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 *Ліс'їх Артема Ігоровича*
(ПІБ)

академічної групи *121М-20-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *«Інженерія програмного забезпечення»*
(назва освітньої програми)

на тему: *Дослідження ефективності синтезу мовлення з урахуванням обмежень обчислювальної потужності*

_____*А.І. Ліс'їх*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
розділів кваліфікаційної роботи				
спеціальний	*Проф. Алексєєв М.О.*			
економічний	*Проф. Вагонова О.Г.*			
Рецензент				
Нормоконтролер	*Доц. Приходченко С.Д.*			

Дніпро
2022

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

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

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 20 22 Року

ЗАВДАННЯ
на виконання кваліфікаційної роботи магістра

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

студенту *121м-20-1* *Ліс'їх Артему Ігоровичу*
(група) (прізвище та ініціали)

Тема дипломного проєкту *Дослідження ефективності синтезу мовлення з урахуванням обмежень обчислювальної потужності*

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від 10.12.2021 р. № 1036-с.

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень – процес синтезу мовлення за допомогою нейронних мереж.

Предмет досліджень – моделі та методи синтезу мовлення.

Мета НДР – дослідження ефективності синтезу українського мовлення з урахуванням обчислювальної потужності.

Вихідні дані для проведення роботи – теоретичні та експериментальні дослідження, дані взяті з аудіокниг для навчання україномовному синтезу.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна отриманих результатів полягає в тому, що удосконалено модель синтезу мовлення FastSpeech, що дозволило підвищити ефективність синтезу українського мовлення з урахуванням обчислювальної потужності.

Практична цінність результатів полягає в тому, що удосконалена модель синтезу мовлення та оцінка результатів синтезу дозволяють накопичувати і використовувати знання для вирішення задач синтезу українського 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 удосконаленої моделі синтезу українського мовлення.

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок – кінець**
Аналіз теми та постановка задачі	12.09.2021 - 30.09.2021
Детальний аналіз методів синтезу мовлення та розробка алгоритму	01.10.2021 - 31.10.2021
Тренування моделей синтезу українського мовлення, проведення оцінювання якості синтезу та аналіз результатів	01.11.2021 - 30.12.2021

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

Завдання прийняв до виконання _____ *Ліс'їх А. І.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *12.09.2021 р.*

Термін подання дипломного проєкту до ЕК *21.01.2021*

РЕФЕРАТ

Пояснювальна записка: 96 стор., 25 рис., 11 таблиць, 3 додатка, 69 джерел.

Об'єкт дослідження: процес синтезу мовлення за допомогою нейронних мереж.

Предмет дослідження: моделі та методи синтезу мовлення.

Мета магістерської роботи: дослідження ефективності синтезу українського мовлення з урахуванням обчислювальної потужності.

Методи дослідження. Для вирішення поставлених задач використані методи: аналізу даних, математичної статистики, нейронних мереж, хмарних технологій, функціонального програмування, об’єктно-орієнтованого програмування.

Наукова новизна отриманих результатів полягає в тому, що удосконалено модель синтезу мовлення FastSpeech, що дозволило підвищити ефективність синтезу українського мовлення з урахуванням обчислювальної потужності.

Практична цінність результатів полягає в тому, що удосконалена модель синтезу мовлення та оцінка результатів синтезу дозволяють накопичувати і використовувати знання для вирішення задач синтезу українського мовлення та подальшого дослідження предметної галузі.

Область застосування. Результати дослідження моделей синтезу мовлення можуть застосовуватись у вдосконаленні якості синтезу мовлення, пошуку нових шляхів зменшення необхідної обчислювальної потужності для їх використання, та у виборі методу синтезу українського мовлення для будь-якої сфери, у якій можна використовувати інформаційні технології.

Значення роботи та висновки. Удосконалена модель синтезу українського мовлення надає нові знання про якість синтезу та вимоги до обчислювальної потужності, що дозволяють продовжувати дослідження у цій сфері та використовувати найбільш ефективні методи у розробці нових програмних продуктів.

Прогнози щодо розвитку досліджень. Дослідити та покращити якість синтезованого мовлення запропонованою моделлю при зміні параметрів моделі або архітектурних рішень.

У розділі «Економіка» проведені розрахунки трудомісткості розробки удосконаленої моделі синтезу мовлення, а також проведені маркетингові дослідження ринку збуту створеного програмного продукту і порахована економічна ефективність його впровадження на підприємстві.

Список ключових слів: синтез мовлення, нейронні мережі, глибинне навчання, нейронні мережі прямого поширення, MOS, Glow-TTS, SpeedySpeech, Tacotron2, FastSpeech, FastSpeech1.1.

ABSTRACT

Explanatory note: 96 pages, 25 figures, 11 tables, 3 applications, 69 sources.

Object of research: the process of speech synthesis using neural networks.

Subject of research: models and methods of speech synthesis.

Purpose of Master's thesis: increasing the efficiency of the synthesis of Ukrainian speech, taking into account computing power.

Research methods. The following methods are used: data analysis, mathematical statistics, neural networks, cloud technology, functional programming, object-oriented programming.

Originality of research is in the improvement of speech synthesis model FastSpeech, which allowed to increase the efficiency of synthesis of Ukrainian speech.

Practical value of the results is that the improved model of speech synthesis and evaluation of synthesis results allow to accumulate and use knowledge to solve problems of synthesis of Ukrainian speech and further study of the subject.

Scope of application. The results of the study of speech synthesis models can be used to improve the quality of speech synthesis, find new ways to reduce the required computing power for their use, and in choosing a method of Ukrainian speech synthesis for any field in which information technology can be used.

The value of the work and conclusions. The improved model of synthesis of Ukrainian speech provides new knowledge about the quality of synthesis and requirements for computing power, which allows to continue research in this area and use the most effective methods in the development of new software products.

Research forecast and development. Investigate and improve the quality of synthesized speech by the proposed model when changing the parameters of the model or architectural solutions.

In the Economics section we calculated the complexity of development of the improved speech synthesis model, and also made marketing research for the created software product and calculated the economic efficiency of its introduction at an enterprise..

Keywords: speech synthesis, neural networks, deep learning, feed-forward neural networks, MOS, Glow-TTS, SpeedySpeech, Tacotron2, FastSpeech, FastSpeech1.1.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

DL – Deep Learning;

F0 – основна частота;

FFT – Feed-Forward Transformer;

MOS – Mean opinion score;

Seq2seq – Sequence-to-sequence;

TTS – Text-to-speech.

ЗМІСТ

ВСТУП

Синтез мовлення – це штучне відтворення людського мовлення. Комп’ютерна система, яка використовується для цієї мети, називається синтезатором мовлення, і може бути реалізована в програмних та апаратних продуктах. Система синтезу мовлення, або TTS перетворює наданий користувачем текст у штучне мовлення.

Синтез мовлення є актуальною задачею, і вже давно є важливим допоміжним технологічним інструментом. Його застосування дозволяє усунути бар’єри доступу для широкого спектру людей з інвалідністю. Області використання включають інформаційні технології, освіту, медицину, телекомунікаційні технології, транспортні системи, мультимедіа тощо.

Існують різні методи синтезу мовлення, серед актуальних: конкатенативний, формантний, артикуляційний, статистичний параметричний, та методи, засновані на нейронних мережах та глибинному навчанні.

Однією з найбільш нагальних проблем систем мовлення, заснованих на глибинному навчанні, є використання великої кількості обчислювальних ресурсів для тренування нейронних мереж. Наявні моделі часто використовують більше кількох діб для навчання англійській мові.

Метою магістерської роботи є дослідження ефективності синтезу українського мовлення з урахуванням обчислювальної потужності. Об'єктом дослідження є процес синтезу мовлення за допомогою глибинних нейронних мереж.

Для виконання кваліфікаційної роботи використовуються методи аналізу даних, математичної статистики, нейронних мереж, хмарних технологій, функціонального програмування, об’єктно-орієнтованого програмування.

Наукова новизна отриманих результатів дипломної роботи визначається тим, що удосконалено модель синтезу мовлення FastSpeech, що дозволило

підвищити ефективність синтезу українського мовлення з урахуванням обчислювальної потужності.

Результати дослідження моделей синтезу мовлення можуть застосовуватись у вдосконаленні якості синтезу мовлення, пошуку нових шляхів зменшення необхідної обчислювальної потужності для їх використання, та у виборі методу синтезу українського мовлення для будь-якої сфери, у якій можна використовувати інформаційні технології.

Кваліфікаційна робота складається з чотирьох розділів. У першому розділі було розглянуто актуальні методи синтезу мовлення, наявні проблеми цієї предметної області та поставлено задачу кваліфікаційної роботи. У другому розділі проведено детальний огляд досліджуваної моделі синтезу мовлення глибинного навчання та сформульована пропозиція з її удосконалення задля розв'язання поставленої задачі. Третій розділ містить у собі опис програмного забезпечення удосконаленої моделі, результати проведених експериментів та висновки. У четвертому розділі було пораховано трудомісткість розробки та економічну ефективність удосконаленої моделі.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Історія предмета дослідження

Упродовж століть науковці з усього світу шукали спосіб відтворювати людське мовлення штучним шляхом. На думку Меттінглі [1], головною мотивацією для дослідження цієї ідеї було бажання досконально зрозуміти, як людям вдається використовувати мовленнєвий апарат для створення зв'язного мовлення. Таким чином, одним з основних засобів наукового дослідження людського мовлення було створення штучних синтезаторів мовлення та збір відповідних даних для їх удосконалення.

Головним принципом роботи синтезаторів мовлення є перетворення вхідного тексту на штучне мовлення. Найважливішими характеристиками таких систем є природність і розбірливість. Природність означає, наскільки результат нагадує людське мовлення, тоді як розбірливість означає, наскільки легко результат можна зрозуміти. В ідеалі система синтезу мовлення повинна створювати таке мовлення, яке одночасно є зрозумілим і природним.

Перші системи, побудовані наприкінці XVIII і на початку XIX століть, були механічними конструкціями, компоненти яких, по суті, утворювали симуляцію людської гортані та голосового апарату. Вони слугували демонстрацією того, що мовлення можна розуміти як фізичну систему. Крім того, скромний успіх цих пристроїв у синтезі штучного мовлення, давав надію, що при адекватному контролі машину можна буде використовувати для «спілкування» з людьми з порушеннями мовлення.

На початку-середині XX століття поява електричних систем спричинила серйозні зміни в синтетичних мовних апаратах. Використання комбінацій електричних елементів дозволило зробити контроль характеристик мовлення більш гнучким. Протягом цього періоду синтезатори мовлення, засновані на

групах фільтрів, резонаторах або лініях зв'язку, разом з розробкою звукового спектрографа використовувалися для дослідження багатьох аспектів відтворення та сприйняття мовлення.

Завдяки цифровим технологіям історія синтезу мовлення знову змінилася. Більше не потрібно було будувати синтезатори як справжні фізичні машини або за допомогою стелажів електричного обладнання. Синтезатор тепер міг бути реалізований як алгоритм з інструкціями для обчислення. Хоча багато цифрових синтезаторів, по суті, є обчислювальними версіями електричних синтезаторів, цифрові алгоритми дозволили внести значні вдосконалення та створювати нові способи синтезу мовлення.

1.2. Актуальність систем синтезу мовлення

У наш час синтез мовлення набув безлічі застосувань. Мабуть, найбільш важливою та корисною сферою застосування є засоби читання та спілкування для людей з обмеженими можливостями. Для читання книжок, до синтезаторів мовлення використовувалися спеціальні аудіокниги, для яких зміст зачитували на аудіокасету. Зрозуміло, що виготовлення такої мовної копії будь-якої великої книги займає багато часу і коштує досить дорого. Завдяки появі синтезаторів, книгу для користувача навчився читати комп’ютер. Також легше стало отримувати інформацію з комп’ютера за допомогою управління голосом, на противагу використанню спеціальної клавіатури бліссимволіки [2], яка є інтерфейсом для читання символів Брайля.

Першою комерційною програмою TTS (англ. Text-to-Speech) була читальна машина Kurzweil [3] для людей з порушенням зору, представлена Раймондом Курцвейлом наприкінці 1970-х років. Вона складалася з оптичного сканера та програмного забезпечення для розпізнавання тексту, і була здатна синтезувати цілком зрозуміле мовлення з написаного багатошрифтового тексту. Ціни на екземпляри цієї машини були занадто високими для пересічного користувача, тому вони використовувалися здебільшого у бібліотеках. Завдяки

технічному прогресу, сьогодні якість таких машин досягла високого рівня, а ціни стали набагато доступнішими.

Синтезоване мовлення дає можливість людям з порушеннями слуху спілкуватися з людьми, які не розуміють мову жестів. Наприклад, відомий науковець Стівен Хокінг через свою важку інвалідність використовував TTS для спілкування з оточуючими. Користувачі таких засобів також часто позбавлені можливості передавати емоції, такі як радість, смуток, злість тощо. Деякі інструменти, такі як HAMLET (англ. Helpful Automatic Machine for Language and Emotional Talk), були розроблені, щоб допомогти користувачам висловлювати свої почуття. HAMLET змінює якість синтезованого мовлення, а також висоту тону та швидкість; разом ці ефекти імітують емоції у голосі.

Синтезоване мовлення використовується також для багатьох навчальних задач. Комп’ютер можна запрограмувати для спеціальних завдань, таких як навчання правопису та вивчення різних мов. Його також можна використовувати з інтерактивними навчальними програмами. Завдяки відповідному програмному забезпеченню, навчання для багатьох дітей з вадами здоров’я може стати легшим і доступнішим. Інший приклад використання синтезу мовлення можна знайти в системах перекладу мови. Вони обладнані цією технологією, щоб запропонувати функціональність вимови перекладеного тексту.

Синтезатор мовлення, підключений до текстового процесора, також є корисним засобом для коректури тексту. Багатьом користувачам легше виявити граматичні та стилістичні проблеми під час прослуховування тексту, аніж читання. Звичайні орфографічні помилки також легше виявити таким чином.

У контексті транспортної галузі, синтезатори мовлення часто використовуються для передачі повідомлень пасажирам, незалежно від того, чи мають вони вади слуху.

Синтез мовлення також широко використовується в області телекомунікацій та мультимедіа. Синтезоване мовлення десятиліттями використовувалося у багатьох видах телефонних довідкових систем, і оскільки

його якість продовжує покращуватись, все більше і більше компаній використовують його для взаємодії з клієнтами. Широке використання голосових асистентів почалося з того, що компанії Google і Apple випустили своїх віртуальних асистентів у 2008 і 2010 роках відповідно. Голосовий асистент – це програмне забезпечення, яке використовує розпізнавання та синтез мовлення, а також алгоритми його обробки для сприйняття певних голосових команд і видачі відповідної інформації або виконання певних функцій за запитом користувача. Сьогодні голосові асистенти інтегровано у безліч пристроїв, які ми використовуємо щодня: смартфони, комп'ютери, музичні колонки, годинники тощо.

Іншим сектором, який інтегрує синтез мовлення у вбудовані або хмарні програми, є досить нова сфера IoT (англ. Internet of Things). Побутові пристрої все частіше обладнують TTS, що підвищує доступність для осіб з особливими потребами та покращує користувацький досвід загалом.

1.3. Методологічні підходи до синтезу мовлення

Далі буде розглянуто актуальні у наш час підходи до синтезу мовлення, їх особливості, переваги та недоліки.

1.3.1. Конкатенативний синтез

Конкатенація (або об'єднання) попередньо записаних фрагментів голосу є основою конкатенативного синтезу мовлення. Загалом, конкатенативний синтез дає найбільш природне за звучанням синтетичне мовлення, якщо мовний корпус є досить великим. Недоліком є те, що всі використовувані мовні сегменти мають бути попередньо записані, що обмежує можливості системи та модифікацій синтезу. Існує три підкатегорії конкатенативного синтезу – синтез

з одиниць довільного розміру, дифонний синтез і синтез, специфічний для галузі.

У синтезі з одиниць довільного розміру використовуються великі набори даних записаного мовлення. Під час процесу створення бази даних кожен записаний аудіофайл розділяється на одну або кілька з наступних категорій: фонеми, дифони, напівфонеми, морфеми, склади, фрази, слова та речення. У базі даних зберігаються кілька екземплярів кожної одиниці з різними просодіями (інтонаціями), взяті з різних контекстів, та дані про їх висоту тону, тривалість, положення в складі тощо. Одиниця, яка найбільше відповідає цільовій специфікації, вибирається та об'єднується з іншими таким чином, щоб модифікації інтонації, необхідні для вибраного блоку, були зведені до мінімуму, або взагалі не потрібні. Оскільки існує кілька екземплярів кожної одиниці, необхідний алгоритм вибору одиниць, що найкраще відповідають цільовій специфікації.

Завдяки великій базі даних, алгоритму вибору одиниць та алгоритму модифікації, результат синтезу часто оцінюється, як близький до натурального.

Водночас цей підхід має ряд недоліків. Він покладається на дуже велику базу даних, що передбачає, з одного боку, значний час розробки та витрати на збір і обробку даних, а з іншого боку, великі вимоги до ресурсів пам'яті для зберігання даних. Другим недоліком є можлива неправильна ручна обробка та поява непередбачуваних цільових контекстів, що знижують якість синтезованого мовлення. Цей феномен непередбачуваних контекстів, можливо, ніколи не буде повністю подоланий за допомогою конкатенативного синтезу, як припускає Бернд Мебіус у своїй науковій публікації [40].

У фонетиці дифон – це сегмент мови між серединами сусідніх фонем. Дифони мають перевагу моделювання коартикуляції, тому що включають перехід до наступної фонеми. Повний список дифонів називається інвентарем дифонів, і після їх визначення їх необхідно виокремити з аудіозаписів мовлення. Для цього, мовлення має бути записано таким чином, щоб були

включені всі фонеми в усіх можливих контекстах, потім дифони повинні бути визначені та сегментовані.

Для синтезу мовлення, у кожного дифона з інвентарю необхідно змінити висоту тону та тривалість, щоб він відповідав специфікації просодії. Далі цільова просодія речення накладається на дифони за допомогою методів цифрової обробки сигналу, таких як кодування з лінійним прогнозуванням, PSOLA або MBROLA або новіші методи, такі як модифікація висоти у вихідній області за допомогою дискретного косинусного перетворення.

Найбільша перевага цього підходу – відносно невелика база даних, адже вона повинна містити лише один приклад кожного дифона. Кількість дифонів залежить від фонотактики мови: наприклад, іспанська має близько 800 дифонів, а німецька – близько 2500. Найбільшим недоліком цього підходу є використання цифрових обробок, через що результат синтезу може бути неприродним та роботизованим.

Іншим типом конкатенативного синтезу є підвид синтезу одиниць – синтез, специфічний для галузі, який об’єднує попередньо записані фрази та слова, щоб створити задані висловлювання. Він використовується, коли результат роботи системи обмежений однією галуззю, наприклад у програмах прогнозу погоди, банківських додатках тощо.

Завдяки використанню специфічних слів, словосполучень та фраз, синтезоване мовлення може мати дуже високу правдоподібність та використовувати мінімальну базу даних. Однак останнє і є недоліком цього підходу, адже він не є універсальним і його не можна застосовувати для генерації будь-яких висловлювань.

1.3.2. Формантний синтез

Формантний синтез не використовує зразки людського мовлення для роботи. Натомість синтезований результат створюється за допомогою адитивного синтезу та акустичної моделі [4]. Такі параметри, як основна

частота, фонація та рівні шуму змінюють, щоб створити форму хвилі штучного мовлення. Цей метод іноді називають синтезом на основі правил; проте багато конкатенативних систем також мають компоненти, засновані на правилах. Багато систем, заснованих на технології формантного синтезу, генерують роботизоване штучне мовлення, яке неможливо сплутати з людським. Проте максимальна природність не завжди є метою системи синтезу мовлення, натомість системи формантного синтезу мають переваги перед конкатенативними системами. Мовлення може бути розбірливим навіть на дуже високій швидкості, уникаючи акустичних збоїв, які часто виникають при конкатенативному синтезі.

Високошвидкісне синтезоване мовлення використовується людьми з вадами зору для швидкої навігації по комп'ютеру. Формантні синтезатори, як правило, займають менше дискового простору, ніж конкатенативні системи, оскільки вони не мають бази даних мовних зразків. Тому їх можна використовувати у вбудованих системах, де пам'ять і потужність мікропроцесора особливо обмежені. Оскільки формантні системи мають повний контроль над усіма аспектами вихідного мовлення, можна виводити найрізноманітніші інтонації, емоції та тони голосу.

1.3.3. Артикуляційний синтез

Артикуляційний синтез створює мовлення шляхом моделювання голосового апарату людини, тому теоретично це найкращий метод створення високоякісного мовлення. На практиці це один із найскладніших методів для реалізації. Параметри артикуляційного контролю включають апертуру губ, виступ губ, позицію кінчика язика, висоту кінчика язика, позицію язика та висоту язика тощо [5]. Попри те, що численні дослідники експериментували з імітацією людського голосового апарату, артикуляційний синтез все ще є найменш дослідженим методом, здебільшого через його складність.

У артикуляційному синтезі є дві труднощі. Перша полягає в отриманні даних для артикуляційної моделі. Ці дані зазвичай отримують з рентгенів. Рентгенологічні дані не характеризують маси чи ступені свободи артикуляторів [6]. Друга складність полягає в тому, щоб знайти баланс між високою точністю моделі та легкістю її проєктування та контролю. Підхід не передбачає великих потреб обчислювальних ресурсів, однак сама розробка є надзвичайно складним процесом і досі не є виправданою якістю результатів. Загалом, якість артикуляційного синтезу не такі висока, як у формантному або конкатенативному синтезі.

1.3.4. Статистичний параметричний синтез

Іншим методом TTS, який є досить розповсюдженим у сфері голосових технологій, є статистичний параметричний синтез мовлення. Він вирішує основне обмеження конкатенативних систем, створюючи мовлення за допомогою статистичних мовних моделей, а не залежно від попередньо записаних одиниць. Ці статистичні моделі створюються за допомогою машинного навчання із мовленнєвого корпусу, що складається з аудіозаписів та відповідних текстових розшифровок.

Одним із найбільш популярних методів статистичного параметричного синтезу є синтез заснований на Прихованій Марковській Моделі (англ. Hidden Markov Model, HMM). Він складається з двох основних фаз: фази навчання та фази синтезу. На етапі навчання слід вирішити, на яких параметрах модель буде тренуватися. Мел-частотні кепстральні коефіцієнти та їх перша та друга похідні є найбільш поширеними типами використовуваних параметрів. Параметри фонем визначаються і додаються у вектор параметрів. Алгоритм Баума-Велча використовується з векторами параметрів для створення моделей для кожної фонеми. Модель зазвичай складається з трьох станів, які представляють початок, середину і кінець фонеми. Фаза синтезу складається з двох кроків: по-перше, необхідно оцінити вектори параметрів для даної

фонемної послідовності. По-друге, використати фільтр для перетворення цих векторів параметрів у звукові сигнали.

Компоненти навчання та синтезу типової системи синтезу мовлення на основі HMM зображені у вигляді блок-схеми на рис. 1.1.

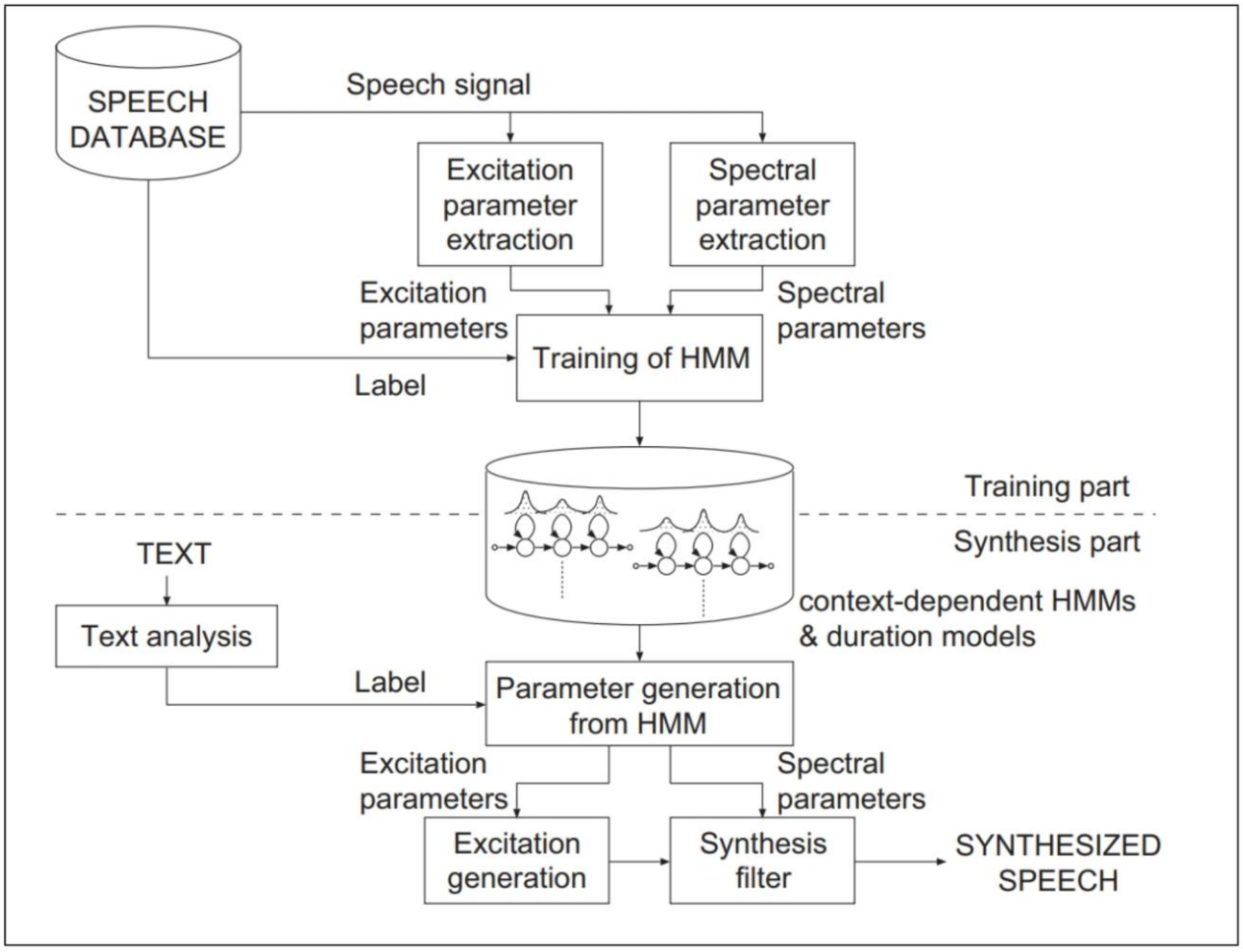

Рис. 1.1. Загальна структура взята з [7].

Переваги генерації на основі HMM включають здатність легко змінювати якість голосу та адаптувати його до різноманітних мов з мінімальними змінами. Також, система займає невелику кількість дискового простору і може використовуватися для синтезу широкого діапазону стилів мовлення та мовлення з емоціями. Якість генерованого мовлення є найбільш істотним недоліком технології синтезу на основі HMM. Через надмірне згладжування, синтезоване мовлення часто звучить неприродно та заглушено.

1.3.5. Синтез, заснований на глибинному навчанні

Глибинне навчання (англ. Deep Learning, DL) – це відносно новий напрям дослідження в області машинного навчання останніх років. Ця технологія може ефективно фіксувати приховані внутрішні структури даних і використовувати потужніші можливості моделювання для характеристики даних. Моделі на основі DL досягли значного прогресу в багатьох областях, таких як розпізнавання рукописного тексту, машинний переклад, розпізнавання мовлення та синтез мовлення тощо.

Метод синтезу на основі DL безпосередньо виконує перетворення лінгвістичних ознак на акустичні за допомогою глибинних нейронних мереж. За небагато років досліджень DL для синтезу мовлення, дослідники вже запропонували численні моделі.

Система TTS зазвичай складається з інтерфейсу аналізу тексту, акустичної моделі та синтезатора мовлення. Оскільки ці компоненти навчаються незалежно і покладаються на великий досвід у предметній області, який є трудомістким, помилки кожного компонента можуть накладатися одна на одну. Щоб вирішити ці проблеми, почали з’являтися методи наскрізного синтезу мовлення, які об’єднують ці компоненти в єдину структуру. Існує багато переваг наскрізної системи TTS:

1. Їх можна навчати на основі великої кількості пар аудіо/текст.
2. Вони не вимагають фонемного вирівнювання.
3. Помилки не можуть накладатися, оскільки це єдина модель.

Далі будуть розглянуті різні методи синтезу мовлення, засновані на глибинному навчанні, включаючи наскрізні системи.

1.3.6. Синтез, заснований на Обмежених Машинах Больцмана

В останні роки для моделювання мовних сигналів у розпізнаванні мовлення, спектрограмному кодуванні та акустико-артикуляційному

інверсійному перетворенні, широко використовуються Обмежені Машини Больцмана (англ. Restrictive Boltzmann Machine, RBM) [13]. У цих програмах RBM часто використовується для попереднього навчання глибинних автокодерів (англ. deep auto-endcoder, DAE) [14,15] або глибинних нейронних мереж. У сфері синтезу мовлення RBM зазвичай розглядають як модель щільності для генерування спектральної оболонки акустичних ознак. Їх використовують для кращого опису розподілу високовимірних спектральних огинаючих, щоб зменшити проблему надмірного згладжування в синтезі мовлення на основі HMM [13]. Після навчання HMM виконується вирівнювання стану для акустичних ознак, а межі стану використовуються для збору спектральних огинаючих, отриманих з кожного стану. Параметри RBM оцінюються з використанням критерію оцінки максимальної правдоподібності (англ. maximum likelihood estimation, MLE). Нарешті, RBM–HMM будуються для моделювання спектральних огинаючих. На фазі синтезу оптимальна послідовність спектральної огинаючої оцінюється на основі вхідного тексту та навчених RBM–HMM. Хоча суб'єктивний результат оцінки цього методу кращий, ніж у традиційних систем HMM, а прогнозована спектральна оболонка ближча до вихідної, цей метод все ще не може вирішити проблему фрагментації навчальних даних, що зустрічається в традиційному методі на основі HMM.

1.3.7. Синтез, заснований на Глибинних Мережах Переконань

Глибинна мережа переконань (англ. Deep Belief Networks, DBN) [16] – це метод моделювання спільного розподілу контекстної інформації та акустичних ознак. Він моделює безперервні спектральні, дискретні звукові/незвукові (V/UV) параметри і основну частоту (F0) одночасно з трьома типами RBM.

Перевага цього методу полягає в тому, що всі склади слів навчаються в одній мережі, а всі дані використовуються для навчання однієї RBM або DBN.

Тому він не може мати проблему фрагментації навчальних даних, але зменшує проблему надмірної плавності. Однак, оскільки цей метод не розрізняє склади в різних контекстах, він усе одно усереднює акустичні параметри, що відповідають одному складу. Крім того, передбачені F0 містять багато шуму, що знижує якість синтезованого мовлення**.**

1.3.8. Синтез, заснований на Глибинних Мережах Щільності Суміші

Хоча моделі синтезу мовлення на основі глибинних нейронних мереж можуть синтезувати мовлення з високою природністю, вони все ще мають деякі обмеження для моделювання параметрів акустичних ознак, наприклад, єдину модальність цільової функції та нездатність передбачити дисперсію. Для вирішення цих проблем Геіґа Зен та Ендрю Сеніор в [17] запропонували метод прогнозування параметрів, заснований на глибинній мережі щільності суміші (англ. Deep Mixture Density, DMD), яка використовує вихідний шар щільності суміші для прогнозування розподілу ймовірностей вихідних ознак за заданими вхідними ознаками.

Мережі щільності суміші [45] можуть не тільки відображати вхідні характеристики параметрів моделі суміші Гаусса (англ. Gaussian Mixture Model, GMM), такі як вагу суміші, середнє значення та дисперсію, але також дають спільну функцію щільності ймовірності y для заданих вхідних характеристик x. Функція щільності спільної ймовірності виражається так:

$$p(y|x,M) = \sum_{m=1}^{M} w_m(x) \cdot N(y;\, \mu_m(x), \sigma_m^2(x)), \qquad (1.1)$$

де M – кількість компонентів суміші, а $w_m(x)$, $\mu_m(x)$ та $\sigma_m^2(x)$ – вага суміші, середнє значення та дисперсія m-го компонента GMM відповідно.

Під час передбачення параметрів мовлення за допомогою глибинної мережі щільності суміші вхідний текст спочатку перетворюється в послідовність лінгвістичних ознак $\{x_1, x_2, \dots, x_T\}$, а потім тривалість кожної

мовної одиниці прогнозується за допомогою моделі передбачення тривалості. Акустичні ознаки, включно з F0, спектральні параметри та відповідні їм динамічні характеристики, оцінюються за допомогою послідовного алгоритму та навченої глибинної мережі. Нарешті, параметри акустичних ознак генеруються за допомогою алгоритму генерації параметрів, а мова синтезується за допомогою вокодера.

Попри те, що ця модель синтезу мовлення може розв'язати проблему єдиної модальності цільової функції та точно передбачити параметри акустичних властивостей, щоб покращити природність синтезованого мовлення, все ще існують деякі проблеми. По-перше, MDN може використовувати лише обмежену контекстну інформацію, оскільки вона може моделювати лише фіксований проміжок часу (наприклад, фіксовану кількість попередніх або наступних контекстів) для вхідних функцій. По-друге, модель може виконувати лише покадрове перетворення (кожен кадр перетворюється незалежно).

1.3.9. Синтез, заснований на Рекурентних Нейронних Мережах

Для вирішення проблем моделі на основі глибинних мереж щільності суміші Алех Ґравес в статті [18] запропонував метод моделювання на основі рекурентних нейронних мереж (англ. Recurrent Neural Network, RNN). Перевагою RNN є можливість використовувати контекстну інформацію при зіставленні вхідних і вихідних даних. Однак традиційні RNN можуть отримати доступ лише до обмеженої контекстної інформації. Крім того, цей алгоритм також не може навчатися довгострокових залежностей.

Для вирішення цих проблем було введено секцію пам’яті та запропонували [19] модель довгої короткочасної пам’яті (англ. Long Short-term Memory, LSTM). Для повного використання контекстної інформації двонапрямний LSTM [20] здебільшого використовується для перетворення вхідних лінгвістичних ознак на акустичні.

BLSTM-RNN – це розширена архітектура двонапрямний рекурентної нейронної мережі (англ. Bidirectional Recurrent Neural Network, BRNN) [21]. Вона замінює елементи в прихованих шарах BRNN на блоки пам'яті LSTM. За допомогою цих блоків пам’яті BLSTM може зберігати інформацію протягом тривалого і короткого часу, а також використовувати відповідні контекстні залежності як у прямому, так і у зворотному напрямках для завдань машинного навчання. Завдяки прямому і зворотному шарам BLSTM може використовувати як минулу, так і майбутню інформацію для моделювання.

При використанні глибинної моделі на основі BLSTM для прогнозування акустичних параметрів, спочатку треба перетворити вхідний текст у вектор ознак, а потім використовувати модель глибинної BLSTM, щоб перетворити вхідні ознаки на акустичні параметри. Нарешті, для генерування акустичних параметрів використовується алгоритм генерації параметрів, а для синтезу мовлення використовується вокодер. Наприклад, Руннан Лі у статті [20] запропонували багатозадачне навчання моделі BLSTM структурованого вихідного рівня для синтезу мовлення, яка здатна збалансувати функції вартості помилок, пов’язані зі спектральними характеристиками та параметрами висоти звуку.

Найбільша перевага BLSTM-RNN – можливість використання повної контекстної інформації. З недоліків методу можна виділити необхідність окремого використання вокодера.

Багатообіцяючих результатів досягли нейронні мережі sequence-to-sequence (seq2seq), що можуть перетворювати вхідну послідовність у вихідну, яка може мати різну довжину та застосовуватися для різних завдань, таких як машинний переклад, розпізнавання мовлення та розпізнавання зображень. Це робиться за допомогою RNN, LSTM або GRU (англ. Gated Recurrent Unit), щоб уникнути проблеми зникнення градієнту. Контекст для кожного елемента є результатом попереднього кроку. Основними компонентами є один кодер і один декодер мережі. Кодер перетворює кожен елемент у відповідний прихований вектор, що містить елемент і його контекст. Декодер змінює процес

у зворотний бік, перетворюючи вектор на вихідний елемент, використовуючи попередній результат як вхідний контекст.

У науковій роботі [24] використали seq2seq структуру із механізмом уваги [25] для моделювання акустичних ознак синтезу мовлення. Char2Wav [26] використовує увагу на основі місцеположення для побудови акустичної моделі кодера-декодера. Щоб вирішити проблему нестабільності відсутніх або повторюваних фонем, від яких досі страждають нинішні моделі seq2seq, в [27] запропонували підхід до акустичного моделювання синтезу мовлення seq2seq. Tacotron, який також є моделлю seq2seq з механізмом уваги, був запропонований для перетворення вхідного тексту на мел-спектрограму для синтезу мовлення.

Мел-спектрограма пов’язана зі спектрограмою лінійної частоти, тобто магнітудою віконного перетворення Фур’є (англ. Short-time Fourier Transform, STFT). Її отримують шляхом застосування нелінійного перетворення до осі частоти STFT, створеного на основі виміряних реакцій слухової системи людини, і підсумовування частотного вмісту з меншими розмірами. Приклад мел-частотної спектрограми взято з [10] і зображено на рис. 1.2.

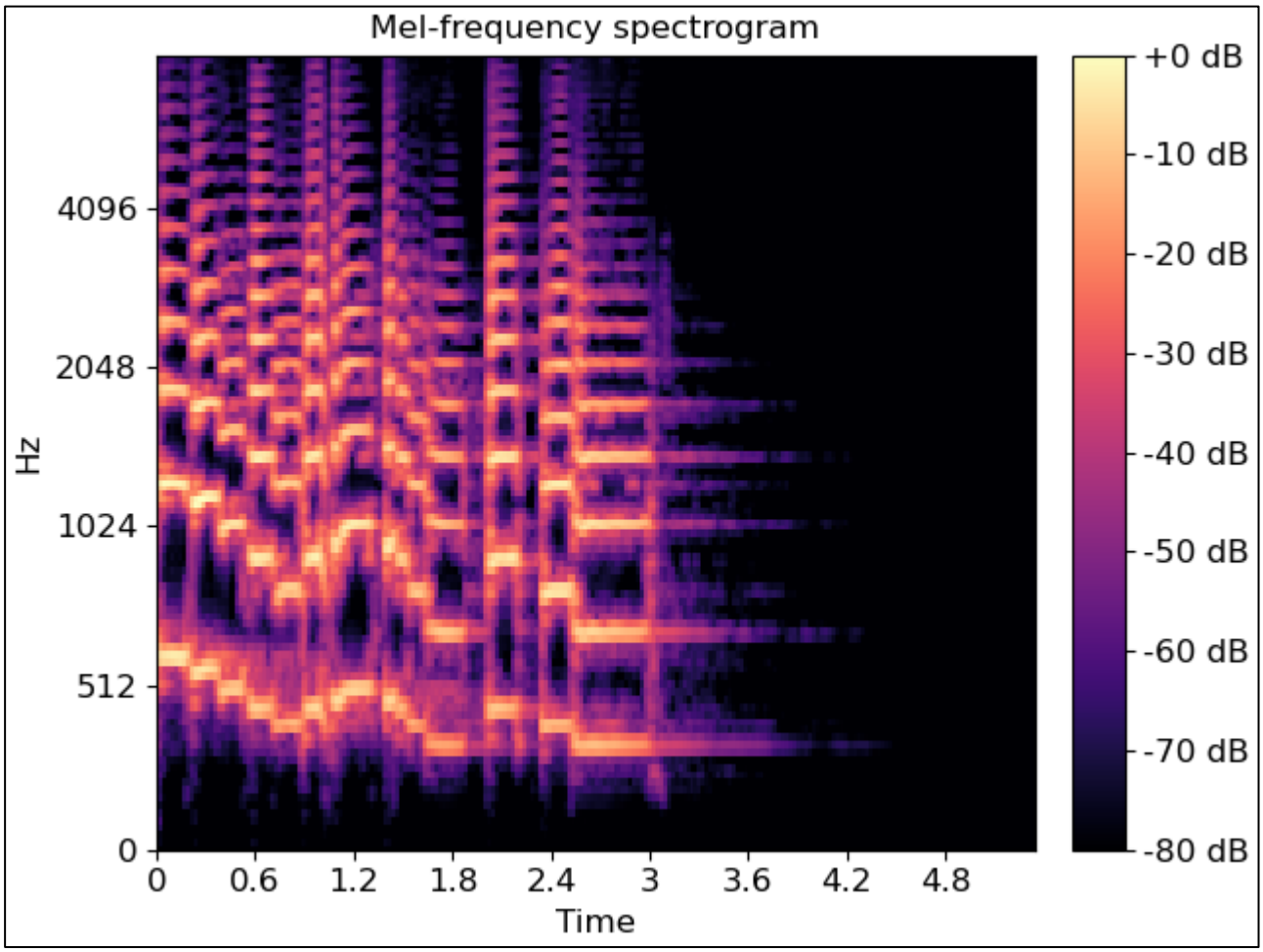

Рис. 1.2. Приклад мел-частотної спектрограми

Використання такої звукової частотної шкали робить акцент на деталях на нижчих частотах, які є важливими для розбірливості мовлення, одночасно зменшуючи акцент на високочастотних деталях, у яких переважають фрикативи та інші шумові сплески і які, як правило, не потрібно моделювати з високою точністю. Через ці властивості, ознаки, отримані від мел-частотної шкали, використовувалися як базове представлення для розпізнавання мови протягом багатьох десятиліть [62].

Tacotron [33] є повністю наскрізною моделлю синтезу мовлення. Вона здатна навчати модель синтезу мовлення за допомогою пар текст-аудіо, таким чином усуваючи потребу в кропіткій розробці ознак. Крім того, оскільки вона застосовується на рівні символів, її можна використовувати майже в усіх мовах, включаючи китайську.

Оскільки Tacotron є повністю наскрізною моделлю, яка безпосередньо відображає вхідний текст у мел-спектрограму, вона привернула велику увагу дослідників, і були запропоновані різні вдосконалені версії. Наприклад, деякі дослідники реалізували Tacotron у проєктах з відкритим кодом [36–38], щоб відтворити мовлення задовільної якості, настільки ж чітке, як і оригінальна робота.

Є також деякі роботи, які поєднують Tacotron і WaveNet [28] для синтезу мовлення, наприклад Deep Voice 2 [41]. У цій системі Tacotron використовується для перетворення вхідного тексту в лінійну спектрограму, тоді як WaveNet використовується як вокодер. Крім того, в [42] також запропонували систему Tacotron2 для генерування аудіосигналів, що призвело до дуже високого середнього бала MOS, порівнянного з людською мовою.

Найбільша перевага методу – дуже високоякісні результати синтезу та використання наскрізного підходу. Найбільший недолік методу – велика ресурсомісткість для тренування моделі та синтезу мовлення.

1.3.10. Синтез, заснований на Згорткових Нейронних Мережах

Хоча наскрізна система на базі Tacotron досягла перспективної продуктивності, вона все ще має недолік – велику кількість повторюваних елементів. Така структура робить навчання моделі досить ресурсомістким, а також унеможливлює її подальші дослідження для людей без високопродуктивних комп’ютерів.

Для вирішення цієї проблеми запропоновано чимало робіт.

WaveNet [28], яка розвинена з моделей PixelCNN [29] або PixelRNN [30], що використовуються в області генерації зображень, є потужною генеративною моделлю необроблених звукових сигналів. Вона була запропонована компанією Deepmind у 2016 році і поклала початок методам наскрізного синтезу мовлення.

Вона здатна генерувати відносно реалістичні людські голоси шляхом безпосереднього моделювання сигналів за допомогою глибинної нейронної мережі, яка тренується із записами реального мовлення. Це повна імовірнісна авторегресивна модель, яка прогнозує розподіл ймовірностей поточного аудіо на основі всіх вибірок, які були згенеровані раніше.

Як і інші моделі синтезу мовлення, модель на основі WaveNet можна розділити на фазу навчання та фазу генерації. На етапі навчання вхідні послідовності є реальними аудіозаписами. На етапі генерації мережа випробовується для створення синтезованого мовлення.

Хоча модель WaveNet може створювати високоякісні аудіо, вона все ще страждає від таких проблем:

1. Вона надто повільна, оскільки передбачення кожної точки вибірки завжди залежить від попередньо передбачених точок вибірки.
2. Вона також залежить від лінгвістичних особливостей існуючого інтерфейсу TTS, і помилки від аналізу тексту безпосередньо впливатимуть на результат синтезу.

Для вирішення цих проблем була запропонована модель паралельного WaveNet для підвищення ефективності синтезу. Вона здатна генерувати високоякісні зразки мовлення більш ніж у 20 разів швидше [31]. Інша нейронна модель, Deep Voice [32], також запропонована, щоб замінити кожен компонент, включаючи інтерфейс аналізу тексту, акустичну модель і синтезатор мовлення, відповідною нейронною мережею.

Гідейукі Тацгібана та ін. у [44] запропонували DCTTS – модель, засновану на глибинних згорткових нейронних мережах (англ. convolutional neural network, CNN) з керованою увагою, яку можна тренувати набагато швидше, ніж найсучаснішу систему на основі RNN, але на жаль вона не надає якісних результатів синтезу.

Повністю згорткову модель Deep Voice 3 для синтезу мовлення запропонували у роботі [46], і вона дає змогу виконувати паралельні обчислення, щоб зробити процес навчання набагато швидшим, ніж при

використанні повторюваних елементів. Схожу модель SpeedySpeech представили у [47] – це повністю згорткова модель синтезу мовлення, яка складається з мережі-вчителя та мережі-учня. Мережа-вчитель є авторегресивною згортковою мережею, яка використовується для вилучення правильних відповідностей між фонемами та відповідними аудіокадрами. Мережа-учень є неавторегресивною, повністю згортковою мережею, яка кодує вхідні фонеми, прогнозує тривалість (кількість необхідних аудіо кадрів) для кожної з них, а потім декодує мел-спектрограму на основі закодованих фонем і прогнозованої тривалості. Мережа-учень поєднана з попередньо підготовленим вокодером MelGAN [49] для досягнення швидкої та якісної інверсії спектрограми.

Хоча моделі повністю засновані на згорткових мережах надають досить швидке тренування моделі, тим не менш, якість синтезованого ними мовлення є зазвичай набагато нижчою, порівняно з іншими наскрізними моделями.

1.3.11. Синтез, заснований на Нейронних Мережах Прямого Поширення

Беручи до уваги монотонне вирівнювання між текстом і мовленням, щоб прискорити генерацію мел-спектрограми, у роботі [48] було запроповано нову модель FastSpeech, яка бере текстову послідовність як вхідну інформацію та генерує мел-спектрограми без авторегресії. Архітектура FastSpeech – це структура з прямим поширенням, заснована на моделі Transformer [52] та одновимірній згортці. Автори назвали цю структуру трансформером прямого поширення (англ. Feed-Forward Transformer, FFT).

Оскільки послідовність мел-спектрограми набагато довша за відповідну послідовність фонем, щоб вирішити проблему невідповідності довжини між двома послідовностями, FastSpeech використовує регулятор довжини, який розширює послідовність вхідних фонем відповідно до довжини вихідної

послідовності мел-спектрограм. Регулятор побудований на предикторі тривалості фонеми, який передбачає тривалість кожної фонеми.

Запропонована модель FastSpeech може вирішити три розглянуті у дослідженні проблеми:

- Завдяки паралельному створенню мел-спектрограм FastSpeech значно прискорює процес синтезу.
- Прогноз тривалості фонеми забезпечує жорстке вирівнювання між фонемою та її мел-спектрограмами, що дуже відрізняється від м'якого та автоматичного вирівнювання уваги в моделях авторегресії. Таким чином, FastSpeech уникає проблем поширення помилок і неправильного вирівнювання уваги, отже, зменшуючи кількість пропущених і повторюваних слів.
- Регулятор довжини може легко регулювати швидкість голосу, подовжуючи або скорочуючи тривалість фонеми, щоб визначити довжину генерованих мел-спектрограм, а також може керувати частиною просодії, додаючи розриви між сусідніми фонемами.

1.3.12. Синтез, заснований на Потокових Генеративних Моделях

Потоковим генеративним моделям (англ. Flow-based generative models) приділяють велику увагу через їх переваги [53, 54]. Вони можуть оцінити точну ймовірність даних, застосовуючи зворотні перетворення. Генеративні потоки просто тренувати, щоб максимізувати ймовірність. Крім ефективної оцінки щільності, алгоритми, запропоновані в [57, 58], гарантують швидкий та ефективний відбір вибірок.

Glow-TTS [58, 59] – це потокова генеративна модель, яка навчається з оцінкою максимальної правдоподібності. Оскільки запропонована модель сама по собі знаходить найбільш вірогідне монотонне вирівнювання між текстом і прихованою репрезентацією мовлення, вся процедура навчання спрощується без необхідності зовнішніх вирівнювачів. У статті [59] продемонстрували, що

Glow-TTS синтезує мел-спектрограми в 15.7 разів швидше, ніж авторегресивний Tacotron2, демонструючи при цьому схожу якість синтезу.

1.4. Проблема обчислювальних потужностей у синтезі мовлення

Завдяки прогресу, процесори стають все швидшими та потужнішими, а запам'ятовувальні пристрої місткішими. Однак, із прогресом також з’являються складніші алгоритми та моделі. У випадку з синтезом мовлення, заснованим на нейронних мережах, частою проблемою стає час, необхідний для тренування моделі. За умови підключення сучасного та потужного графічного процесора, для деяких моделей тренування може займати від декількох годин до декількох днів. Це значно знижує доступність до використання таких моделей, простоту подальших досліджень. У випадку з методами конкатенативного синтезу та іншими, нема потреби у потужному процесорі, адже результат синтезу залежатиме від самої розробки алгоритмів та підготовки даних. Такі моделі не розглядатимуться далі, адже вважаються застарілими та неперспективними.

Щодо пам’яті, необхідної для алгоритмів синтезу мовлення, ситуація дещо інша. Алгоритми, засновані на глибинних мережах не потребують багато постійної пам’яті. Щодо оперативної пам’яті, в залежності від параметрів навчання, зазвичай вистачає навіть 12 Гб для тренування моделі.

Наостанок, чи не найважливішим ресурсом для синтезу мовлення є використовуваний мовний корпус (для моделей що його потребують). У випадку з методами, заснованими на глибинних мережах, постає необхідність у великій кількості ретельно підготовлених аудіофайлів з текстовими розшифровками. Часто за основу використовуються аудіокниги. Для англійської мови найпопулярнішим набором даних є LJ Speech Dataset [60]. Для української мови на момент написання роботи існує лише M-AILABS [61] з даними взятими з різних аудіокниг начитаних професійними акторами озвучування. Сумарно, мовний корпус складає 87 годин, однак для розглянутих

алгоритмів треба використовувати лише один голос, що значно зменшує кількість доступних аудіофайлів. Докладна структура мовного корпусу M-AILABS наведена у таблиці 1.1.

Таблиця 1.1

Структура мовного корпусу M-AILABS

Стать	Автор озвучування	Тривалість	Загальна тривалість
Жіноча	Ольга Сумська	10:28	10:28
Чоловіча	Борис Лобода	19:02	76:40
	Михайло Міскун	11:26	
	Василь Обручов	24:29	
	Владислав Писарєв	02:50	
	Олександр Шепель	18:53	

1.5. Україномовні синтезатори мовлення

До середини 2016 року, існували напівпрофесійні TTS системи з підтримкою української мови:

1. Розмовлялка [39] (1 чоловічий голос, 1 жіночий).
2. CyberMova/VymovaPlus/VymovaPro [35] (3 чоловічі голоси, 1 жіночий).
3. UkrVox [23] (1 чоловічий голос).
4. RHVoice [22] (1 чоловічий голос, 1 жіночий).
5. Google TTS [12].
6. Yandex TTS [68].

У 2016-2019 роках з'явилася перша професійна система українського мовлення на основі Google WaveNet Text-to-Speech [12]: спочатку 2016 року

Google додала україномовний голос лише до вебверсії Google Translate, 5 квітня 2017 року Google також оновила свій офіційний застосунок Google Text-to-Speech для Android до версії 3.11.1, додавши підтримку україномовного WaveNet жіночого голосу, а 21 лютого 2019 року Google також додала україномовний голос до вебверсії Google Cloud Text-to-Speech.

У вересні 2018 року, разом з оновленням для Nuance Vocalizer TTS [11] для Android до версії 3.1.7, з'явилася друга професійна система української мови Text-to-Speech під назвою Lesya розроблена компанією Cerence/Nuance. Цей новий український голос Lesya також доступний у TTS продуктах незалежних пере-постачальників, як от Code Factory, NextUp, KobaVision/KobaSpeech тощо.

Серед розглянутих україномовних синтезаторів, лише RHVoice має відкритий код і він реалізує метод статистичного параметричного синтезу.

1.6. Постановка задачі

Отже, в результаті розгляду різних підходів до синтезу мовлення, стає зрозумілим, що серед усіх методів синтезу мовлення, найбільш актуальним та досліджуваним на цей момент є процес синтезу мовлення за допомогою глибинних нейронних мереж. У даній кваліфікаційній роботі необхідно вирішити наступні задачі:

1. Обрати та детально розглянути метод синтезу мовлення, заснований на глибинному навчанні.
2. Запропонувати удосконалення цієї моделі задля зменшення кількість необхідних обчислювальних ресурсів на навчання та синтез мовлення.
3. Створити програмне забезпечення вдосконаленої моделі.
4. Провести навчання оригінальної моделі, удосконаленої моделі та кількох інших, для подальшого порівняння у ході експериментів.

5. Провести оцінювання якості синтезованого мовлення, кількості помилок у випадку зі складними реченнями, підрахувати кількість витрачених ресурсів.
6. Обробити отримані дані, візуалізувати їх за допомогою таблиць та графіків та зробити висновки.

1.7. Висновки

Було розглянуто наступні методи синтезу мовлення: конкатенативний, формантний, артикуляційний, статистичний параметричний та синтез заснований на глибинному навчанні. Були розглянуті основні актуальні проблеми синтезу мовлення, наявні україномовні синтезатори мовлення та мовленнєві корпуси. Завдяки детальному огляду цих тем, було поставлено задачу кваліфікаційної роботи.

РОЗДІЛ 2
УДОСКОНАЛЕННЯ МОДЕЛІ FASTSPEECH

2.1. Вибір моделі

Серед розглянутих методів і моделей синтезу мовлення для удосконалення була обрана модель, заснована на Нейронних Мережах Прямого Поширення – FastSpeech. FastSpeech було вперше представлено у науковій статті “FastSpeech: Fast, Robust and Controllable Text to Speech” [48] у 2019 році. Розгляньмо детальніше її архітектуру та ефективність роботи, з огляду на обчислювальні витрати, а потім опишемо її удосконалення.

2.2. Архітектура моделі FastSpeech

Архітектура FastSpeech – це структура прямого поширення, заснована на механізмі самоуваги в Transformer [52] та одновимірній згортковій мережі [46, 34]. Автори назвали цю структуру трансформер прямого поширення (англ. Feed-Forward Transformer, FFT), як показано на рис. 2.1. Трансформер прямого поширення об’єднує кілька блоків FFT для перетворення фонеми в спектрограму з N блоків на стороні фонеми та N блоків на стороні мел-спектрограми з регулятором довжини між ними, щоб подолати розрив у довжині між фонемою та послідовністю мел-спектрограми. Кожен блок FFT складається з самоуваги та одновимірної згорткової мережі. Мережа самоуваги складається з багатоголової уваги для вилучення інформації про перехресне положення. На відміну від 2-шарової мережі щільності в Transformer, автори використали 2-шарову одновимірну згорткову мережу з активацією ReLU. Після Transformer залишкові зв’язки, нормалізація шару та виключення додаються після мережі самоуваги та одновимірної згорткової мережі відповідно.

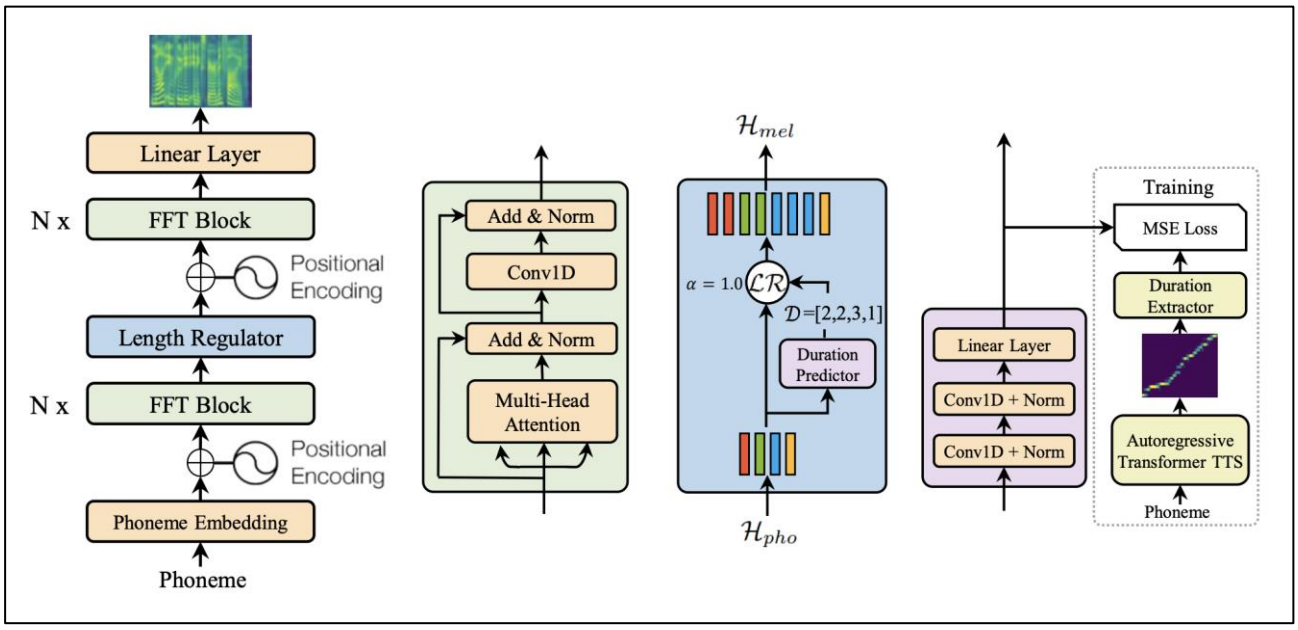

Рис. 2.1. Загальна архітектура FastSpeech: трансформер прямого поширення, блок FFT, регулятор довжини, предиктор тривалості.

Регулятор довжини використовується для вирішення проблеми невідповідності довжини між фонемою та послідовністю спектрограми у FFT, а також для управління швидкістю голосу та частиною просодії.

Передбачення тривалості фонеми важливе для регулятора довжини. Як показано на рис. 2.1, предиктор тривалості складається з 2-шарової одновимірної згорткової мережі з активацією ReLU, за кожним шаром якої слідують нормалізація та шар виключення, а також додатковий лінійний шар для виведення скаляра, який є точно передбаченою тривалістю фонеми. Важливо зазначити, що цей модуль укладено поверх блоків FFT на стороні фонеми і він тренується разом з моделлю FastSpeech, щоб передбачати довжину мел-спектрограм для кожної фонеми з втратою середньоквадратичної похибки. Навчений предиктор тривалості використовується лише на етапі синтезу, оскільки можна безпосередньо використовувати тривалість фонеми, витягнуту з авторегресивної моделі-вчителя під час навчання.

2.3. Ефективність моделі FastSpeech

У науковій статті [60] проводилися експерименти з набором даних LJSpeech, який містить 13 100 аудіозаписів англійською мовою та відповідних текстових транскриптів із загальною тривалістю аудіо приблизно 24 години. Набір даних було випадковим чином розділено на 3 набори: 12500 зразків для навчання, 300 зразків для тестування та 300 зразків для оцінювання. Щоб оцінити надійність FastSpeech, також було обрано 50 речень, які є особливо важкими для системи TTS, дотримуючись практики зі статті [46].

Спочатку було навчено авторегресивну модель Transformer TTS на 4 графічних процесорах NVIDIA V100 з пакетом по 16 речень на кожному GPU. Для оптимізатора Adam було обрано параметри $\beta_1 = 0{,}9$, $\beta_2 = 0{,}98$, $\varepsilon = 10^{-9}$ і використано той же графік швидкості навчання, що й у статті [52]. У навчанні знадобилось 80 тисяч кроків до зближення. Для кожної послідовності тексту автори згенерували мел-спектрограми за допомогою авторегресивної моделі Transformer TTS і взяли вихідний текст і згенеровані мел-спектрограми як парні дані для навчання моделі FastSpeech.

Далі автори навчали модель FastSpeech разом із предиктором тривалості. Оптимізатор та інші гіперпараметри для FastSpeech були використані такі ж, як і в авторегресивній моделі Transformer TTS. Навчання моделі FastSpeech зайняло близько 80 тисяч кроків на 4 графічних процесорах NVIDIA V100. У процесі виведення вихідні мел-спектрограми моделі FastSpeech перетворюються на аудіо зразки за допомогою попередньо натренованого вокодера WaveGlow [67].

Було проведено оцінювання за допомогою суб’єктивної середньої оцінки MOS на тестовому наборі, щоб оцінити якість звуку. Кожне аудіо прослуховували щонайменше 20 тестувальників, які є носіями англійської мови. Автори порівняли MOS згенерованих аудіо зразків за моделлю FastSpeech з іншими системами, які включають:

1. Ground truth – справжній запис мовлення.
2. Ground Truth (Mel + WaveGlow) – справжній запис мовлення, де спочатку було перетворено аудіо в мел-спектрограми, а потім перетворено мел-спектрограми в аудіо за допомогою WaveGlow.
3. Tacotron2 [42] (Mel + WaveGlow).
4. Transofrmer TTS [66] (Mel + WaveGlow).
5. Merlin [65] (WORLD), популярна параметрична система TTS з WORLD [64] у якості вокодера.

Результати наведені на рис. 2.2. Можна побачити, що FastSpeech наближується до якості моделей Transformer TTS і Tacotron2.

Method	MOS
GT	4.41 ± 0.08
GT (Mel + WaveGlow)	4.00 ± 0.09
Tacotron 2 [22] (Mel + WaveGlow)	3.86 ± 0.09
Merlin [28] (WORLD)	2.40 ± 0.13
Transformer TTS [14] (Mel + WaveGlow)	3.88 ± 0.09
FastSpeech (Mel + WaveGlow)	3.84 ± 0.08

Рис. 2.2. Оцінка MOS з 95% довірчими інтервалами.

Також було оцінено швидкість синтезу FastSpeech в порівнянні з авторегресивною моделлю Transformer TTS, яка має аналогічну кількість параметрів з FastSpeech. Дані прискорення генерації мел-спектрограми наведено на рис. 2.3. Можна побачити, що FastSpeech прискорює генерацію мел-спектрограми у 269 разів у порівнянні з моделлю Transformer TTS. Також наведено швидкість синтезу моделей з WaveGlow у якості вокодера. Можна помітити, що FastSpeech все ще може досягти 38-кратного прискорення генерації звуку.

Method	Latency (s)	Speedup
Transformer TTS [14] (Mel)	6.735 ± 3.969	/
FastSpeech (Mel)	0.025 ± 0.005	$269.40\times$
Transformer TTS [14] (Mel + WaveGlow)	6.895 ± 3.969	/
FastSpeech (Mel + WaveGlow)	0.180 ± 0.078	$38.30\times$

Рис. 2.3. Порівняння часу синтезу з 95% довірчими інтервалами.

Було також візуалізовано зв'язок між затримкою виводу та довжиною передбаченої послідовності мел-спектрограми в тестовому наборі. На рис. 2.4. показано, що затримка виводу майже не збільшується з довжиною передбачуваної мел-спектрограми для FastSpeech, тоді як вона значно збільшується в Transformer TTS. Це вказує на те, що швидкість виводу FastSpeech не чутлива до довжини згенерованого аудіо завдяки паралельному генеруванню.

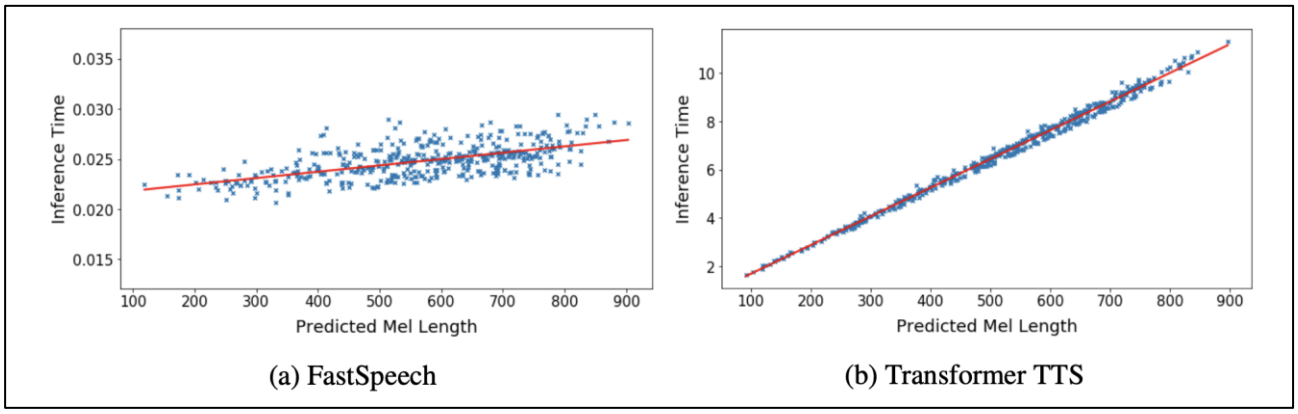

Рис. 2.4. Час синтезу (в секундах) проти довжини мел-спектрограм: (a) для FastSpeech, (b) для Transformer TTS

Щоб оцінити надійність FastSpeech, автори обрали 50 речень, які є особливо важкими для системи TTS. Кількість помилок наведено на рис. 2.5. Можна помітити, що Transformer TTS не стійкий до цих важких випадків і

отримує 34% помилок, тоді як FastSpeech може ефективно виключити повторення та пропуск слів для покращення розбірливості.

Method	Repeats	Skips	Error Sentences	Error Rate
Tacotron 2	4	11	12	24%
Transformer TTS	7	15	17	34%
FastSpeech	0	0	0	0%

Рис. 2.5. Порівняння кількості помилок між FastSpeech та іншими системами на 50 важких реченнях.

2.4. Запропонована модель FastSpeech1.1.

Була запропонована нова модель FastSpeech1.1, що є покращенням оригінальної моделі FastSpeech, для того, щоб прискорити швидкість її навчання.

Структура моделі “вчитель-учень” була видалена у предикторі тривалостей, і мел-спектрограми справжніх аудіозаписів були безпосередньо використані як ціль для навчання моделі, таким чином можна уникнути втрати інформації в мел-спектрограмах і підвищити верхню межу якості синтезованого голосу. Оновлена схема архітектури моделі зображена на рис. 2.6.

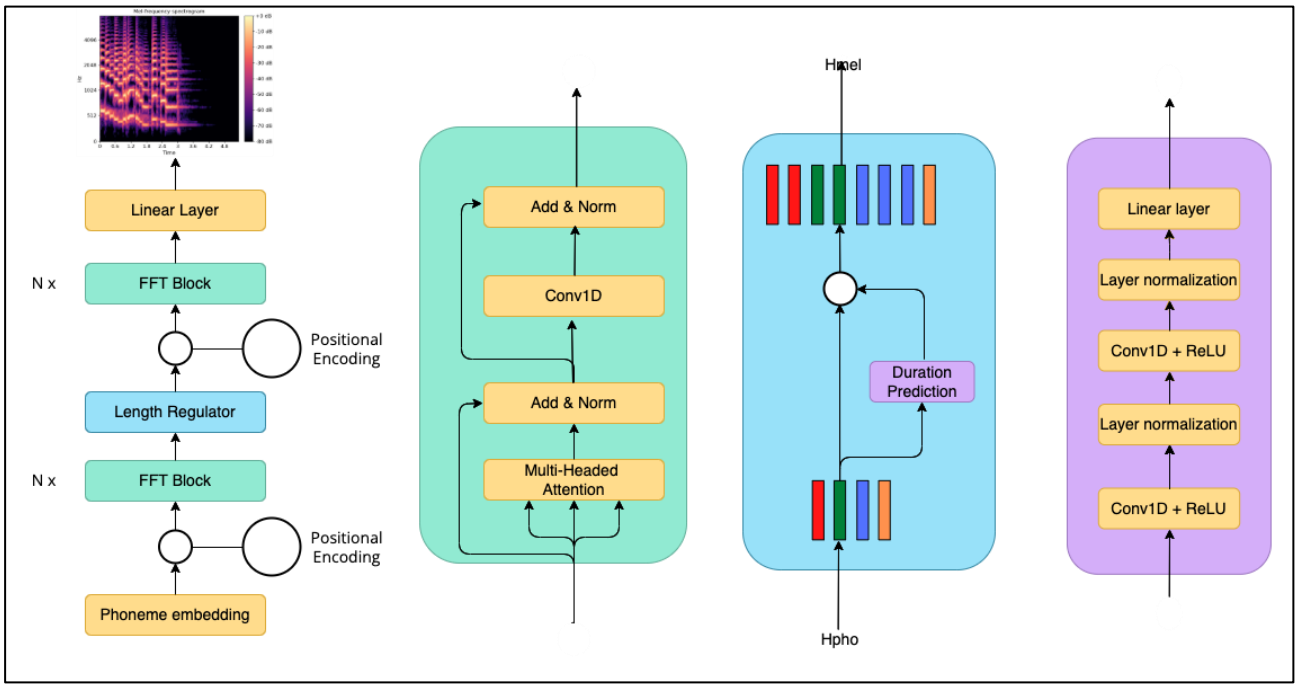

Рис. 2.6. Запропонована модель FastSpeech1.1: трансформер прямого поширення, блок FFT, регулятор довжини, предиктор тривалості.

Предиктор тривалості удосконаленої моделі використовує тривалість фонеми, отриману шляхом примусового вирівнювання [8], як ціль для тренування, яка є більш точною, ніж отримана з асоціативного масива уваги авторегресивної моделі-вчителя, як підтверджено експериментально.

Як показано на рис. 2.6, предиктор тривалості має оновлену структуру моделі (але різні параметри моделі), яка складається з 2-шарової одновимірної згорткової мережі з активацією ReLU, за кожним з яких слідують нормалізація шару, і додатковий лінійний шар для перетворення прихованих станів у вихідну послідовність.

Предиктор тривалості приймає приховану послідовність фонеми як вхідні дані та прогнозує тривалість кожної фонеми, яка відповідає кількості кадрів мел-спектрограми. Предиктор тривалості оптимізовано з урахуванням втрат середньоквадратичної похибки, беручи витягнуту тривалість як цільове значення для навчання. Замість того, щоб витягувати тривалість фонеми за допомогою попередньо навченої авторегресивної моделі у FastSpeech, буде

використано інструмент MFA (англ. Montreal forced alignment) [8] для вилучення тривалості фонеми, щоб покращити точність вирівнювання і таким чином зменшити інформаційний розрив між входом і виходом моделі.

2.5. Висновки

Було розглянуто модель синтезу мовлення FastSpeech, засновану на глибинному мовленні, її архітектуру та показники ефективності. Також було запропоновано удосконалену модель FastSpeech1.1 зі змінами у предикторі тривалості, які повинні надати покращення швидкості синтезу мовлення. Результати розробки та оцінювання якості запропонованої моделі представлені у розділі 3 цієї кваліфікаційної роботи.

РОЗДІЛ 3
ПРОВЕДЕННЯ ЕКСПЕРИМЕНТІВ З УДОСКОНАЛЕНОЮ МОДЕЛЛЮ FASTSPEECH1.1

3.1. Опис класів розробленої моделі FastSpeech1.1

Для розробки моделі була обрана мова програмування Python, а у якості середовища розробки – PyCharm.

Модель містить наступні класи:

1. FastSpeech1_1 – клас запропонованої моделі FastSpeech1.1.
2. LengthRegulatorModule – клас модуля регулятора довжини.
3. DurationPredictorModule – клас модуля предиктора тривалості.
4. BatchNormConv1dModule – клас модуля одновимірного згорткового шару та пакетної нормалізації.
5. ConvolutionModule – клас модуля одновимірного згорткового шару.
6. LinearModule – клас модуля лінійного шару.
7. HighwayModule – клас модуля магістралевої мережі.
8. PreNetModule – клас модуля пре-мережі.
9. CBHGModule – клас модуля CBHG (англ. Convolution Bank Higway GRU), що складається з групи одновимірних згорток, магістралевої мережі та двонапрямної GRU.
10. EncoderModule – клас модуля кодера моделі.
11. DecoderModule – клас модуля декодера моделі.
12. LinearLayer – клас лінійного шару.
13. PreNetLayer – клас шару пре-мережі.
14. ConvolutionalLayer – клас згорткового шару.
15. FFTBlockLayer – клас шару FFT блоку.
16. ConvolutionNormalizationLayer – клас шар згорткової нормалізації.
17. PostNetLayer – клас шару пост-мережі.

Повний код проєкту наведено в Додатку А.

3.2. Алгоритм проведення експериментів

Алгоритм проведення практичних експериментів складається з наступних етапів:

1. Вибір імплементації різних методів синтезу у проєктах з відкритим кодом.
2. Вибір способу тренування та запуску моделей: локальний, хмарний тощо.
3. Вибір конфігурації для комп’ютера, на якому будуть проводитись дослідження.
4. Вибір параметрів моделей для їх навчання та запуску.
5. Вибір вокодеру для моделей синтезу та їх навчання, якщо це необхідно.
6. Пошук та вибір мовного корпусу для тренування моделей та проведення досліджень.
7. Підготовка мовного корпусу, якщо необхідна.
8. Розподіл мовного корпусу для різних цілей: тренування, тестування, оцінювання.
9. Запуск навчання моделей.
10. Фіксація необхідного часу для тренування.
11. Ручна перевірка тренованих моделей.
12. Підготовка необхідних даних для проведення оцінювання результатів синтезованого мовлення.
13. Вибір способу та шкали оцінювання результатів.
14. Проведення оцінювання.
15. Обробка даних результатів оцінювання методів синтезу мовлення.
16. Представлення результатів оцінювання, аналізу ресурсомісткості та

їх порівняння.

17. Формування висновків проведеного дослідження.

3.3. Підготовка до тренування моделей

Серед проєктів з відкритим кодом, що надають імплементацію відібраних методів та є популярними, було знайдено наступні:

- **Tacotron2**: Mozilla TTS [63], Coqui TTS [56], Tacotron2 by Rayhane Mama [55].
- **FastSpeech**: Coqui TTS [56].
- **SpeedySpeech**: Coqui TTS [56], PaddleSpeech [51].
- **Glow-TTS**: Coqui TTS [56], Glow-TTS by Jaehyeon Kim [50].

Оскільки усі чотири моделі представлені в одному проєкті Coqui TTS, задля простоти саме його й було обрано для використання у дослідженні.

3.3.1. Огляд проєкту Coqui-TTS

Coqui-TTS – це бібліотека для синтезу мовлення. Вона створена на основі останніх досліджень і була розроблена для досягнення найкращого балансу між простотою навчання, швидкістю та якістю синтезованого мовлення. Coqui-TTS постачається з попередньо підготовленими моделями, інструментами для вимірювання якості наборів даних і вже використовується більш ніж 20 мовами для продуктів і дослідницьких проєктів. Вона покладається на використання програмної платформи для машинного навчання Tensorflow та візуалізацію моделей за допомогою Tensorboard.

Tensorflow – відкрита програмна бібліотека для машинного навчання цілій низці задач, розроблена компанією Google. Вона підтримує наступне:

- числові обчислення на основі багатовимірних масивів;
- GPU та розподілені обчислювання;

- автоматичне диференціювання;
- побудова моделей, їх навчання та використання;
- та ін.

Tensorboard надає візуалізацію та інструменти, необхідні для дослідження проєктів з машинним навчанням, такі як:

- відстеження та візуалізація таких показників, як втрати та точність;
- візуалізація графу моделі;
- перегляд гістограм ваг, зміщень або інших тензорів, та їх зміну з часом;
- демонстрація зображень, тексту та аудіоданих;
- профілювання програм Tensorflow та ін.

3.3.2. Вибір обчислювальних ресурсів

Обрані моделі потребують значних обчислювальних потужностей (графічний процесор), але під час проведення роботи доступ до фізичного комп’ютера з затребуваними характеристиками був відсутній. Тому було вирішено проводити тренування та синтез моделей у хмарному середовищі. Для цього було обрано один з найпопулярніших сервісів хмарних обчислень для проведення досліджень – Google Colaboratory.

Google Colaboratory (або Google Colab) це IDE, яка дозволяє будь-якому користувачеві писати вихідний код у своєму редакторі та запускати його з браузера. Зокрема, він підтримує мову програмування Python і орієнтований на завдання машинного навчання, аналіз даних, навчальні проєкти тощо. Цей сервіс засновано на Jupyter Notebook, і його можна використовувати безкоштовно за допомогою облікового запису Gmail. Він не вимагає налаштування, а також користувачу не потрібно завантажувати чи встановлювати Jupyter. Таким чином цей сервіс вважається одним з найпростіших у використанні для проведення досліджень.

Google Colab надає безкоштовно обчислювальні ресурси для запуску та тестування коду, наприклад GPU, RAM тощо. Для кожного користувача сервіс створює окрему віртуальну машину, на якій можна запускати свій код, ізольований від інших користувачів і ресурсів. Тому віртуальну машину можна відновити до початкового стану, якщо у користувача виникають проблеми. Це також означає, що якщо користувач виконує код у своїй віртуальній машині та закриває браузер, машина буде видалена після певного періоду бездіяльності, щоб звільнити ресурси. Щоб зберегти результати роботи сесії та сам проєкт та завантажити їх після відключення середовища, їх можна зберігати у Google Drive. Завдяки формату Jupyter ".ipynb", проєкт можна потім завантажувати локально будь-якою програмою, що підтримує це розширення.

Цей сервіс також має три варіанти підписки з перевагами вказаними у таблиці 3.1.

Таблиця. 3.1

Порівняння підписок у Google Colab

	Colab	Colab Pro	Colab Pro+
Гарантія ресурсів	Низька	Висока	Дуже висока
Доступні GPU	NVIDIA K80	NVIDIA K80, NVIDIA T4, NVIDIA P100	
RAM	16 GB	32 GB	52 GB
Тривалість сесії	До 12 годин	До 24 годин	До 24 годин
Виконання у фоновому режимі	Не доступне	Не доступне	Доступне
Вартість	Безкоштовно	$9.99/місяць	$49.99/місяць

У безкоштовній версії Google Colab, кількість ресурсів та їхня якість майже не гарантується, тобто при підключенні користувач може отримати менше RAM, GPU нижчої потужності або нижчий пріоритет у доступі до GPU тощо. При купівлі підписки Colab Pro у користувача з'являється більше RAM

(до 32 GB), проєкти можуть довше бути активними та з’являються нові, потужніші GPU. Через те, що у безкоштовній версії доступний GPU досить низької потужності, та максимальна тривалість сесії досить коротка, було вирішено скористатися підпискою Colab Pro, щоб отримати доступ до потужнішого процесора NVIDIA P100 та спростити процес навчання п’ятьох моделей. Процесор NVIDIA P100 зрівнюється за потужністю з тими, що зазначаються у наукових роботах – NVIDIA V100, тому саме він був використаний у дослідженні.

3.3.3. Визначення вимог до навчання

Одним з найважливіших параметрів навчання будь-якої нейронної мережі є кількість ітерацій навчання. Зазвичай, лише емпіричним способом можна дізнатись, скільки ітерацій знадобиться для якісного навчання нейронної мережі. Тому виходячи з інформації, наданої у наукових статтях, були розглянуті наступні дані (табл. 3.2.)

Таблиця 3.2

Порівняння кількості ітерацій наведених у наукових статтях

Метод	Кількість ітерацій
Tacotron2	не вказана
SpeedySpeech	не вказана
FastSpeech	80 000
Glow-TTS	240 000

Оскільки лише у двох наукових статтях вказана кількість ітерацій, то для завершення тренування моделей був обраний алгоритм [9], наданий у проекті Coqui TTS. Алгоритм виглядає наступним чином:

1. Перевірити згенероване аудіо фази тестування, чи продовжує

покращуватись його якість?

2. Перевірити графіки механізму уваги фази тестування, чи вони виглядають чіткими та діагональними?
3. Перевірити графік втрат, чи він плавно опустився і стабілізувався.
4. Якщо відповідь “так” на всі три пункти, то модель можна вважати навченою. Тепер можна перевірити її з набором складних речень вручну.

3.3.4. Вибір вокодера

Кожна модель, описана у наукових статтях, використовувала різні конфігурації з вокодерами (наприклад, SpeedySpeech + Griffin-Lim порівнювався з SpeedySpeech + MelGAN). Найбільш популярним та простим у використанні, є вокодер на основі алгоритму Griffin-Lim, і він не потребує окремого тренування, на відміну від інших вокодерів. Для того, щоб ізольовано від тренування вокодерів розглянути моделі, буде використовуватися саме алгоритм Griffin-Lim.

3.3.5. Підготовка мовного корпусу

Як вже згадано в розділі 1.4, було віднайдено лише один архів з україномовними аудіозаписами та розшифровками у відкритому доступі – M-AILABS. Для тренування моделей будуть використовуватись аудіокниги озвучені Ольгою Сумською, взяті з архіву M-AILABS. Аудіозаписи цього архіву мають частоту дискретизації 16 кГц. Дані у цьому архіві вже підготовлені правильним чином, тому нам просто необхідно розділити записи на три категорії: дані для тренування моделей, тестування та проведення оцінювання. Отже, з 3947 записів випадковим чином були відібрані 3907 для тренування, 20 для тестування та 20 для оцінювання.

3.4. Тренування моделей

Для того, щоб почати тренування моделей у Google Colab, було створено обліковий запис Google, та придбано підписку Colab Pro. Окрім цього, було придбано підписку на 100 GB у Google Drive, щоб підключити його до проєкту в Google Colab і зберігати там проєкти без втрати прогресу.

Було створено проєкт у Google Colab з назвою “TTS”. У ньому було створено наступні блоки з кодом:

1. Блок з підключенням необхідних пакетів.
2. Блок з підключенням проєкт до Google Drive.
3. Блок завантаження Coqui TTS з Github у Google Drive.
4. Блок встановлення пакетів Python проєкт Coqui TTS.
5. Блок завантаження FastSpeech1.1 з Github у Google Drive.
6. Блок встановлення пакетів Python проєкта FastSpeech1.1.
7. Блок запуску тренування Glow-TTS.
8. Блок запуску тренування SpeedySpeech.
9. Блок запуску тренування Tacotron2.
10. Блок запуску тренування FastSpeech.
11. Блок запуску тренування FastSpeech1.1.
12. Блок підключення Tensorboard у log-директорії Glow-TTS.
13. Блок підключення Tensorboard у log-директорії SpeedySpeech.
14. Блок підключення Tensorboard у log-директорії Tacotron2.
15. Блок підключення Tensorboard у log-директорії FastSpeech.
16. Блок підключення Tensorboard у log-директорії FastSpeech1.1.
17. Блок запуску Glow-TTS для синтезу мовлення з набором складних речень для ручної перевірки.
18. Блок запуску SpeedySpeech для синтезу мовлення з набором складних речень для ручної перевірки.
19. Блок запуску Tacotron2 для синтезу мовлення з набором складних речень для ручної перевірки.

20. Блок запуску FastSpeech для синтезу мовлення з набором складних речень для ручної перевірки.
21. Блок запуску FastSpeech1.1 для синтезу мовлення з набором складних речень для ручної перевірки.

Інтерфейс проєкту зображено на рис. 3.1-3.4.

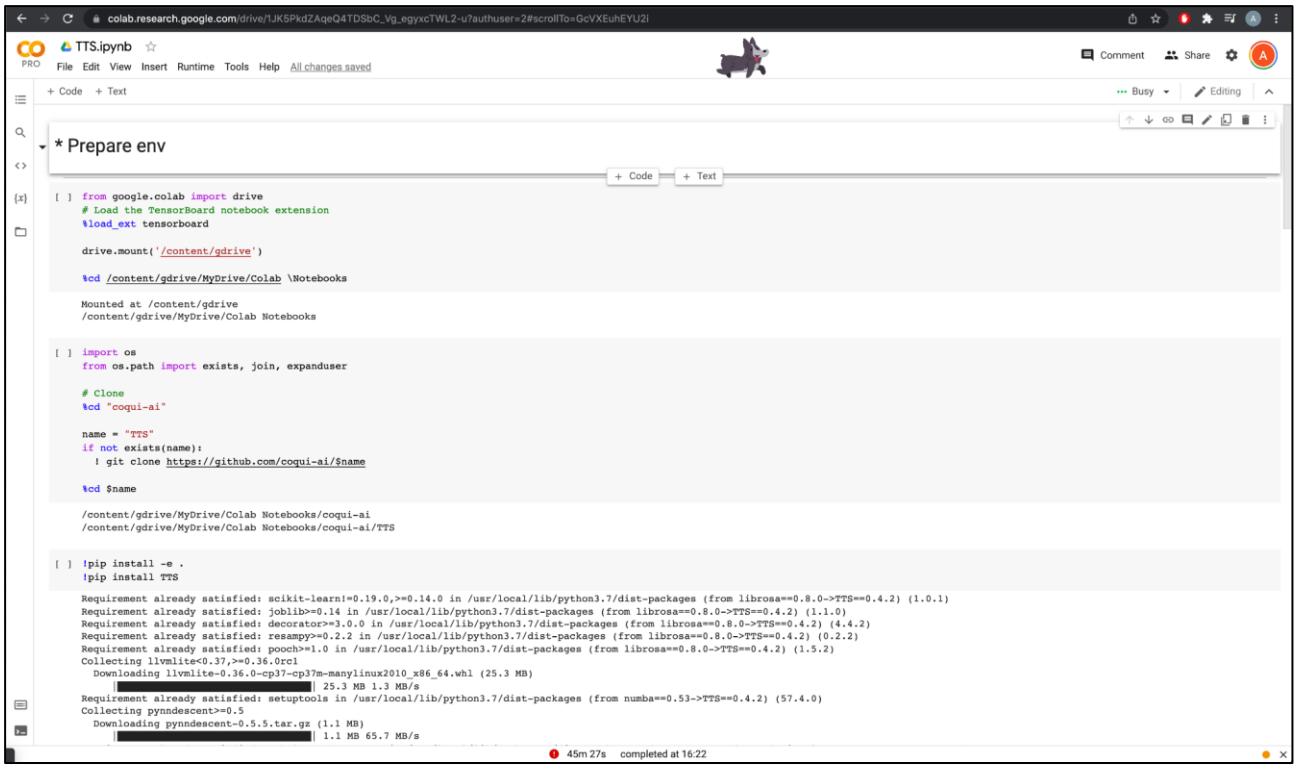

Рис. 3.1. Інтерфейс проєкту Google Colab

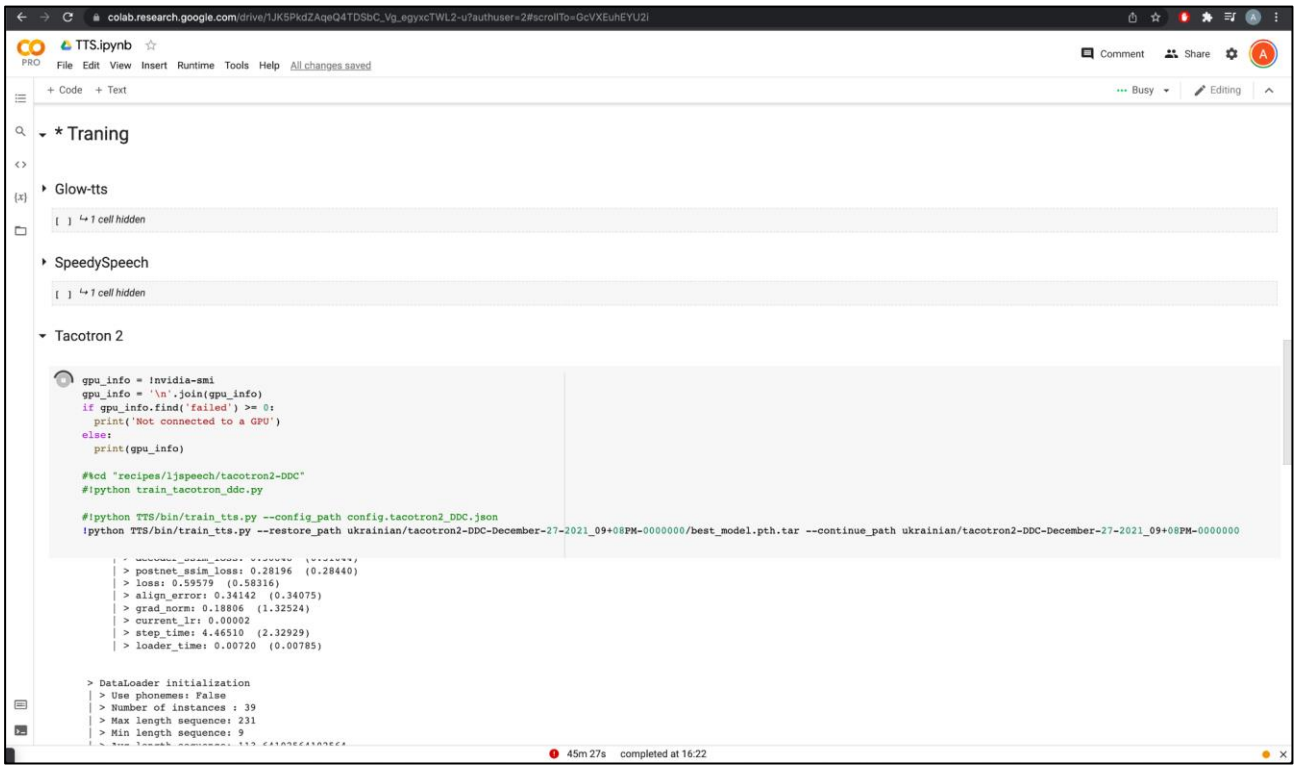

Рис. 3.2. Інтерфейс проєкту Google Colab

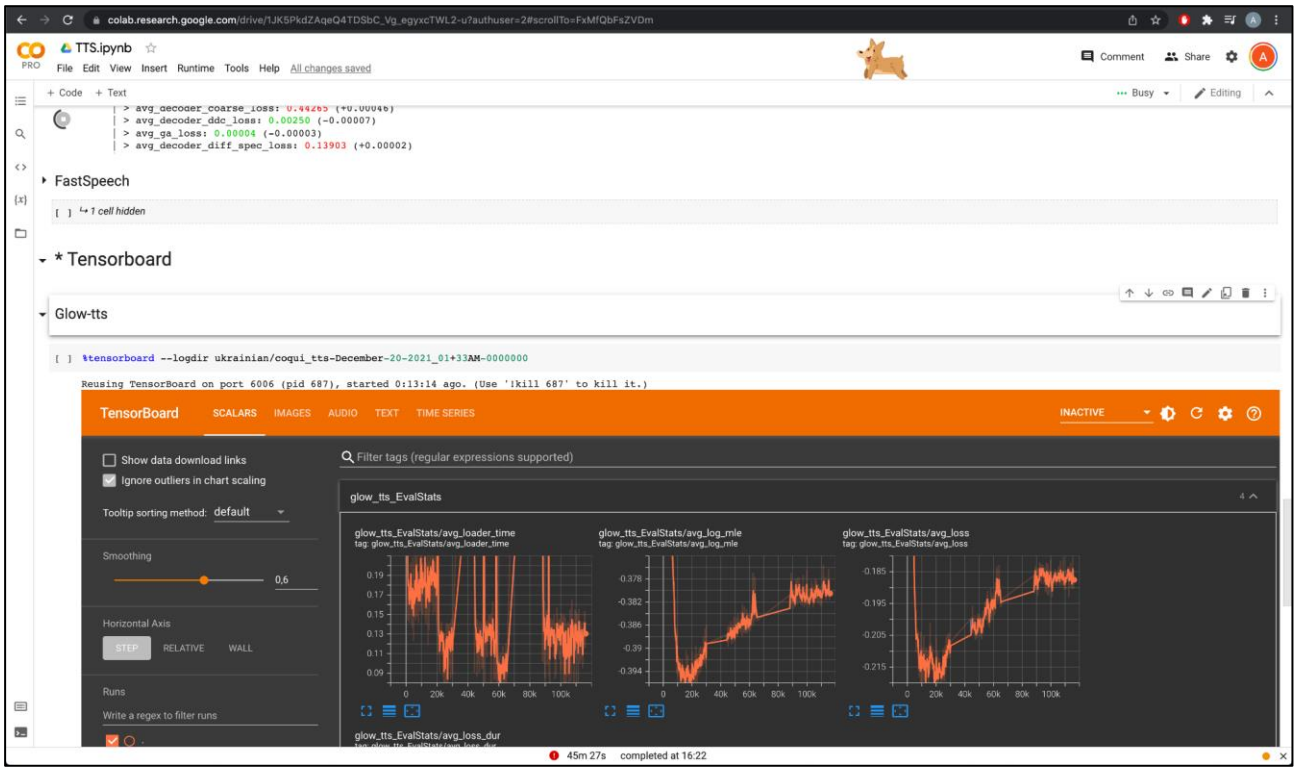

Рис. 3.3. Інтерфейс проєкту Google Colab

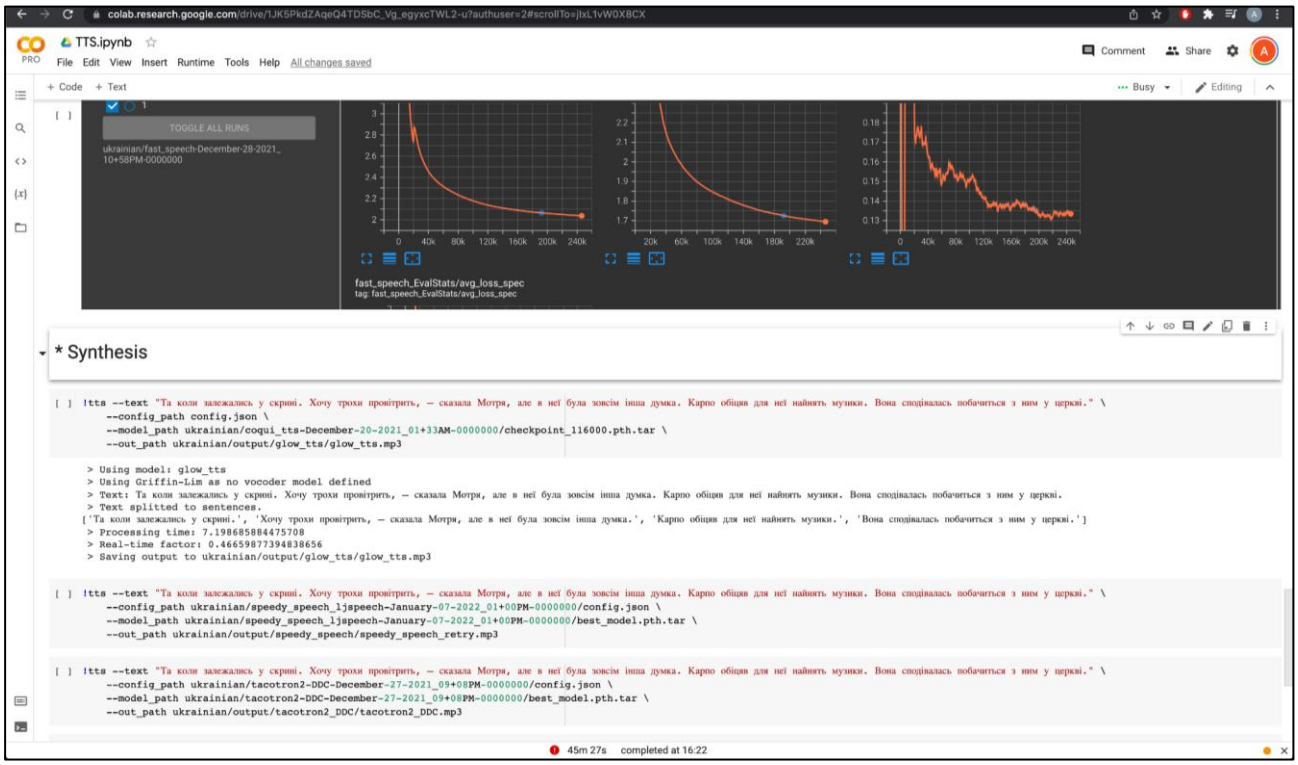

Рис. 3.4. Інтерфейс проєкту Google Colab

Стандартні конфігурації для тренування кожної з моделей були змінені, щоб відобразити параметри мовного корпусу.

Блоки коду були запущено у середовищі з графічним процесором NVIDIA P100 з 12 GB ОЗП і за декілька днів усі з моделей завершили тренування, згідно з алгоритмом перевірки:

1. Якість аудіо фази тестування у кожного з методів перестала покращуватись.
2. Графіки механізму уваги фази тестування виглядають чіткими та діагональними для Glow-TTS (рис. 3.5), SpeedySpeech (рис. 3.6), Tacotron2 (рис. 3.7), FastSpeech (рис. 3.8), FastSpeech1.1 (рис. 3.9).
3. Графіки втрат плавно опустилися і стабілізувалися для Glow-TTS (рис. 3.10), SpeedySpeech (рис. 3.11), Tacotron2 (рис. 3.12), FastSpeech (рис. 3.13), FastSpeech1.1 (рис. 3.14).
4. Моделі було перевірено на складних реченнях вручну.

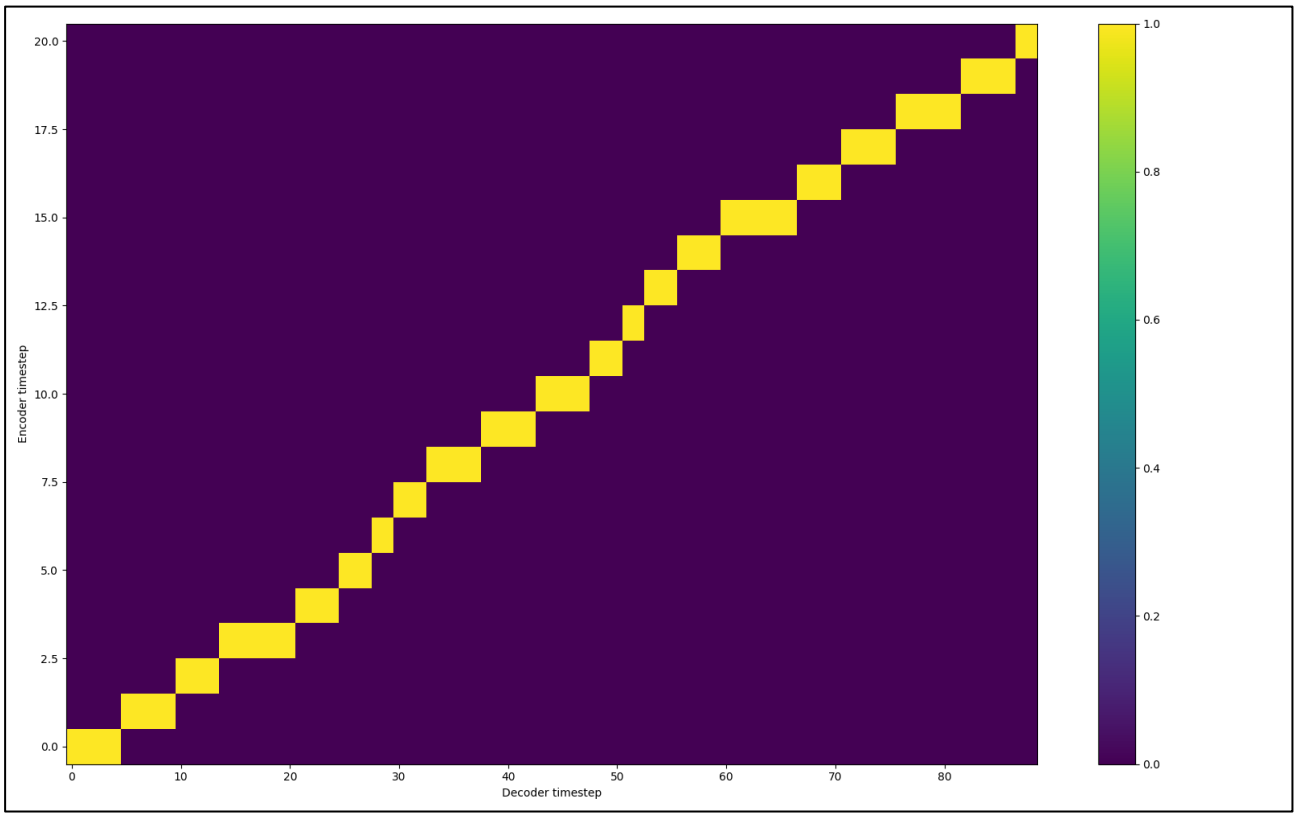

Рис. 3.5. Графік механізму уваги Glow-TTS

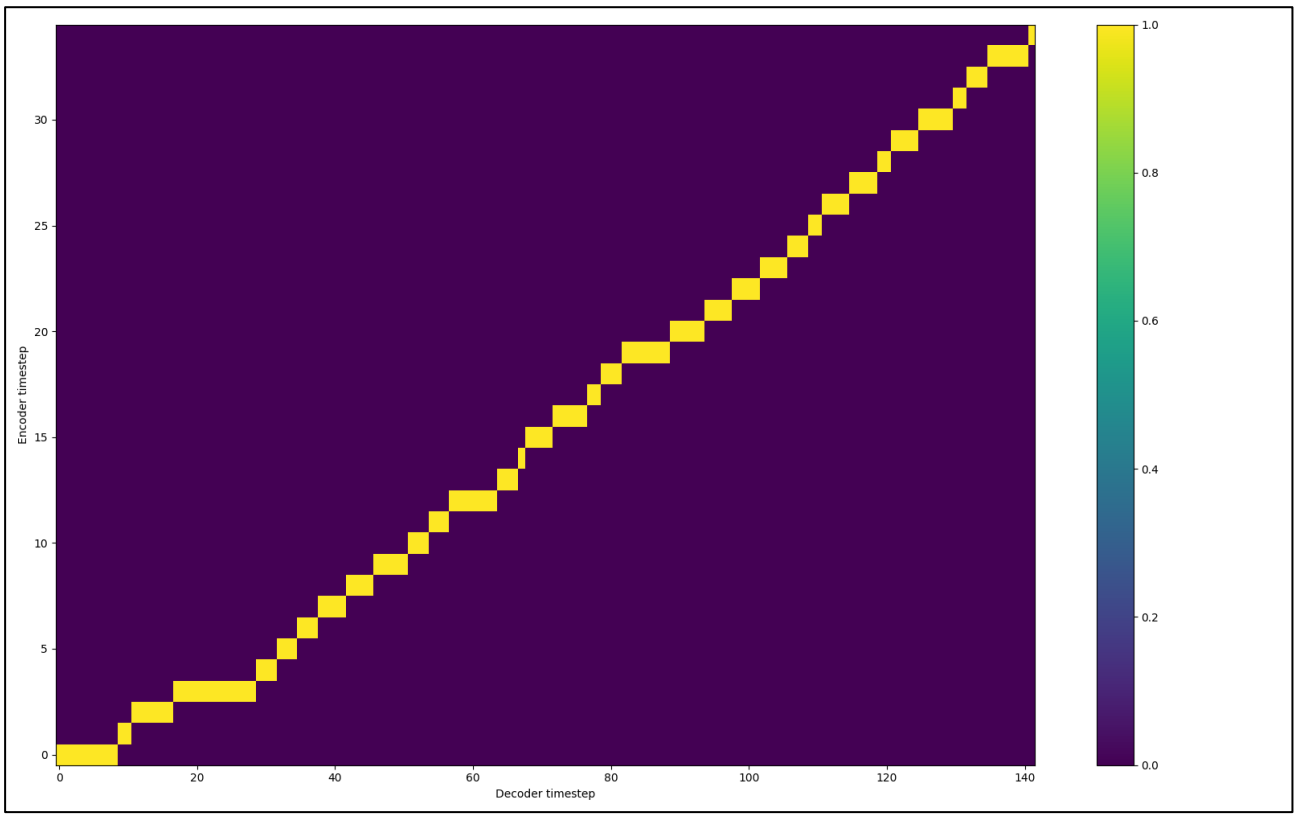

Рис. 3.6. Графік механізму уваги SpeedySpeech

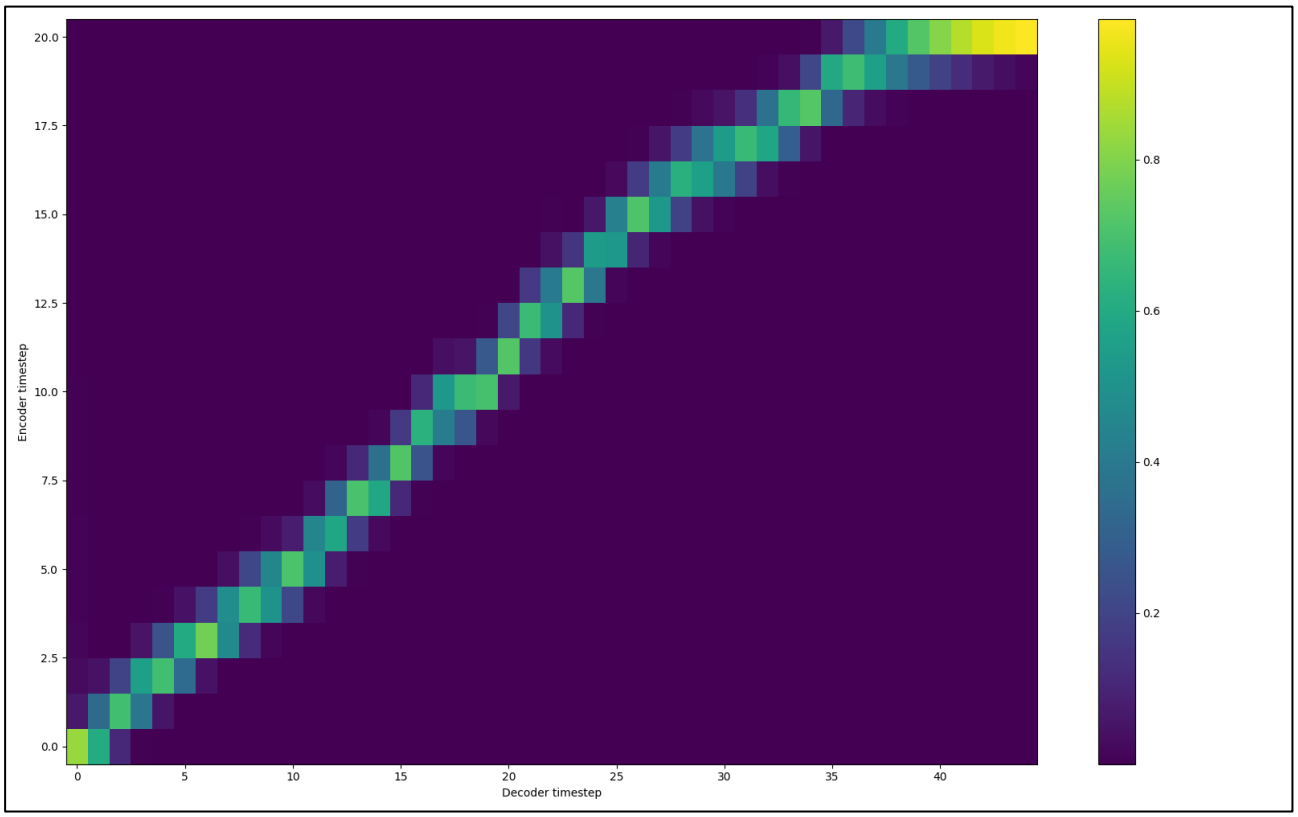

Рис. 3.7. Графік механізму уваги Tacotron2

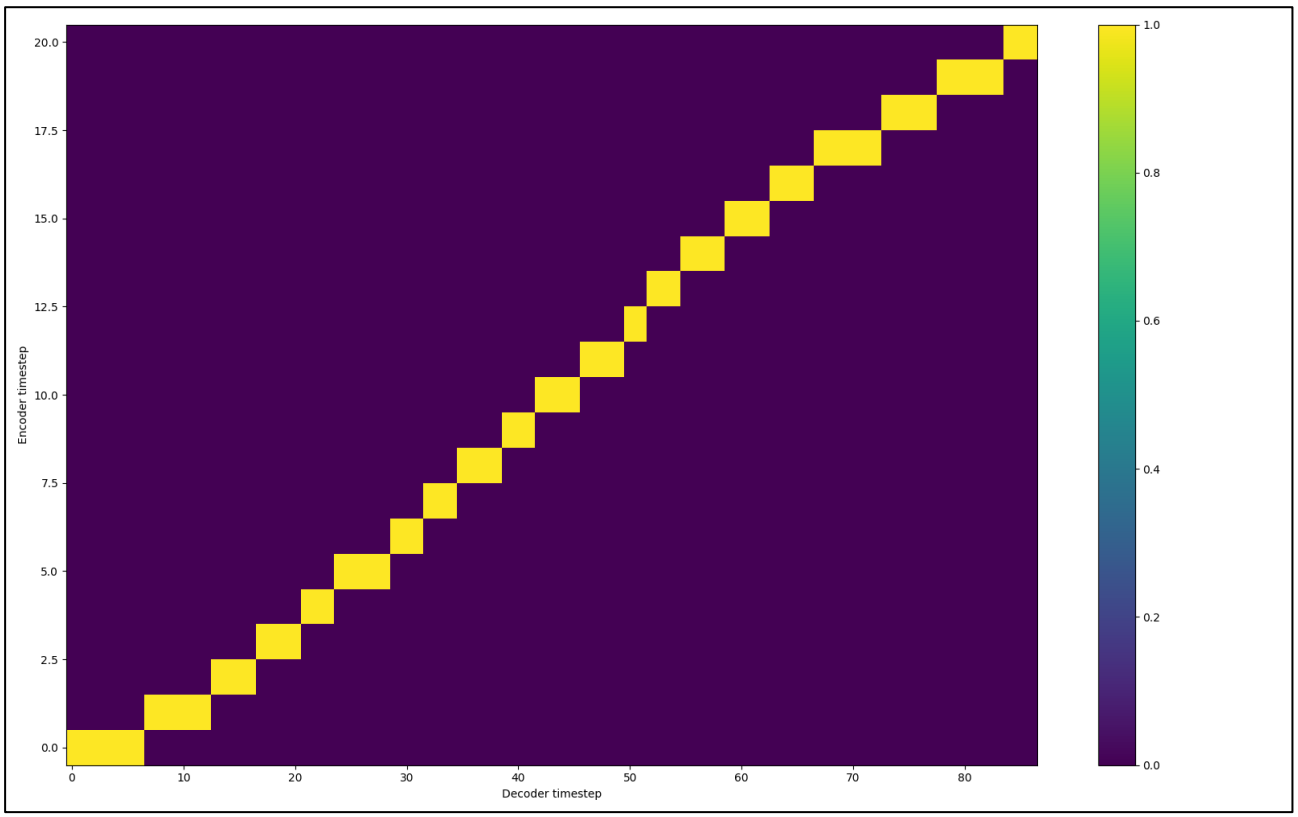

Рис. 3.8. Графік механізму уваги FastSpeech

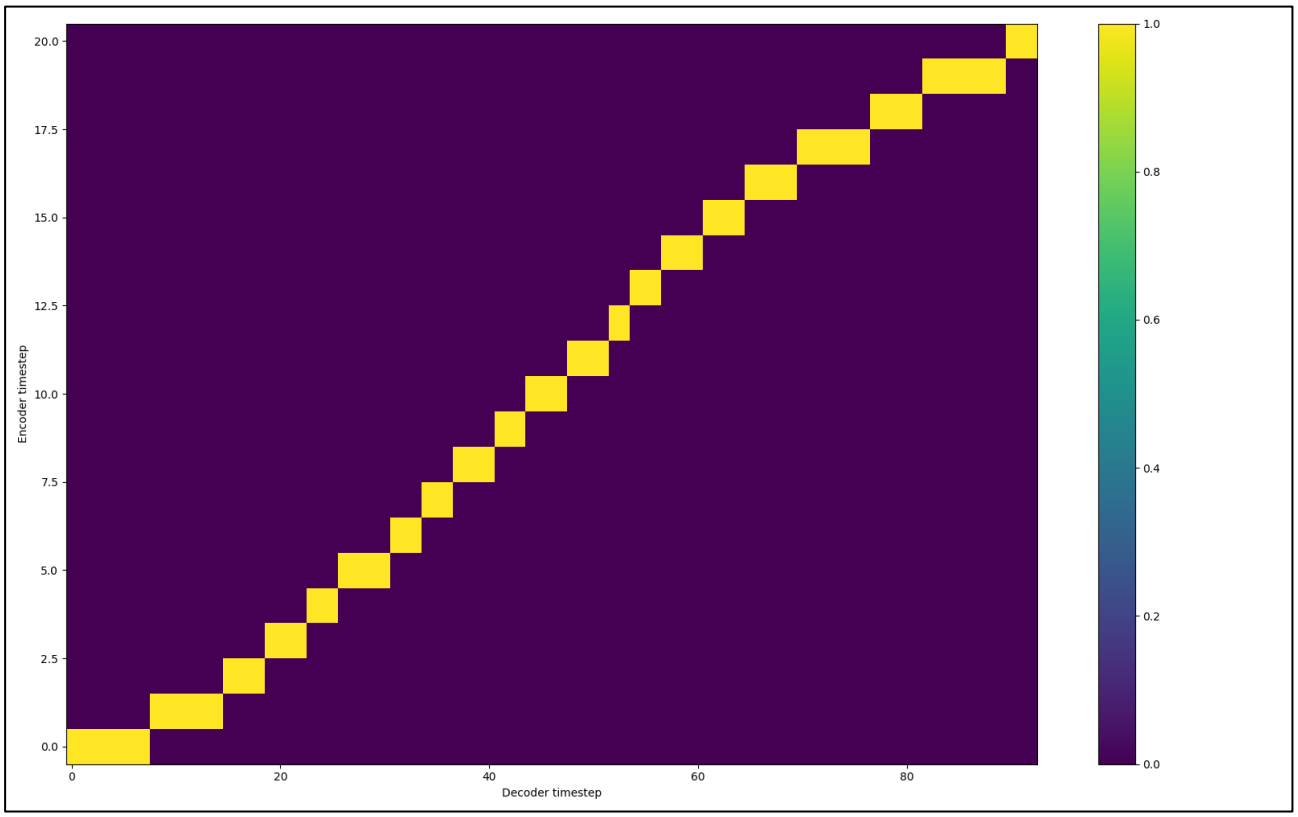

Рис. 3.9. Графік механізму уваги FastSpeech1.1

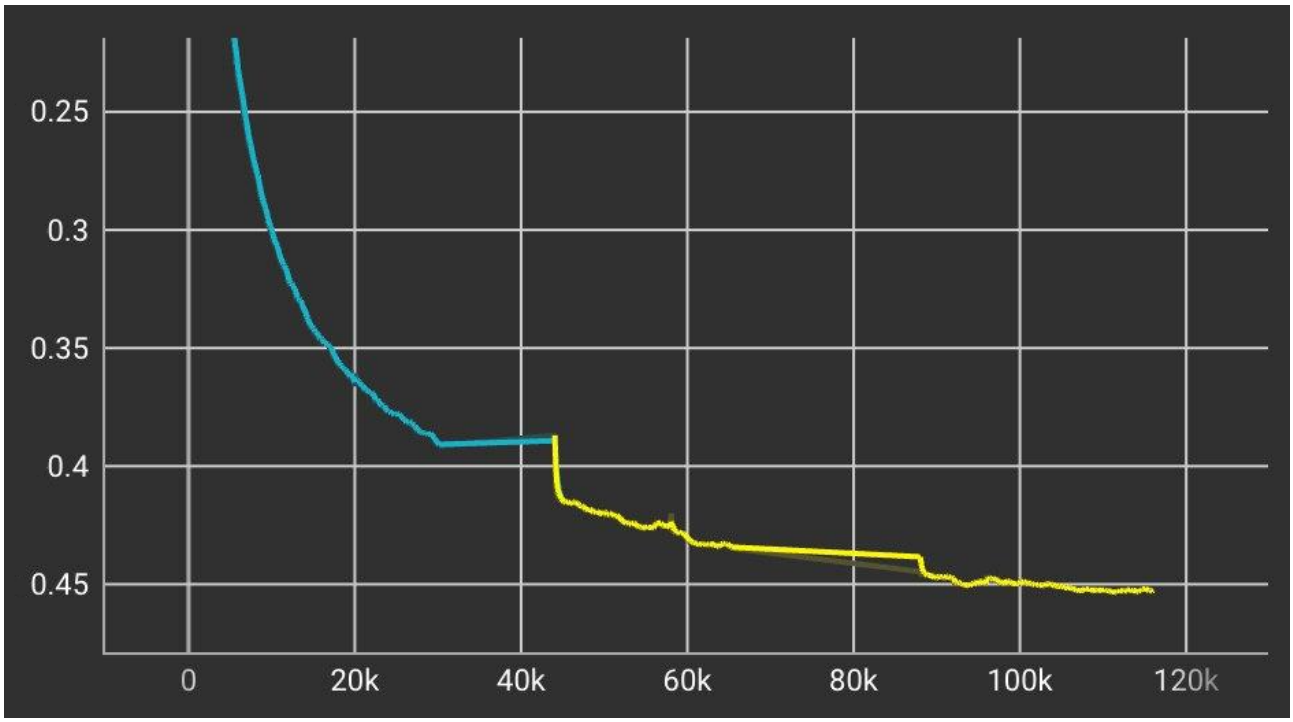

Рис. 3.10. Графік механізму уваги Glow-TTS

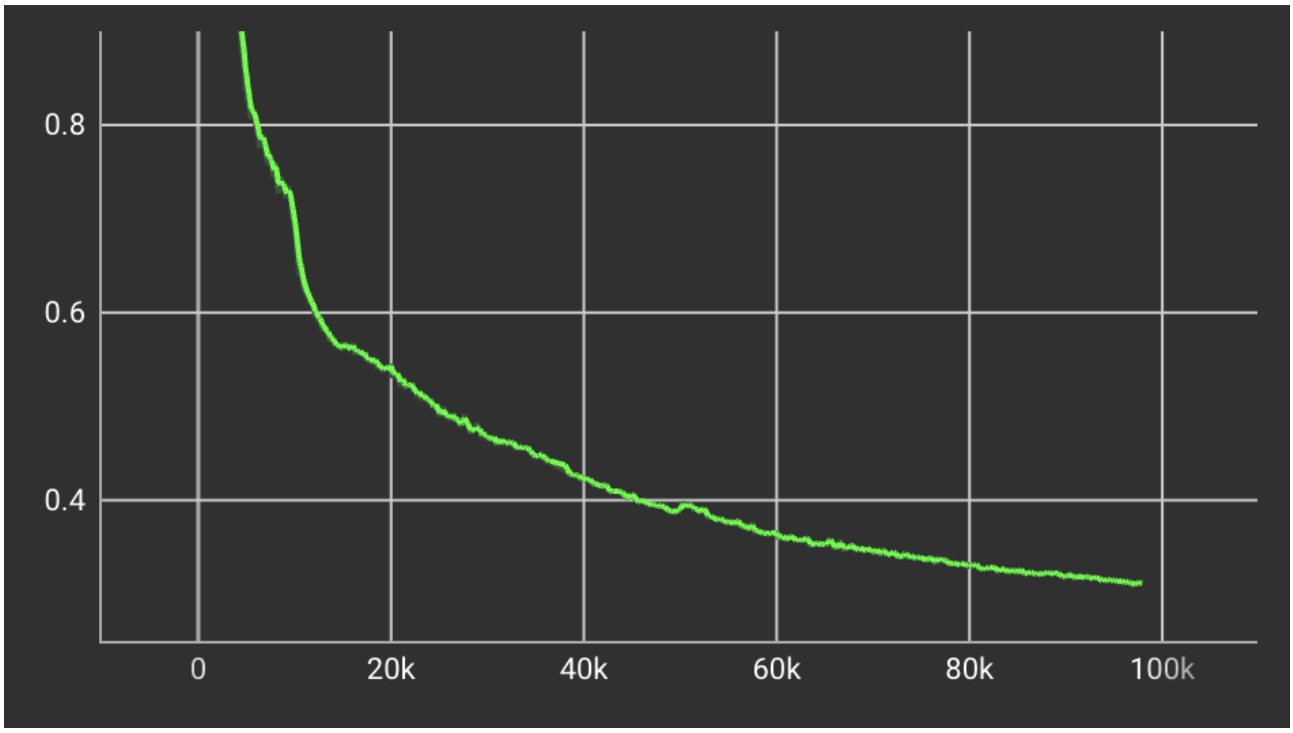

Рис. 3.11. Графік механізму уваги SpeedySpeech

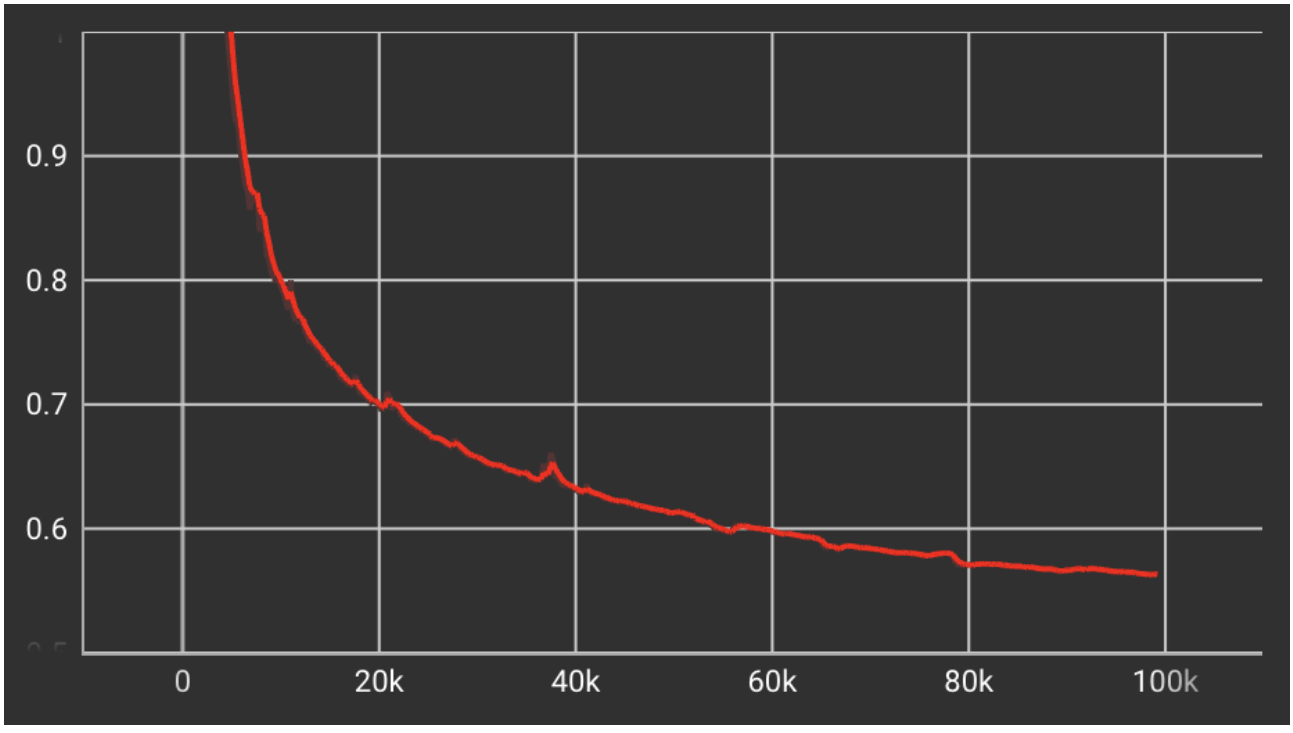

Рис. 3.12. Графік механізму уваги Tacotron2

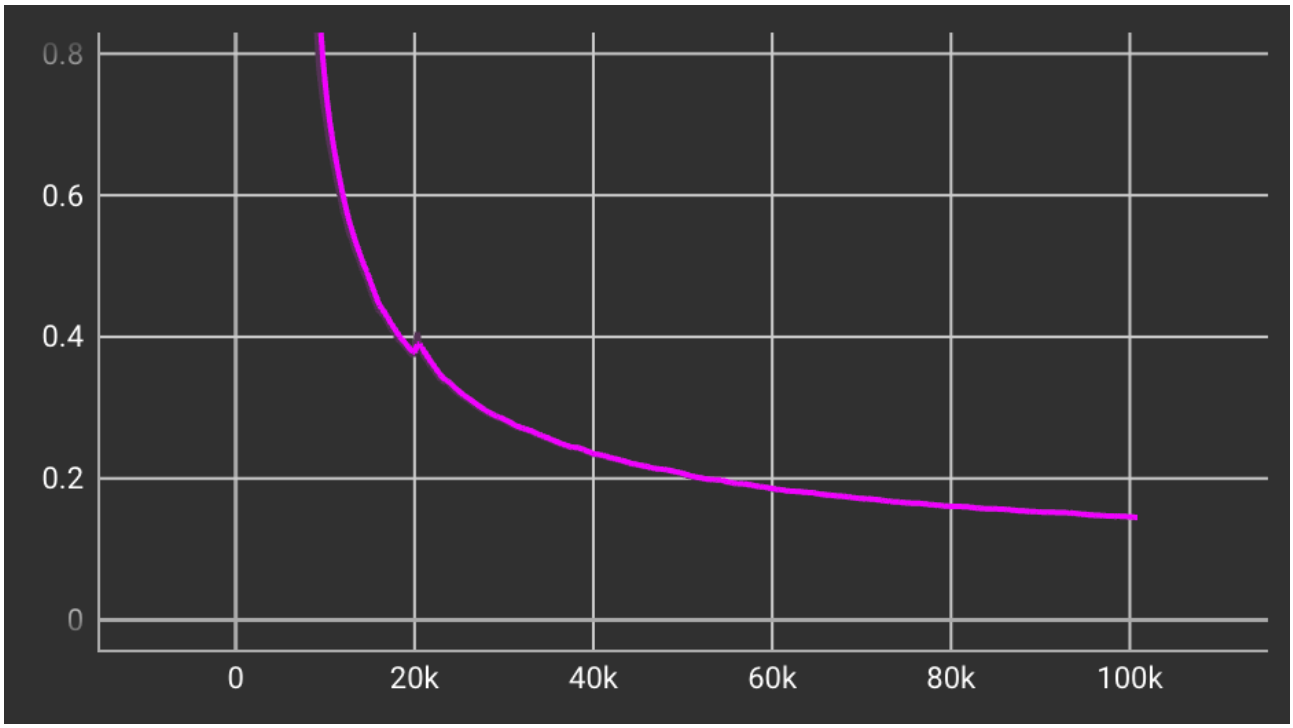

Рис. 3.13. Графік механізму уваги FastSpeech

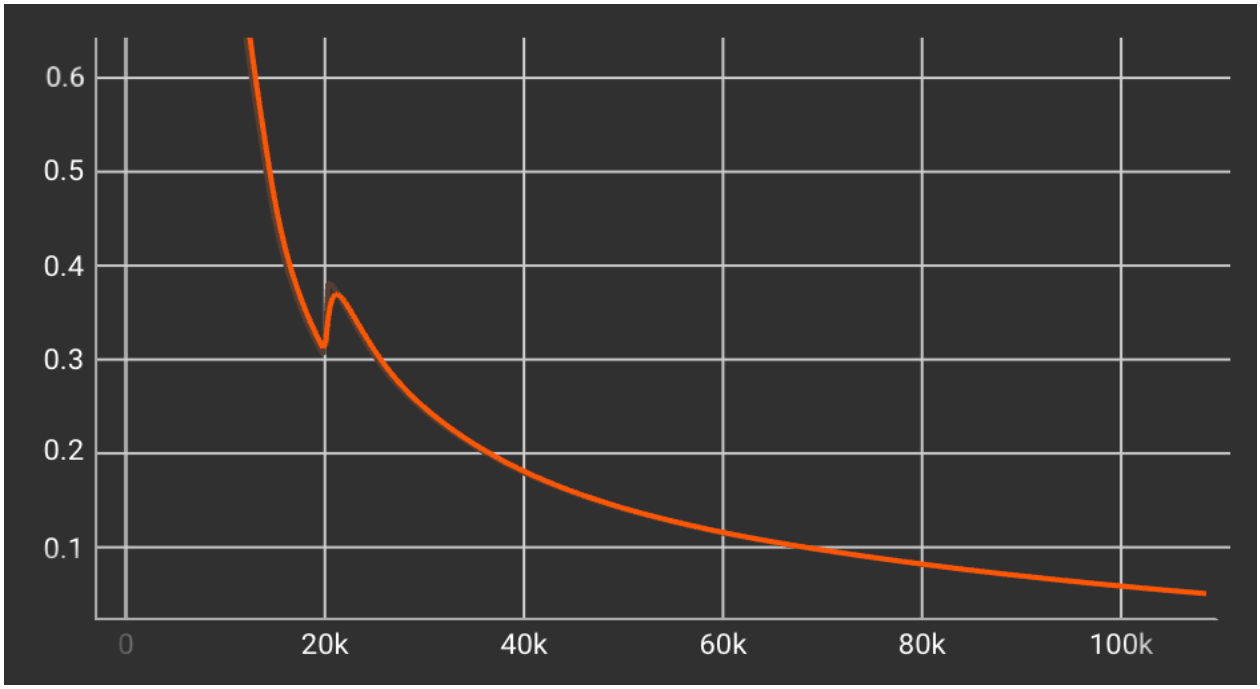

Рис. 3.14. Графік механізму уваги FastSpeech1.1

Було зафіксовано кількість часу, що знадобився для тренування кожної з моделей. Результати наведені у розділі 3.6 кваліфікаційної роботи.

3.5. Проведення оцінювання якості синтезованого мовлення

Для оцінювання якості синтезованого мовлення було синтезовано аудіо для попередньо випадково обраних 20 речень з мовного корпусу, що вказані у таблиці 3.3.

Таблиця 3.3

Речення для оцінювання якості синтезу моделі

Текст речення	К-сть символів
1	2
А як я свисну за садком, чи вийдеш?	35
Вже пізненько, як усі богомольці поснули таким сном, що їх не розбудили б і лаврські дзвони, чернець вийшов з келії.	116
Мотря оступилась.	17
Погана, бо іродів шинкар розводить водою, – проморив Балаш. – Бодай йому смерть така, як оця горілка.	101
Лаврін прикрив Мелашчині плечі своєю свитою і обняв її за стан.	63
А ти чого оступаєшся за своєю жінкою? – крикнув він на Карпа. – Коли хочеш, то я тобі носа втру.	96
За що їм давати половину? Чи то можна? Це вони схотять, щоб ми давали їм половину картоплі та буряків. Це все свекруха наговорює в волості.	139
Вони втрьох ходили, розпитували, чи не чув хто, чи не бачив молодої чорнявої молодиці, чи не шукала, чи не питала вона про своїх людей.	135
Вже й знайшов красуню! Та в неї лице, як тріска, стан, наче кописткa, руки, як кочерги, сама, як дошка, а як іде, то аж кістки торохтять.	137
Але як тільки вона трохи сердилась, з неї спадала та солодка луска, і вона лаялась і кричала на ввесь рот. Маруся була сердита.	127
З того часу Лаврін загарбав хазяйство в свої руки. Батько мусив мовчати і рідко коли вмикувався в хазяйство.	108

Продовження таблиці 3.3

1	2
Не встиг він закласти заноза в ярмо і ненароком кинув очима за Рось: за Россю, коло скелі на долині, вкритій зеленим житом, червоніла якась велика квітка.	154
Якого це ти нечистого так ляпаєш? Ще мало сміття в хаті, то нехай буде грязь, – крикнула Кайдашиха. – Чом ти своїй жінці нічого не скажеш? – сказала Кайдашиха до Карпа.	168
Балаш з Кайдашем посідали кінець стола і почали розмовляти за жито та пшеницю, за сіно та за ярину. Балашиха послала найстаршого хлопця в шинок по горілку.	155
Оцього я вже не люблю! – крикнула Мотря.	40
Хоч між дровами, аби з чорними бровами! – сказав Лаврін. – Хоч під лавкою, аби з гарною панянкою.	97
Ого-го, наша пані економша вилізла трохи не під небо! – загомоніли чоловіки. – Видно, що їде на розглядини.	107
Лаврін знав парубоцький звичай і повів усю парубочу ватагу в шинок. Він поставив їм могорича і вже в згоді з ними вернувся на вулицю.	133
Даруй же, боже, нам і нашим дітям вік довгий та щасливий, щоб ти, моя доню, була здорова, як вода, щоб цвіла довіку, як рожа.	125
Боже мій! Де це я? – говорив сам до себе Кайдаш.	48

Була обрана шкала оцінювання суб’єктивної середньої оцінки MOS. MOS – це показник, який використовується в області забезпечення якості та телекомунікаційній інженерії, що представляє загальну якість системи. Це середнє арифметичне за всіма окремими «значеннями у заздалегідь визначеній шкалі, яку суб’єкт зіставляє зі своєю думкою про якість системи» [43]. Такі оцінки зазвичай збираються за допомогою суб’єктивного тесту оцінки якості, але вони також можуть бути оцінені алгоритмічно.

MOS є широко використовуваним способом для оцінки якості відео, аудіо та аудіовізуального зображення, але не обмежується цими галузями.

MOS виражається як єдине дійсне число, як правило, в діапазоні 1–5, де 1 – це найнижча суб’єктивна якість, а 5 – найвища суб’єктивна якість. Інші

діапазони MOS також можливі, залежно від шкали оцінок, яка була використана в тесті. Дуже часто використовується шкала оцінок абсолютної категорії, яка відображає оцінки від “погано” до “відмінно” як числа від 1 до 5, як продемонстровано в таблиці 3.4.

Таблиця 3.4

Відповідність оцінки MOS до позначки

Оцінка	Позначка
5	Відмінно
4	Дуже добре
3	Добре
2	Незадовільно
1	Погано

MOS розраховується як середнє арифметичне для окремих оцінок, визначених людьми для даної системи в тесті суб’єктивної оцінки якості. Таким чином:

$$MOS = \frac{\sum_{m=1}^{M} R_n}{N}, \quad (3.1)$$

де R – це індивідуальні оцінки системи, визначені N суб’єктами.

Щоб оцінити та порівняти якість синтезованого звуку, було проведено внутрішнє опитування з 40 учасників. Кожна особа є носієм української мови, тому може справедливо оцінити правдоподібність та розбірливість синтезованого мовлення. 20 аудіозаписів було синтезовано для кожної моделі, і разом з 20 аудіозаписами справжнього мовлення їх було випадковим чином розміщено у опитуванні Google Forms. Разом опитування склало 120 анонімізованих аудіозаписів. Кожен учасник їх прослухав і оцінив природність та розбірливість від 1 до 5. Підрахована суб’єктивна середня оцінка MOS для кожної з моделей вказана у розділі 3.6.

Також, під час синтезу аудіозаписів для проведення оцінювання, було зібрано дані щодо швидкості синтезу. Оброблені дані представлені у розділі 3.6.

Для перевірки моделей на помилки синтезу у цій роботі був запропонований набір з 20 складних речень, наведених у таблиці 3.5. Ці речення було синтезовано моделями Tacotron2, FastSpeech та FastSpeech1.1, щоб підрахувати як часто у цих моделей виникатимуть помилки. Результати наведені у розділі 3.6.

Таблиця 3.5

Речення для перевірки моделі на помилки

Текст речення	К-сть символів
Бабин біб розцвів у дощ – буде бабі біб у борщ.	47
Був собі цебер, та переполуцебрився на полуцебренята.	53
Бобер на березі з бобренятами бублики пік.	42
Боронила борона по боронованому полю.	37
Дзижчить над житом жвавий жук, бо жовтий він вдягнув кожух.	59
Біжать стежини поміж ожини.	27
Захар заліз на перелаз. – Захарку, злізь! – Захарку, злазь! Зумів залізти – Знай, як злізти.	92
Кинув кріп Прокіп в окріп. У окропі, окрім кропу кипить короп для Прокопа.	74
Коваль кулю кував, кував і перековував.	39
Ходить квочка коло кілочка, водить діток, дрібних квіток.	57

3.6. Результати дослідження

Після завершення навчання моделей було зафіксовано витрачений на це час та кількість необхідних ітерацій. Ці дані зібрані у таблиці 3.6.

Таблиця 3.6

Витрачений час на навчання моделей

Модель	Кількість ітерацій	Час
Glow-TTS	116 000	32 години (1.3 доби)
SpeedySpeech	98 000	34 години (1.4 доби)
Tacotron2	105 000	52 години (2.1 доби)
FastSpeech	101 000	16.5 години (0.7 доби)
FastSpeech1.1	110 000	15 годин (0.6 доби)

Можемо побачити, що найшвидшим у навчанні виявився запропонований у роботі FastSpeech1.1, якому знадобилося лише 15 годин для навчання за допомогою графічного процесора NVIDIA P100. Найбільш ресурсоємним виявилася, як і очікувалося, модель Tacotron2, заснована на рекурентних нейронних мережах. Для завершення її навчання знадобилося 52 години.

Дані щодо швидкості синтезу були зібрані під час синтезу аудіозаписів для проведення оцінювання. Результати представлені у таблиці 3.7 та на рис. 3.15. Тривалість синтезу означає кількість секунд, що знадобилися для синтезу 1 символу в середньому.

Таблиця 3.7

Дані про швидкість синтезу

Модель	Тривалість синтезу одного символу з 95% довірчим інтервалом
Glow TTS	0.04 ± 0.013
SpeedySpeech	0.023 ± 0.003
Tacotron2	0.059 ± 0.009
FastSpeech	0.037 ± 0.008
FastSpeech1.1	0.047 ± 0.012

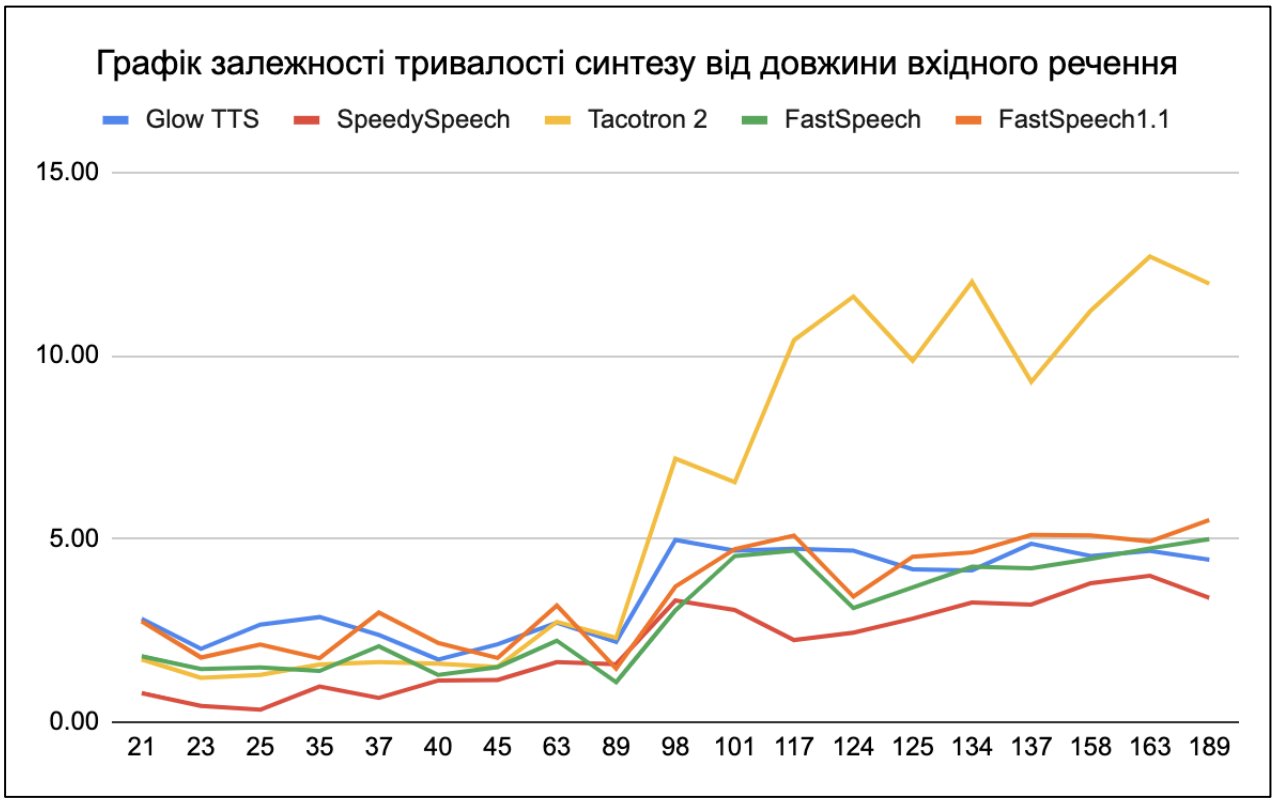

Рис. 3.15. Графік залежності тривалості синтезу від довжини вхідного речення

Згідно з цими результатами, можна зазначити, що найшвидшою моделлю у синтезі мовлення виявилася SpeedySpeech, а найповільнішою – Tacotron2. Ці результати збігаються з результатами, наданими у наукових роботах, присвячених цим моделям. Запропонована модель FastSpeech1.1, на жаль, показала негативні результати: порівняно з оригінальним FastSpeech, FastSpeech1.1 сповільнилася у 1.2 рази.

Результати оцінювання якості синтезованого мовлення наведені у таблиці 3.8 та на рис. 3.16. Можна побачити, що удосконалена модель показує приблизно такі ж самі результати якості синтезованого мовлення. Усі моделі, на жаль, показали загалом досить поганий результат природності та розбірливості синтезованого мовлення. Найкращий результат показала модель FastSpeech з оцінкою MOS 0.33, а найгіршою була визначена модель SpeedySpeech, що має оцінку MOS 1.120.

Таблиця 3.8

Оцінка якості синтезованого мовлення

Модель	MOS з 95% довірчими інтервалами
Справжні аудіозаписи	4.690 ± 0.045
Glow-TTS	1.318 ± 0.072
SpeedySpeech	1.120 ± 0.061
FastSpeech	3.330 ± 0.087
FastSpeech1.1	3.263 ± 0.082
Taco2	2.053 ± 0.08

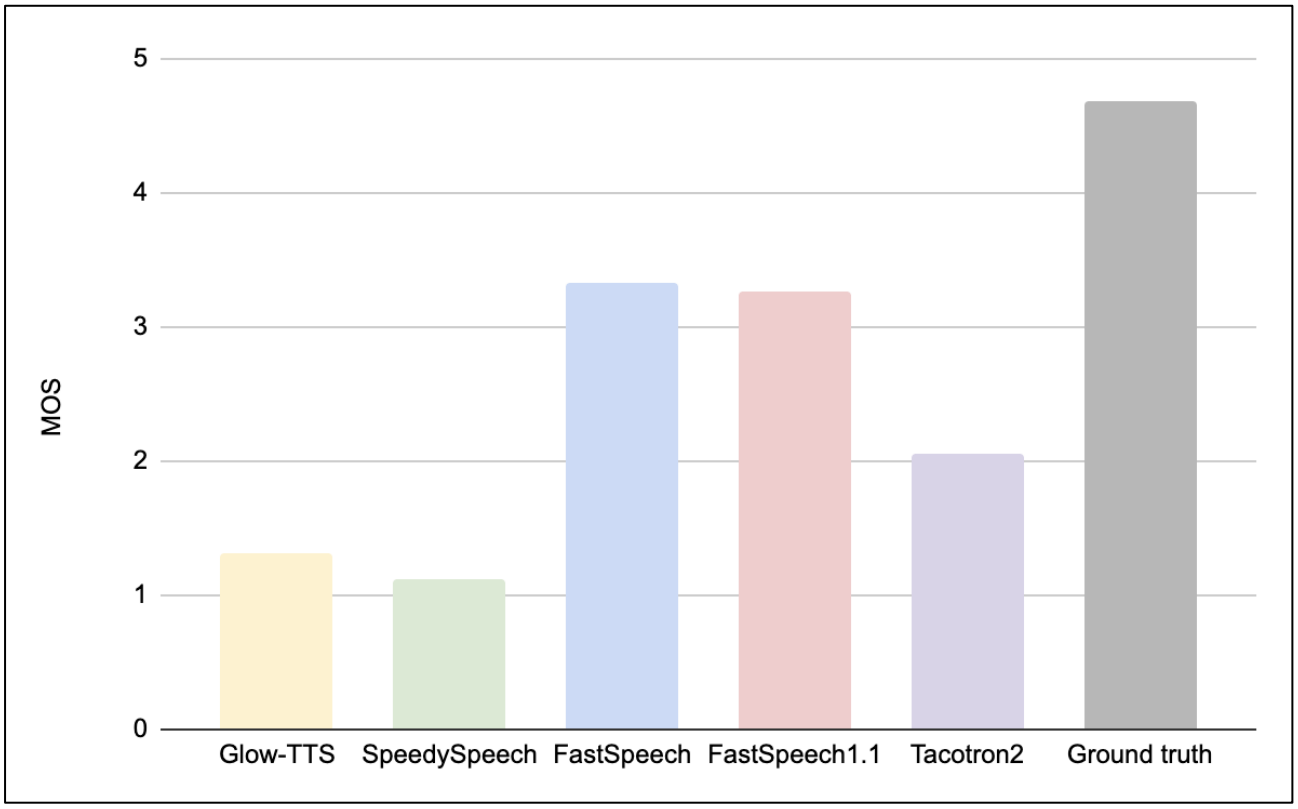

Рис. 3.16. Графік оцінки MOS для моделей синтезу

Результати тестування моделей синтезу на складних реченнях вказані у таблиці 3.9 та на рис. 3.17. Результати показали, що україномовний синтез має досить велику кількість помилок, можливо, через маленький мовленнєвий корпус. FastSpeech1.1 має лише на одну помилку менше, порівняно з FastSpeech.

Таблиця 3.9

Помилки у синтезі мовлення

Модель	Проблема наголосу	Порушення синтезу	Відсоток помилок
FastSpeech	6	1	35%
FastSpeech1.1	5	1	30%
Tacotron2	10	2	60%

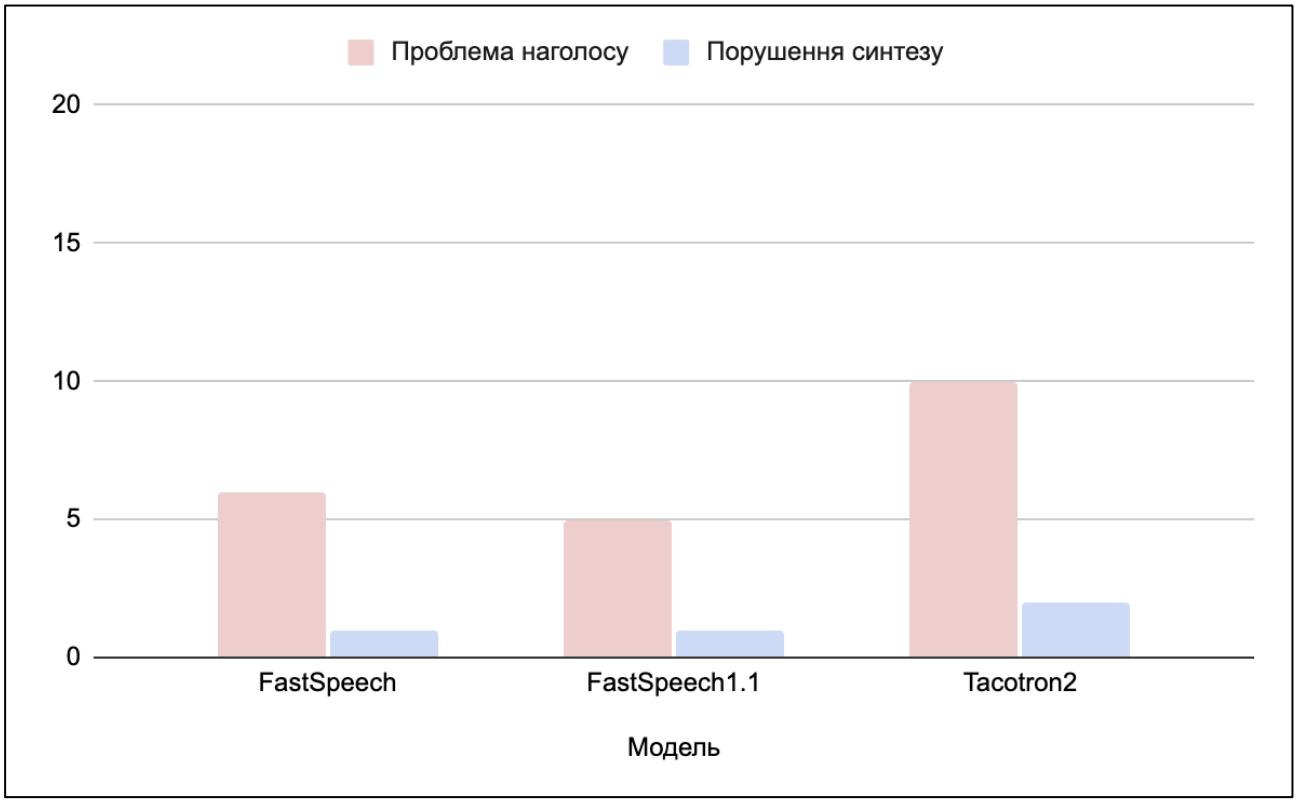

Рис. 3.17. Графік кількості помилок синтезу для моделей синтезу

3.7. Висновки

Було створено покращену модель синтезу мовлення, засновану на потокових нейронних мережах – FastSpeech1.1. Було проведено експеримент з п’ятьма моделями синтезу мовлення глибинного навчання: Glow-TTS, SpeedySpeech, Tacotron2, FastSpeech, FastSpeech1.1. Для навчання моделей було обрано невеликий корпус українського мовлення. Усі моделі було навчено успішно. Дані щодо витрачених ресурсів було зібрано і проаналізовано, а якість

синтезу була оцінена 40 респондентами за шкалою MOS. Запропонована у роботі модель FastSpeech1.1 продемонструвала покращення у швидкості тренування – 15 годин, проти 16.5 годин у оригінальній FastSpeech. Щодо швидкості синтезу, FastSpeech1.1 навпаки отримав невелике погіршення – 0.047 проти 0.037 у FastSpeech. Якість синтезу мовлення FastSpeech1.1, на жаль, також оцінена трохи гірше – 3.263 проти 3.330 у FastSpeech.

Однією з причин такої низької якості синтезу в усіх моделях може бути мовленнєвий корпус. З одного боку він є набагато меншим за ті, що використовувалися у розглянутих наукових роботах, з іншого він має меншу частоту дискретизації – 16 кГц, на противагу 20 кГц у LJSpeech. Іншою проблемою може бути використання досить простого вокодера на основі алгоритму Гріффіна-Ліма, на противагу окремо натренованим вокодерам WaveNet, MelGAN, ParallelWaveGAN, WaveGrad, WaveRNN тощо.

Завдяки проведенню цього дослідження, було виявлено, що такі моделі, як FastSpeech, можна навчити лише на одному графічному процесорі менш ніж за добу. Щобільше, було запропонувано удосконалення моделі FastSpeech, і отримали ще більш високу швидкість навчання моделі. Це не лише знижує вимоги до ресурсів електроенергії та часу, але й сприяє подальшій адаптації таких моделей для різних мов світу. На противагу цьому, такі моделі як Tacotron2 потребують набагато більше часу та потужностей, тому це варто мати на увазі при їх використанні чи подальшому дослідженні.

Загалом, результат дослідження є успішним, оскільки була досягнена мета - підвищення ефективності синтезу мовлення з урахуванням обмежень обчислювальної потужності. Для поліпшення якості україномовних синтезаторів, варто проводити пошук кращих мовленнєвих корпусів, подальші дослідження з використанням інших вокодерів та параметрів моделей.

РОЗДІЛ 4
ЕКОНОМІКА

4.1. Визначення трудомісткості розробки

Початкові параметри:

1. Прогнозоване число операторів – 1320.
2. Коефіцієнт складності програми – 1,5.
3. Коефіцієнт корекції програми в ході її розробки – 0,2.
4. Погодинна заробітна плата програміста – 200 грн/год.
5. Коефіцієнт збільшення витрат праці внаслідок неповного та/або недостатнього опису задачі – 1,4.
6. Коефіцієнт кваліфікації програміста, обумовлений стажем роботи в даній спеціальності – 1,2.
7. Вартість машино-години ЕОМ – 11 грн/год.

Процес нормування праці в ході створення програмного забезпечення є достатньо складним через творчий характер роботи програміста. Відповідно трудомісткість розробки програмного забезпечення може бути розрахована на основі системи декількох моделей з різною точністю оцінки.

Трудомісткість розробки програмного забезпечення можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{\text{отл}} + t_{\text{д}}\text{, людино-годин,} \qquad (4.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

t_n – витрати праці на програмування по готовій блок-схемі;

$t_{\text{отл}}$ – витрати праці на налагодження програми на ЕОМ;

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмному забезпеченні, яке розробляється.

Умовне число операторів визначається за формулою:

$$Q = q \cdot C \cdot (1 + p), \tag{4.2}$$

де q – передбачуване число операторів (1320);

C – коефіцієнт складності програми (1,5);

p – коефіцієнт корекції програми в ході її розробки (0,2).

Підставивши у формулу (4.2) початкові дані, отримуємо умовне число операторів в програмі:

$$Q = 1320 \cdot 1{,}5 \cdot (1 + 0{,}2) = 2376$$

Витрати праці на вивчення опису задачі t_u визначаються з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}, \text{ людино-годин}, \tag{4.3}$$

де B – коефіцієнт збільшення витрат праці внаслідок неповного та/або недостатнього опису задачі;

k – коефіцієнт кваліфікації програміста, обумовлений стажем роботи на даній спеціальності. При стажі роботи від 3 до 5 років він складає 1,2.

З урахуванням початкових даних, використавши формулу (4.3) отримуємо витрати праці на вивчення опису завдання:

$$t_u = \frac{2376 \cdot 1{,}4}{75 \cdot 1{,}2} = 36{,}96, \text{ людино-години},$$

Витрати праці на розробку алгоритму рішення задачі визначаються за наступною формулою:

$$t_a = \frac{Q}{(20...25)\cdot k}, \text{ людино-годин,} \quad (4.4)$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (4.4), отримуємо:

$$t_a = \frac{2376}{20\cdot 1,2} = 99 \text{ людино-годин.}$$

Витрати на складання програми по готовій схемі становлять:

$$t_n = \frac{Q}{(20...25)\cdot k}, \text{ людино-годин,} \quad (4.5)$$

Підставивши відповідні значення в формулу (4.5), отримуємо:

$$t_n = \frac{2376}{25\cdot 1,2} = 79,2, \text{ людино-годин,}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{\text{отл}} = \frac{Q}{(4...5)\cdot k}, \text{ людино-годин,} \quad (4.6)$$

- за умови комплексного налагодження завдання:

$$t_{\text{отл}}^{k} = 1,5 \cdot t_{\text{отл}}, \text{ людино-години,} \quad (4.7)$$

З формули (4.6) виходить:

$$t_{\text{отл}} = \frac{2376}{5 \cdot 1,2} = 396, \text{ людино-годин,}$$

Підставивши значення в формулу (4.7), отримуємо:

$$t_{\text{отл}}^{k} = 1,5 \cdot 396 = 594, \text{ людино-години,}$$

Витрати праці на підготовку документації визначаються за формулою:

$$t_{\text{д}} = t_{\text{др}} \cdot t_{\text{до}}, \text{ людино-годин,} \quad (4.8)$$

де $t_{др}$-трудомісткість підготовки матеріалів і рукопису:

$$t_{\text{др}} = \frac{Q}{(15..20) \cdot k}, \text{ людино-годин,} \quad (4.9)$$

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{\text{до}} = 0,75 \cdot t_{\text{др}}, \text{ людино-годин,} \quad (4.10)$$

Підставляючи відповідні значення в формули (4.8-4.10) відповідно отримаємо:

$$t_{\text{др}} = \frac{2376}{18 \cdot 1,2} = 110, \text{ людино-годин,}$$

$$t_{\text{до}} = 0,75 \cdot 110 = 82,5, \text{ людино-годин,}$$

$$t_д = 110 + 82,5 = 192,5, \text{ людино-годин,}$$

Повертаючись до формули (4.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

$$t_д = 50 + 36,96 + 99 + 79,2 + 396 + 192,5 = 853,66 \approx 854,$$

людино-годин.

4.2. Розрахунок витрат на створення програмного забезпечення

Витрати на створення програмного забезпечення $K_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрати машинного часу, необхідного на налагодження програми на ЕОМ:

$$K_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн.} \quad (4.11)$$

Заробітна плата виконавців розраховується за наступною формулою:

$$З_{ЗП} = t \cdot C_{ПР}, \text{ грн} \quad (4.12)$$

де t – загальна трудомісткість, людино-годин;

$C_{ПР}$ – середня годинна заробітна плата програміста, грн/година.

Підставивши дані у формулу (4.12) отримуємо:

$$З_{ЗП} = 853,66 \cdot 200 = 170732 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{МВ} = t_{отл} \cdot C_{мч}\text{, грн.} \quad (4.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год;

$C_{мч}$ – вартість машино-години ЕОМ, грн/год (11 грн/год).

Вартість необхідного для налагодження машинного часу за (3.13):

$$З_{МВ} = 396 \cdot 11 = 4356, \text{ грн}$$

Звідси витрати на створення програмного продукту:

$$К_{ПО} = 170732 + 4356 \approx 175088 \text{ грн.}$$

Очікуваний період створення програмного забезпечення:

$$T = \frac{t}{B_k \cdot F_p}\text{, міс,}$$

де B_k – число виконавців (дорівнює 1);

F_p – місячний фонд робочого часу (при 40 годинному робочому тижні F_p=176 годин).

Звідси очікуваний період створення програмного забезпечення дорівнює:

$$T = \frac{854}{1 \cdot 176} \approx 4{,}8\text{, міс}$$

4.3. Маркетингові дослідження ринку збуту розробленого програмного продукту

Порівняно з іншими проєктами з відкритим кодом, що реалізують моделі синтезу мовлення, удосконалена модель FastSpeech1.1 пропонує

користувачам україномовний синтез, більш високу швидкість навчання, та гнучкість у вдосконаленнях. Серед наявних професійних систем синтезу українського мовлення, якість мовлення може бути дещо нижчою, але ці системи не мають відкритого коду, тож користувачі не мають жодного доступу до модифікацій систем. З цього висновок – удосконалена модель синтезу мовлення є актуальною для використання на підприємствах, що мають департамент розробки програмного забезпечення і хочуть мати можливість модифікування системи під себе за потреби. Для таких компаній впровадження системи не буде дуже коштовним, адже для навчання моделі достатньо одного не найбільш потужного графічного процесора, що не коштує дорого у рамках бюджету підприємств. Також, модель є актуальною для подальших удосконалень в інших наукових роботах, задля покращення якості мовлення та пришвидшення синтезу.

4.4. Оцінка економічної ефективності впровадження програмного забезпечення

Впровадження програмного забезпечення, наприклад, у банку, може дозволити підприємству зменшити штат своїх працівників завдяки поліпшенню взаємодії зі своїми клієнтами, а також покращити їх користувацький досвід, що має збільшити приток нових користувачів.

Числовий розрахунок економічного ефекту від впровадження розробленого програмного забезпечення на підприємстві показаний у таблиці 4.1.

Розраховані коефіцієнти економічної ефективності:

Чиста поточна вартість доходів:

$$NPU = 329\ 724 - 122\ 400 = 207\ 324 \text{ грн.} > 0$$

Строк окупності:

$$T = 122\ 400/65\ 944 = 1{,}9 \text{ року}$$

Індекс прибутковості:

ИД = 9264/3300 = 2,7

Таблиця 4.1

Розрахунок чистих грошових надходжень від розробки ПЗ

Показники, грн	За роками						Усього за 5 років	Середнє за 5 років
	0	1	2	3	4	5		
1. Інвестиції на ПЗ	122 400	-	-	-	-	-	122 400	24 480
- на придбання GPU	118 000	-	-	-	-	-	118 000	23 600
- на електро-енергію	700	-	-	-	-	-	700	140
2. Витрати до впровад-ження ПЗ	-	89 400	89 400	89 400	89 400	89 400	447 000	89 400
- на оплату праці операторів банку	-	88 000	88 000	88 000	88 000	88 000	440 000	88 000
- на електро-енергію	-	1400	1400	1400	1400	1400	7000	1400
3. Витрати після впровад-ження ПО	-	101 250	15 180	5180	5180	5180	131 970	26 394
- на корекцію помилок	-	95 000	10 000	-	-	-	105 000	21 000
- на щорічну перевірку	-	5000	5000	5000	5000	5000	25 000	5000
- на електро-енергію	-	1250	180	180	180	180	1970	394
4. Економія	-	-11 850	74 220	84 220	84 220	84 220	315 030	63 006
5. Амортиза-ція	-	28 000	28 000	28 000	28 000	10 400	122 400	24 480
6. Чисті грошові надход-ження	-	16 150	102 220	112 220	112 220	112 220	455 030	91 006
7. Коефіцієнт диском-тування	-	0,909	0,826	0,751	0,683	0,621	-	-
Дисконтові	-	14 680	84 433	84 277	76 646	69 688	329 724	65 944

грошові надход-ження								

Показник економічної ефективності (NPU – чиста поточна вартість доходів за роки реалізації впровадження (3-5 років) складе 207 324 грн тобто відповідає умовам ефективності, тому що NPU > 0.

Середній строк окупності капвкладень складе 1,8 року.

Індекс прибутковості за 5 років складе 2,7, тобто ИД > 1, проєкт варто прийняти.

Таким чином, показник ефективності свідчить про те, що дане впровадження є економічно вигідним.

4.5. Висновки

Були визначені витрати на реалізацію програмного забезпечення удосконаленої моделі синтезу мовлення FastSpeech1.1. Трудомісткість склала 854 людино-години. Вартість даного продукту склала 175 088 грн. Очікуваний час створення – 4,8 місяця. Також була порахована економічна ефективність впровадження програмного забезпечення в банку: чиста поточна вартість доходів склала 207 324 грн, строк окупності склав 1,9 року, а індекс прибутковості склав 2,7. Показник ефективності свідчить про те, що впровадження даного програмного забезпечення є економічно вигідним.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було проведено дослідження моделей синтезу мовлення, розроблено удосконалену модель синтезу мовлення FastSpeech1.1 для підвищення ефективності синтезу мовлення, проведено оцінювання якості синтезу та підраховано необхідні обчислювальні ресурси для використання різних моделей. Оцінювання якості отриманих результатів синтезу мовлення було проведено за шкалою MOS завдяки опитуванню 40 респондентів. Для візуалізації отриманих результатів було представлено таблиці та графіки.

Для вирішення поставлених задач були використані методи аналізу даних, математичної статистики, нейронних мереж, хмарних технологій, функціонального програмування, об’єктно-орієнтованого програмування.

Результати дослідження моделей синтезу мовлення можуть застосовуватись у вдосконаленні якості синтезу мовлення, пошуку нових шляхів зменшення необхідної обчислювальної потужності для їх використання, та у виборі методу синтезу українського мовлення для будь-якої сфери, у якій можна використовувати інформаційні технології.

Удосконалена модель синтезу українського мовлення FastSpeech1.1 надає нові знання про якість синтезу та вимоги до обчислювальної потужності, що дозволяють продовжувати дослідження у цій сфері та використовувати найбільш ефективні методи у розробці нових програмних продуктів. Щоб прискорити швидкість тренування FastSpeech, структура моделі “вчитель-учень” була видалена у предикторі тривалостей, і були використані мел-спектрограми справжніх аудіозаписів як ціль для навчання моделі, таким чином значним чином зменшені втрати інформації в мел-спектрограмах і швидкість синтезу мовлення була прискорена.

У економічному розділі були визначені витрати на реалізацію програмного забезпечення удосконаленої моделі синтезу мовлення FastSpeech1.1. Трудомісткість склала 854 людино-години. Вартість даного

продукту склала 175088 грн. Очікуваний час створення – 4,8 місяця. Також була порахована економічна ефективність впровадження програмного забезпечення в банку: чиста поточна вартість доходів склала 207 324 грн, строк окупності склав 1,9 року, а індекс прибутковості склав 2,7. Показник ефективності свідчить про те, що впровадження даного програмного забезпечення є економічно вигідним.

Подальші дослідження синтезу мовлення за допомогою глибинного навчання дозволять створити нові моделі синтезу мовлення, покращити натуральність та розбірливість синтезованого мовлення, зменшити частоту виникнення помилок синтезу та підвищити ефективність моделей синтезу з урахуванням обчислювальних ресурсів.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Story B. H. History of speech synthesis [Електронний ресурс] / Brad H. Story // The routledge handbook of phonetics. – Abingdon, Oxon ; New York, NY : Routledge, 2019. | Series: Routledge handbooks in linguistics, 2019. – С. 9–33. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
2. Bliss C. K. Semantography (blissymbolics) / C. K. Bliss. – 3-тє вид. – [Б. м.] : Semantography-Blissymbolics Publications, 1978. – 882 с.
3. Kurzweil reading machine (KRM) [Електронний ресурс] // Insight. – 1984. – Т. 2, № 3. – С. 89–91. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
4. Bundy A. Formant synthesis [Електронний ресурс] / Alan Bundy, Lincoln Wallen // Catalogue of artificial intelligence tools. – Berlin, Heidelberg, 1984. – С. 37. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
5. Kröger B. J. Articulatory synthesis of speech and singing: state of the art and suggestions for future research [Електронний ресурс] / Bernd J. Kröger, Peter Birkholz // Multimodal signals: cognitive and algorithmic issues. – Berlin, Heidelberg, 2009. – С. 306–319. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
6. Klatt D. H. Review of text-to-speech conversion for English [Електронний ресурс] / Dennis H. Klatt // The journal of the acoustical society of america. – 1987. – Т. 82, № 3. – С. 737–793. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
7. Black A. W. Statistical parametric speech synthesis [Електронний ресурс] / Alan W. Black, Heiga Zen, Keiichi Tokuda // 2007 IEEE international conference on acoustics, speech and signal processing - ICASSP '07, Honolulu, HI, USA, 15–20 квіт. 2007 р. – [Б. м.], 2007. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
8. Montreal forced aligner: trainable text-speech alignment using kaldi [Електронний ресурс] / Michael McAuliffe [та ін.] // Interspeech 2017. –

ISCA, 2017. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.

9. Humble FAQ - TTS 0.5.0 documentation [Електронний ресурс] // TTS 0.5.0 documentation. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
10. Librosa.feature.melspectrogram – librosa 0.8.1 documentation [Електронний ресурс] // Librosa. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
11. Text-to-Speech (TTS) engine in 119 voices | nuance | nuance [Електронний ресурс] // Nuance Communications. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
12. Text-to-Speech documentation | cloud text-to-speech documentation | google cloud [Електронний ресурс] // Google Cloud. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
13. Ling Z.-H. Modeling spectral envelopes using restricted boltzmann machines and deep belief networks for statistical parametric speech synthesis [Електронний ресурс] / Zhen-Hua Ling, Li Deng, Dong Yu // IEEE transactions on audio, speech, and language processing. – 2013. – Т. 21, № 10. – С. 2129–2139. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
14. Deng L. Binary coding of speech spectrograms using a deep auto-encoder / Li Deng, Michael Seltzer, Dong Yu // Proceedings of the 11th annual conference of the international speech communication association, INTERSPEECH 2010 : Наук. конф. – [Б. м.], 2010. – С. 1692–1695.
15. Extracting deep bottleneck features using stacked auto-encoders [Електронний ресурс] / Jonas Gehring [та ін.] // ICASSP 2013 - 2013 IEEE international conference on acoustics, speech and signal processing (ICASSP), Vancouver, BC, Canada, 26–31 трав. 2013 р. – [Б. м.], 2013. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.

16. Kang S. Multi-distribution deep belief network for speech synthesis [Електронний ресурс] / Shiyin Kang, Xiaojun Qian, Helen Meng // ICASSP 2013 - 2013 IEEE international conference on acoustics, speech and signal processing (ICASSP), Vancouver, BC, Canada, 26–31 трав. 2013 р. – [Б. м.], 2013. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
17. Zen H. Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis [Електронний ресурс] / Heiga Zen, Andrew Senior // ICASSP 2014 - 2014 IEEE international conference on acoustics, speech and signal processing (ICASSP), Florence, Italy, 4–9 трав. 2014 р. – [Б. м.], 2014. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
18. Graves A. Framewise phoneme classification with bidirectional LSTM networks [Електронний ресурс] / A. Graves, J. Schmidhuber // International joint conference on neural networks 2005, Montreal, Que., Canada. – [Б. м.]. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
19. Connectionist temporal classification [Електронний ресурс] / Alex Graves [та ін.] // The 23rd international conference, Pittsburgh, Pennsylvania, 25–29 черв. 2006 р. – New York, New York, USA, 2006. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
20. Multi-task learning of structured output layer bidirectional LSTMS for speech synthesis [Електронний ресурс] / Runnan Li [та ін.] // 2017 IEEE international conference on acoustics, speech and signal processing (ICASSP), New Orleans, LA, 5–9 берез. 2017 р. – [Б. м.], 2017. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
21. Learning cross-lingual information with multilingual BLSTM for speech synthesis of low-resource languages [Електронний ресурс] / Quanjie Yu [та ін.] // 2016 IEEE international conference on acoustics, speech and signal processing (ICASSP), Shanghai, 20–25 берез. 2016 р. – [Б. м.], 2016. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.

22. RHVoice.org [Електронний ресурс] // RHVoice.org. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
23. UkrVox 4.2 завантажити безкоштовно - Безкоштовні програми [Електронний ресурс] // Скачать бесплатные программы. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
24. Wang W. First step towards end-to-end parametric TTS synthesis: generating spectral parameters with neural attention [Електронний ресурс] / Wenfu Wang, Shuang Xu, Bo Xu // Interspeech 2016. – [Б. м.], 2016. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
25. Bahdanau D. Neural machine translation by jointly learning to align and translate / Dzmitry Bahdanau, Kyunghyun Cho // 3rd international conference on learning representations. – [Б. м.], 2015.
26. Sotelo J. Char2Wav: end-to-end speech synthesis / Jose Sotelo, Soroush Mehri, Kundan Kumar // In proceedings of the international conference on learning representations workshop. – [Б. м.], 2017.
27. Acoustic fall detection using Gaussian mixture models and GMM supervectors [Електронний ресурс] / Xiaodan Zhuang [та ін.] // ICASSP 2009 - 2009 IEEE international conference on acoustics, speech and signal processing, Taipei, Taiwan, 19–24 квіт. 2009 р. – [Б. м.], 2009. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
28. Aäron van den Oord. WaveNet: a generative model for raw audio [Електронний ресурс] / Aäron van den Oord, Sander Dieleman, Heiga Zen // arXiv.org. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
29. Aaron van den Oord. Conditional image generation with pixelcnn decoders [Електронний ресурс] / Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals // arXiv.org. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.

30. Aaron van den Oord. Pixel recurrent neural networks [Електронний ресурс] / Aaron van den Oord, Nal Kalchbrenner, Koray Kavukcuoglu // arXiv.org. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
31. Aaron van den Oord. Parallel wavenet: fast high-fidelity speech synthesis [Електронний ресурс] / Aaron van den Oord, Yazhe Li, Igor Babuschkin // arXiv.org. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
32. Arik S. O. Deep voice: real-time neural text-to-speech [Електронний ресурс] / Sercan O. Arik, Mike Chrzanowski, Adam Coates // arXiv.org. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
33. Wang Y. Tacotron: towards end-to-end speech synthesis [Електронний ресурс] / Yuxuan Wang, Rj Skerry-Ryan, Daisy Stanton // arXiv.org. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
34. Gehring J. Convolutional sequence to sequence learning [Електронний ресурс] / Jonas Gehring, Michael Auli, David Grangier // arXiv.org. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
35. CyberMova/VymovaPlus/VymovaPro [Електронний ресурс] // Ukrainian Speech and Language Resources and Software - CyberMova. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
36. GitHub - kyubyong/tacotron: a tensorflow implementation of tacotron: a fully end-to-end text-to-speech synthesis model [Електронний ресурс] // GitHub. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
37. GitHub - keithito/tacotron: A TensorFlow implementation of Google's Tacotron speech synthesis with pre-trained model (unofficial) [Електронний ресурс] // GitHub. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
38. GitHub - r9y9/tacotron_pytorch: pytorch implementation of tacotron speech synthesis model. [Електронний ресурс] // GitHub. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.

39. Розмовлялка [Електронний ресурс] // archive.ph. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
40. Möbius B. Rare events and closed domains: two delicate concepts in speech synthesis / Bernd Möbius // Proceedings of the 4th ESCA workshop on speech synthesis. – [Б. м.], 2001.
41. Arik S. Deep Voice 2: Multi-Speaker Neural Text-to-Speech [Електронний ресурс] / Sercan Arik, Gregory Diamos, Andrew Gibiansky // arXiv.org. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
42. Natural TTS synthesis by conditioning wavenet on MEL spectrogram predictions [Електронний ресурс] / Jonathan Shen [та ін.] // ICASSP 2018 - 2018 IEEE international conference on acoustics, speech and signal processing (ICASSP), Calgary, AB, 15–20 квіт. 2018 р. – [Б. м.], 2018. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
43. P.10 : vocabulary for performance, quality of service and quality of experience [Електронний ресурс] // ITU: Committed to connecting the world. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
44. Tachibana H. Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention [Електронний ресурс] / Hideyuki Tachibana, Katsuya Uenoyama, Shunsuke Aihara // ICASSP 2018 - 2018 IEEE international conference on acoustics, speech and signal processing (ICASSP), Calgary, AB, 15–20 квіт. 2018 р. – [Б. м.], 2018. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
45. Singing voice synthesis based on deep neural networks [Електронний ресурс] / Masanari Nishimura [та ін.] // Interspeech 2016. – [Б. м.], 2016. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
46. Ping W. Deep voice 3: scaling text-to-speech with convolutional sequence learning [Електронний ресурс] / Wei Ping, Kainan Peng, Andrew Gibiansky // arXiv.org. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.

47. Vainer J. SpeedySpeech: efficient neural speech synthesis [Електронний ресурс] / Jan Vainer, Ondřej Dušek // Interspeech 2020. – ISCA, 2020. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
48. Ren Y. FastSpeech: fast, robust and controllable text to speech [Електронний ресурс] / Yi Ren, Yangjun Ruan, Xu Tan // arXiv.org. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
49. Kumar K. MelGAN: generative adversarial networks for conditional waveform synthesis [Електронний ресурс] / Kundan Kumar, Rithesh Kumar, Thibault de Boissiere // arXiv.org. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
50. GitHub - jaywalnut310/glow-tts: a generative flow for text-to-speech via monotonic alignment search [Електронний ресурс] // GitHub. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
51. GitHub - paddlepaddle/paddlespeech: easy-to-use speech toolkit including SOTA ASR pipeline, influential TTS with text frontend and end-to-end speech simultaneous translation. [Електронний ресурс] // GitHub. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
52. Vaswani A. Attention is all you need / Ashish Vaswani, Noam Shazeer, Niki Parmar // Advances in neural information processing systems 30 (NIPS 2017). – [Б. м.], 2017.
53. Durkan C. Neural spline flows [Електронний ресурс] / Conor Durkan, Artur Bekasov, Iain Murray // arXiv.org. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
54. Hoogeboom E. Emerging convolutions for generative normalizing flows [Електронний ресурс] / Emiel Hoogeboom, Rianne van den Berg, Max Welling // arXiv.org. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
55. GitHub - rayhane-mamah/tacotron-2: deepmind's tacotron-2 tensorflow implementation [Електронний ресурс] // GitHub. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.

56. GitHub - coqui-ai/TTS: - a deep learning toolkit for Text-to-Speech, battle-tested in research and production [Електронний ресурс] // GitHub. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
57. Dinh L. Density estimation using Real NVP [Електронний ресурс] / Laurent Dinh, Jascha Sohl-Dickstein, Samy Bengio // arXiv.org. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
58. Kingma D. P. Glow: generative flow with invertible 1x1 convolutions [Електронний ресурс] / Diederik P. Kingma, Prafulla Dhariwal // arXiv.org. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
59. Kim J. Glow-TTS: a generative flow for text-to-speech via monotonic alignment search [Електронний ресурс] / Jaehyeon Kim, Sungwon Kim, Jungil Kong // arXiv.org. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
60. The LJ speech dataset [Електронний ресурс] // Keith Ito. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
61. The M-AILABS Speech Dataset – caito [Електронний ресурс] // caito – Independent? Yes – Intelligent? Maybe – Impartial? Never. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
62. Davis S. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences [Електронний ресурс] / S. Davis, P. Mermelstein // IEEE transactions on acoustics, speech, and signal processing. – 1980. – Т. 28, № 4. – С. 357–366. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
63. GitHub - mozilla/tts: deep learning for text to speech (discussion forum: https://discourse.mozilla.org/c/tts) [Електронний ресурс] // GitHub. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
64. Morise M. WORLD: a vocoder-based high-quality speech synthesis system for real-time applications [Електронний ресурс] / Masanori Morise, Fumiya Yokomori, Kenji Ozawa // IEICE transactions on information and systems. –

2016. – E99.D, № 7. – С. 1877–1884. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.

65. Wu Z. Merlin: an open source neural network speech synthesis system [Електронний ресурс] / Zhizheng Wu, Oliver Watts, Simon King // 9th ISCA speech synthesis workshop. – [Б. м.], 2016. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
66. Li N. Neural speech synthesis with transformer network [Електронний ресурс] / Naihan Li, Shujie Liu, Yanqing Liu // arXiv.org. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
67. Prenger R. Waveglow: a flow-based generative network for speech synthesis [Електронний ресурс] / Ryan Prenger, Rafael Valle, Bryan Catanzaro // ICASSP 2019 - 2019 IEEE international conference on acoustics, speech and signal processing (ICASSP), Brighton, United Kingdom, 12–17 трав. 2019 р. – [Б. м.], 2019. – Режим доступу: (дата звернення: 14.01.2022). – Назва з екрана.
68. Text to speech, voice recognition – Yandex SpeechKit [Електронний ресурс]. – Режим доступу: https://cloud.yandex.com/en/services/speechkit (дата звернення: 17.01.2022). – Назва з екрана.
69. 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.

Додаток А

КОД ПРОГРАМИ

Блок з підключенням проєкт до Google Drive

```
from google.colab import drive
# Load the TensorBoard notebook extension
%load_ext tensorboard

drive.mount('/content/gdrive')

%cd /content/gdrive/MyDrive/Colab \Notebooks
```

Блок завантаження Coqui TTS з Github у Google Drive

```
import os
from os.path import exists, join, expanduser

# Clone
%cd "coqui-ai"

name = "TTS"
if not exists(name):
  ! git clone https://github.com/coqui-ai/$name

%cd $name
```

Блок встановлення пакетів Python проєкт Coqui TTS

```
!pip install -e .
!pip install TTS
```

Блок запуску тренування Glow-TTS

```
gpu_info = !nvidia-smi
gpu_info = '\n'.join(gpu_info)
if gpu_info.find('failed') >= 0:
  print('Not connected to a GPU')
else:
  print(gpu_info)

!python TTS/bin/train_tts.py --restore_path ukrainian/coqui_tts-December-20-2021_01+33AM-
0000000/checkpoint_92000.pth.tar --continue_path ukrainian/coqui_tts-December-20-2021_01+33AM-
0000000
```

Блок запуску тренування SpeedySpeech

```
gpu_info = !nvidia-smi
gpu_info = '\n'.join(gpu_info)
if gpu_info.find('failed') >= 0:
  print('Not connected to a GPU')
else:
  print(gpu_info)

#%cd "recipes/ljspeech/speedy_speech"
#!python train_speedy_speech.py

#!python TTS/bin/train_tts.py --config_path config.speedy_speech.json #can be continued
!python TTS/bin/train_tts.py --restore_path ukrainian/speedy_speech_ljspeech-January-07-
2022_01+00PM-0000000/best_model.pth.tar --continue_path ukrainian/speedy_speech_ljspeech-January-07-
2022_01+00PM-0000000
```

Блок запуску тренування Tacotron2

```
gpu_info = !nvidia-smi
gpu_info = '\n'.join(gpu_info)
if gpu_info.find('failed') >= 0:
  print('Not connected to a GPU')
else:
```

```
  print(gpu_info)

#%cd "recipes/ljspeech/tacotron2-DDC"
#!python train_tacotron_ddc.py

#!python TTS/bin/train_tts.py --config_path config.tacotron2_DDC.json
!python TTS/bin/train_tts.py --restore_path ukrainian/tacotron2-DDC-December-27-2021_09+08PM-
0000000/checkpoint_99000.pth.tar --continue_path ukrainian/tacotron2-DDC-December-27-2021_09+08PM-
0000000
```

Блок запуску тренування FastSpeech

```
gpu_info = !nvidia-smi
gpu_info = '\n'.join(gpu_info)
if gpu_info.find('failed') >= 0:
  print('Not connected to a GPU')
else:
  print(gpu_info)

#%cd "recipes/ljspeech/fast_speech"
#!python train_fast_speech.py

#!python TTS/bin/train_tts.py --config_path config.fast_speech.json
!python TTS/bin/train_tts.py --restore_path ukrainian/fast_speech-December-28-2021_10+58PM-
0000000/checkpoint_200000.pth.tar --continue_path ukrainian/fast_speech-December-28-2021_10+58PM-
0000000
```

Блок підключення Tensorboard у log-директорії Glow-TTS

```
%tensorboard --logdir ukrainian/coqui_tts-December-20-2021_01+33AM-0000000
```

Блок підключення Tensorboard у log-директорії SpeedySpeech

```
%tensorboard --logdir ukrainian/speedy_speech_ljspeech-December-24-2021_07+38PM-0000000
```

Блок підключення Tensorboard у log-директорії Tacotron2

```
%tensorboard --logdir ukrainian/tacotron2-DDC-December-27-2021_09+08PM-0000000
```

Блок підключення Tensorboard у log-директорії FastSpeech

```
%tensorboard --logdir ukrainian/fast_speech-December-28-2021_10+58PM-0000000
```

Блок запуску Glow-TTS для синтезу мовлення з набором речень для оцінювання синтезу

```
arr = ["А як я свисну за садком, чи вийдеш?",
       "Вже пізненько, як усі богомольці поснули таким сном, що їх не розбудили б і лаврські дзвони,
чернець вийшов з келії.",
"Мотря оступилась.",
"Погана, бо іродів шинкар розводить водою, – проморив Балаш. – Бодай йому смерть така, як оця
горілка.",
"Лаврін прикрив Мелашчині плечі своєю свитою і обняв її за стан.",
"А ти чого оступаєшся за своєю жінкою? – крикнув він на Карпа. – Коли хочеш, то я тобі носа втру.",
"За що їм давати половину? Чи то можна? Це вони схотять, щоб ми давали їм половину картоплі та
буряків. Це все свекруха наговорює в волості.",
"Вони втрьох ходили, розпитували, чи не чув хто, чи не бачив молодої чорнявої молодиці, чи не
шукала, чи не питала вона про своїх людей.",
"Вже й знайшов красуню! Та в неї лице, як тріска, стан, наче копистка, руки, як кочерги, сама, як
дошка, а як іде, то аж кістки торохтять.",
"Але як тільки вона трохи сердилась, з неї спадала та солодка луска, і вона лаялась і кричала на
ввесь рот. Маруся була сердита.",
"З того часу Лаврін загарбав хазяйство в свої руки. Батько мусив мовчати і рідко коли вмикувався в
хазяйство.",
"Не встиг він закласти заноза в ярмо і ненароком кинув очима за Рось: за Россю, коло скелі на
долині, вкритій зеленим житом, червоніла якась велика квітка.",
"Якого це ти нечистого так ляпаєш? Ще мало сміття в хаті, то нехай буде грязь, – крикнула Кайдашиха.
– Чом ти своїй жінці нічого не скажеш? – сказала Кайдашиха до Карпа.",
"Балаш з Кайдашем посідали кінець стола і почали розмовляти за жито та пшеницю, за сіно та за ярину.
Балашиха послала найстаршого хлопця в шинок по горілку.",
"Оцього я вже не люблю! – крикнула Мотря.",
"Хоч між дровами, аби з чорними бровами! – сказав Лаврін. – Хоч під лавкою, аби з гарною панянкою.",
"Ого-го, наша пані економша вилізла трохи не під небо! – загомоніли чоловіки. – Видно, що їде на
розглядини.",
"Лаврін знав парубоцький звичай і повів усю парубочу ватагу в шинок. Він поставив їм могорича і вже
```

```
в згоді з ними вернувся на вулицю.",
"Даруй же, боже, нам і нашим дітям вік довгий та щасливий, щоб ти, моя доню, була здорова, як вода,
щоб цвіла довіку, як рожа.",
"Боже мій! Де це я? – говорив сам до себе Кайдаш."
       ]
i = 0
for item in arr:
  i = i + 1
  !tts --text "$item" \
    --config_path config.json \
    --model_path ukrainian/coqui_tts-December-20-2021_01+33AM-0000000/checkpoint_116000.pth.tar \
      --out_path ukrainian/output/glow_tts/"$i".mp3
```

Блок запуску SpeedySpeech для синтезу мовлення з набором речень для оцінювання синтезу

```
arr = ["А як я свисну за садком, чи вийдеш?",
       "Вже пізненько, як усі богомольці поснули таким сном, що їх не розбудили б і лаврські дзвони,
чернець вийшов з келії.",
"Мотря оступилась.",
"Погана, бо іродів шинкар розводить водою, – проморив Балаш. – Бодай йому смерть така, як оця
горілка.",
"Лаврін прикрив Мелашчині плечі своєю свитою і обняв її за стан.",
"А ти чого оступаєшся за своєю жінкою? – крикнув він на Карпа. – Коли хочеш, то я тобі носа втру.",
"За що їм давати половину? Чи то можна? Це вони схотять, щоб ми давали їм половину картоплі та
буряків. Це все свекруха наговорює в волості.",
"Вони втрьох ходили, розпитували, чи не чув хто, чи не бачив молодої чорнявої молодиці, чи не
шукала, чи не питала вона про своїх людей.",
"Вже й знайшов красуню! Та в неї лице, як тріска, стан, наче копистка, руки, як кочерги, сама, як
дошка, а як іде, то аж кістки торохтять.",
"Але як тільки вона трохи сердилась, з неї спадала та солодка луска, і вона лаялась і кричала на
ввесь рот. Маруся була сердита.",
"З того часу Лаврін загарбав хазяйство в свої руки. Батько мусив мовчати і рідко коли вмикувався в
хазяйство.",
"Не встиг він закласти заноза в ярмо і ненароком кинув очима за Рось: за Россю, коло скелі на
долині, вкритій зеленим житом, червоніла якась велика квітка.",
"Якого це ти нечистого так ляпаєш? Ще мало сміття в хаті, то нехай буде грязь, – крикнула Кайдашиха.
– Чом ти своїй жінці нічого не скажеш? – сказала Кайдашиха до Карпа.",
"Балаш з Кайдашем посідали кінець стола і почали розмовляти за жито та пшеницю, за сіно та за ярину.
Балашиха послала найстаршого хлопця в шинок по горілку.",
"Оцього я вже не люблю! – крикнула Мотря.",
"Хоч між дровами, аби з чорними бровами! – сказав Лаврін. – Хоч під лавкою, аби з гарною панянкою.",
"Ого-го, наша пані економша вилізла трохи не під небо! – загомоніли чоловіки. – Видно, що їде на
розглядини.",
"Лаврін знав парубоцький звичай і повів усю парубочу ватагу в шинок. Він поставив їм могорича і вже
в згоді з ними вернувся на вулицю.",
"Даруй же, боже, нам і нашим дітям вік довгий та щасливий, щоб ти, моя доню, була здорова, як вода,
щоб цвіла довіку, як рожа.",
"Боже мій! Де це я? – говорив сам до себе Кайдаш."
       ]
i = 0
for item in arr:
  i = i + 1
  !tts --text "$item" \
    --config_path ukrainian/speedy_speech_ljspeech-January-07-2022_01+00PM-0000000/config.json \
    --model_path ukrainian/speedy_speech_ljspeech-January-07-2022_01+00PM-0000000/best_model.pth.tar
\
      --out_path ukrainian/output/speedy_speech/"$i".mp3
```

Блок запуску Tacotron2 для синтезу мовлення з набором речень для оцінювання синтезу

```
arr = ["А як я свисну за садком, чи вийдеш?",
       "Вже пізненько, як усі богомольці поснули таким сном, що їх не розбудили б і лаврські дзвони,
чернець вийшов з келії.",
"Мотря оступилась.",
"Погана, бо іродів шинкар розводить водою, – проморив Балаш. – Бодай йому смерть така, як оця
горілка.",
"Лаврін прикрив Мелашчині плечі своєю свитою і обняв її за стан.",
"А ти чого оступаєшся за своєю жінкою? – крикнув він на Карпа. – Коли хочеш, то я тобі носа втру.",
"За що їм давати половину? Чи то можна? Це вони схотять, щоб ми давали їм половину картоплі та
буряків. Це все свекруха наговорює в волості.",
"Вони втрьох ходили, розпитували, чи не чув хто, чи не бачив молодої чорнявої молодиці, чи не
шукала, чи не питала вона про своїх людей.",
"Вже й знайшов красуню! Та в неї лице, як тріска, стан, наче копистка, руки, як кочерги, сама, як
дошка, а як іде, то аж кістки торохтять.",
"Але як тільки вона трохи сердилась, з неї спадала та солодка луска, і вона лаялась і кричала на
ввесь рот. Маруся була сердита.",
"З того часу Лаврін загарбав хазяйство в свої руки. Батько мусив мовчати і рідко коли вмикувався в
```

```
хазяйство.",
"Не встиг він закласти заноза в ярмо і ненароком кинув очима за Рось: за Россю, коло скелі на
долині, вкритій зеленим житом, червоніла якась велика квітка.",
"Якого це ти нечистого так ляпаєш? Ще мало сміття в хаті, то нехай буде грязь, – крикнула Кайдашиха.
– Чом ти своїй жінці нічого не скажеш? – сказала Кайдашиха до Карпа.",
"Балаш з Кайдашем посідали кінець стола і почали розмовляти за жито та пшеницю, за сіно та за ярину.
Балашиха послала найстаршого хлопця в шинок по горілку.",
"Оцього я вже не люблю! – крикнула Мотря.",
"Хоч між дровами, аби з чорними бровами! – сказав Лаврін. – Хоч під лавкою, аби з гарною панянкою.",
"Ого-го, наша пані економша вилізла трохи не під небо! – загомоніли чоловіки. – Видно, що їде на
розглядини.",
"Лаврін знав парубоцький звичай і повів усю парубочу ватагу в шинок. Він поставив їм могорича і вже
в згоді з ними вернувся на вулицю.",
"Даруй же, боже, нам і нашим дітям вік довгий та щасливий, щоб ти, моя доню, була здорова, як вода,
щоб цвіла довіку, як рожа.",
"Боже мій! Де це я? – говорив сам до себе Кайдаш."
       ]
i = 0
for item in arr:
  i = i + 1
  !tts --text "$item" \
      --config_path ukrainian/tacotron2-DDC-December-27-2021_09+08PM-0000000/config.json \
      --model_path ukrainian/tacotron2-DDC-December-27-2021_09+08PM-0000000/checkpoint_99000.pth.tar
\
      --out_path ukrainian/output/tacotron2_DDC/"$i".mp3
```

Блок запуску FastSpeech для синтезу мовлення з набором речень для оцінювання синтезу

```
arr = ["А як я свисну за садком, чи вийдеш?",
       "Вже пізненько, як усі богомольці поснули таким сном, що їх не розбудили б і лаврські дзвони,
чернець вийшов з келії.",
"Мотря оступилась.",
"Погана, бо іродів шинкар розводить водою, – проморив Балаш. – Бодай йому смерть така, як оця
горілка.",
"Лаврін прикрив Мелашчині плечі своєю свитою і обняв її за стан.",
"А ти чого оступаєшся за своєю жінкою? – крикнув він на Карпа. – Коли хочеш, то я тобі носа втру.",
"За що їм давати половину? Чи то можна? Це вони схотять, щоб ми давали їм половину картоплі та
буряків. Це все свекруха наговорює в волості.",
"Вони втрьох ходили, розпитували, чи не чув хто, чи не бачив молодої чорнявої молодиці, чи не
шукала, чи не питала вона про своїх людей.",
"Вже й знайшов красуню! Та в неї лице, як тріска, стан, наче кописткa, руки, як кочерги, сама, як
дошка, а як іде, то аж кістки торохтять.",
"Але як тільки вона трохи сердилась, з неї спадала та солодка луска, і вона лаялась і кричала на
ввесь рот. Маруся була сердита.",
"З того часу Лаврін загарбав хазяйство в свої руки. Батько мусив мовчати і рідко коли вмикувався в
хазяйство.",
"Не встиг він закласти заноза в ярмо і ненароком кинув очима за Рось: за Россю, коло скелі на
долині, вкритій зеленим житом, червоніла якась велика квітка.",
"Якого це ти нечистого так ляпаєш? Ще мало сміття в хаті, то нехай буде грязь, – крикнула Кайдашиха.
– Чом ти своїй жінці нічого не скажеш? – сказала Кайдашиха до Карпа.",
"Балаш з Кайдашем посідали кінець стола і почали розмовляти за жито та пшеницю, за сіно та за ярину.
Балашиха послала найстаршого хлопця в шинок по горілку.",
"Оцього я вже не люблю! – крикнула Мотря.",
"Хоч між дровами, аби з чорними бровами! – сказав Лаврін. – Хоч під лавкою, аби з гарною панянкою.",
"Ого-го, наша пані економша вилізла трохи не під небо! – загомоніли чоловіки. – Видно, що їде на
розглядини.",
"Лаврін знав парубоцький звичай і повів усю парубочу ватагу в шинок. Він поставив їм могорича і вже
в згоді з ними вернувся на вулицю.",
"Даруй же, боже, нам і нашим дітям вік довгий та щасливий, щоб ти, моя доню, була здорова, як вода,
щоб цвіла довіку, як рожа.",
"Боже мій! Де це я? – говорив сам до себе Кайдаш."
       ]
i = 0
for item in arr:
  i = i + 1
  !tts --text "$item" \
      --config_path ukrainian/fast_speech-December-28-2021_10+58PM-0000000/config.json \
```

```
      --model_path  ukrainian/fast_speech-December-28-2021_10+58PM-0000000/checkpoint_246000.pth.tar
\
      --out_path ukrainian/output/fast_speech/"$i".mp3
```

Блок запуску Glow-TTS для синтезу мовлення з набором складних речень для ручної перевірки

```
arr = ["Бабин біб розцвів у дощ - буде бабі біб у борщ.",
       "Був собі цебер, та переполуцебрився на полуцебренята.",
       "Бобер на березі з бобренятами бублики пік.",
       "Боронила борона по боронованому полю.",
       "Бук бундючився перед дубом, тряс над дубом бурим чубом. Дуб пригнув до чуба бука. Буде
букові наука.",
       "Варка варила вареника, Василь взяв вареника. Варка Василя варехою. Василь Варку вареником.",
       "Вередували вереднички, що не зварили вареничків. Не вередуйте, вередниченьки, ось поваряться
варениченьки.",
       "Викис, вимок, виліз, висох, став на колоду, та знов - бовть у воду.",
       "Ворона проворонила вороненя.",
       "Наш садівник розсадівникувався.",
       "Вовк-вовцюг вівцю волік. Вова вовку - вила в бік. Як завив же вовк-вовцюг, миттю випустив
вівцю.",
       "Галасливі ґави й галки в гусенят взяли скакалки. Гусенята їм ґелґочуть, що й вони скакати
хочуть.",
       "Дрова рубали два дроворуби.",
       "У нас надворі погода розмокропогодилася.",
       "Дзижчить над житом жвавий жук, бо жовтий він вдягнув кожух.",
       "Біжать стежини поміж ожини.",
       "Захар заліз на перелаз. - Захарку, злізь! - Захарку, злазь! Зумів залізти - Знай, як
злізти.",
       "Кинув кріп Прокіп в окріп. У окропі, окрім кропу кипить короп для Прокопа.",
       "Коваль кулю кував, кував і перековував.",
       "Ходить квочка коло кілочка, водить діток, дрібних квіток."
       ]
i = 0
for item in arr:
  i = i + 1
  !tts --text "$item" \
    --config_path config.json \
    --model_path ukrainian/coqui_tts-December-20-2021_01+33AM-0000000/checkpoint_116000.pth.tar \
      --out_path ukrainian/output/glow_tts/"err_$i".mp3
```

Блок запуску SpeedySpeech для синтезу мовлення з набором складних речень для ручної перевірки

```
arr = ["Бабин біб розцвів у дощ - буде бабі біб у борщ.",
       "Був собі цебер, та переполуцебрився на полуцебренята.",
       "Бобер на березі з бобренятами бублики пік.",
       "Боронила борона по боронованому полю.",
       "Бук бундючився перед дубом, тряс над дубом бурим чубом. Дуб пригнув до чуба бука. Буде
букові наука.",
       "Варка варила вареника, Василь взяв вареника. Варка Василя варехою. Василь Варку вареником.",
       "Вередували вереднички, що не зварили вареничків. Не вередуйте, вередниченьки, ось поваряться
варениченьки.",
       "Викис, вимок, виліз, висох, став на колоду, та знов - бовть у воду.",
       "Ворона проворонила вороненя.",
       "Наш садівник розсадівникувався.",
       "Вовк-вовцюг вівцю волік. Вова вовку - вила в бік. Як завив же вовк-вовцюг, миттю випустив
вівцю.",
       "Галасливі ґави й галки в гусенят взяли скакалки. Гусенята їм ґелґочуть, що й вони скакати
хочуть.",
       "Дрова рубали два дроворуби.",
       "У нас надворі погода розмокропогодилася.",
       "Дзижчить над житом жвавий жук, бо жовтий він вдягнув кожух.",
       "Біжать стежини поміж ожини.",
       "Захар заліз на перелаз. - Захарку, злізь! - Захарку, злазь! Зумів залізти - Знай, як
злізти.",
       "Кинув кріп Прокіп в окріп. У окропі, окрім кропу кипить короп для Прокопа.",
       "Коваль кулю кував, кував і перековував.",
       "Ходить квочка коло кілочка, водить діток, дрібних квіток."
       ]
i = 0
for item in arr:
  i = i + 1
  !tts --text "$item" \
    --config_path ukrainian/speedy_speech_ljspeech-January-07-2022_01+00PM-0000000/config.json \
    --model_path ukrainian/speedy_speech_ljspeech-January-07-2022_01+00PM-0000000/best_model.pth.tar
\
      --out_path ukrainian/output/speedy_speech/"err_$i".mp3
```

Блок запуску Tacotron2 для синтезу мовлення з набором складних речень для ручної перевірки

```
arr = ["Бабин біб розцвів у дощ - буде бабі біб у борщ.",
       "Був собі цебер, та переполуцебрився на полуцебренята.",
       "Бобер на березі з бобренятами бублики пік.",
       "Боронила борона по боронованому полю.",
       "Бук бундючився перед дубом, тряс над дубом бурим чубом. Дуб пригнув до чуба бука. Буде
букові наука.",
       "Варка варила вареника, Василь взяв вареника. Варка Василя варехою. Василь Варку вареником.",
       "Вередували вередничкі, що не зварили вареничків. Не вередуйте, вередниченьки, ось поваряться
варениченьки.",
       "Викис, вимок, виліз, висох, став на колоду, та знов - бовть у воду.",
       "Ворона проворонила вороненя.",
       "Наш садівник розсадівникувався.",
       "Вовк-вовцюг вівцю волік. Вова вовку - вила в бік. Як завив же вовк-вовцюг, миттю випустив
вівцю.",
       "Галасливі ґави й ґалки в гусенят взяли скакалки. Гусенята їм ґелґочуть, що й вони скакати
хочуть.",
       "Дрова рубали два дроворуби.",
       "У нас надворі погода розмокропогодилася.",
       "Дзижчить над житом жвавий жук, бо жовтий він вдягнув кожух.",
       "Біжать стежини поміж ожини.",
       "Захар заліз на перелаз. - Захарку, злізь! - Захарку, злазь! Зумів залізти - Знай, як
злізти.",
       "Кинув кріп Прокіп в окріп. У окропі, окрім кропу кипить короп для Прокопа.",
       "Коваль кулю кував, кував і перековував.",
       "Ходить квочка коло кілочка, водить діток, дрібних квіток."
       ]
i = 0
for item in arr:
  i = i + 1
  !tts --text "$item" \
    --config_path ukrainian/tacotron2-DDC-December-27-2021_09+08PM-0000000/config.json \
    --model_path ukrainian/tacotron2-DDC-December-27-2021_09+08PM-0000000/best_model.pth.tar \
      --out_path ukrainian/output/tacotron2_DDC/"err_$i".mp3
```

Блок запуску FastSpeech для синтезу мовлення з набором складних речень для ручної перевірки

```
arr = ["Бабин біб розцвів у дощ - буде бабі біб у борщ.",
       "Був собі цебер, та переполуцебрився на полуцебренята.",
       "Бобер на березі з бобренятами бублики пік.",
       "Боронила борона по боронованому полю.",
       "Бук бундючився перед дубом, тряс над дубом бурим чубом. Дуб пригнув до чуба бука. Буде
букові наука.",
       "Варка варила вареника, Василь взяв вареника. Варка Василя варехою. Василь Варку вареником.",
       "Вередували вередничкі, що не зварили вареничків. Не вередуйте, вередниченьки, ось поваряться
варениченьки.",
       "Викис, вимок, виліз, висох, став на колоду, та знов - бовть у воду.",
       "Ворона проворонила вороненя.",
       "Наш садівник розсадівникувався.",
       "Вовк-вовцюг вівцю волік. Вова вовку - вила в бік. Як завив же вовк-вовцюг, миттю випустив
вівцю.",
       "Галасливі ґави й ґалки в гусенят взяли скакалки. Гусенята їм ґелґочуть, що й вони скакати
хочуть.",
       "Дрова рубали два дроворуби.",
       "У нас надворі погода розмокропогодилася.",
       "Дзижчить над житом жвавий жук, бо жовтий він вдягнув кожух.",
       "Біжать стежини поміж ожини.",
       "Захар заліз на перелаз. - Захарку, злізь! - Захарку, злазь! Зумів залізти - Знай, як
злізти.",
       "Кинув кріп Прокіп в окріп. У окропі, окрім кропу кипить короп для Прокопа.",
       "Коваль кулю кував, кував і перековував.",
       "Ходить квочка коло кілочка, водить діток, дрібних квіток."
       ]
i = 0
for item in arr:
```

```
i = i + 1
!tts --text "$item" \
  --config_path ukrainian/fast_speech-December-28-2021_10+58PM-0000000/config.json \
  --model_path ukrainian/fast_speech-December-28-2021_10+58PM-0000000/checkpoint_246000.pth.tar \
    --out_path ukrainian/output/fast_speech/"err_$i".mp3
```

Додаток Б

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ДНІПРОВСЬКА ПОЛІТЕХНІКА»

Факультет інформаційних технологій
Кафедра програмного забезпечення комп'ютерних систем

ВІДГУК

Керівника
економічної
частини Ваганової О.Г, д.е.н., професора каф. ПЕП та ПУ
(прізвище, ім'я, по батькові, вчене звання, посада, місце роботи)

На кваліфікаційну роботу
студента Ліс'їх Артема Ігоровича
(прізвище, ім'я, по батькові)

курсу II групи _____121м-20-1_____
спеціальності 121 Інженерія програмного забезпечення
на тему Дослідження ефективності синтезу мовлення з урахуванням обчислювальної потужності

.

«___»_____2022 р. _____
(підпис)

Додаток В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом_Ліс’їх.doc	Пояснювальна записка роботи. Документ Word.
Диплом_Ліс’їх.pdf	Пояснювальна записка роботи в форматі PDF.
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму.
Презентація	
Презентація_Ліс’їх.ppt	Презентація кваліфікаційної роботи.