі трансформера інформацію про порядок вхідної послідовності. Це робиться шляхом асоціювання кожної позиції у вхідній послідовності з вектором, який допомагає трансформеру вивчати позиційні зв’язки.

Позиційне кодування можна вивчити під час навчання або попередньо визначити, і його можна закодувати у відносних або абсолютних значеннях для завдань SP.

2.3. Синтез мовлення з використанням штучного інтелекту

2.3.1. Автоматичне розпізнавання мовлення (ASR)

ASR дозволяє машинам розпізнавати вимовлену мову та перетворювати її у відповідну послідовність тексту (слів або підслів). Сучасні системи ASR досягли підвищеної продуктивності завдяки використанню RNN з блоками LSTM [7] як магістральних мереж. Останнім часом зростає інтерес до використання трансформерів [14] для ASR, натхненний їхнім успіхом у різних завданнях NLP, таких як мовне моделювання [56] та машинний переклад [57]. RNN обробляють вхідний сигнал у послідовний спосіб, використовуючи дорогий алгоритм зворотного поширення в часі [58] для вивчення часових залежностей. Трансформери обходять це за допомогою механізму самоуваги, щоб вловлювати часові кореляції між послідовними даними. Це дає змогу трансформерам фіксувати довші часові кореляції з меншою складністю обчислень. Ще однією перевагою використання трансформера є можливість розпаралелювати обчислення в трансформерах, що може скоротити час для навчання глибших моделей на більших наборах даних.

У ASR трансформери досягли конкурентоспроможного рівня розпізнавання порівняно з базовими моделями на основі RNN. Наприклад, Karita et al. [59] експериментально порівнювали трансформери зі звичайними RNN. Ґрунтуючись на результатах, вони продемонстрували різноманітні переваги в навчанні та продуктивності, досягнуті за допомогою трансформерів у порівнянні з RNN. У [60] Zeyer et al. провели порівняння трансформерної моделі кодера декодера-уваги з RNN і виявили, що трансформери є більш стабільними порівняно з LSTM, однак вони стикаються з проблемою переобладнання та узагальнення. Вони також виявили, що попереднє навчання призводить до швидшої конвергенції та підвищення продуктивності. Лі та ін. [61] провели порівняння між RNN і наскрізними моделями на основі трансформерів, використовуючи 65 тисяч годин анонімних навчальних даних Microsoft. Вони виявили, що архітектура кодера-декодера уваги на основі трансформера досягла

найкращої точності. Подібним чином дослідження [62], [63] також провели порівняння трансформерів з різними системами ASR і підкреслили переваги трансформерів і вказівки для майбутніх досліджень.

Повторювані моделі послідовності до послідовності досягли значного прогресу в ASR. Ці моделі базуються на архітектурі кодера-декодера, де кодер перетворює послідовність мовних ознак у приховані представлення та генерує вихідну послідовність. Звичайні моделі послідовності на основі RNN страждають від проблем із повільним навчанням і паралелізацією навчання. У моделі послідовності до послідовності на основі трансформера мережа кодера та декодера складається з мереж концентрації уваги та позиції, а не RNN. Крім того, виходи кодера супроводжуються кожним блоком декодера відповідно. Це робить навчання моделей послідовності на основі трансформерів швидшим і дозволяє проводити паралельне навчання.

Трансформери також продемонстрували багатообіцяючі результати у великомасштабному ASR. Чен та ін. [64] оцінили потенціал моделей трансформерних перетворювачів для декодування першого проходу з низькою затримкою та високою швидкістю на великомасштабному наборі даних ASR. На основі експериментів вони показали, що модель Transformer Transducer перевершує RNN Transducer (RNN-T) [65], потоковий трансформатор і гібридну модель у потоковому сценарії. Wang та ін. [62] виконали порівняльне дослідження акустичної моделі на основі трансформера на великомасштабній ASR. Вони виявили, що модель ASR на основі трансформера забезпечує кращу продуктивність порівняно з LSTM для завдань голосового помічника. Вищезазначені дослідження показують ефективність трансформерів для ASR.

2.3.2. Нейронний синтез мовлення

Нейронний синтез мовлення (TTS), є важливою галуззю досліджень, метою якої є синтез мовлення з введеного тексту. Традиційні системи TTS складаються зі складних компонентів, включаючи акустичні інтерфейси, модель

тривалості, модель акустичного прогнозування та моделі вокодера [66]. Складність TTS нещодавно подолано за допомогою глибоких наскрізних архітектур TTS [11], [8]. Ці системи можуть синтезувати мовлення з реалістичним звучанням, тренуючись на парах <текст, аудіо>, і усувають потребу в складних підкомпонентах і їх окремому навчанні. Відомі моделі включають Tacotron [11], Tacotron 2 [10], Deep Voice 3 [67] і Clarinet [68]. Ці моделі генерують Mel-спектрограму з введеного тексту, який потім використовується для синтезу мови за допомогою вокодерів, таких як Griffin-Lim [52], WaveNet [70] і Waveglow [71].

Останнім часом трансформери стають популярними для генерації Mel-спектрограми в системах TTS. Зокрема, вони замінюють структури RNN у наскрізному TTS для підвищення ефективності навчання та логічного висновку. У [13] Лі та ін. спробував використати механізм уваги з кількома головами для заміни структур RNN, а також механізм уваги у Tacotron 2. Це допомагає покращити множинність шляхом вирішення проблеми залежності на великій відстані. Вони створили Mel-спектрограму, використовуючи послідовності фонем як вхідні дані, і використали WaveNet як вокодер для синтезу зразків мови. На основі результатів вони показали, що трансформер TTS зміг прискорити навчання в 4,25 рази порівняно з Tacotron 2 і досягти подібної продуктивності MOS.

Fastspeech 2 [14] — це ще одна система TTS на основі трансформера, яка вирішила проблеми у Fastspeech і краще вирішила проблему відображення один до багатьох у TTS. Він використовує більш різноманітну інформацію про мовлення (наприклад, енергію, висоту та точнішу тривалість) як умовні вхідні дані та безпосередньо навчає систему на наземній цілі. Fastspeech 2s є ще одним варіантом, запропонованим у [14], який додатково спрощує конвеєр синтезу мовлення шляхом безпосереднього генерування мовлення з тексту в логічному висновку без використання Mel-спектрограм як проміжного виходу. Експерименти з даними LJSpeech показали, що FastSpeech 2 і FastSpeech 2s

представляють спрощену систему навчання зі швидким, надійним і керованим синтезом мовлення порівняно з FastSpeech.

Однак нейронні моделі TTS можуть страждати від проблем із надійністю та створювати погані зразки звуку для невидимого або незвичайного тексту. Щоб подолати ці проблеми, Li et al. запропонував RobuTrans, надійний трансформер, який перетворює введені тексти на лінгвістичні функції перед подачею їх у кодер [72]. Вони також модифікували механізм уваги та вбудовування позиції, щоб покращити вивчення цілісної інформації з вхідних даних, що призвело до кращих показників MOS порівняно з іншими популярними моделями TTS. Іншим підходом до досягнення стійкості в системах TTS є сегментний трансформер (s-Transformer) [73], запропонований Wang et al. S-Transformer здатний моделювати мовлення на рівні сегмента, дозволяючи йому фіксувати довготривалі залежності та використовувати кодер-декодер на рівні сегмента для обробки довгих пар послідовностей. Цей підхід дозволяє sTransformer досягти подібної продуктивності до стандартного трансформера, водночас демонструючи надійність наддовгих речень. Нарешті, Zheng et al. [74] запропонували підхід, який включає локальну рекурентну нейронну мережу в трансформер для захоплення як послідовної, так і локальної інформації в послідовностях. Оцінка 20-годинного корпусу мовлення китайською мовою показала, що ця модель перевершує сам трансформер з точки зору продуктивності.

Останні дослідження продовжують розширювати межі систем синтезу мовлення, вивчаючи різні підходи до покращення продуктивності.

2.3.3. Переклад мовлення

Переклад мовлення (ST) — це процес перекладу усного мовлення мовою оригіналу на мову перекладу. Системи ST зазвичай діляться на дві категорії: каскадні системи та наскрізні системи. Каскадні системи ST складаються з системи автоматичного розпізнавання мови (ASR) і системи машинного

перекладу. Система ASR генерує текст із вимовленого речення, який потім використовується системою машинного перекладу для перекладу на цільову мову. Каскадні системи ST стикаються з проблемою поєднання помилок між компонентами, наприклад, помилки розпізнавання, що призводять до більших помилок перекладу. Навпаки, наскрізні системи ST оптимізують єдину модель, яка безпосередньо перекладає усне висловлювання на цільову мову. Різноманітні дослідження досліджували методи та методи покращення продуктивності як каскадних систем ST [75]–[77], так і наскрізних систем ST [78]–[80].

В даний час дослідження ST використовують трансформери для вирішення різних проблем. Віла та ін. [81] використали трансформер для наскрізного перекладу мови. Було проведено оцінювання завдання перекладу з іспанської на англійську, а також було розраховано двомовну базову оцінку, яка показала, що наскрізна архітектура змогла перевершити конкатеновані системи. Чжан та ін. [82] представили решітчастий трансформер для перекладу мови, який також використовує ґратчасте представлення на додаток до традиційного послідовного введення. Вони оцінили запропоновану модель Корпусу іспано-англійського перекладу мовлення та досягли кращих результатів порівняно з базовими результатами.

2.4. Використання трансформерів у моделях синтезу мовлення

Трансформери також стають все більш популярними в мовленні через їхню придатність для виконання різноманітних завдань, включаючи розпізнавання мовлення, покращення, синтез тексту в мовлення, розпізнавання мовця та обробку з кількома мікрофонами. Різноманітні бібліотеки з відкритим кодом, зокрема Hugging Face, SpeechBrain і torch audio, прискорюють дослідження в області мовлення. Такі великі технологічні компанії, як Google, Meta, Amazon тощо, створюють великі моделі трансформаторів, пов’язаних із мовленнєвою областю. Хоча трансформери спочатку були розроблені для завдань NLP, згодом вони були адаптовані для інших типів даних, включаючи

мову. BERT [39] — це мовна модель, яка використовує моделювання замаскованої мови (MLM) як мету попереднього навчання. BERT складається з двох модулів: рівня вбудовування, який зіставляє токени з векторами, і рівня кодувальника, який застосовує мережі самоконтролю та прямого зв’язку для вивчення контекстуалізованих представлень. У той час як BERT і подібні текстові моделі великих мов (наприклад, GPT [40], XLNet [41], T5 [42]) тощо) були успішними в різних завданнях NLP, їх застосування для обробки мови обмежене через кілька недоліків. Наприклад, вони вимагають дискретних маркерів як вхідних даних, що означає, що їм потрібен токенізатор або система розпізнавання мовлення для перетворення необроблених аудіосигналів у текст, вносячи помилки та шум у процес, що може негативно вплинути на продуктивність [43]. Крім того, ці моделі попередньо навчені на великих текстових корпусах, які можуть не відповідати домену чи стилю мовних даних, що призводить до проблем з невідповідністю домену.

wav2vec

Wav2vec використовує підхід до самоконтрольованого навчання, який використовує функцію втрати контрастного прогнозного кодування для вивчення репрезентації мовлення без необхідності транскрипції чи сегментації [44], [45]. Wav2vec використовує мультишарові згорткові шари нейронної мережі для кодування аудіо і далі маскує відрізки отриманого мовлення, так само як і масковане моделювання мови. Далі, отриманні приховані уявлення передаються в мережу трансформера для побудування контекстових репрезентацій і модель будується на задачі визначення контрасту, де прихована істина визначається завдяки дистракторам [45].

Цей підхід дозволяє wav2vec досягти найсучаснішої продуктивності в кількох завданнях обробки мовлення, включаючи розпізнавання мовлення, розпізнавання мовця та розуміння усної мови [44, 45].

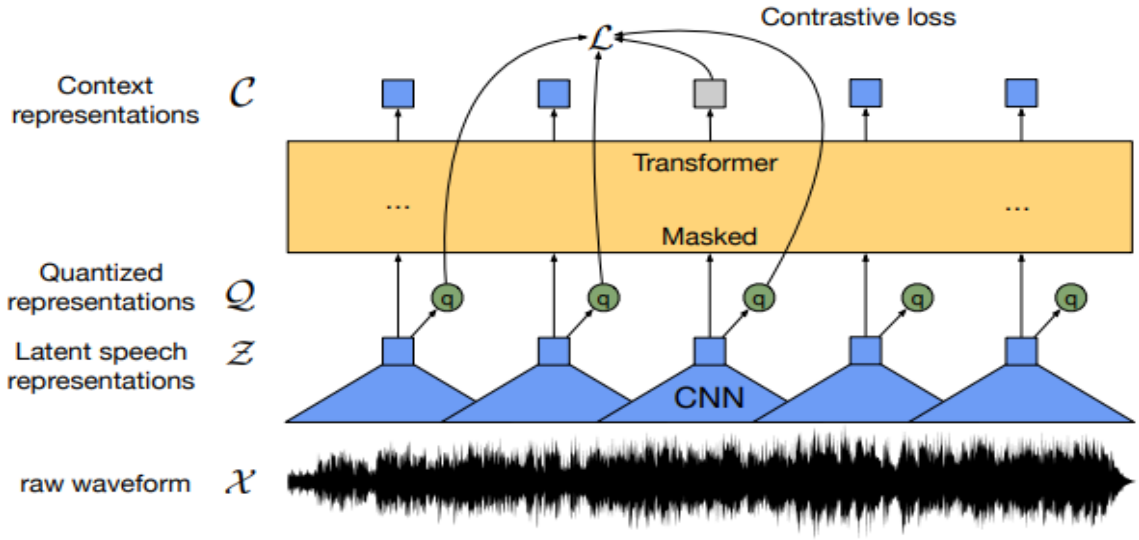


Рис. 2.2. Приклад роботи моделі Wav2vec [45]

w2v-BERT — це структура, яка поєднує контрастне навчання та MLM для попереднього навчання мовленню під самоконтролем і спирається на успіх wav2vec. w2v-BERT складається з трьох модулів: кодера функцій, модуля квантування та модуля замаскованого передбачення. Функціональний кодувальник подібний до wav2vec, але модуль квантування дискретизує безперервні представлення мови в кінцевий набір одиниць мови за допомогою Gumbel-softmax. Основні відмінності між wav2vec і w2v-BERT полягають у їхніх цілях попереднього навчання та типах даних, з якими вони працюють. wav2vec зосереджується на необроблених аудіосигналах і використовує контрастне навчання для вивчення представлень мови. Навпаки, w2v-BERT працює з окремими токенами та використовує як MLM, так і контрастне навчання як цілі попереднього навчання.

data2vec

data2vec спрямований на вивчення мультимодальних представлень даних, включаючи мову, зображення та текст, використовуючи порівняльну навчальну мету [46]. Подібно до wav2vec, data2vec використовує підхід до самостійного навчання, який не потребує міток чи анотацій і вивчає представлення шляхом максимальної узгодженості між різними доповненими представленнями однієї вибірки даних. Однак, на відміну від wav2vec, який зосереджується виключно на

мовних сигналах, data2vec може працювати з різними типами даних і вивчати спільні представлення, які фіксують кросмодальні кореляції та передають знання між модальностями. Підхід Data2vec до навчання з самоконтролем дозволяє вивчати представлення без потреби в позначених даних, що робить його масштабованим і економічно ефективним рішенням для багатьох програм.

Whisper

Whisper [47] — це модель загального призначення, призначена для розпізнавання мовлення в умовах шуму або з низьким ресурсом, і здатна виконувати кілька завдань, пов’язаних з мовленням. Whisper використовує слабкий контроль і мінімалістичний підхід до попередньої обробки даних. Він досягає найсучасніших результатів, демонструючи потенціал використання передових методів машинного навчання в обробці мовлення. Whisper здатний виконувати багатомовне розпізнавання мовлення, переклад мовлення та ідентифікацію мови. Він навчений на великому наборі різноманітних аудіоданих і є багатозадачною моделлю, яка може виконувати різноманітні завдання, пов’язані з мовленням, як-от транскрипція, голосові помічники, навчання, розваги та доступність. Модель Whisper унікальна тим, що використовує мінімалістичний підхід до попередньої обробки даних, що дозволяє моделям передбачати необроблений текст стенограм без значної стандартизації. Це усуває потребу в окремому етапі інверсної нормалізації тексту для створення натуралістичних транскрипцій, спрощуючи конвеєр розпізнавання мовлення. Отримані моделі можна добре узагальнити для стандартних тестів і є конкурентоспроможними з попередніми повністю контрольованими результатами без тонкого налаштування.

Tacotron

Tacotron [11] використовує архітектуру послідовності до послідовності з механізмами уваги для створення високоякісної мови з введеного тексту. Обмеження алгоритму Гріффіна-Ліма, який використовується для генерації аудіосигналу, призвели до розробки Tacotron 2 [10] компанією Google AI у 2018 році, який використовував WaveNet для генерації необроблених звукових

сигналів безпосередньо з мел-спектрограм, що призвело до більш природного звучання мови. Крім того, у 2019 році Microsoft представила Transformer TTS [13], у якому використовується трансформерна мережа замість згорткових і рекурентних мереж, що використовуються в Tacotron 2, разом із предиктором тривалості та новим методом навчання, який використовує примусове вчителя для швидшої конвергенції та кращої продуктивності.

Класична модель Tacotron2 має в собі двонаправлений шар RNN в якості кодувальника. У покращеній версії він заміняється одним трансформером.

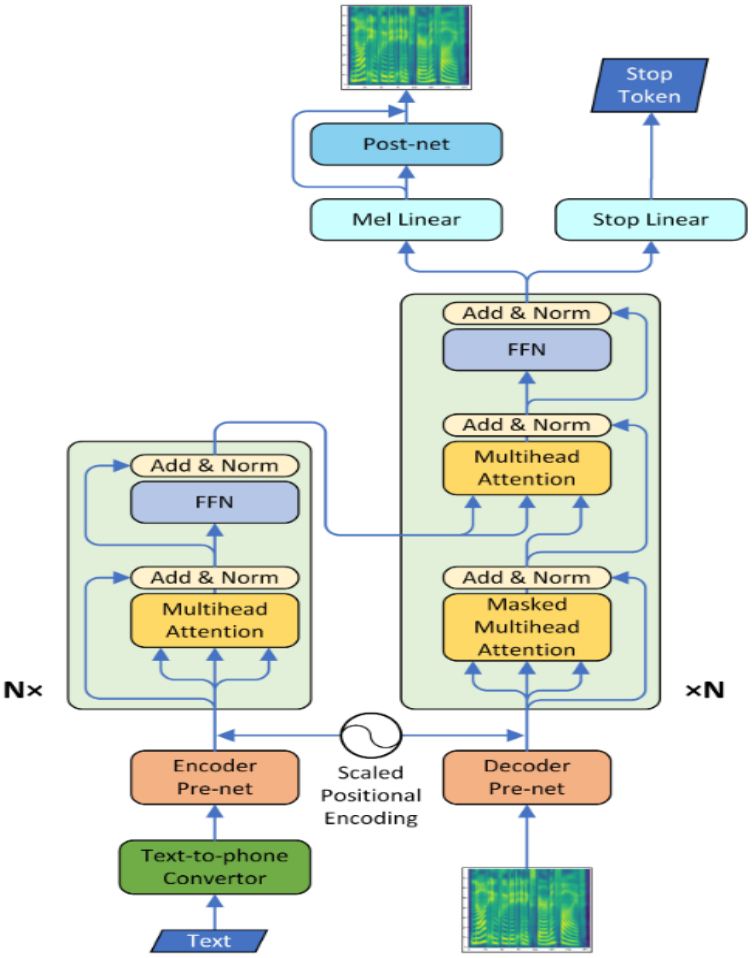


Рис. 2.3. Приклад моделі Tacotron2, яка базується на трансформерах [13]

У порівнянні з двонаправленим шаром RNN - фокус багатосторонньої уваги розділяється на два простора, таким чином що кодувальник отримує змогу інтерпретувати окремі відношення у різних аспектах та будувати зв’язок між двома часовими рамками завдяки глобального контексту усієї послідовності. Це

важливо для згенерованої просодії, бо таким чином, особливо у випадках, коли згенеровані речення є довгими - звук здається більш однорідним та природним. До того ж використання багатосторонньої уваги в порівнянні з оригінальним рішенням може дати приріст в паралельній обробці, що напряму впливає на час, що модель має витратити на навчання.

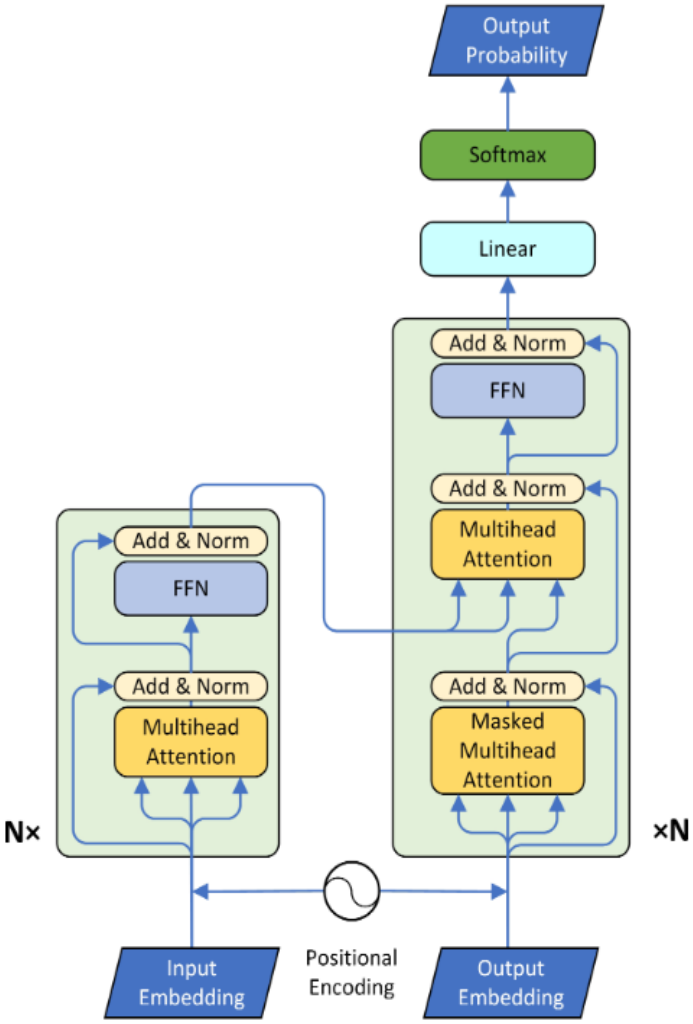


Рис. 2.4. Архітектура трансформера, який використовується в покращеній моделі Tacotron2 [13]

Можна виділити дві головні переваги у використанні трансформера над звичайними двома шарами RNN:

- додавання механізму самоуваги;

- використання багатосторонньої уваги, замість уваги, яка б орієнтувалась на розташуванні окремих звуків.

Механізм багатосторонньої уваги може розглядати сховані стани кодувальника з різних точок зору і таким чином генерувати кращі контекстуальні вектори [13].

Незважаючи на ці досягнення, поточні системи все ще мають обмеження у створенні природного звучання мовлення для неанглійських мов, обробки складних інтонацій і акцентів, а також синтезу мовлення в реальному часі для таких програм, як голосові помічники та автоматизовані телефонні системи.

VALL-E

VALL-E [48] — це система синтезу тексту в мову з нульовим ударом, яка використовує підхід моделювання мови, розглядаючи TTS як завдання моделювання умовної мови, а не безперервний сигнал регресії. Він навчений за допомогою окремих кодів із стандартної моделі нейронного аудіокодека та попередньо навчена на 60 000 годин даних мовлення англійською мовою, що забезпечує потужні можливості навчання в контексті. На відміну від попередніх систем TTS, VALL-E не вимагає додаткового проектування конструкції, попередньо розроблених акустичних функцій або тонкого налаштування. Вона може синтезувати високоякісну персоналізовану мову лише за допомогою 3-секундної звукової підказки від невидимого динаміка. Модель також забезпечує різноманітні виходи з однаковим вхідним текстом і може зберегти акустичне середовище та емоції мовця від акустичного підказки. Розмір динаміків VALLE побудований на узагальненій системі TTS, яка використовує велику кількість напівконтрольованих даних. Цей підхід важливий, оскільки масштабування напівконтрольованих даних було недооцінено для TTS. Результати оцінки показують, що VALL-E значно перевершує найсучаснішу систему TTS з нульовим ударом на наборах даних LibriSpeech і VCTK з точки зору природності мови та схожості мовця.

VALL-E X [49] було розроблено як природне розширення VALL-E для вирішення проблеми міжмовного синтезу мовлення. Міжмовний синтез

мовлення передбачає генерування мовлення цільовою мовою за допомогою мовної підказки вихідної мови та текстової підказки цільової мови. Хоча VALL-E був розроблений для нульового TTS англійською мовою, існувала потреба в моделі, яка могла б обробляти міжмовний синтез мовлення кількома мовами. Таким чином, VALL-E X розширює VALL-E для підтримки міжмовного синтезу, навчаючи багатомовну модель передбачати послідовності акустичних токенів цільовою мовою, використовуючи підказки як вихідною, так і цільовою мовами. Це дозволяє моделі генерувати високоякісне мовлення цільовою мовою, зберігаючи голос, емоції та акустичне середовище невидимого мовця та ефективно усуваючи проблему іноземного акценту, яку можна контролювати за допомогою ідентифікатора мови.

Conformer

Останні досягнення в трансформерах для обробки мови також призвели до появи моделей Conformer [50]. Архітектура Conformer поєднує в собі згортковий і трансформерний рівні, що дозволяє отримувати як локальну, так і глобальну контекстну інформацію. Це робить моделі Conformer добре придатними для завдань обробки мовлення, таких як розпізнавання мовлення та ідентифікація мовця, де фіксація довгострокових залежностей має вирішальне значення.

Кодувальником Conformer також є згортковий шар, за якими йде низка блоків Conformer. Приклад кодувальника даної моделі наведений на рис. 2.4. [50]

Як можна побачити з рисунка, спочатку йде лінійний шар, після якого слідує шар з dropout (що доданий для протидії перенавчанню даної моделі), після якого - сам Conformer Block, який містить всередині 5 блоків, які йдуть один за одним: модуль передачі, модуль багатосторонньої самоуваги, згортковий модуль, модуль передачі, модуль нормалізації значень.

Завдяки відносному позиційному кодуванню, механізм самоуваги має змогу краще навчатись на різних розмірах вхідних даних, і в результаті він буде більш стійкий до різних можливих варіантів довжин висловлювань.

Модуль багатосторонньої самоуваги складається з шару нормалізації, шару, який використовує механізм уваги з відносним позиціюванням, та шару dropout.

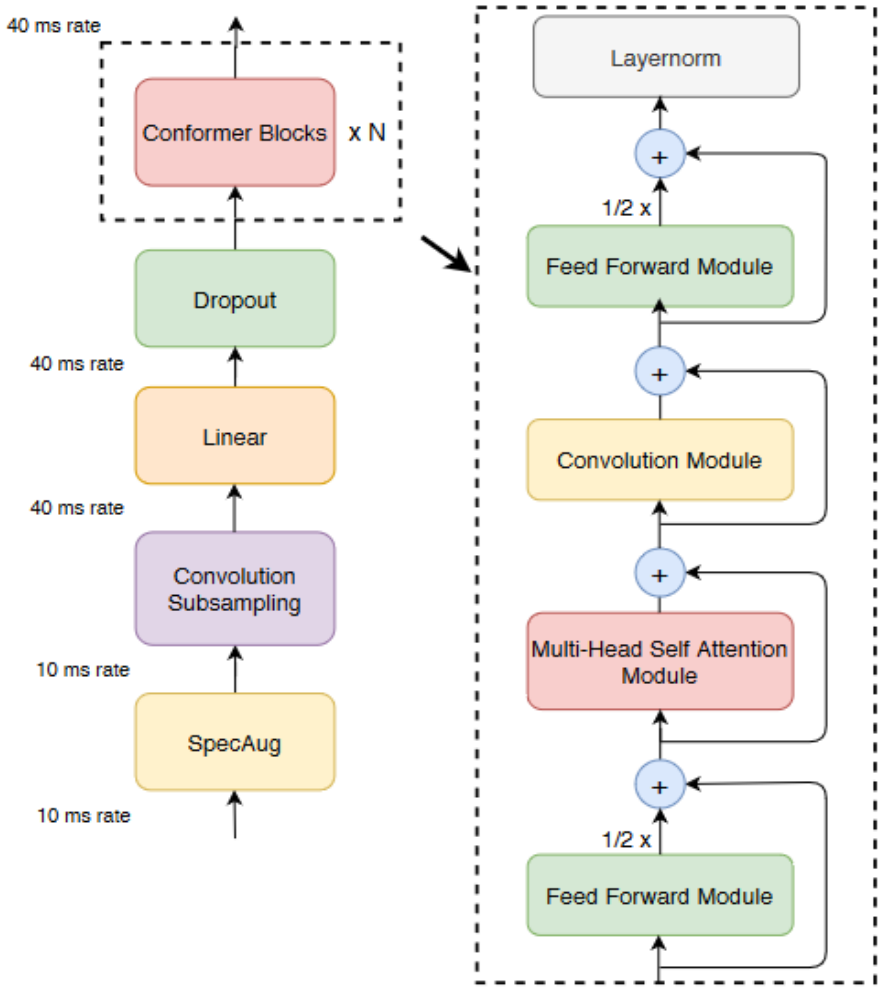


Рис. 2.5. Приклад архітектури моделі Conformer [50]

Conformer досяг найсучаснішої продуктивності в таких тестах, як LibriSpeech та AISHELL-1. Однак попередні обмеження в синтезі та розпізнаванні мовлення, такі, як боротьба за створення природного мовлення мовами, відмінними від англійської, і створення мовлення в режимі реального часу, залишилися. У 2021 році Ван та ін. [51] розширили Conformer і розробили Conformer-LHUC, який використовував внесок прихованої одиниці навчання (LHUC) для адаптації динаміків. Conformer-LHUC продемонстрував чудову ефективність у розпізнаванні мовлення людей похилого віку та має потенційні наслідки для клінічної діагностики та лікування хвороби Альцгеймера.

UniSpeech

Підхід Microsoft UniSpeech [52] пропонує уніфікований метод попереднього навчання, який поєднує контрольоване та неконтрольоване навчання для навчання репрезентації мовлення. У цьому підході використовується фонетичне навчання CTC і контрастне самоконтрольоване навчання з усвідомленням фонетики, щоб отримати більше фонетичної інформації та краще узагальнювати між мовами та доменами. У статті [53] автори пропонується UniSpeechSAT, універсальний метод навчання репрезентації мовлення з попереднім навчанням мовця. Метод покращує існуюче самоконтрольоване навчання для навчання репрезентації мовця за допомогою контрастного навчання щодо висловлювання та доповнення змішування висловлювань. Цей метод досягає найсучаснішої продуктивності в навчанні універсального представлення, особливо для завдань ідентифікації мовця, і може бути легко адаптований до подальших завдань з мінімальним тонким налаштуванням.

Speechformer

У червні 2021 року було запропоновано Speechformer [54], самоконтрольовану попередньо навчену модель для наскрізного розпізнавання мовлення, яка використовує замасковане акустичне моделювання та контрастне прогнозне кодування. На відміну від попередніх моделей, які використовували або згорточні, або рекурентні нейронні мережі, Speechformer використовує трансформерну архітектуру кодера-декодера з кодуванням відносного положення та нормалізацією шару. Він попередньо навчений на 53 тисячах годин немаркованих мовних даних і досягає конкурентоспроможних результатів у кількох тестах ASR.

WavLM

Microsoft Research Asia випустила WavLM [55], широкомасштабну попередньо навчену модель із самоконтролем, яка може вирішувати повний стек мовленнєвих завдань, таких як ASR, TTS і перевірка мовця. WavLM спільно вивчає передбачення замаскованого мовлення та усунення шумів під час

попереднього навчання та використовує стробований зміщення відносного положення для структури трансформера, щоб краще фіксувати порядок послідовності вхідного мовлення. Він навчається на величезному наборі даних про 94 тисячі годин мовлення та досягає найсучасніших результатів у кількох подальших мовних завданнях для 10 мов.

В таблиці 2.1 наведені узагальнені результати аналізу мовних моделей, що використовують трансформер та розв'язують проблему TTS, ST або ASR.

Таблиця 2.1.

Моделі мовлення на основі трансформера [35].

Назва моделі	Рік випуску	Число параметрів	Задачі			Мульти-модальна
			Синтез мови (TTS)	Переклад мови (ST)	Автоматичне розпізнавання мови (ASR)	
Tacotron	2017	13 млн	✓	×	×	×
Tacotron 2	2017	28.2 млн	✓	×	×	×
Transformer-TTS	2018	30.7 млн	✓	×	×	×
vq-wav2vec	2019	34 млн	×	×	✓	×
Mockingjay	2019	85 млн	×	×	✓	×
FastSpeech	2019	23 млн	✓	×	×	×
wav2vec	2019	16 млн	×	×	✓	×
wav2vec 2.0	2020	317 млн	×	×	✓	×
FastSpeech 2	2020	27 млн	✓	×	×	×
FastPitch	2020	26.8 млн	✓	×	×	×
Conformer	2020	1 блн	×	✓	✓	×
DeCoAR 2.0	2020	317 млн	×	×	✓	×
w2v-Conformer	2021	1 блн	×	×	✓	×
w2v-BERT	2021	1 блн	×	×	✓	×
HuBERT	2021	317 млн	×	×	✓	×
XLS-R	2021	2 блн	×	✓	✓	×
UniSpeech	2021	317 млн	×	×	✓	×
UniSpeech-SAT	2021	317 млн	×	×	✓	×
BigSSL	2021	8 блн	×	×	✓	×
WavLM	2021	317 млн	×	×	✓	×
DeltaLM	2021	360 млн	✓	✓	×	✓
SpeechT5	2021	11 блн	✓	✓	✓	✓
data2vec	2022	*	×	×	✓	✓
data2vec 2.0	2022	*	×	×	✓	✓
SpeechFormer	2022	3.5 млн	×	×	✓	✓
Whisper	2022	1.6 блн	×	✓	✓	×
VALL-E	2023	*	✓	×	×	×
VALL-E X	2023	*	✓	×	×	×

2.5. Архітектура та принцип роботи моделі VALL-E

2.5.1. Розгляд TTS як моделювання мови умовного кодеку

Маючи набір даних $D = \{x_i, y_i\}$, де y — вибірка аудіо, а $x = \{x_0, x_1, ..., x_L\}$ — відповідна транскрипція фонеми, попередньо навчена модель нейронного кодека буде використана для кодування кожного аудіо вибірку в дискретні акустичні коди, що позначаються як $Encodec(y) = C^{T\times 8}$, де C представляє двовимірну матрицю акустичного коду, а T є довжиною вислову з пониженою дискретизацією. Вектор-рядок кожної матриці акустичного коду $c_{t,:}$ представляє вісім кодів для кадру t, а вектор-стовпець кожної матриці акустичного коду $c_{:,j}$ представляє послідовність кодів з j-ї кодової книги, де $j \in \{1,.. .,8\}$. Після квантування нейронний кодек-декодер здатний реконструювати сигнал, позначений як $Decodec(C) \approx \hat{y}$.

Zero-shot TTS [83] вимагає, щоб модель синтезувала високоякісну мову для невидимих динаміків. У кваліфікаційній роботі буде розглянуто нульовий TTS як умовне завдання моделювання мови кодека. Модель нейронної мови буде навчено генерувати акустичну кодову матрицю C, обумовлену послідовністю фонеми x і акустичною матрицею підказок $\tilde{C}^{\hat{T}\times 8}$ з метою оптимізації $maxp(\mathbf{C}|x,\tilde{\mathbf{C}})$. Тут $\tilde{\mathbf{C}}$ отримано тим самим нейронним кодеком із зареєстрованим записом як вхід. Очікується, що модель нейронної мови навчиться витягувати вміст і інформацію про мовця з послідовності фонем і акустичного підказки відповідно. Під час висновку, враховуючи послідовність фонем і 3-секундний зареєстрований запис невидимого мовця, матриця акустичного коду з відповідним вмістом і голосом мовця спочатку оцінюється навченою мовною моделлю. Потім нейронний кодек-декодер синтезує високоякісну мову.

2.5.2. Моделювання мови умовного кодеку

Модель нейронного мовного кодека дозволяє працювати з дискретними звуковими представленнями. Через залишкове квантування в моделі нейронного

кодека токени мають ієрархічну структуру (рис.2.6): токени з попередніх квантувачів відновлюють акустичні властивості, як-от ідентичність мовця, тоді як послідовні квантувачі вивчають тонкі акустичні деталі. Кожен квантувач навчений моделювати залишки від попередніх квантувачів. Виходячі з цього, проектуються дві моделі умовної мови в ієрархічній манері.

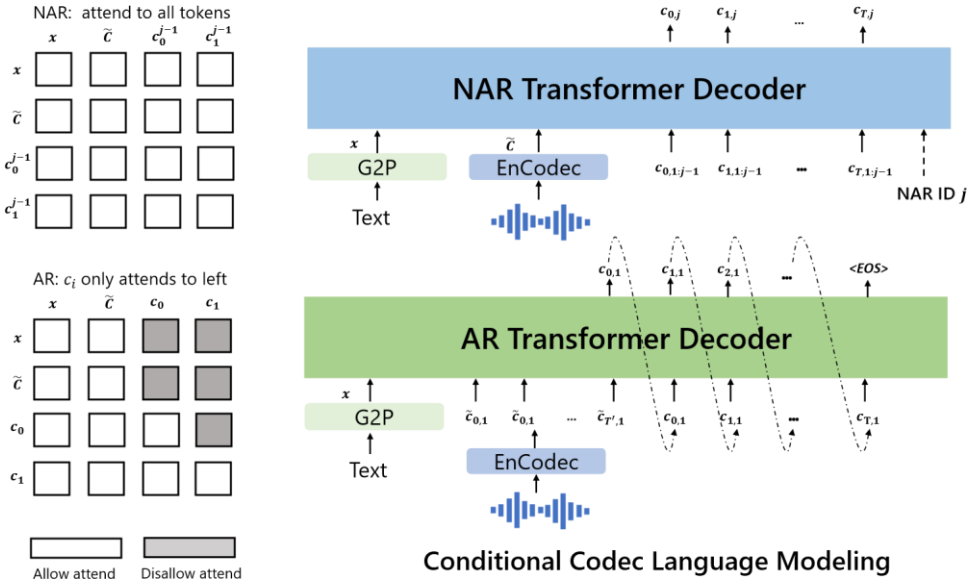


Рис. 2.6. Структура моделювання мови умовного кодека, яка побудована в ієрархічній формі.

Для дискретних токенів з першого квантувача $c_{:,1}$ навчається лише авторегресійна (AR) мовна модель декодера. Він обумовлений послідовністю фонем x і акустичною підказкою $C\tilde{}_{:,1}$, сформульованою як

$$p(\mathbf{c}_{:,1}|\mathbf{x}, \tilde{\mathbf{C}}_{:,1}; \theta_{AR}) = \prod_{t=0}^{T} p(\mathbf{c}_{t,1}|\mathbf{c}_{<t,1}, \tilde{\mathbf{c}}_{:,1}, \mathbf{x}; \theta_{AR}) \tag{2.7}$$

Оскільки VALL-E є лише декодером LM, конкатенація $\tilde{c}_{:,1}$ і $c_{:,1}$ є цілою послідовністю, і вони не розрізняються і не відмічаються певними маркерами під час навчання. Передбачається лише $c_{:,1}$, тоді як префікс $\tilde{c}_{:,1}$ дається під час висновку.

Для дискретних токенів від другого до останнього квантувача, $c_{:,j\in[2,8]}$, навчається неавторегресійна (NAR) модель мови. Оскільки маркери не можуть отримати доступ один до одного за допомогою NAR, для обмеження

ідентичності мовця в якості акустичного підказки використовується матриця $C^{\sim}$ акустичних підказок. Таким чином, модель є залежною від послідовності фонем *x*, акустична підказка $C^{\sim}$ і передбачені акустичні токени належать до попередніх кодових книг $C_{:,<j}$:

$$p(\mathbf{C}_{:,2:8}|\mathbf{x}, \tilde{\mathbf{C}}; \theta_{NAR}) = \prod_{j=2}^{8} p(\mathbf{c}_{:,j}|\mathbf{C}_{:,<j}, \mathbf{x}, \tilde{\mathbf{C}}; \theta_{NAR}) \tag{2.8}$$

Поєднання моделі AR і моделі NAR забезпечує хороший компроміс між якістю мовлення та швидкістю логічного висновку. З одного боку, швидкість згенерованого мовлення має відповідати зареєстрованому запису, і важко навчити передісторію довжини для різних ораторів, оскільки їх швидкість мовлення може бути дуже різною. У цьому випадку модель AR є більш природним вибором із її гнучкістю для передбачення довжини акустичної послідовності. З іншого боку, для послідовних етапів, оскільки кількість вихідних слотів відповідає довжині послідовності першого етапу, NAR може зменшити часову складність від *O(T)* до *O(1)*. Загалом, прогноз *C* можна змоделювати як:

$$p(\mathbf{C}|\mathbf{x}, \tilde{\mathbf{C}}; \theta) = p(\mathbf{c}_{:,1}|\tilde{\mathbf{C}}_{:,1}, \mathbf{X}; \theta_{AR}) \prod_{j=2}^{8} p(\mathbf{c}_{:,\mathbf{j}}|\mathbf{c}_{:,<j}, \mathbf{x}, \tilde{\mathbf{C}}; \theta_{NAR}) \tag{2.9}$$

2.5.3. Авторегресійне моделювання мови кодеків

Модель авторегресійної мови генерує лексеми з першого квантувача. Він містить вбудовування фонеми W_x, акустичне вбудовування W_a, декодер трансформатора та рівень передбачення. Щоб створити мову з певним вмістом, використовується послідовність фонем як підказка фонеми мовної моделі. Таким чином, вхідними даними моделі є конкатенація *x* і $c_{:,1}$, а після кожного з них додається два спеціальних маркера <EOS>. Обчислюється вбудовування звивистої позиції окремо для підказок і вхідних токенів. Для моделі причинного

трансформера кожен маркер $c_{t,1}$ може відповідати $(x,c_{\leq t,1})$, як показано в лівій частині рисунка 2.4. Модель оптимізовано для максимізації ймовірності наступного маркера в першій кодовій книзі. Параметри вихідного проекційного шару поділяються з параметрами акустичного вбудовування W_a.

У моделі AR аудіозапис явно не витягується як підказка під час навчання. Навчальний процес — це звичайне навчання моделі мови. Таким чином, будь-яка послідовність префікса $c_{<t,1}$ розглядається як підказка для останньої частини послідовності $c_{\geq t,1}$. Під час висновку, враховуючи зареєстрований запис, потрібно повинні об'єднати послідовність фонем зареєстрованого запису та послідовність фонем для синтезу. Тим часом послідовність акустичних маркерів зареєстрованого запису використовується як префікс у декодуванні AR, як сформульовано в рівнянні (2.7). Перевага цього налаштування буде досліджена в розділі 3.

2.5.4. Моделювання мови кодеків без авторегресії

Коли будуть отримані перші коди квантователя за допомогою моделі AR, буде використано неавторегресійну (NAR) модель для генерації кодів інших семи квантувачів. Модель NAR має подібну архітектуру до моделі AR, за винятком того, що вона містить вісім окремих шарів акустичного вбудовування. На кожному кроці навчання випадково вибирається етап навчання $i \in [2,8]$. Модель навчена максимізувати акустичні маркери з i-ї кодової книги квантователя. Акустичні токени від стадії 1 до стадії $i-1$ вбудовуються та підсумовуються як вхідні дані моделі:

$$e_{c_{t,j}} = W_a^j \odot c_{t,j} \tag{2.10}$$

$$\mathbf{e_{c_t}} = \sum_{j=1}^{i-1} e_{c_t,j} \tag{2.11}$$

де ○ позначає вибір індексу. Послідовність фонем також розглядається як підказка мовної моделі. Крім того, щоб клонувати унікальний голос даного оратора, також використовуються акустичні токени із зареєстрованої промови як акустична підказка. Зокрема, спочатку маркується зареєстроване мовлення за допомогою моделі нейронного кодека як $C^{\sim T\times 8}$. Вбудовані представлення з усіх восьми кодових книг підсумовуються як акустична підказка

$$\mathbf{e}_{\tilde{\mathbf{c}}_\mathbf{t}} = \sum_{j=1}^{8} e_{\tilde{c}_{t,j}} \quad (2.12)$$

Щоб передбачити акустичні токени з *i*-ї кодової книги, вхід трансформатора є конкатенацією $(e_x, e_{\tilde{c}}, e_{c:,<i})$. Позиційні вбудовування також обчислюються окремо для підказок і акустичної послідовності. Поточний етап і вводиться в мережу за допомогою оператора адаптивної нормалізації рівня [84], тобто $\mathrm{AdaLN}(h,i) = a_i \mathrm{LayerNorm}(h) + b_i$, де h — проміжні активації, a_i і b_i — отримані з лінійної проекції вбудовування сцени. На відміну від AR, модель NAR дозволяє кожному маркеру звертати увагу на всі вхідні маркери на рівні самоконтролю. Також розділяються параметри рівня акустичного вбудовування та вихідного рівня передбачення, що означає, що ваги j-го рівня передбачення такі ж, як і $(j + 1)$-го шару акустичного вбудовування.

2.6. Висновки

У другому розділі були розглянуті приклади використання інструментів, обраних у першому розділі, та прийняте остаточне рішення щодо особливостей реалізації обраної моделі синтезу мовлення.

Було визначено, що ключова інновація трансформерів полягає в їхній здатності фіксувати далекі залежності між вхідними послідовностями за допомогою рівнів самоконтролю, і її ефективність була продемонстрована в різних завданнях обробки мовлення, таких як автоматичне розпізнавання мовлення, синтез мовлення з тексту, розпізнавання мовця, мульти -обробка мікрофона та переклад мови.

Трансформери використовують рівні самоконтролю для захоплення довгострокових залежностей між вхідними послідовностями, що дозволяє їм складати конкурентоспроможну альтернативу моделям на основі рекурентної нейронної мережі в завданнях автоматичного розпізнавання мовлення і синтезу мовлення.

РОЗДІЛ 3

РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ МОДЕЛІ СИНТЕЗУ МОВЛЕННЯ НА ОСНОВІ ТРАНСФОРМЕРА

3.1. Обґрунтування вхідних даних

Спираючись на результати аналізу мовних моделей, що використовують трансформер та розв'язують проблему TTS, ST або ASR, наведені у розділі 2, у якості моделі для навчання було обрано модель VALL-E, через те, що дана модель є найновішою з розглянутих і найперспективнішою для майбутнього покращення. Даний тип системи мовленнєвого синтезу виконує задачі умовного моделювання, на відміну від потокової регресії аудіо сигналу. Оригінальна модель навчалась за допомогою дискретних кодів з аудіо кодеку та пре тренована 60 тисячами годинами англійського мовлення, які зберегли сильне контекстуальне значення. На відміну від попередніх систем TTS, VALL-E не вимагає додаткову інженерну конструкцію, попередньо спроєктованих акустичних характеристик, або тонких налаштувань. Більш того, завдяки трансформеру, модель може генерувати різні вихідні комбінації з однієї вхідної послідовності та створювати емоційне забарвлення.

Оскільки аудіо зазвичай зберігається як послідовність 16-бітних цілих значень, для синтезу необробленого аудіо потрібна генеративна модель для виведення 2 16 = 65 536 ймовірностей на крок у часі. Крім того, частота дискретизації аудіо, що перевищує десять тисяч, призводить до надзвичайно великої довжини послідовності, що робить її більш складною для синтезу необробленого аудіо. З цією метою потрібне квантування мови для стиснення цілих значень і довжини послідовності. Перетворення μ-закону може квантувати кожен часовий крок до 256 значень і реконструювати високоякісний необроблений звук. Він широко використовується в моделях генерування мови,

таких як WaveNet [9], але швидкість логічного висновку все ще низька, оскільки довжина послідовності не зменшується.

Останнім часом векторне квантування широко застосовується в моделях самоконтрольованого мовлення для виділення ознак, таких як vq-wav2vec [44] та HuBERT [85]. Наступна робота [86, 87] показує, що коди з самоконтрольованих моделей також можуть реконструювати вміст, а швидкість висновку є вищою, ніж у WaveNet. Однак ідентичність мовця була відкинута, а якість реконструкції низька [88]. AudioLM [88] навчає мовні моделі синтезу мовлення як на токенах k-means із самоконтрольованої моделі, так і на акустичних токенах із моделі нейронного кодека, що призводить до високоякісного генерування мовлення в мовлення.

У даній кваліфікаційній роботі наслідується AudioLM [88], щоб використовувати моделі нейронних кодеків для представлення мови в дискретних токенах. Щоб стиснути аудіо для мережевої передачі, моделі кодеків здатні кодувати форму хвилі в дискретні акустичні коди та реконструювати високоякісну форму сигналу, навіть якщо мовця не видно під час навчання. Порівняно з підходами до традиційних аудіокодеків, нейронний кодек значно кращий при низьких бітрейтах, і вважається, що квантовані маркери містять достатньо інформації про динаміка та умови запису. Порівняно з іншими методами квантування, аудіокодек демонструє наступні переваги:

1) він містить велику кількість інформації про мовця та акустичну інформацію, яка може підтримувати ідентичність мовця під час реконструкції порівняно з кодами HuBERT [85];

2) є готовий кодек-декодер для перетворення дискретних токенів у форму сигналу без додаткових зусиль щодо навчання вокодера, як-от методи на основі VQ, які працюють зі спектром [87];

3) це може зменшити тривалість кроків за часом для ефективності вирішення проблеми перетворення μ-закону [9].

У якості маркера використовується попередньо навчена модель нейронного аудіокодека, EnCodec [89]. EnCodec — це модель згорткового

кодера-декодера, вхід і вихід якого є аудіо 24 кГц зі змінною швидкістю потоку. Кодер створює вбудовування на 75 Гц для вхідних сигналів на 24 кГц, що є 320-кратним зменшенням частоти дискретизації. Кожне вбудовування моделюється залишковим векторним квантуванням (RVQ), у якому вибирається вісім ієрархічних квантувачів із 1024 записами кожен. Ця конфігурація відповідає EnCodec із 6K бітрейтом для реконструкції звуку 24 кГц. У цьому параметрі, враховуючи 10-секундну форму хвилі, дискретне представлення є матрицею з 750 × 8 записами, де 750 = 24 000 × 10 320 – це часовий крок зі зниженою дискретизацією, а 8 – кількість квантувачів.

Також можливо вибрати інші налаштування бітрейту. Більший бітрейт відповідає більшій кількості квантизаторів і кращій якості реконструкції. Наприклад, якщо обрати EnCodecc із швидкістю 12 КБ, потрібно 16 квантувачів, а 10-секундна форма сигналу відповідає матриці з 750 × 16 записами. Завдяки дискретним кодам від усіх квантувачів згортковий декодер EnCodec генерує реальні вбудовування та реконструює сигнал на частоті 24 кГц.

3.2. Вхідні дані

Для навчання моделі був використаний набір даних LibriTTS [90]. Даний набір розміщений на порталі Google research та має модель відкритого доступу, з назвою ліцензії “CC BY 4.0”.

Корпус містить 585 годин матеріалу записаного англійською мовою з частотою дискретизації 24 кілогерц. Згідно з описом, набір даних був спеціально розроблений для навчання моделей TTS, матеріал якого був отриманий з іншого набору даних (LibriSpeech, який містить 1000 годин матеріалів), а саме з аудіо файлів LibriVox (платформа, на якій можна розташовувати публічні аудіокниги) та великої електронної бібліотеки Project Gutenberg.

Основними перевагами LibriTTS над LibriSpeech можна вважати:

- частоту дискретизації 24 кілогерц, на відміну від 16 кілогерців у LibriSpeech;
- відсутність записів з високим рівнем шуму;
- кожний запис набору представляє окреме речення.

Даний набір даних був опублікований у 2019 році і є дуже розповсюдженим в роботах по синтезу людського мовлення.

3.3. Опис моделі

Навчання в контексті є важливою здатністю текстової моделі мови, яка здатна передбачати мітки для невидимих вхідних даних без додаткових оновлень параметрів. Для TTS, якщо модель може синтезувати високоякісне мовлення для невидимих мовців без тонкого налаштування, вважається, що модель має можливість навчання в контексті. Однак здатність до навчання в контексті існуючих систем TTS є слабкою, оскільки вони або потребують додаткового тонкого налаштування, або різко погіршуються для невидимих мовців.

Для мовних моделей підказки необхідні, щоб увімкнути навчання в контексті в сценарії нульового удару. Підказки та висновки були розроблені наступним чином. Спочатку текст перетворюється у послідовність фонем і зареєстрований запис кодується в акустичну матрицю, утворюючи підказку фонеми та акустичну підказку. Обидва підказки використовуються в моделях AR і NAR.

Для моделі AR використовується декодування на основі вибірки, яке обумовлюється підказками, оскільки передбачалося, що пошук променя може привести LM до нескінченного циклу. Крім того, метод на основі вибірки може значно збільшити різноманітність результату.

Для моделі NAR використовується жадібне декодування, щоб вибрати токен із найвищою ймовірністю. Нарешті, декодер нейронного кодека використовується для генерації сигналу, який обумовлюється вісьмома кодовими послідовностями. Акустична підказка може або не може бути

семантично пов’язаною з мовою, яку потрібно синтезувати, що призводить до двох випадків:

1. Оскільки головна мета полягає в тому, щоб створювати певний контент для невидимих спікерів, моделі надається текстове речення, фрагмент записаного мовлення та його відповідна транскрипція. Фонема транскрипції зареєстрованого мовлення додається до послідовності фонем даного речення як підказка фонеми та використовується акустичний токен першого рівня зареєстрованого мовлення $c^{\sim}_{:,1}$ як акустичний префікс. За допомогою підказки фонеми та акустичного префікса VALL-E генерує акустичні токени для даного тексту, копіюючи голос цього мовця.

2. У налаштуванні VALL-E-continual використовується вся транскрипція та перші 3 секунди висловлювання як фонема та акустичні підказки відповідно, і модель повинна створити продовження. Процес висновку такий самий, як і налаштування VALL-E, за винятком того, що зареєстроване мовлення та згенероване мовлення семантично безперервні.

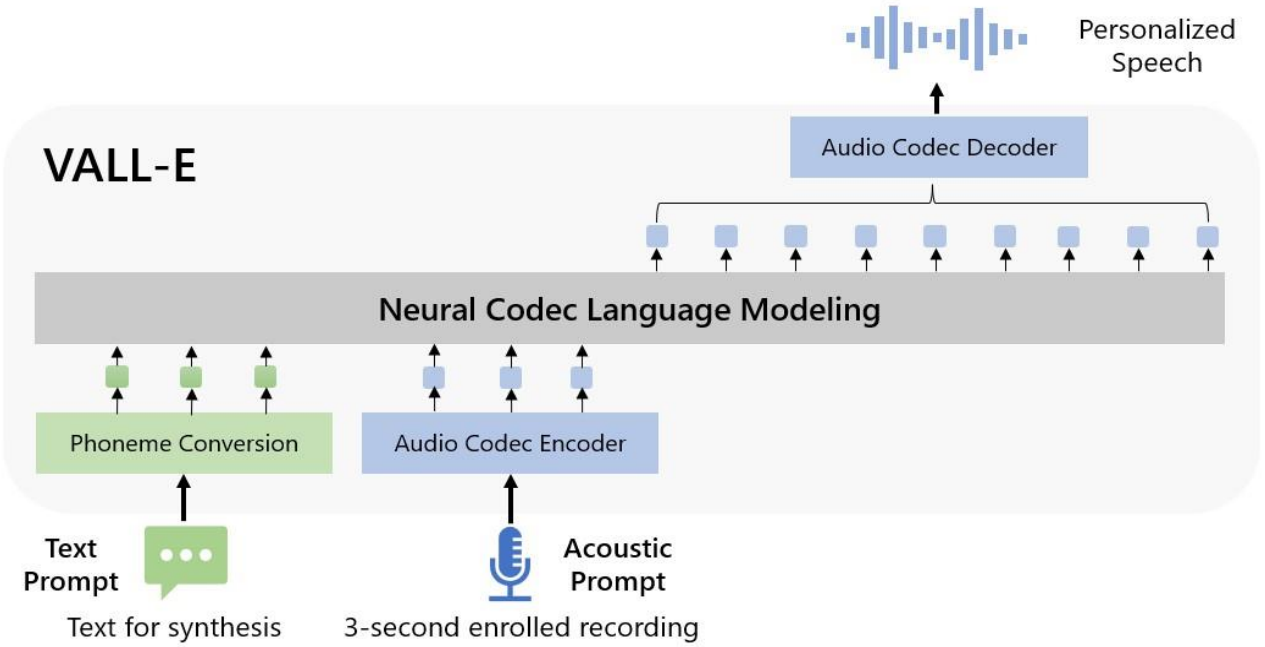


Рис. 3.1. Принцип роботи моделі VALL-E [91]

У випадку з VALL-E X принцип є досить схожим, але на відміну від звичайного VALL-E вхідною послідовністю VALL-E X є текст двома мовами та

ідентифікатор мови. Далі, після перетворень вхідних даних вони вже переходять до межи мовного мовленнєвого кодека, після якого декодується в мовленнєвий сигнал.

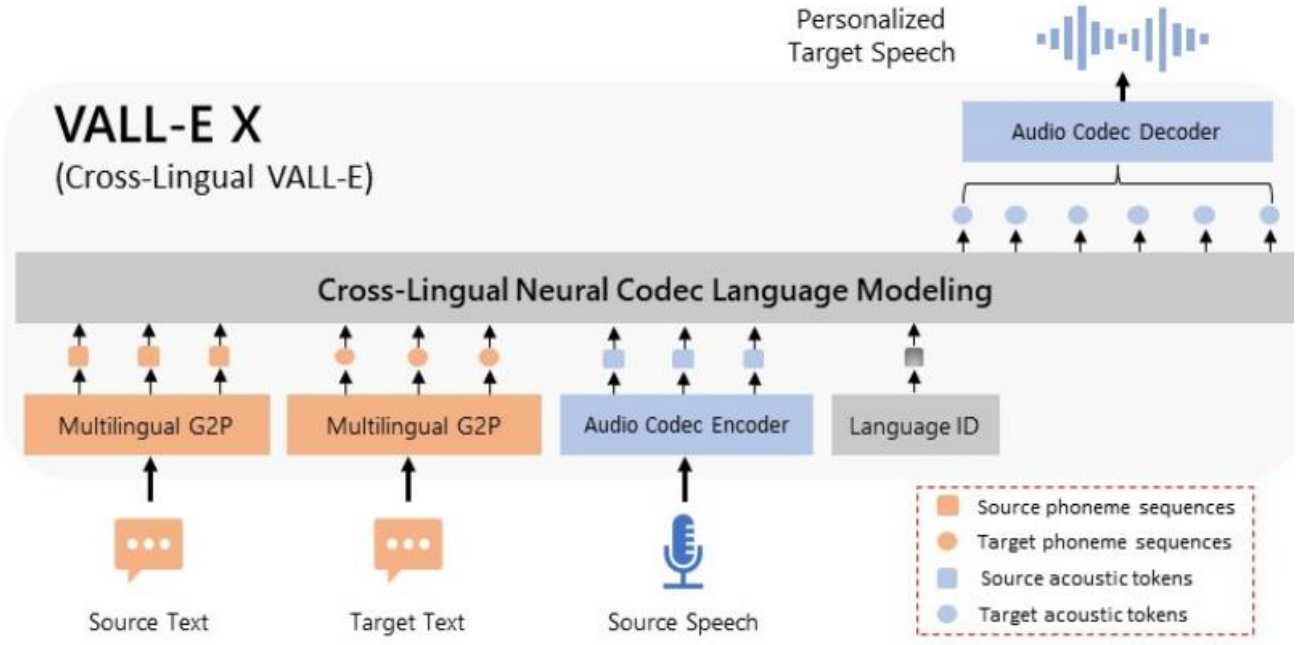


Рис. 3.2. Структура VALL-E X [92]

3.4. Дослідження моделі

Після навчання оригінальної моделі VALL-E - були проаналізовані результати та зроблено висновок, що модель має показники краще за найсучасніші моделі TTS при навчанні на однакових наборах даних (LibriSpeech та VCTK) та має більш природне мовлення. [35]

Таблиця 3.1.

Аналіз результатів моделі VALL-E

model	WER	SPK
GroundTruth	2.2	0.754
Speech-to-Speech Systems		
GSLM	12.4	0.126
AudioLM*	6.0	-
TTS Systems		
YourTTS	7.7	0.337
VALL-E	5.9	**0.580**
VALL-E-continual	**3.8**	0.508

Згідно з табл. 3.2, у якій подано аналіз результатів з опитування людей, які прослухали сгенероване мовлення, VALL-E показав найкращі результати у схожості з GroundTruth, бо відстав від нього усього на 0.12 SMOS, в той час як YourTTS відстав від VALL-E моделі на 0.93 SMOS або на 1.05 SMOS від GroundTruth.

Таблиця 3.2.

Аналіз результатів за прослуховуванням

	SMOS	CMOS (v.s. VALL-E)
YourTTS	$3.45_{\pm 0.09}$	-0.12
VALL-E	$4.38_{\pm 0.10}$	0.00
GroundTruth	$4.5_{\pm 0.10}$	+0.17

Нижче наведені графіки отриманих результатів для навченої моделі VALL-E.

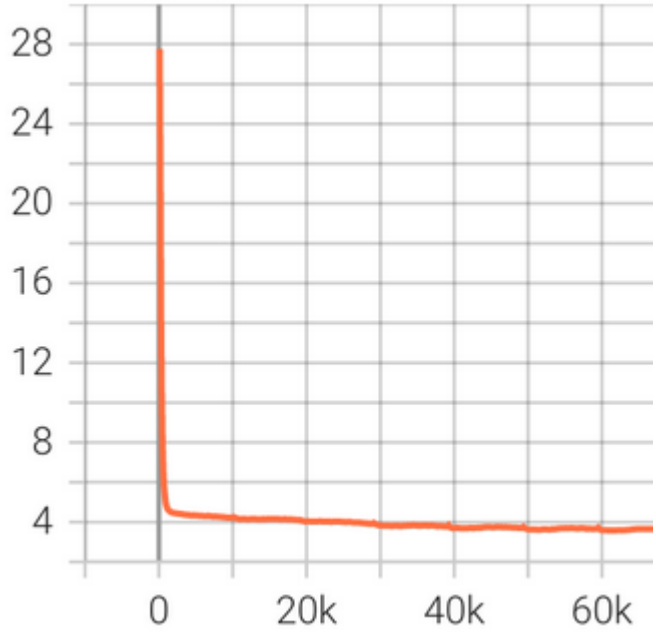


Рис. 3.3. Графік зміни загальної loss функції

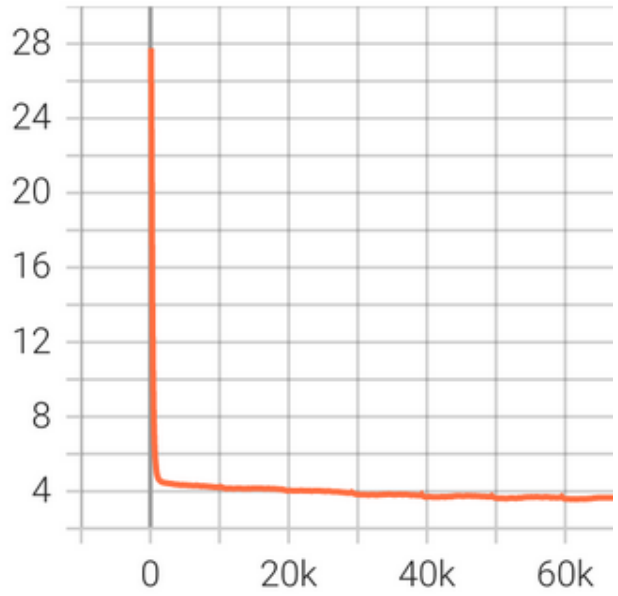


Рис. 3.4. Графік зміни загальної loss функції

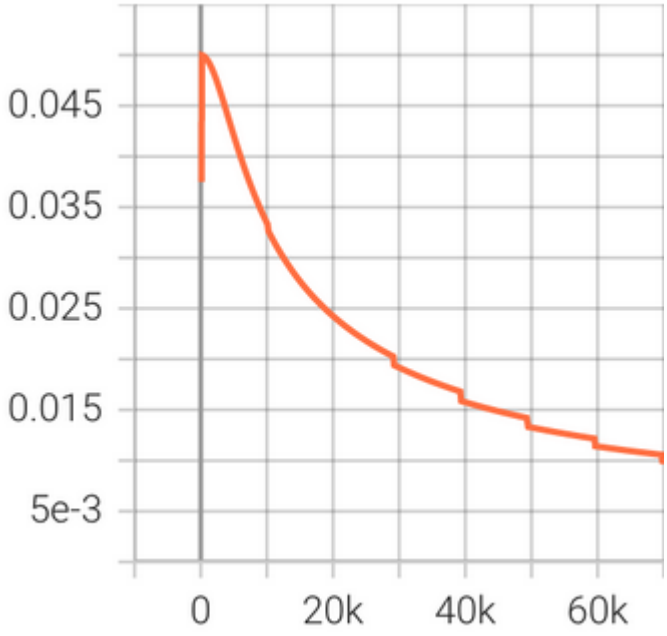
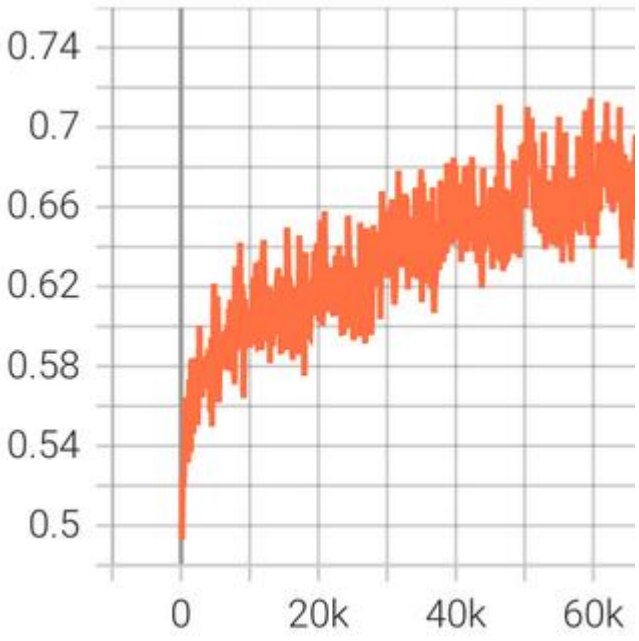
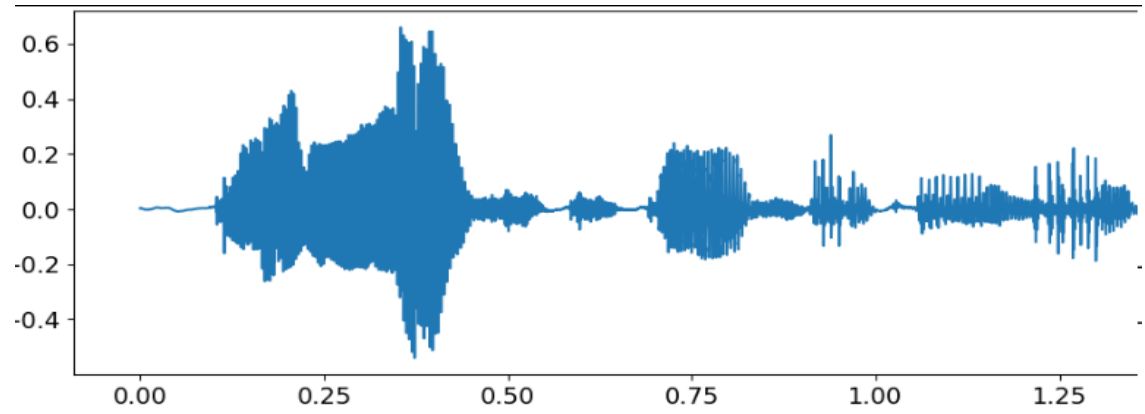


Рис. 3.5. Графік зміни динамічного значення швидкості навчання

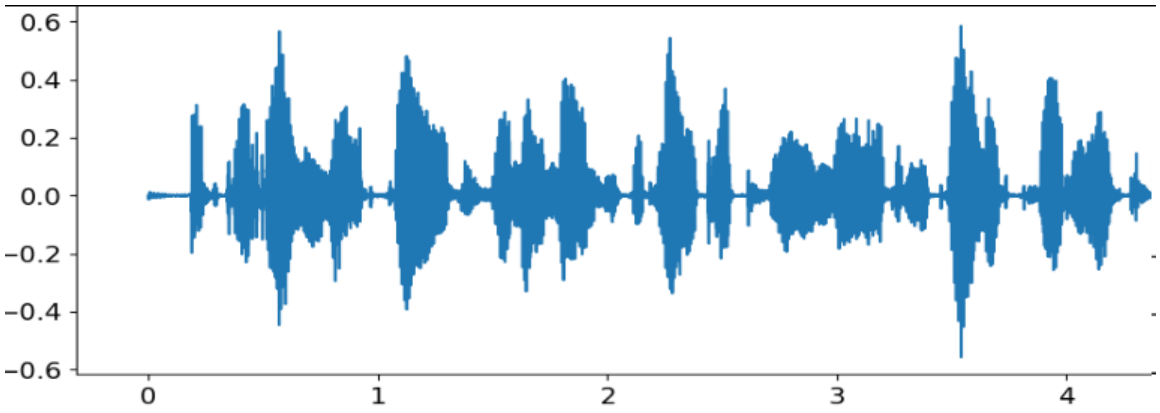


.

Рис. 3.6 Значення топ 10 точності синтезовани



а



б

Рис. 3.7. Приклади синтезованого сигналу

3.5. Висновки

В даному розділі було протестовано модель звукового синтезу VALL-E та продемонстрований аналіз результатів її роботи в межах графіків, таблиць та текстового аналізу. Проведений аналіз результатів синтезу мовлення відносно моделей YourTTS та GroundTruth та продемонстровані графіки отриманих сигналів.

Було досліджено VALL-E, підхід мовної моделі для TTS із кодами аудіокодеків як проміжними представленнями. VALL-E була попередньо навчана з 60 тисячами годин мовних даних, і показана можливість навчання в контексті в сценаріях нульового удару. VALL-E може зберігати акустичне середовище та емоції мовця в синтезі, а також забезпечувати різноманітні результати в різних процесах декодування на основі семплювання.

Незважаючи на значну ефективність, VALL-E страждає від кількох проблем, серед яких:

1. Надійність синтезу: деякі слова можуть бути незрозумілими, пропущеними або дубльованими під час синтезу мовлення. Це головним чином тому, що частина мови «фонема-акустика» є авторегресійною моделлю, в якій існують невпорядковані вирівнювання уваги та немає обмежень для вирішення проблеми.

2. Покриття даних: навіть за умови використання 60 тисяч годин даних для навчання, не можливо охопити голос кожного, особливо мовців з акцентом. Гірший результат на VCTK, ніж на LibriSpeech, також означає недостатнє охоплення мовців з акцентом. Крім того, різноманітності стилів мовлення недостатньо, оскільки LibriLight — це набір даних аудіокниг, у якому більшість висловлювань є у стилі читання.

ВИСНОВКИ

У кваліфікаційній роботі були розглянуті та порівняні між собою сучасні засоби синтезу мовлення. Наведений аналіз демонструє, що найефективнішими для розв’язання задач синтезу мовлення з тексту та перетворення мовлення є методи штучного інтелекту на основі глибокого навчання. Для вирішення проблеми, окресленій в даній кваліфікаційній роботі, була обрана модель на основі трансформера.

У другому розділі були розглянуті приклади використання інструментів генеративного штучного інтелекту, обраних у першому розділі, та обґрунтовано вибір способу розв’язання задач TTS з метою адаптивного перетворення тексту в мову для синтезу високоякісного мовлення для невидимих мовців без тонкого налаштування.

В третьому розділі було протестовано модель звукового синтезу VALL-E та продемонстрований аналіз результатів її роботи в межах графіків, таблиць та текстового аналізу. VALL-E була попередньо навчана з 60 тисячами годин мовних даних, і показана можливість навчання в контексті в сценаріях нульового удару. Згідно результатів аналізу, модель VALL-E генерує одночасно більш зрозуміле мовлення і мовлення, яке має більшу схожість до оригінального мовця, відносно моделей GroundTruth та YourTTS.

Удосконалена процедура навчання моделі VALL-E, яка грунтується на масштабуванні моделі та даних, потенціально може сприяти вирішенню завдання TTS з нульовим ударом.

У майбутньому необхідно додатково розширити навчальні дані, щоб покращити продуктивність моделі з точки зору просодії, стилю мовлення та схожості мовців.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. A. Triantafyllopoulos et al., "An Overview of Affective Speech Synthesis and Conversion in the Deep Learning Era," in Proceedings of the IEEE, vol. 111, no. 10, pp. 1355-1381, Oct. 2023, doi: 10.1109/JPROC.2023.3250266.

2. K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura, “Speech parameter generation algorithms for HMM-based speech synthesis,” in Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, vol. 3, Istanbul, Turkey, 2000, pp. 1315–1318.

3. H. Zen, K. Tokuda, and A. W. Black, “Statistical parametric speech synthesis,” Speech Communication, vol. 51, no. 11, pp. 1039–1064, 2009.

4. M. Morise, F. Yokomori, and K. Ozawa, “WORLD: a vocoder-based high-quality speech synthesis system for real-time applications,” IEICE Transactions on Information and Systems, vol. 99, no. 7, pp. 1877–1884, 2016.

5. H. Kawahara, “STRAIGHT, exploitation of the other aspect of VOCODER: Perceptually isomorphic decomposition of speech sounds,” Acoustical Science and Technology, vol. 27, no. 6, pp. 349– 353, 2006.

6. M. I. Jordan, “Serial order: A parallel distributed processing approach,” in Advances in Psychology, vol. 121, Elsevier, 1997, pp. 471–495.

7. S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

8. S. O. Arik, M. Chrzanowski, A. Coates, G. F. Diamos, A. Gibiansky,¨ Y. Kang, X. Li, J. Miller, A. Y. Ng, J. Raiman, S. Sengupta, and M. Shoeybi, “Deep voice: Real-time neural text-to-speech,” in Proceedings of the International Conference on Machine Learning (ICML), D. Precup and Y. W. Teh, Eds., vol. 70, Sydney, NSW, Australia: PMLR, 2017, pp. 195–204.

9. A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, “Wavenet: A generative model for raw audio,” in The 9th ISCA Speech Synthesis Workshop, Sunnyvale, CA, USA: ISCA, 2016, p. 125

10. J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R.-S. Ryan, R. A. Saurous, Y. Agiomyrgiannakis, and Y. Wu, “Natural TTS synthesis by conditioning wavenet on Mel spectrogram predictions,” in Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada: IEEE, 2018, pp. 4779–4783.

11. Y. Wang, R. J. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. V. Le, Y. Agiomyrgiannakis, R. Clark, and R. A. Saurous, “Tacotron: Towards end-to-end speech synthesis,” in Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH), Stockholm, Sweden: ISCA, 2017, pp. 4006–4010

12. K. Peng, W. Ping, Z. Song, and K. Zhao, “Non-autoregressive neural text-to-speech,” in Proceedings of the International Conference on Machine Learning (ICML), PMLR, Vienna, Austria, 2020, pp. 7586– 7598.

13. N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu, “Neural speech synthesis with transformer network,” in Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA: AAAI, 2019, pp. 6706– 6713.

14. Y. Ren, C. Hu, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, “FastSpeech 2: Fast and high-quality end-to-end text to speech,” in Proceedings of the International Conference on Learning Representations (ICLR), Addis Ababa, Ethiopia: PMLR, 2020.

15. D. P. Kingma and M. Welling, “Auto-encoding variational Bayes,” in Proceedings of the International Conference on Learning Representations (ICLR), Scottsdale, AZ, USA: PMLR, 2013.

16. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” Advances in Neural Information Processing Systems, vol. 27, 2014.

17. A. Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. Driessche, E. Lockhart, L. Cobo, F. Stimberg, et al., “Parallel wavenet: Fast high-fidelity speech synthesis,” in Proceedings of the International Conference on Machine Learning (ICML), PMLR, Stockholm, Sweden, 2018, pp. 3918–3926.

18. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” Advances in Neural Information Processing Systems, vol. 30, 2017.

19. K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brebisson, Y. Bengio, and A. C. Courville, “MelGAN:´ Generative adversarial networks for conditional waveform synthesis,” in Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 2019, pp. 14881–14892.

20. R. Yamamoto, E. Song, and J.-M. Kim, “Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram,” in Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain: IEEE, 2020, pp. 6199–6203.

21. J. Kong, J. Kim, and J. Bae, “Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis,” Advances in Neural Information Processing Systems, vol. 33, pp. 17022–17033, 2020.

22. C. Donahue, J. McAuley, and M. Puckette, “Adversarial audio synthesis,” in Proceedings of the International Conference on Learning Representations (ICLR), Vancouver, BC, Canada: PMLR, 2018.

23. B. Sisman, J. Yamagishi, S. King, and H. Li, “An overview of voice conversion and its challenges: From statistical modeling to deep learning,” IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 29, pp. 132–157, 2020.

24. T. Kinnunen and H. Li, “An overview of text-independent speaker recognition: From features to supervectors,” Speech communication, vol. 52, no. 1, pp. 12–40, 2010.

25. Y. Saito, Y. Ijima, K. Nishida, and S. Takamichi, “Non-parallel voice conversion using variational autoencoders conditioned by phonetic posteriorgrams and d-vectors,” in Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada: IEEE, 2018, pp. 5274–5278.

26. C.-C. Hsu, H.-T. Hwang, Y.-C. Wu, Y. Tsao, and H.-M. Wang, “Voice conversion from non-parallel corpora using variational auto-encoder,” in Proceedings

of the Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA), Jeju, South Korea: IEEE, 2016, pp. 1–6.

27. H. Kameoka, T. Kaneko, K. Tanaka, and N. Hojo, “StarGAN-VC: Non-parallel many-to-many voice conversion with star generative adversarial networks,” in Proceedings of the IEEE Spoken Language Technology Workshop (SLT), Stuttgart, Germany: IEEE, 2018, pp. 266–273.

28. T. Kaneko and H. Kameoka, “CycleGAN-VC: Non-parallel voice conversion using cycle-consistent adversarial networks,” in Proceedings of the IEEE European Signal Processing Conference (EUSIPCO), Rome, Italy: IEEE, 2018, pp. 2100–2104.

29. S. Liu, Y. Cao, S. Kang, N. Hu, X. Liu, D. Su, D. Yu, and H. Meng, “Transferring source style in non-parallel voice conversion,” in Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH), H. Meng, B. Xu, and T. F. Zheng, Eds., Shanghai, China: ISCA, 2020, pp. 4721–4725.

30. Y. Li, K.-A. Lee, Y. Yuan, H. Li, and Z. Yang, “Many-to-many voice conversion based on bottleneck features with variational autoencoder for non-parallel training data,” in Proceedings of the Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA), Honolulu, HI, USA: IEEE, 2018, pp. 829–833.

31. J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired imageto-image translation using cycle-consistent adversarial networks,” in Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy: IEEE, 2017, pp. 2242–2251.

32. T. Kaneko, H. Kameoka, K. Tanaka, and N. Hojo, “CycleGANVC2: Improved CycleGAN-based non-parallel voice conversion,” in Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, United Kingdom: IEEE, 2019, pp. 6820–6824.

33. Y. Choi, M.-J. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, “StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation,” Salt Lake City, UT, USA: IEEE, 2018, pp. 8789–8797.

34. O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, “Convolutional neural networks for speech recognition,” IEEE/ACM Transactions on audio, speech, and language processing, vol. 22, no. 10, 2014.

35. S. Latif, A. Zaidi, H. Cuayahuitl, F. Shamshad, M. Shoukat, J. Qadir, “Transformers in Speech Processing: A Survey”, arXiv preprint arXiv: 2303.11607, 2023.

36. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, pp. 6000–6010.

37. E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov, “Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned,” arXiv preprint arXiv:1905.09418, 2019.

38. Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, “Efficient transformers: A survey,” ACM Computing Surveys, vol. 55, no. 6, pp. 1–28, 2022.

39. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” arXiv preprint arXiv:1810.04805, 2018.

40. A. Radford, K. Narasimhan, T. Salimans, I. Sutskever et al., “Improving language understanding by generative pre-training,” 2018.

41. Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, “XLNet: Generalized autoregressive pretraining for language understanding,” Advances in neural information processing systems, vol. 32, 2019.

42. C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, “Exploring the limits of transfer learning with a unified text-to-text transformer,” The Journal of Machine Learning Research, vol. 21, no. 1, pp. 5485–5551, 2020.

43. N. R. Wu and I. Gurevych, “Sentence-BERT: Sentence Embeddings using Siamese BERT-networks,” in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020, p. 3982–3992.

44. A. Baevski, M. A. Schneider, and M. Auli, “vq-wav2vec: Selfsupervised learning of discrete speech representations,” in International Conference on Learning Representations, 2020. [Online]. Available: https://openreview.net/ forum?id=r1gEjCNYwS

45. A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, “wav2vec 2.0: A framework for self-supervised learning of speech representations,” Advances in Neural Information Processing Systems, vol. 33, pp. 12449– 12460, 2020.

46. A. Baevski, W.-N. Hsu, Q. Xu, A. Babu, J. Gu, and M. Auli, “Data2vec: A general framework for self-supervised learning in speech, vision and language,” in International Conference on Machine Learning. PMLR, 2022, pp. 1298–1312.

47. A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, “Robust speech recognition via large-scale weak supervision,” arXiv preprint arXiv:2212.04356, 2022.

48. Z. Chen, J. Zhang, Z. Liu, X. Chen, and X. Liang, “Neural codec language models are zero-shot text to speech synthesizers,” arXiv preprint arXiv:2301.02111, 2022.

49. Z. Zhang, L. Zhou, C. Wang, S. Chen, Y. Wu, S. Liu, Z. Chen, Y. Liu, H. Wang, J. Li, L. He, S. Zhao, and F. Wei, “Speak foreign languages with your own voice: Cross-lingual neural codec language modeling,” arXiv preprint arXiv: 2303.03926, 2023.

50. A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu et al., “Conformer: Convolution-augmented transformer for speech recognition,” arXiv preprint arXiv:2005.08100, 2020.

51. T. Wang, J. Deng, M. Geng, Z. Ye, S. Hu, Y. Wang, M. Cui, Z. Jin, X. Liu, and H. Meng, “Conformer based elderly speech recognition system for alzheimer’s disease detection,” arXiv preprint arXiv:2206.13232, 2022.

52. C. Wang, Y. Wu, Y. Qian, K. Kumatani, S. Liu, F. Wei, M. Zeng, and X. Huang, “UniSpeech: Unified speech representation learning with labeled and unlabeled data,” in International Conference on Machine Learning. PMLR, 2021, pp. 10937–10947.

53. S. Chen, Y. Wu, C. Wang, Z. Chen, Z. Chen, S. Liu, J. Wu, Y. Qian, F. Wei, J. Li et al., “UniSpeech-SAT: Universal speech representation learning with speaker aware pre-training,” in ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2022, pp. 6152–6156.

54. S. Papi, M. Gaido, M. Negri, and M. Turchi, “Speechformer: Reducing information loss in direct speech translation,” in Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, 2021, pp. 1698–1706.

55. S. Chen, C. Wang, Z. Chen, Y. Wu, S. Liu, Z. Chen, J. Li, N. Kanda, T. Yoshioka, X. Xiao et al., “Wavlm: Large-scale self-supervised pretraining for full stack speech processing,” IEEE Journal of Selected Topics in Signal Processing, vol. 16, no. 6, pp. 1505–1518, 2022.

56. Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. Le, and R. Salakhutdinov, “Transformer-xl: Attentive language models beyond a fixed-length context,” in Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019, pp. 2978–2988.

57. A. Vaswani, S. Bengio, E. Brevdo, F. Chollet, A. Gomez, S. Gouws, L. Jones, Ł. Kaiser, N. Kalchbrenner, N. Parmar et al., “Tensor2tensor for neural machine translation,” in Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track), 2018, pp. 193–199.

58. P. J. Werbos, “Backpropagation through time: what it does and how to do it,” Proceedings of the IEEE, vol. 78, no. 10, pp. 1550–1560, 1990.

59. S. Karita, N. Chen, T. Hayashi, T. Hori, H. Inaguma, Z. Jiang, M. Someki, N. E. Y. Soplin, R. Yamamoto, X. Wang et al., “A comparative study on transformer vs RNN in speech applications,” in 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU). IEEE, 2019, pp. 449–456.

60. A. Zeyer, P. Bahar, K. Irie, R. Schluter, and H. Ney, “A comparison of transformer and LSTM encoder decoder models for ASR,” in 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU). IEEE, 2019, pp. 8–15.

61. J. Li, Y. Wu, Y. Gaur, C. Wang, R. Zhao, and S. Liu, “On the comparison of popular end-to-end models for large scale speech recognition,” Proc. Interspeech 2020, pp. 1–5, 2020.

62. Y. Wang, Y. Shi, F. Zhang, C. Wu, J. Chan, C.-F. Yeh, and A. Xiao, “Transformer in action: a comparative study of transformer-based acoustic models for large scale speech recognition applications,” arXiv preprint arXiv:2010.14665, 2020.

63. S. Zhou, L. Dong, S. Xu, and B. Xu, “A comparison of modeling units in sequence-to-sequence speech recognition with the transformer on mandarin chinese,” in International Conference on Neural Information Processing. Springer, 2018, pp. 210–220.

64. X. Chen, Y. Wu, Z. Wang, S. Liu, and J. Li, “Developing real-time streaming transformer transducer for speech recognition on large-scale dataset,” arXiv preprint arXiv:2010.11395, 2020.

65. A. Graves, “Sequence transduction with recurrent neural networks,” arXiv preprint arXiv:1211.3711, 2012.

66. P. Taylor, Text-to-speech synthesis. Cambridge university press, 2009.

67. W. Ping, K. Peng, A. Gibiansky, S. O. Arik, A. Kannan, S. Narang, J. Raiman, and J. Miller, “Deep voice 3: 2000-speaker neural text-tospeech,” Proc. ICLR, pp. 214–217, 2018.

68. W. Ping, K. Peng, and J. Chen, “Clarinet: Parallel wave generation in end-to-end text-to-speech,” in International Conference on Learning Representations, 2018.

69. D. Griffin and J. Lim, “Signal estimation from modified short-time Fourier transform,” IEEE Transactions on acoustics, speech, and signal processing, vol. 32, no. 2, pp. 236–243, 1984.

70. A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, “Wavenet: A generative model for raw audio,” https://arxiv.org/abs/1609.03499, 2016.

71. R. Prenger, R. Valle, and B. Catanzaro, “Waveglow: A flow-based generative network for speech synthesis,” in ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019, pp. 3617–3621.

72. N. Li, Y. Liu, Y. Wu, S. Liu, S. Zhao, and M. Liu, “Robutrans: A robust transformer-based text-to-speech model,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 05, 2020, pp. 8228–8235.

73. X. Wang, H. Ming, L. He, and F. K. Soong, “s-transformer: Segmenttransformer for robust neural speech synthesis,” arXiv preprint arXiv:2011.08480, 2020.

74. Y. Zheng, X. Li, F. Xie, and L. Lu, “Improving end-to-end speech synthesis with local recurrent neural network enhanced transformer,” in ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2020, pp. 6734–6738.

75. H. Ney, “Speech translation: Coupling of recognition and translation,” in 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No. 99CH36258), vol. 1. IEEE, 1999, pp. 517–520.

76. E. Matusov, S. Kanthak, and H. Ney, “On the integration of speech recognition and statistical machine translation,” in Ninth European Conference on Speech Communication and Technology, 2005.

77. Q. T. Do, S. Sakti, and S. Nakamura, “Toward expressive speech translation: A unified sequence-to-sequence lstms approach for translating words and emphasis.” in INTERSPEECH, 2017, pp. 2640–2644.

78. A. Berard, O. Pietquin, L. Besacier, and C. Servan, “Listen and translate: A proof of concept for end-to-end speech-to-text translation,” in NIPS Workshop on end-to-end learning for speech and audio processing, 2016.

79. A. Berard, L. Besacier, A. C. Kocabiyikoglu, and O. Pietquin, “End-´ to-end automatic speech translation of audiobooks,” in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2018, pp. 6224–6228.

80. R. J. Weiss, J. Chorowski, N. Jaitly, Y. Wu, and Z. Chen, “Sequence-tosequence models can directly translate foreign speech,” Proc. Interspeech 2017, pp. 2625–2629, 2017.

81. L.-C. Vila, C. Escolano, J. A. Fonollosa, and M.-R. Costa-Jussa, “End-` to-end speech translation with the transformer,” Proc. IberSPEECH 2018, pp. 60–63, 2018.

82. P. Zhang, N. Ge, B. Chen, and K. Fan, “Lattice transformer for speech translation,” in Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019, pp. 6475–6484.

83. E. Casanova, J. Weber, C. D Shulby, A. C. Junior, E. Gölge, and M. A Ponti. Yourtts: Towards zero-shot multi-speaker tts and zero-shot voice conversion for everyone. In ICML, pages 2709–2720. PMLR, 2022b.

84. J. Xu, X. Sun, Zh. Zhang, G. Zhao, and J. Lin. Understanding and improving layer normalization. In Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pages 4383–4393, 2019.

85. W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 29:3451–3460, 2021.

86. K. Lakhotia, E. Kharitonov, W.-N. Hsu, Y. Adi, A. Polyak, B. Bolte, T. A. Nguyen, J. Copet, A. Baevski, A. Mohamed, and E. Dupoux. Generative spoken language modeling from raw audio. CoRR, abs/2102.01192, 2021.

87. C. Du, Y. Guo, X. Chen, and K. Yu. VQTTS: high-fidelity text-to-speech synthesis with self-supervised VQ acoustic feature. In Interspeech 2022, 23rd Annual Conference of the International Speech Communication Association, Incheon, Korea, 18-22 September 2022, pages 1596–1600. ISCA, 2022. doi: 10.21437/Interspeech.2022-489.

88. Z. Borsos, R. Marinier, D. Vincent, E. Kharitonov, O. Pietquin, M. Sharifi, O. Teboul, D. Grangier, M. Tagliasacchi, and N. Zeghidour. Audiolm: a language modeling approach to audio generation. CoRR, abs/2209.03143, 2022.

89. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In NAACL, pages 4171–4186, 2019.

90. LibriTTS [Online]: https://research.google/resources/datasets/libri-tts/

91. Ch. Wang, S. Chen, Y. Wu, Z. Zhang, L. Zhou, S. Liu, Zh. Chen, Y. Liu, H. Wang, J. Li, L. He, Sh. Zhao, F. Wei, “Neural Codec Language Models are Zero-Shot Text to Speech Synthesizers”, arXiv preprint arXiv: 2301.02111, 2023.

92. VALL-E-X [Online]: https://www.microsoft.com/en-us/research/project/vall-e-x/vall-e-x/

93. Методичні рекомендації до виконання кваліфікаційних робіт здобувачами другого (магістерського) рівня вищої освіти спеціальностей 121 «Інженерія програмного забезпечення» та 122 «Комп’ютерні науки» / Б.І. Мороз, О.В. Іванченко, О.В. Реута, О.С. Шевцова; М-во освіти і науки України, Нац. техн. ун-т “Дніпровська політехніка”. – Дніпро : НТУ «ДП», 2021. – 56 с.

Додаток А

КОД ПРОГРАМИ

```
import json
import logging
from collections import defaultdict

import torch
import os
import pickle
from tqdm import tqdm

from torch.utils.data import DataLoader, Dataset
from torch import Tensor, einsum, nn

_logger = logging.getLogger(__name__)


def load_engines():
    model = get_model(cfg.model)

    engines = dict(
        model=trainer.Engine(
            model=model,
            config=cfg.ds_cfg,
        ),
    )
    engines = trainer.load_engines(engines, cfg)
    return engines

def main():
    setup_logging(cfg.log_dir)

    print('datasets start')
    if os.path.isfile('./vall_e/dataloaders/train_dl.pkl'):
        with open('./vall_e/dataloaders/train_dl.pkl', 'rb') as f:
            train_dl = pickle.load(f)
        with open('./vall_e/dataloaders/subtrain_dl.pkl', 'rb') as f:
            subtrain_dl = pickle.load(f)
        with open('./vall_e/dataloaders/val_dl.pkl', 'rb') as f:
            val_dl = pickle.load(f)

        if not os.path.exists('./vall_e/dataloaders'):
            os.makedirs('./vall_e/dataloaders')

        with open('./vall_e/dataloaders/train_dl.pkl', 'wb') as f:
            pickle.dump(train_dl, f)
        print('train loader saved')
        with open('./vall_e/dataloaders/subtrain_dl.pkl', 'wb') as f:
            pickle.dump(subtrain_dl, f)
        print('subtrain loader saved')
        with open('./vall_e/dataloaders/val_dl.pkl', 'wb') as f:
            pickle.dump(val_dl, f)
        print('validation loader saved')

    print('datasets done')

def train_feeder(engines, batch, name):
    model = engines["model"]
```

```
    if cfg.model.startswith("ar"):
        _ = model(
            text_list=batch["text"],
            proms_list=batch["proms"],
            resp_list=batch["resp"],
        )
    elif cfg.model.startswith("nar"):
        _ = model(
            text_list=batch["text"],
            proms_list=batch["proms"],
            resps_list=batch["resps"],
        )
    else:
        raise NotImplementedError(cfg.model)

    losses = model.gather_attribute("loss")

    loss = torch.stack([*losses.values()]).sum()

    stats = {}
    stats |= {k: v.item() for k, v in losses.items()}
    stats |= engines.gather_attribute("scalar")

    return loss, stats

def _create_dataloader(dataset, training):
    return DataLoader(
        dataset=dataset,
        batch_size=cfg.batch_size,
        shuffle=training,
        drop_last=training,
        num_workers=cfg.nj,
        collate_fn=collate_fn,
        persistent_workers=True,
        worker_init_fn=_seed_worker,
    )

def _load_train_val_paths():
    paths = []
    train_paths = []
    val_paths = []

    for data_dir in cfg.data_dirs:
        paths.extend(tqdm(data_dir.rglob("*.qnt.pt")))

    if len(paths) == 0:
        raise RuntimeError(f"Failed to find any .qnt.pt file in
{cfg.data_dirs}.")

    pairs = sorted([(cfg.get_spkr(p), p) for p in paths])
    del paths

    for _, group in groupby(pairs, lambda pair: pair[0]):
        paths = sorted([p for _, p in group])
        random.seed(0)
        random.shuffle(paths)
        n = round(len(paths) * 0.95)
        train_paths.extend(paths[:n])
        val_paths.extend(paths[n:])

    train_paths, val_paths = map(sorted, [train_paths, val_paths])

    return train_paths, val_paths
```

```
@cfg.diskcache()
def create_datasets():
    train_paths, val_paths = _load_train_val_paths()

    train_dataset = VallEDataset(
        train_paths,
        training=True,
    )

    val_dataset = VallEDataset(
        val_paths,
        train_dataset.phone_symmap,
        train_dataset.spkr_symmap,
        extra_paths_by_spkr_name=train_dataset.paths_by_spkr_name,
    )

    val_dataset.interleaved_reorder_(cfg.get_spkr)
    val_dataset.head_(cfg.max_num_val)

    return train_dataset, val_dataset

def create_train_val_dataloader():
    train_dataset, val_dataset = create_datasets()

    train_dl = _create_dataloader(train_dataset, training=True)
    val_dl = _create_dataloader(val_dataset, training=False)

    _logger.info(str(train_dataset.phone_symmap))
    _logger.info(str(train_dataset.spkr_symmap))

    _logger.info(f"#samples (train): {len(train_dataset)}.")
    _logger.info(f"#samples (val): {len(val_dataset)}.")

    subtrain_dataset = copy.deepcopy(train_dataset)
    subtrain_dataset.interleaved_reorder_(cfg.get_spkr)
    subtrain_dataset.head_(cfg.max_num_val)
    subtrain_dataset.training_(False)
    subtrain_dl = _create_dataloader(subtrain_dataset, training=False)
    assert isinstance(subtrain_dl.dataset, VallEDataset)

    return train_dl, subtrain_dl, val_dl

@dataclass(frozen=True)
class Config(ConfigBase):
    data_root: Path = Path("data")
    data_dirs: list[Path] = field(default_factory=lambda: [])

    @property
    def sample_rate(self):
        return 24_000

    is_style_layer=True

    p_additional_prompt: float = 0.8
    max_prompts: int = 3

    max_num_val: int = 20
    max_val_ar_steps: int = 300

    token_dim: int = 256
    num_tokens: int = 1024
```

```
    nj: int = 8
    batch_size: int = 32
    eval_batch_size: int = 32
    warmup_min_lr: float = 1e-6
    warmup_max_lr: float = 2e-4
    dis_warmup_max_lr: float = 4e-4
    warmup_num_steps: int = 1_000
    max_iter: int = 1_000_000
    gradient_clipping: float = 1
    eval_every: int = 2_000
    save_ckpt_every: int = 2_000

    model: str = "ar-quarter"
    spkr_name_getter: str = "lambda p: p.parts[-2]"

    min_phones: int = 10
    max_phones: int = 50

    use_fp16: bool = True
    gradient_accumulation_steps: int = 1
    sampling_temperature: float = 1.0

    cache_dataloader: bool = False

    @cached_property
    def get_spkr(self):
        return eval(self.spkr_name_getter)

    @property
    def fp16_cfg(self):
        return {
            "enabled": self.use_fp16,
        }

    @property
    def ds_cfg(self):
        return {
            "train_micro_batch_size_per_gpu": self.batch_size,
            "gradient_accumulation_steps":
self.gradient_accumulation_steps,
            "optimizer": {
                "type": "Adam",
                "lr": self.warmup_min_lr,
            },
            "scheduler": {
                "type": "WarmupDecayLR",
                "params": {
                    "warmup_min_lr": self.warmup_min_lr,
                    "warmup_max_lr": self.warmup_max_lr,
                    "warmup_num_steps": self.warmup_num_steps,
                    "total_num_steps": self.max_iter,
                    "warmup_type": "linear",
                },
            },
            "gradient_clipping": self.gradient_clipping,
            "fp16": self.fp16_cfg,
        }

    @property
    def cache_dir(self):
        return ".cache" / self.relpath

    @cached_property
```

```
    def diskcache(self):
        if self.cache_dataloader:
            return diskcache.Cache(self.cache_dir).memoize
        return lambda: lambda x: x

cfg = Config.from_cli()

@torch.inference_mode()
def run_eval(engines, name, dl):
    log_dir = cfg.log_dir / str(engines.global_step) / name

    model = engines["model"]
    log_dir = cfg.log_dir / str(engines.global_step) / name
    stats = defaultdict(list)
    for batch in tqdm(dl):
        batch: dict = to_device(batch, cfg.device)

        if cfg.model.startswith("ar"):
            resp_list = model(
                text_list=batch["text"],
                proms_list=batch["proms"],
                max_steps=cfg.max_val_ar_steps,
                sampling_temperature=cfg.sampling_temperature,
            )
            resps_list = [r.unsqueeze(-1) for r in resp_list]

        losses = model.gather_attribute("loss")
        batch_stats = {k: v.item() for k, v in losses.items()}
        for k, v in batch_stats.items():
            stats[k].append(v)

        for path, ref, hyp in zip(batch["path"], batch["resps"],
resps_list):
            hyp_path = (log_dir / "hyp" /
cfg.data_root).with_suffix(".wav")
            ref_path = (log_dir / "ref" /
cfg.data_root).with_suffix(".wav")
            hyp_path.parent.mkdir(parents=True, exist_ok=True)
            ref_path.parent.mkdir(parents=True, exist_ok=True)
            qnt.decode_to_file(ref, ref_path)
            if len(hyp) > 0:
                qnt.decode_to_file(hyp, hyp_path)

    stats = {k: sum(v) / len(v) for k, v in stats.items()}
    stats["global_step"] = engines.global_step
    stats["name"] = name

def eval_fn(engines):
    run_eval(engines, "subtrain", subtrain_dl)
    run_eval(engines, "val", val_dl)

    print('training start')
    trainer.train(
        engines_loader=load_engines,
        train_dl=train_dl,
        train_feeder=train_feeder,
        eval_fn=eval_fn,
        is_style_layer=True
    )

class MultiHeadAttention(nn.Module):

    def __init__(self, query_dim, key_dim, num_units, num_heads):
```

```
        super().__init__()
        self.num_units = num_units
        self.num_heads = num_heads
        self.key_dim = key_dim

        self.W_query = nn.Linear(in_features=query_dim,
out_features=num_units, bias=False)
        self.W_key = nn.Linear(in_features=key_dim,
out_features=num_units, bias=False)
        self.W_value = nn.Linear(in_features=key_dim,
out_features=num_units, bias=False)

    def forward(self, query, key):
        querys = self.W_query(query)  # [N, T_q, num_units]
        keys = self.W_key(key)  # [N, T_k, num_units]
        values = self.W_value(key)

        split_size = self.num_units // self.num_heads
        querys = torch.stack(torch.split(querys, split_size, dim=2),
dim=0)  # [h, N, T_q, num_units/h]
        keys = torch.stack(torch.split(keys, split_size, dim=2), dim=0)  #
[h, N, T_k, num_units/h]
        values = torch.stack(torch.split(values, split_size, dim=2),
dim=0)  # [h, N, T_k, num_units/h]

        # score = softmax(QK^T / (d_k ** 0.5))
        scores = torch.matmul(querys, keys.transpose(2, 3))  # [h, N, T_q,
T_k]
        scores = scores / (self.key_dim ** 0.5)
        scores = F.softmax(scores, dim=3)

        # out = score * V
        out = torch.matmul(scores, values)  # [h, N, T_q, num_units/h]
        out = torch.cat(torch.split(out, 1, dim=0), dim=3).squeeze(0)  #
[N, T_q, num_units]

        return out

class STL(nn.Module):

    def __init__(self, E, token_num, num_heads):
        super().__init__()
        self.embed = nn.Parameter(torch.FloatTensor(token_num, E //
num_heads))
        d_q = E // 2
        d_k = E // num_heads
        self.attention = MultiHeadAttention(query_dim=d_q, key_dim=d_k,
num_units=E, num_heads=num_heads)
        # style controllable vall-e test
        self.lastlin = nn.Linear(E, E//2)

        init.normal_(self.embed, mean=0, std=0.5)

    def forward(self, inputs, is_scale = False):
        N = inputs.size(0)
        query = inputs.unsqueeze(1)  # [N, 1, E//2]
        if is_scale:
            multiplier = torch.tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
dtype=torch.float16)
            scaled_embed = self.embed * multiplier.unsqueeze(1).cuda()
            keys = torch.tanh(scaled_embed).unsqueeze(0).expand(N, -1, -1)
# [N, token_num, E // num_heads]
        else:
```

```
            keys = torch.tanh(self.embed).unsqueeze(0).expand(N, -1, -1)
# [N, token_num, E // num_heads]

        style_embed = self.attention(query, keys)

        style_embed = self.lastlin(style_embed)

        return style_embed

class SinusodialEmbedding(nn.Module):

    def __init__(self, d_model):
        super().__init__()
        self.d_model = d_model
        exponent = torch.arange(self.d_half, dtype=torch.float32)
        exponent = exponent / self.d_half
        omega = torch.exp(-math.log(1e4) * exponent)
        self.omega: torch.Tensor
        self.register_buffer("omega", omega, persistent=False)

    @property
    def d_half(self):
        assert self.d_model % 2 == 0, "Only support even d_model."
        return self.d_model // 2

    def forward(self, x):
        omega = self.omega

        while omega.dim() <= x.dim():
            omega = omega.unsqueeze(0)  # (... d)

        x = x.unsqueeze(-1)  # (... 1)
        x = omega * x
        x = torch.cat([x.sin(), x.cos()], dim=-1)

        return x

    def get_pe(self, n: int):
        device = self.omega.device
        return self.forward(torch.arange(n, device=device))

    def add_pe(self, x):
        e = self.get_pe(x.shape[1])  # t d
        e = e[None]  # b t d
        x = x + e
        return x


class Attention(nn.Module):

    def __init__(self, d_model, n_heads, casual):
        super().__init__()
        assert d_model % n_heads == 0
        dim_head = d_model // n_heads
        self.casual = casual
        self.n_heads = n_heads
        self.scale = dim_head**-0.5
        self.to_qkv = nn.Linear(d_model, d_model * 3, bias=False)
        self.to_out = nn.Linear(d_model, d_model)

    def softmax(self, input_tensor, dimension=2):
        # Logit scaling
```

```
        max_value, _ = torch.max(input_tensor, dim=dimension,
keepdim=True)
        scaled_input = input_tensor - max_value

        # Exponential and sum
        exp_values = torch.exp(scaled_input)
        sum_exp = torch.sum(exp_values, dim=dimension, keepdim=True)

        # Softmax
        softmax_output = exp_values / sum_exp

        # Apply exception handling for specific vectors
        if dimension == 2:
            invalid_mask = torch.isnan(softmax_output)
            softmax_output[invalid_mask] = 0.0

        return softmax_output

    def forward(self, x, m):
        h = self.n_heads

        q, k, v = self.to_qkv(x).chunk(3, dim=-1)
        q, k, v = map(lambda t: rearrange(t, "b t (h d) -> b t h d", h=h),
(q, k, v))

        e = einsum("b i h d, b j h d -> b i j h", q, k)
        e = e * self.scale
        kpm = m.unsqueeze(1) * m.unsqueeze(2)  # b i j 1

        if self.casual:
            kpm = kpm.squeeze(-1).tril().unsqueeze(-1)  # b i j 1

        e = e.masked_fill(kpm == 0, -torch.finfo(e.dtype).max)
        a = self.softmax(e, dimension = 2)
        o = einsum("b i j h, b j h d -> b i h d", a, v)
        o = o.flatten(-2)
        o = self.to_out(o)  # b t c
        o = o * m

        return o

if __name__ == "__main__":
    main()
```

Додаток Б

ВІДГУК КЕРІВНИКА
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ДНІПРОВСЬКА ПОЛІТЕХНІКА»

Факультет інформаційних технологій
Кафедра програмного забезпечення комп'ютерних систем

ВІДГУК
на магістерську роботу

Наукового керівника Алексєєва Михайла Олександровича, д.т.н., проф. каф. ПЗКС
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання, посада, місце роботи)

студента Снісара Андрія Владиславовича
(прізвище, ім'я, по батькові)

курсу II групи 122м-22-1
спеціальності 122 Комп'ютерні науки
освітньої програми
на тему Дослідження ефективності застосування генеративних методів штучного інтелекту до синтезу звуку

Актуальність теми Представлена магістерська кваліфікаційна робота присвячена підвищенню ефективності процесу синтезу мовлення для невидимих мовців, які знаходяться поза набором даних із обмеженими довідковими даними. В роботі обґрунтовано вибір способу розв'язання задач TTS з метою адаптивного перетворення тексту в мову для синтезу високоякісного мовлення. З огляду на це, магістерська робота характеризується актуальністю та своєчасністю.

Мета досліджень Полягає у дослідження моделі синтезу мовлення з використанням нейронної мережі

Коротка характеристика розділів роботи У першому розділі було розглянуто визначення поняття та характеристику процесу синтезу мови, проаналізовані методи розв'язання подібних задач, проведено порівняльний аналіз програмних інструментів задля виявлення їх слабких та сильних сторін. У другому розділі розглянуто види методи генеративного штучного інтелекту та обґрунтовано вибір способу розв'язання задач TTS з метою адаптивного перетворення тексту в мову для синтезу високоякісного мовлення для невидимих мовців без тонкого налаштування. Третій розділ містить опис та реалізацію навчання моделі синтезу мовлення на основі трансформера VALL-E.

Практичне значення роботи полягає у тому, що навчена модель синтезу мовлення дозволяє накопичувати і використовувати знання для вирішення задач синтезу мовлення та подальшого дослідження предметної галузі.

Зауваження та недоліки В роботі відсутній більш детальний порівняльний

аналіз методів навчання моделі трансформера в контексті в сценаріях нульового удару.

Висновки та оцінка Магістром було проведено аналіз та порівняння можливих методів розв'язання поставленої задачі та обрано оптимальний варіант. Під час виконання магістерської кваліфікаційної роботи студент Снісар А.В. проявив себе грамотним, кваліфікованим спеціалістом, здатним приймати самостійно складні технічні рішення. Вважаю, що магістерська кваліфікаційна робота заслуговує оцінку «відмінно», а Снісар А.В. – присвоєння кваліфікації «магістра» з комп'ютерних наук.

Науковий керівник Алексєєв М.О., док. техн. наук, проф., зав. каф. ПЗКС

(прізвище, ім'я, по батькові, посада, місце роботи)

«___»_____20__ р. _____

(підпис)

Додаток В

РЕЦЕНЗІЯ
на магістерську роботу

студента Снісара Андрія Владиславовича
(прізвище, ім'я, по батькові)
курсу II групи _122м-22-1___
кафедри програмного забезпечення комп'ютерних систем
спеціальності 122 Комп'ютерні науки
освітньої програми
Тема роботи Дослідження ефективності застосування генеративних методів штучного інтелекту до синтезу звуку

Стисла характеристика розділів роботи В першому розділі розглянуто визначення поняття та характеристику процесу синтезу мови, проаналізовані методи розв'язання подібних задач, проведено порівняльний аналіз програмних інструментів. У другому розділі розглянуто види методи генеративного штучного інтелекту та обґрунтовано вибір способу розв'язання задач TTS з метою адаптивного перетворення тексту в мову для синтезу високоякісного мовлення. Третій розділ містить опис та реалізацію навчання моделі синтезу мовлення на основі трансформера VALL-E.

Пропозиції, внесені студентом, рівень їх наукового обґрунтування В даній кваліфікаційній роботі студентом надано декілька пропозицій щодо вирішення поставлених задач. Кожна з пропозицій була обґрунтована та підкріплена науковими даними.

Практичне значення роботи Результати роботи можуть бути застосовані для подальших наукових досліджень в даній сфері, а також вони можуть бути корисними для практичного використання.

Якість оформлення роботи Магістерська кваліфікаційна робота, яку подано на рецензію, виконана у повному обсязі у встановлений термін. Робота є добре структурованою та достатньо проілюстрованою. Викладена основна суть проблеми, що вирішується в ході виконання роботи, і шляхів її вирішення.

Недоліки в роботі відсутність детального математичного опису процесу квантування вхідного звукового сигналу. Проте вказаний недолік не впливає на позитивне враження від роботи.

Загальний висновок Магістерська кваліфікаційна робота виконана у
(підготовленість студента до самостійної роботи як спеціаліста)
відповідності з завданням із дотриманням всіх вимог.

Оцінка магістерської роботи Робота заслуговує оцінки «відмінно», а студент Снісар А.В. – присвоєння кваліфікації «магістра» з комп'ютерних наук.

Рецензент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання, посада, місце роботи)

«___»_____20__ р. _____
(підпис)

Додаток Г

ПЕРЕЛІКЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Снісар_АВ.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Снісар_АВ.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF.
Програма	
VALL-E.rar	Архів. Містить коди програми і откомпільовану програму.
Презентація	
Презентація_Снісар.ppt	Презентація кваліфікаційної роботи.