НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

На правах рукопису
УДК 004.896

До захисту допущено
В. о. завідувача кафедри ММСА
О.Л.Тимощук
«___» _____ 2018 р.

Магістерська дисертація

на здобуття ступеня магістра за спеціальністю 122 Комп'ютерні науки
на тему: «Застосування методів штучного інтелекту в задачі автоматичного розпізнавання мовлення»

Виконав:
студент II курсу, групи КА-74 мп
Бех Петро Васильович _____

Керівник: Професор кафедри ММСА
д.т.н., професор, Зайченко Ю.П. _____

Рецензент: Старший науковий співробітник
кафедри програмного забезпечення комп'ютерних систем
КПІ ім. Ігоря Сікорського
д.т.н., с.н.с, Вішталь Д.М. _____

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

Київ
2018

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Рівень вищої освіти – другий (магістерський)
Спеціальність (спеціалізація) – 122 «Комп'ютерні науки» («Інтелектуальний аналіз даних в управлінні проектами»)

ЗАТВЕРДЖУЮ
В. о. завідувача кафедри ММСА
О. Л. Тимощук
«___» _____ 2018 р.

ЗАВДАННЯ

на магістерську дисертацію студенту Беху Петру Васильовичу

1. Тема дисертації: «Дослідження застосування методів штучного інтелекту в задачі автоматичного розпізнавання мовлення», науковий керівник дисертації Зайченко Юрій Петрович, доктор технічних наук, професор кафедри ММСА, затверджені наказом по університету від «07» листопада 2018 р. № 4121-с

2.Термін подання студентом дисертації: _____

3. Об'єкт дослідження: алгоритми автоматичного розпізнавання мовлення.

4. Предмет дослідження: алгоритми автоматичного розпізнавання мовлення.

5. Перелік завдань, які потрібно розробити:

1) провести аналіз методів штучного інтелекту для вирішення прикладних задач;
2) вибрати навчальний набір даних;
3) синтезувати архітектуру нейронної мережі;
4) провести навчання синтезованої мережі;
5) розробити веб-інтерфейс для завантаження звукових файлів;

6) виконати тестування програмної частини системи

6. Орієнтовний перелік публікацій:

(1) Science and Technology of the XXI Century : the XVIII All-Ukrainian Students R&D Conference Proceeding, (Kyiv, December 07, 2017) / National Technical University of Ukraine „Igor Sikorsky Kyiv Polytechnic Institute". – Part IV. – Kyiv, 2017. – pp. 22-25

7. Дата видачі завдання: _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Написання вступу магістерської дисертації	07.09.2018–13.09.2018	
2.	Порівняльний аналіз підходів по вирішенню поставленої задачі, вибір архітектур моделей для порівняння	14.09.2018–23.09.2018	
3.	Підготовка навчальної вибірки, навчання моделей	24.09.2018–30.09.2018	
4.	Підготовка матеріалів першого розділу магістерської дисертації	01.10.2018–08.10.2018	
5.	Проведення порівняльного аналізу побудованих моделей	09.10.2018–17.10.2018	
6.	Підготовка матеріалів другого розділу магістерської дисертації	18.10.2018–26.10.2018	
7.	Підготовка графічної частини магістерської дисертації	27.10.2018–04.11.2018	
8.	Підготовка матеріалів третього розділу магістерської дисертації	05.11.2018–14.11.2018	
9.	Підготовка матеріалів розділу стартап-проекту магістерської дисертації	15.11.2018–22.11.2018	
10.	Написання висновку магістерської дисертації	23.11.2018–26.11.2018	

Студент П.В. Бех

Науковий керівник дисертації Ю.П. Зайченко

РЕФЕРАТ

Магістерська дисертація: 133 с., 17 Рисунків, 23 табл., 39 джерел літератури, 1 додаток.

Об’єкт дослідження: алгоритми автоматичного розпізнавання мовлення.

Предмет дослідження: обробка природних мов.

Цілі дослідження: створення системи з перетворення аудіофайлів у текстові записи в контексті нарад.

Задачі роботи: розробити штучну нейронну мережу для автоматичного розпізнавання мовлення, тобто розробити інформаційну систему з веб-інтерфейсом, що могла б за командою користувача провести автоматичне розпізнавання усного мовлення з певного файлу-контейнера, а також надавати можливість індексованого пошуку тексту в уже оброблених записах.

Під час виконання роботи було проведено аналіз сучасних аудіоформатів та методів з розпізнавання інформаційних сигналів, серед методів була обрана комбінація попередньої обробки через MFCC, інтелектуального аналізу через LSTM.

Була розроблена архітектура мережі для розпізнавання мовних символів у звуковому потоці.

Актуальність проекту мотивується відсутністю аналогічних рішень і потенційною зацікавленістю з боку бізнесу.

Результати роботи можуть бути використані як комерційними організаціями, так і ентузіастами, що прагнуть розробити щось подібне.

АНАЛІЗ MP3, АВТОМАТИЧНЕ РОЗПІЗНАВАННЯ МОВЛЕННЯ, PYTHON, TENSORFLOW, KERAS, MFCC, LSTM, RNN, CTC.

ABSTRACT

Master’s thesis: 133 p., 17 fig., 23 tabl., 39 ref., 1 appendix.

Object of research: algorithms of automatic speech recognition.

Subject of study: natural languages processing.

Objectives of the study: creation of a system for making text records of meetings.

Tasks of the work: to develop an artificial neural network for automatic speech recognition, that is, to develop an information system with a web interface that could, after a user's command, perform automatic speech recognition from a specific container file, and provide an indexed text search in already processed records.

During the work, the analysis of modern audio formats and methods for recognition of information signals was done; among the formats MPEG-1 Layer 3 was chosen as the most popular, and among the methods a combination of preprocessing through MFCC, intelligent analysis through LSTM were chosen.

A network architecture was developed for the recognition of speech symbols in an audio stream, based on which a system that fulfilled the task was developed.

The relevance of the project is motivated by the potential business interest and lack of similar solutions.

The results of the work could be used both by commercial organizations and by enthusiasts who want to develop something similar.

MP3 ANALYSIS, AUTOMATIC SPEECH RECOGNITION, PYTHON, TENSORFLOW, KERAS, MFCC, LSTM, RNN, CTC.

ЗМІСТ

УМОВНІ ПОЗНАЧЕННЯ

Кодування – перетворення будь-якої інформації на послідовність імпульсів, що мають властивість самосинхронізації для передачі через телекомунікаційні канали.

ІКМ (імпульсно-кодова модуляція) – процес перетворення аналогового сигналу у цифровий сигнал, коли через певні інтервали часу беруться відліки аналогового сигналу і незалежно один від одного квантуються і далі кодуються цифрами.

ЦАП – цифро-аналоговий перетворювач.

GNU General Public License (Загальна публічна ліцензія GNU або Загальна громадська ліцензія GNU) – одна з найпопулярніших[1] ліцензій на вільне програмне забезпечення, створена Річардом Столменом для проекту GNU.

Дискретне перетворення Фур'є (ДПФ, англ. Discrete Fourier Transform) – це математична процедура, що використовується для визначення гармонічного, або частотного, складу дискретних сигналів.

Швидке перетворення Фур'є (часто FFT від англ. Fast Fourier Transform) – швидкий алгоритм обчислення дискретного перетворення Фур'є.

АЧХ (амплітудно-частотна характеристика) – графік залежності амплітуди вихідного сигналу передавача від частоти вхідного сигналу сталої амплітуди.

Спектрограма – зображення, відображаюче залежність спектральної щільності потужності сигналу від часу.

ВСТУП

Ведення бізнесу сучасних установ зазвичай передбачає проведення нарад. Рішення, що приймаються на нарадах, несуть стратегічний характер, тому неприпустимою є ситуація, коли рішення, прийняте на нараді, має слабку мотивацію. Однак, подібне стається і це пов’язане з людським фактором. Певні особи можуть змінювати думку або забувати факти, що привели до тих чи інших висновків. Це приводить до необхідності повторно проводити наради з тих самих питань, що веде до зниження ефективності роботи керівників підрозділів.

Рішенням подібних проблем часто стає підбиття висновків по основним питанням у вигляді резюме зустрічі. Однак, на це не завжди вистачає часу, а коли він з’являється, щось уже може бути забуте. На сьогоднішній день не існує автоматизованих засобів ведення резюме зустрічей. Навіть якщо боротися з подібним шляхом записування зустрічей у вигляді відео-/звукозаписів, їх перегляд займатиме майже стільки ж часу, скільки й проведення повторних нарад.

У якості альтернативи зазначеним вище методам ведення записів нарад пропонується система, що могла б не тільки вести звукозапис наради, а й перевести його в форму тексту. Текстова форма передбачає не тільки можливість читати записи наради, а й можливість пошуку реплік за ключовими словами, що значно зменшує час, необхідний на аналіз.

Система має спрощувати задачу ведення обліку матеріалів нарад і таким чином підвищувати ефективність підприємства шляхом зменшення кількості повторних нарад.

У якості файлу-контейнера слід розглядати файл формату MPEG-1 Layer 3 через його значну поширеність і відсутності необхідності ліцензування, так як усі ліцензійні збори завершено.

Типовий сценарій використання готового продукту: на певному сервері розгорнута база даних та нейронна мережа, навчена розпізнавати мову, що застосовується на нарадах в установі. Перед проведенням наради особа, відповідальна за підготовку, вмикає записуючий пристрій. Після завершення наради її звукозапис передається до системи, що після деякого часу обробки зберігала б у базі даних текстовий файл із текстовою формою сказаного на нараді. За необхідності будь-хто з осіб, маючих доступ до подібної інформації, має змогу звернутися до запису (як звукозапису, так і тексту) і зберегти, що дозволить знайти необхідну інформацію.

Актуальність роботи мотивується тим, що подібних продуктів на ринку ще не має, а бізнес завжди зацікавлений у підвищенні власної ефективності.

1 ОГЛЯД ПРОБЛЕМАТИКИ ОБРОБКИ ЗВУКУ

Важливою частиною аналізу будь-яких даних є їх попередня обробка. Розглянемо попередню обробку звукозаписів.

1.1. MP3

MP3 – кодек розроблений командою MPEG, формат файлу для зберігання аудіоінформації . Формат був ліцензованим, але 23 квітня 2017 року термін дії всіх патентів закінчився і лицензійні збори припинені[1]. MP3 є одним из найбільш поширених і популярних форматів цифрового кодування звукової информації з втратами. Формат може програватися практично в усіх популярних операційних системах, на більшості портативних аудіоплеєрів, а також підтримується всіма сучасними моделями музичних центрів і DVD-плеєрів.

Через високу популярність, неліцензованість, постійну частоту дискретизації і порівняно низьку вагу файлу формату MP3 було вирішено в якості основного формату розглядати саме його. Розглянемо детальніше структуру MP3-файлу, щоб зрозуміти, що саме варто з нього отримати для подальшого аналізу.

1.2. Кодування MP3

У цьому форматі звуки кодуються частотним чином (без дискретних партій), є підтримка стерео. Згідно зі специфікацією[2], MP3 є форматом стиснення з втратами, тобто частина звукової інформації, яку вухо людини сприйняти не може або сприймається не всіма людьми, знищується. Ступінь стиснення можна варіювати, зокрема в межах одного файлу. Інтервал можливих значень бітрейту становить 8 – 320 кбіт/с. Для порівняння, потік даних із звичайного компакт-диска формату AUDIO-CD дорівнює 1411,2 кбіт/с при частоті дискретизації 44100 Гц.

Оскільки формат MP3 підтримує двохканальне кодування (стерео), існує 3 режими кодування звукових каналів[10]:

– Стерео – двохканальне кодування, при якому канали кодуються незалежно один від одного. Таким чином, заданий бітрейт ділиться на два канали. Наприклад, якщо заданий бітрейт 192 кбіт/с, то для кожного каналу він буде рівний тільки 96 кбіт/с.

– Моно – одноканальне кодування. Якщо закодувати двохканальний матеріал в цей спосіб, відмінності між каналами будуть повністю стерті, оскільки два канали змішуються в один, він кодується і він же відтворюється в обох каналах стереосистеми. Єдиним плюсом даного режиму може бути тільки вихідна якість в порівнянні з режимом Стерео при однаковому бітрейті, оскільки на один канал припадає удвічі більша кількість бітів, ніж в режимі Стерео. Але відмінностей між каналами ви не почуєте, оскільки канал тут тільки один.

– Об'єднане стерео (Joint Stereo) – оптимальний спосіб двохканального кодування, при якому лівий і правий канали перетворяться в їх суму і різницю. Для більшості звукових файлів канал з різницею виходить набагато тихіше за канал з сумою, тому на суму відводиться більша частина

бітрейта. Таким чином, якість вихідного файлу разюче відрізняється в кращу сторону від режиму Стерео при однаковому бітрейті, особливо при низькому. Існує думка, що даний режим не підходить для звукового стереоматеріалу, в якому в двох каналах відтворюється суб'єктивно абсолютно різний матеріал, оскільки він стирає відмінності між каналами. Це помилкова думка, оскільки насправді MP3-кодек оперує частотами, а певні частоти в більшості випадків перетинаються в обох каналах, тобто ідентична інформація все ж таки присутня, а різна кодується окремо. Особливо ефективний цей спосіб двохканального кодування при використанні змінного бітрейта.

1.3. Отримання ІКМ з MP3-файлу

Імпульсно-кодова модуляція (ІКМ або PCM – англ. Pulse Code Modulation) - процес перетворення аналогового сигналу у цифровий сигнал, коли через певні інтервали часу беруться відліки аналогового сигналу і незалежно один від одного квантуються і далі кодуються цифрами[3]. ІКМ використовується для оцифровки аналогових сигналів перед їхньою передачею. Практично всі види аналогових даних (відео, голос, музика, дані телеметрії) допускають застосування ІК-модуляції (рис 1.1).

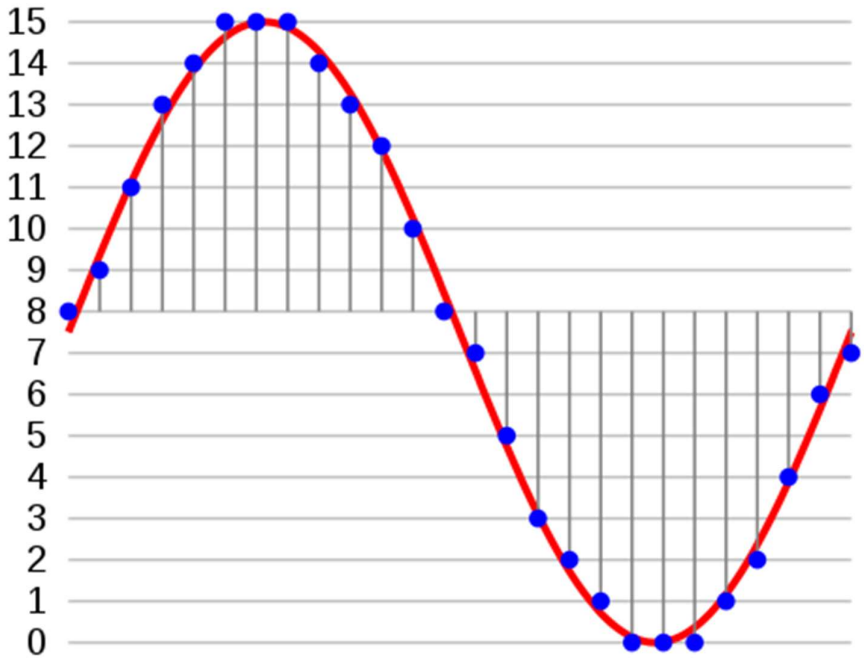

Рисунок 1.1 – ІКМ гармонічного сигналу

Щоб одержати на вході каналу зв'язку (передавальний кінець) ІК-модульований сигнал з аналогового, амплітуда аналогового сигналу вимірюється через рівні проміжки часу. Кількість оцифрованих значень у секунду (або швидкість оцифрування) кратна максимальній частоті (Гц) у спектрі аналогового сигналу. Миттєве обмірюване значення аналогового сигналу округляється до найближчого рівня з декількох заздалегідь певних значень. Цей процес називається квантуванням, а кількість рівнів завжди береться кратним ступеню двійки, наприклад, 8, 16, 32 або 64. Номер рівня може бути відповідно представлений 3, 4, 5 або 6 бітами. Таким чином, на виході модулятора виходить набір бітів (0 або 1) [13].

На прийомному кінці каналу зв'язку демодулятор перетворює послідовність бітів в імпульси з тим же рівнем квантування, що використав модулятор. Ці імпульси поступають на ЦАП, що складається з декодуючого пристрою, який перетворює прийняту закодовану послідовність в квантовану послідовність відліків, а також з згладжуючого фільтра, що відновлює переданий аналоговий сигнал по квантованим значенням відліків.

1.4. Дискретне перетворення Фур’є

Дискретне перетворення Фур'є (ДПФ, англ. Discrete Fourier Transform) – це математична процедура, що використовується для визначення гармонічного, або частотного, складу дискретних сигналів. ДПФ є однією з найбільш розповсюджених і потужних процедур цифрової обробки сигналів. ДПФ (Рисунок 1.2) дозволяє аналізувати, перетворювати і синтезувати сигнали такими способами, які неможливі при неперервній (аналоговій) обробці.

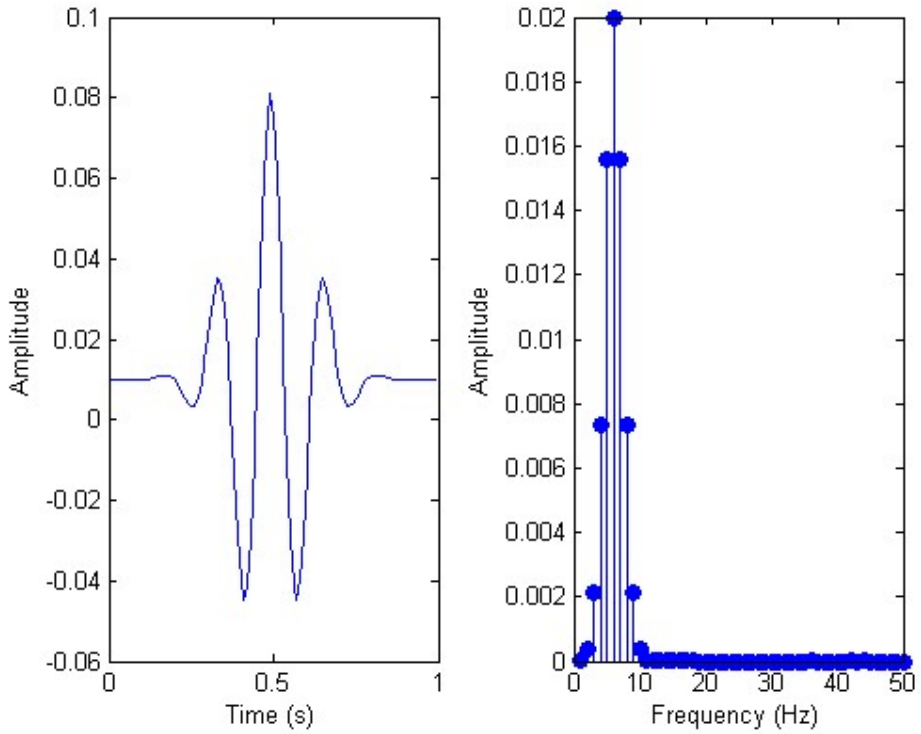

Рисунок 1.2 – Візуалізація переходу від часової до частотної області

Витоком ДПФ є неперервне перетворення Фур'є $X(f)$ (1.1), яке визначається:

$$X(f) = \int_{-\infty}^{+\infty} x(t) e^{-j2\pi ft}\, dt \tag{1.1}$$

Експоненціальна форма (1.2):

$$X(m) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nm/N} \tag{1.2}$$

Тригонометрична форма (1.3):

$$X(m) = \sum_{n=0}^{N-1} x(n)(\cos(2\pi nm/N) - j\sin(2\pi nm/N)) \tag{1.3}$$

де:

– X(m) – m-ий компонент ДПФ, тобто X(0), X(1), X(2),…,

– m – індекс ДПФ в частотній області, m = 0, 1, 2,…, N – 1,

– x(n) – послідовність вхідних відліків, x(0), x(1), x(2),…,

– n – часовий індекс вхідних відліків, n=0, 1, 2, 3,…, N-1,

– N – кількість відліків вхідної послідовності і кількість частотних відліків результату ДПФ.

Якщо представити довільний відлік ДПФ X(m) як суму дійсних і уявних частин (1.4):

$$X(m)=X_{re}(m)+jX_{im}(m)=X_{mag} \text{ з кутом } X_{\varphi}(m) \tag{1.4}$$

то амплітуда X(m) обчислюється (1.5):

$$X_{mag}(m) = |X(m)| = \sqrt{X_{re}(m)^2 + X_{im}(m)^2} \quad (1.5)$$

Фазовий кут X(m), $X_{\varphi}(m)$, обчислюється так (1.6):

$$X_{\varphi}(m) = tan^{-1}(\frac{X_{im}(m)}{X_{mag}(m)}) \quad (1.6)$$

Потужність відліків X(m), яка називається спектром потужності, являє собою амплітуду, піднесену до квадрату (1.7):

$$X_{PS}(m) = X_{mag}(m)^2 = X_{re}(m)^2 + X_{im}(m)^2 \quad (1.7)$$

1.5. Швидке перетворення Фур'є

Швидке́ перетво́рення Фур'є́ (часто FFT від англ. Fast Fourier Transform) – швидкий алгоритм обчислення дискретного перетворення Фур'є. Якщо для прямого обчислення дискретного перетворення Фур'є з N точок даних потрібно $O(N^2)$ арифметичних операцій, то FFT дозволяє обчислити такий же результат використовуючи O(N log N) операцій. Алгоритм FFT часто використовується для цифрової обробки сигналів для перетворення дискретних даних з часового у частотний діапазон.

Покажемо як виконати дискретне перетворення Фур'є за $O(N(p_1 + \dots + p_n))$ обчислень при $N=p_1p_2\dots p_n$. Зокрема, при $N=2^n$ знадобиться O(Nlog(N)) обчислень.

Дискретне перетворення Фур'є перетворює набір чисел $a_0,\dots ,a_{n-1}$ в набір чисел $b_0,\dots,b_{n-1}$, такий, що $b_i = \sum_{j=0}^{n-1} a_j \varepsilon^{ij}$, де $\varepsilon^n = 1$ і $\varepsilon^k \neq 1$ при $0<k<n$. Алгоритм швидкого перетворення Фур'є може бути застосований до будь-яких

комутативних асоціативних кілець з одиницею. Найчастіше цей алгоритм застосовують до поля комплексних чисел (з $\varepsilon = e^{2\pi i/n}$) і до кілець залишків за модулем n.

Основний крок алгоритму полягає у зведенні задачі для N чисел до задачі для p=N/q чисел, де q – дільник N. Нехай ми вже вміємо вирішувати задачу для N/q чисел. Застосуємо перетворення Фур'є до наборів a_i, a_{q+i}, … , $a_{q(p-1)+i}$ для i=0, 1, … ,q-1. Тепер покажемо, як за O(Np) обчислень розв'язати вихідну задачу. Зауважимо, що $b_i = \sum_{j=0}^{q-1} \varepsilon^{ij} (\sum_{k=0}^{p-1} a_{kq+j} \varepsilon^{kiq})$. Вирази в дужках нам уже відомі – це (i mod p)-те число після перетворення Фур'є j-тої групи. Таким чином, для обчислення кожного b_i потрібно O(q) обчислень, а для обчислення всіх b_i – O(Nq) обчислень.

1.6. Мел-кепстральні коефіцієнти

Мел-кепстральні коефіцієнти (MFCC) – представлення логарифму потужності спектру в мел-частотній області. Вони описують потужність огинаючої спектру, що характеризує модель мовного тракту.

Типова схема побудови MFCC:[4][5]

1. Проведення перетворення Фур’є над початковим сигналом;
2. Відображення потужностей спектру сигналу на мел-шкалу за допомогою трикутних вікон, що перетинаються;
3. Розрахунок логарифмів потужностей на кожній мел-частоті;
4. Проведення дискретного косинусного розкладу отриманих вище логарифмів потужностей;
5. Амплітуди отриманого спектру – мел-кепстральні коефіцієнти.

У системі буде застосований мел-кепстральний розклад через те, що:

- він забезпечує додаткову фільтрацію сигналу під час обробки;
- у нього закладена психоакустична модель, що дозволить отримувати більш прогнозовані результати відносно використання коефіцієнтів розкладу Фур'є.

Висновки до розділу

У розділі було описано підходи по збереженню й обробці звукозаписів.
Ключові моменти:

- у якості контейнера аудіо було обрано формат MP3;
- введене поняття імпульсно-кодової модуляції (ІКМ), описаний перехід між контейнером аудіо і ІКМ;
- описані ключові підходи отримання інформації про початковий сигнал з ІКМ (фур'є-розклад, мел-кепстральний розклад);
- у подальшому в роботі будуть застосовуватися коефіцієнти мел-кепстрального розкладу, отримані на ковзному вікні розміром 10 мс з кроком 5 мс [6].

2 ОГЛЯД ПРОБЛЕМАТИКИ ОБРОБКИ ЗВУКУ

У задачі автоматичного розпізнавання мовлення за останні роки з’явилося багато підходів. У розділі будуть розглянуті ключові підходи.

2.1. Приховані Марковські моделі

Прихована марковська модель (ПММ, hidden Markov model, HMM) є статистичною марковською моделлю, де модельована система, розглядається в якості марковського процесу із прихованими (неспостережуваними) станами. Приховану марковську модель можна представити як динамічну баєсову мережу. Математичний апарат для ПММ був розроблений Леонардом Баумом зі співробітниками. Він тісно пов'язаний з ранньою працею про оптимальну нелінійну задачу фільтрування Руслана Стратоновича[7], який найпершим описав послідовно-зворотній алгоритм.

У більш простих марковських моделях (таких як ланцюги Маркова) стан є видимим спостерігачеві, і тому єдиними параметрами є ймовірності переходів між станами станів. У ПММ стан є невидимим безпосередньо, але вихід стану видимим є. Кожен стан має деякий ймовірнісний розподіл усіх можливих вихідних значень. Таким чином, послідовність символів, що генерується ПММ, дає деяку інформацію про послідовність станів. Прикметник «прихований» стосується лише послідовності станів моделі, а не йй параметрів; модель все одно називається «прихованою» марковською моделлю, якщо ці параметри відомі точно.

Прихована модель Маркова – це такий ланцюг Маркова, для якого стан може спостерігаєтися лише частково. Іншими словами, спостереження певним

чином пов'язані зі станом системи, але їх зазвичай, недостатньо, щоб точно визначити стан. Існує декілька відомих алгоритмів для реалізації прихованих моделей Маркова. Наприклад, враховуючи наявну послідовність спостережень, алгоритм Вітербо обчислює найбільш вірогідну відповідну послідовність станів, далі алгоритм обчислення вірогідності послідовності спостережень і алгоритм Баума-Уелча будуть оцінювати стартові вірогідності, перехід функції, та функцію спостереження прихованої марковської моделі.

Діаграма нижче (рис 2.1) показує загальну архітектуру прихованої марковської моделі. Кожен овальний елемент представляє випадкову змінну, що може приймати значне число значень. Випадкова змінна *x(t)* – прихований стан у момент часу *t* (у моделі з діаграми нижче *x(t) ∈ (x1, x2, x3))*. Випадкова змінна *y(t)* – спостереження виходу процесу у момент часу *t* (у моделі на діаграмі нижче *y(t) ∈ (y1, y2, y3, y4)*). Переходи (стрілки) у цій діаграмі (часто називають ґратковою діаграмою) позначають стохастичні залежності.

З діаграми нижче видно, що умовна функція розподілу прихованої змінної *x(t)* у момент часу *t*, якщо задані значення прихованої змінної в будь-які моменти часу, залежить лише від значення попередньої прихованої змінної *x(t − 1)*: значення у попередні моменти часу (*t − 2* і т.д.) не мають впливу. Це явище називається марковською властивістю. Так само значення змінної спостереження *y(t)* залежить лише від поточного значення прихованої змінної *x(t)*.

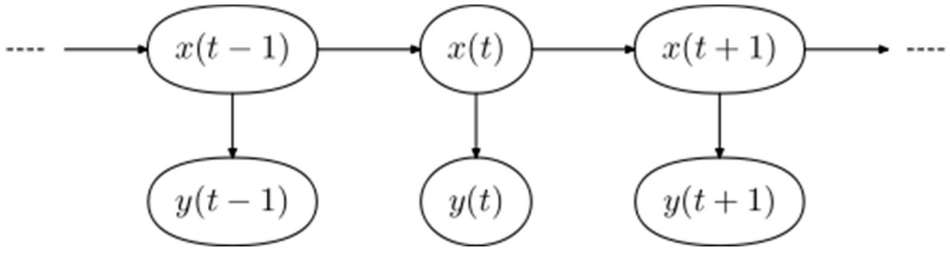

Рисунок 2.1 – загальна архітектура ПММ

У стандартному вигляді прихованої марковської моделі, що розглядається, простір прихованих станів є дискретним, водночас самі спостереження можуть бути або дискретними (вони зазвичай генеруються з

категорійного розподілу) або неперервними (тоді вони зазвичай генеруються з нормального розподілу). Параметри прихованої марковської моделі розділяються на два типи: вірогідності переходів та вірогідності виходів. Вірогідності переходів керують тим, як саме прихований стан у момент часу *t* обирається в залежності від прихованого стану в попередній момент часу *t-1*.

Вважається, що простір прихованих станів прихованої марковської моделі складається з одного з *N* можливих значень і може бути змодельований як категорійний розподіл. Це значить, що для кожного з можливих станів, у якому прихована змінна може знаходитися в момент часу *t*, є вірогідність переходу з нього до кожного з *N* інших можливих станів прихованої змінної в наступний момент часу *t+1*, отже N^2 ймовірностей переходів. При цьому сума ймовірностей переходів для переходів з деякого заданого стану мусить дорівнювати 1. Отже, матриця *NxN* вірогідностей переходів є марковською матрицею. Оскільки будь-яку одну вірогідність переходу можливо визначити, коли відомо решту, загальна кількість параметрів переходу розраховується як *N(N-1)*

Окрім цього, для кожного з можливих станів є набір вірогідностей виходів, що визначає розподіл спостережуваної змінної у деякий момент часу для заданого стану прихованої змінної у цей момент часу. Розмірність такого набору залежить від природи поведінки спостережуваної змінної.

Приклад: якщо спостережувана змінна дискретна (з *M* можливих значень), що розподілені за категорійним розподілом, то буде *M-1* окремих параметрів, отже загалом буде *N(M-1)* параметрів виходів для всіх прихованих станів моделі.

Однак, якщо спостережувана змінна є деяким багатовимірним (розмірністю *M*) вектором з довільним багатовимірним нормальним розподілом, то буде *M* параметрів, що контролюють математичні сподівання, та *M(M+1)/2* параметрів, що контролюватимуть матрицю коваріацій, всього буде $N(M+M(M+1)/2) = NM(M+3)/2 = O(NM^2)$ параметрів виходів. (У такому випадку, якщо значення *M* є немалим, зручніше буде обмежити природу

коваріацій між конкретними елементами вектора спостережень, припустивши, що елементи назалежні один від одного, або, менш локалізуючи, не залежать від усіх, окрім деякого (фіксованого) числа сусідніх елементів)

Перевагою використання подібної моделі є її наочність та можливість роботи з великими інформаційними сигналами.

До недоліків можна віднести необхідність уточнення особливостей розподілів випадкових величин, що характеризують виходи моделі, а також те, що алгоритми її навчання зазвичай дозволяють досягти лише локального оптимуму і малоефективні на великих навчальних вибірках.

2.1.1. Алгоритм навчання ПММ

При розділенні початкової вибірки на навчальну та тестову задача навчання зводиться до двох підзадач:

– мінімізація помилок при класифікації;

– максимізація узгодженості моделі (схожі результати на навчальній та випробувальній вибірках).

Розглянемо наступну модель:

$\lambda=(A, B, \pi)$,

де $A = |a_{ij}| = P[q_{t+1}=S_j \mid q_t = S_i]$ – вірогідність переходу зі стану i в стан j,

$B = |b_j(k)| = P[v_k \mid q_t = S_j]$ – вірогідність виходу k-го символу у стані j.

В якості цих ймовірностей зазвичай розглядається багатовимірний нормальний розподіл, тому:

$b_j(X) \sim N_k(\mu, \sum)$, де μ – k позначає координату вектору-виходу моделі.

$\pi = |\pi_i| = P[q_1 = S_i]$ – вірогідність того, що стан i буде початковим станом моделі.

Тоді така модель буде навчатися за наступною модифікацією ЕМ-алгоритму:

– Послідовна рекурсія:

Ініціалізація:

$\alpha_0(s_i) = \pi_i$

Індукція (2.1):

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(\mathbf{x}_t) \qquad 1 \le j \le N, 1 \le t \le T \tag{2.1}$$

Завершення (2.2):

$$p(\mathbf{X} \mid \lambda) = \alpha_T(s_E) = \sum_{i=1}^{N} \alpha_T(i) a_{iE} \tag{2.2}$$

,

Де s_i – початковий стан, а s_E – кінцевий стан.

– Зворотна рекурсія:

Ініціалізація рекурсії:

$\beta_T(i) = a_{iE}$

Індукція рекурсії (2.3):

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(\mathbf{x}_{t+1}) \beta_{t+1}(j) \qquad t = T-1, \ldots, 1 \tag{2.3}$$

,

Завершення рекурсії (2.4):

$$p(\mathbf{X}|\boldsymbol{\lambda}) = \beta_0(I) = \sum_{j=1}^{N} a_{Ij} b_j(\mathbf{x}_1)\beta_1(j) = \alpha_T(s_E) \quad (2.4)$$

,

– Налаштування параметрів моделі:

Після обчислення попередніх етапів можна отримати наступне:

Вірогідність появи стану j у час t (2.5):

$$\gamma_t(j) = P(S(t)=j|\mathbf{X},\boldsymbol{\lambda}) = \frac{1}{\alpha_T(s_E)}\alpha_t(j)\beta_t(j) \quad (2.5)$$

,

Вірогідність, що модель буде знаходитися в стані і в момент t і в стані j в момент t + 1 (2.6):

$$\begin{aligned}\xi_t(i,j) &= P(S(t)=i, S(t+1)=j|\mathbf{X},\boldsymbol{\lambda}) = \\ &= \frac{p(S(t)=i, S(t+1)=j, \mathbf{X}|\boldsymbol{\lambda})}{p(\mathbf{X}|\boldsymbol{\lambda})} = \\ &= \frac{\alpha_t(i)a_{ij}b_j(\mathbf{x}_{t+1})\beta_{t+1}(j)}{\alpha_T(s_E)}\end{aligned} \quad (2.6)$$

,

Можемо отримати нову матрицю ймовірностей переходів (2.7):

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T}\xi_t(i,j)}{\sum_{k=1}^{N}\sum_{t=1}^{T}\xi_t(i,k)} \quad (2.7)$$

Нові імовірності початку роботи моделі з певним станом *i* (2.8):

$$\bar{\pi}_i = \gamma_i(1) \tag{2.8}$$

Обчислюємо нові параметри розподілу виходів моделі (2.9):

$$\begin{aligned} \xi_t(i,j) &= P(S(t)=i, S(t+1)=j \mid \mathbf{X}, \boldsymbol{\lambda}) = \\ &= \frac{p(S(t)=i, S(t+1)=j, \mathbf{X} \mid \boldsymbol{\lambda})}{p(\mathbf{X} \mid \boldsymbol{\lambda})} = \\ &= \frac{\alpha_t(i) a_{ij} b_j(\mathbf{x}_{t+1}) \beta_{t+1}(j)}{\alpha_T(s_E)} \end{aligned} \tag{2.9}$$

Таким чином за деяку кількість кроків буде досягнуто локальний оптимум, що дозволить розв’язати обидві частини поставленої задачі.

2.2. Мережі глибокого навчання

Штучна нейронна мережа (рисунки 2.2 та 2.3) – математична модель, її програмна та/або апаратна реалізація, що побудовані за аналогією до роботи реальних біологічних нейронних мереж – мереж з нервових клітин у живому організмі. Це поняття з’явилося під час вивчення процесів, які відбуваються в головному мозку, та при спробах змоделювати ці процеси.

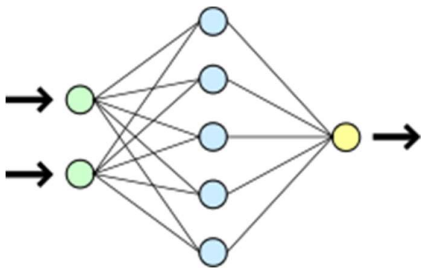

Рисунок 2.2 – умовна схема одношарової нейронної мережі з двома входами та одним виходом

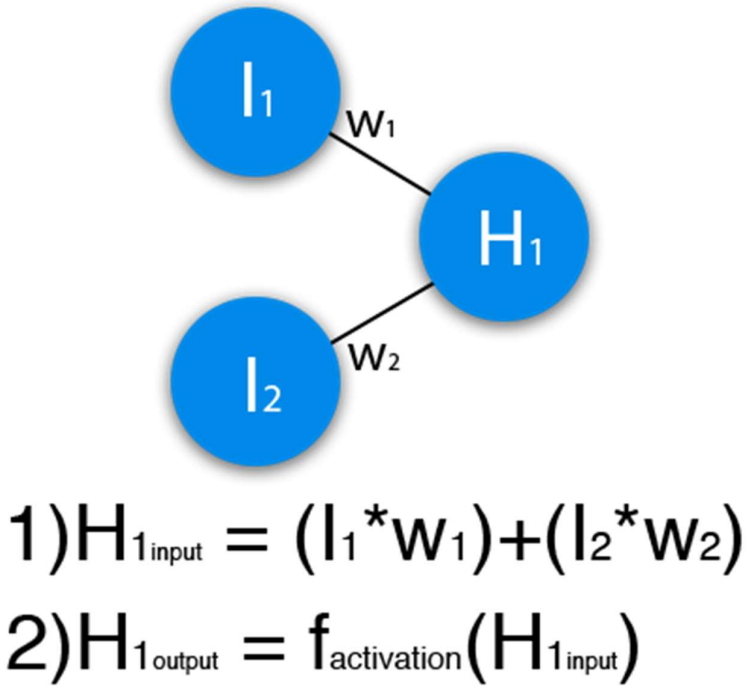

Рисунок 2.3 – модель синапсу нейронної мережі

У випадку з мережами глибокого навчання мається на увазі, що кількість прихованих шарів (рисунок 2.4) є значною. При цьому для спрощення процесу навчання зазвичай у якості активуючих функцій нейронів застосовують так звані випрямлячі (ReLU), що обчислюється як (2.10):

$$f(x) = x^+ = \max(0, x), \tag{2.10}$$

Гладким наближенням випрямляча є гладка функція(2.11):

$$f(x) = \log(1 + \exp x). \tag{2.11}$$

Плюсом випрямляча є простота розрахунку його похідної, що дозволяє при навчанні мереж із великою кількістю нейронів витрачати менше часу на розрахунки.

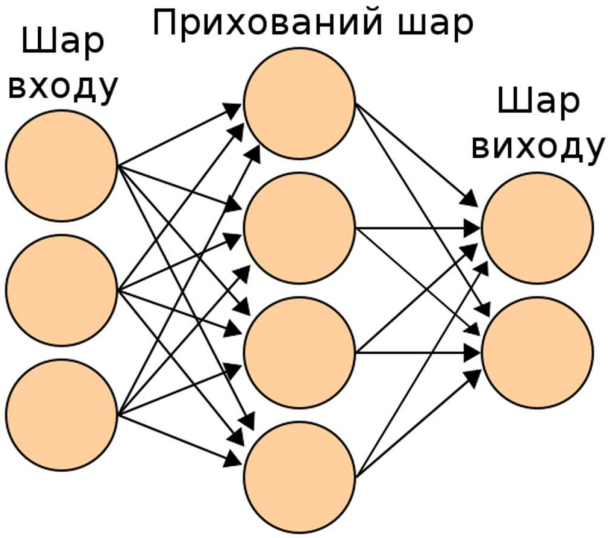

Рисунок 2.4 – нейронна мережа з одним прихованим шаром

Нейронна мережа може виступати в якості класифікатора інформаційних сигналів, приймаючи на вхід фрагмент певної довжини і генеруючи на виході ймовірність належності до певного класу. Таким чином можна порахувати статистику належності фрагментів сигналу до різних класів і обрати один із них як фінальний клас.

2.3. Згорткові нейронні мережі

Згорткові нейронні мережі (також відомі як convolutional neural network або CNN) у методах штучного інтелекту – це клас штучних нейронних мереж з прямим поширенням, який успішно застосовується в аналізі візуальних зображень.

Згорткові нейронні мережі використовують особливий вид багатошарових перцептронів: він розроблений так, щоби працювати за умов використання мінімальної кількості операцій попередньої обробки.[8] Також вони відомі як просторово інваріантні або інваріантні відносно зсуву штучні

нейронні мережі, через їхню архітектуру спільних ваг та характеристики інваріантності від паралельного перенесення.[9][10]

Згорткові мережі було створено під враженням від біологічних процесів,[4] у яких схему з'єднання нейронів запозичено з організації зорової кори тварин. Окремі нейрони зорової кори реагують на подразники лише в обмеженій поля зору (відомій як рецептивне поле). Рецептивні поля окремих нейронів частково перекриваються, таким чином вони загалом покривають усе зорове поле.

Згорткові нейронні мережі використовують відносно мало попередньої обробки, якщо порівнювати з іншими алгоритмами класифікації зображень. Це значить, що мережа самостійно визначає фільтри, що в інших алгоритмах розроблялися розробниками вручну. Така незалежність у конструюванні ознак від початкових знань та зусиль розробників є великою перевагою.

Вони знайшли своє застосування в наступних сферах: розпізнавання зображень та відео, рекомендаційні системи[11] та обробка природних мов.[12]

2.4. Архітектура ЗНМ

Згорткова нейронна мережа будується з відного та вихідного шарів, а також із деякої кількості прихованих шарів. Приховані шари (рисунок 2.5), як правило, складаються з наступних елементів: згорткові шари, агрегувальні шари, повнозв'язні шари та шари нормалізації.

Цей процес в нейронних мережах називають згорткою за домовленістю. З точки зору математики він – скоріше взаємна кореляція, ніж згортка. Це важливо лише для індексів матриці, а саме які ваги розташовуються на якому індексі.

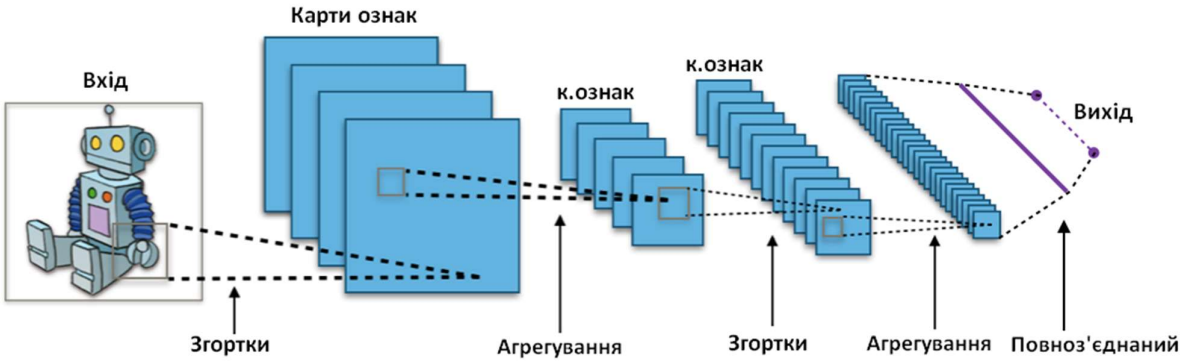

Рисунок 2.5 – типова архітектура ЗНМ

2.4.1. Згорткові шари

Згорткові шари ЗНМ застосовують до вхідних даних операцію згортки, генеруючи результат згортки до наступного шару. Згортка реалізує реакцію певного нейрону на деякий зоровий стимул.[13]

Кожен нейрон згортки обробляє вхідні дані лише для визначеного для нього рецептивного поля.

Хоча повнозв'язні нейронні мережі з прямим поширенням і можливо застосувати як для навчання ознакам, так і для класифікації даних, застосування такої архітектури до візуальних зображень не є практичним. Було б необхідно дуже багато нейронів, навіть у поверхневій (не глибинній) архітектурі, через значні розміри входу НМ, пов'язані з графічними зображеннями (кожен піксель є окремою відповідною змінною). Наприклад, повнозв'язний шар для (відносно маленького) зображення з розміром 100 на 100 пікселів має 10,000 ваг. Операція згортки надає змогу вирішити цю проблему, оскільки вона зменшує розмірність простору вільних параметрів, таким чином дозволяючи мережі бути ще глибшою, маючи при цьому меншу кількість параметрів[14]. Наприклад, незалежно від розміру оригінального зображення, області замощування згорткового шару розміром 5 на 5 пікселів з одними й

тими ж спільними вагами, мають лише 25 параметрів. Отже, це вирішує проблеми зникання або стрімкого зростання градієнтів під час тренування традиційних багатошарових нейронних мереж з великою кількістю шарамів за допомогою backpropagation.

2.4.2. Агрегувальні шари

ЗНМ можуть включати локальногі або глобальні агрегувальні шари, які об'єднують виходи сегментів нейронів одного шару до одного нейрону на виході.[15][16] Наприклад, агрегування з максимізацією (max pooling) повертає максимальне значення виходів з кожного з сегментів нейронів попереднього шару.[17] Інший приклад: агрегування усередненням (average pooling), що повертає усереднене значення виходів з кожного з сегментів нейронів попереднього шару.

2.4.3. Повнозв'язні шари

У повнозв'язних шарах реалізується з'єднання всіх нейронів попереднього шару з усіма нейронами наступного шару. Формально, це є аналогом традиційної нейронної мережі у вигляді багатошарового перцептрону.

2.4.4. Ваги

Згорткові нейронні мережі застосовують спільні ваги у шарах згорток, що значить, що для кожного шару рецептивного поля застосовується один і той самий фільтр (банк ваг), що зменшує необхідний обсяг пам'яті та покращує продуктивність.

2.5. Довга короткочасна пам’ять

Довга короткочасна пам'ять (також відома як ДКЧП, long short-term memory або LSTM) – це відома архітектура рекурентних нейронних мереж, розроблена 1997 року Зеппом Хохрайтером та Юргеном Шмідгубером.[18] Як і більшість рекурсивних нейронних мереж, мережа довгої короткочасної пам’яті є універсальною в сенсі, що за достатньо великої кількості вузлів у мережі вона може обчислювати будь-які операції, що може обчислювати простий комп'ютер, але за умови, що вона має відповідну матрицю вагових коефіцієнтів. Матриця вагових коефіцієнтів може розглядатися як програма ДКЧП. На відміну від традиційних рекурентних нейронних мереж, мережа довгої короткочасної пам’яті добре підходить для тренування з досвіду з метою класифікації, обробки або передбачення значень часових рядів за умови, що між ключовими подіями є деякі часові затримки непостійної та невідомої тривалості. Ця відноста нечутливість до розмірів затримок дає довгій короткочасній пам’яті перевагу в багатьох застосуваннях над альтернативними рекурентними нейронними мережами, прихованими марковськими моделями, а також іншими методами навчання послідовностей. З-поміж інших успіхів, довга короткочасна пам’ять досягла найліпших із відомих результатів у

наступних задачах: стиснення тексту з природної мови,[19] розпізнавання несегментованого неперервного тексту-рукопису,[20]. Також у 2009 ДКЧП році перемогла в змаганні з розпізнавання рукописного тексту «ICDAR». Мережі довгої короткочасної пам’яті також застосовуються в задачах автоматичного розпізнавання мовлення. Вони були головною складовою штучної нейронної мережі, яка в 2003 році досягла рекордного фонемних похибок у 17.7 відсотків на широко відомому наборі даних з природного мовлення «TIMIT».[21] На сьогоднішній день найбільші технологічні компанії, в тому числі «Google», «Apple», «Microsoft» і «Baidu», застосовують мережі довгої короткочасної пам’яті як ключові складові своїх нових продуктів.[22][23]

2.5.1. Архітектура ДКЧП

Мережа довгої короткочасної пам'яті – ШНМ, яка містить вузли довгої короткочасної пам'яті замість, або разом з іншими (більш традиційними) вузлами мережі. Вузол довгої короткочасної пам'яті – це вузол РНМ, який відзначається своїм запам'ятовуванням проміжних значень з довгих або коротких інтервалів часу. Важливим для досягнення цієї здатності є відмова віл використання функції активації у своїх рекурентних компонентах. Отже, збережене значення, не страждає від ітеративного згладження з часом, тому член градієнту або вини не схильний до розмиваття, коли під час його тренування використовується зворотне поширення в часі.

Вузли довгої короткочасної пам'яті часто реалізують у так званих «блоках», які містять кілька вузлів довгої короткочасної пам'яті. Ця конструкція типова для глибинних багатошарових ШНМ, вона сприяє реалізаціям на апаратному забезпеченні з паралелізмом. У рівняннях, що наведені нижче, кожна змінна, написана курсивом та в нижньому регістрі, є вектором з

розмірністю рівною кількості вузлів довгої короткочасної пам'яті у відповідному блоці.

Блоки довгої короткочасної пам'яті можуть містити три або чотири так званих «вентилі», які вони застосовують для регулювання плину інформації до або з їх пам'яті. Дані вентилі реалізують за допомогою логістичної функції що обчислює значення між 0 та 1. Для реалізації часткового дозволу або повної заборони перетікання інформації до/з цієї пам'яті застосовують множення на значення «вентиля». Наприклад, «входовий вентиль» регулює міру, до досягнення якої нове значення може входити до пам'яті. Так званий «забувальний вентиль» регулює міру, до досягнення якої значення може залишатися в пам'яті. І так званий «виходовий вентиль» регулює міру, до досягнення якої значення всередині пам'яті можеть використовуватися для обчислення активації виходу всього блоку.

Слід зазначити, що у деяких реалізаціяї «входовий» та «забувальний» вентилі можуть бути об'єднані в один. Ідея такого рішення полягає в тому, що слід забувати настає лише тоді, коли з'являються нові дані, які слід запам'ятати.

У блоці ДКЧП (рисунок 2.6) є єдині ваги: $\underline{W}$ та $\underline{U}$ – використовуються в цілях кординації дії вентилів. Такі ваги застосовуються до значень, які потрапляють до блоку (до них входять: входовий вектор x_t та вихід блоку в попередній моменту часу) под час їх плину між вентилями.

Блоки довгої короткочасної пам'яті як правило тренують із застосуванням алгоритму зворотного поширення в часі.

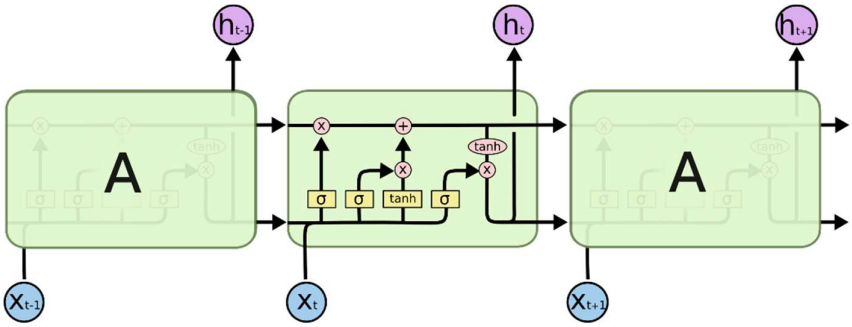

Рисунок 2.6 – Архітектура ДКЧП

2.5.2. Типові ДКЧП

Традиційна довга короткочасна пам’ять із забувальними вузлами (2.12).[18][24]

$c_0 = 0$ і $h_0 = 0$.

Символ «○» позначає добуток Адамара (поелементний добуток матриць однакових розмірностей).

$$\begin{aligned} f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\ i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\ o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\ c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\ h_t &= o_t \circ \sigma_h(c_t) \end{aligned} \tag{2.12}$$

де:

- x_t – входовий вектор;
- h_t – виходовий вектор;
- c_t – вектор стану комірки;
- $\boldsymbol{W}$, $\boldsymbol{U}$ і $\boldsymbol{b}$ – матриці та вектор параметрів;
- f_t – Вектор забувального вентиля. Вага пам'ятання старої інформації
- i_t – Вектор входового вентиля. Вага отримання нової інформації.
- o_t – Вектор виходового вентиля. Кандидатність на вихід;
- σ_g – сигмоїдна функція активації;
- σ_c – гіперболічний тангенс активації;
- σ_h – гіперболічний тангенс активації.

Також відомі такі модифікації ДКЧП як вічкова[25][26](2.13) і згорткова[27](2.14)(* позначає оператор згортки).

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f c_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i c_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o c_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + b_c) \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
\tag{2.13}
$$

$$
\begin{aligned}
f_t &= \sigma_g(W_f * x_t + U_f * h_{t-1} + V_f \circ c_{t-1} + b_f) \\
i_t &= \sigma_g(W_i * x_t + U_i * h_{t-1} + V_i \circ c_{t-1} + b_i) \\
o_t &= \sigma_g(W_o * x_t + U_o * h_{t-1} + V_o \circ c_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c * x_t + U_c * h_{t-1} + b_c) \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
\tag{2.14}
$$

2.6. Алгоритми навчання

Важливим аспектом побудови штучних НМ є вибір алгоритму навчання. Від нього залежить як швидкість навчання, так і близькість моделі до глобального оптимуму. Розглянемо деякі з них.

2.6.1. Метод зворотного поширення помилки

Метод зворотного поширення помилки (більш відомий як backpropagation) – відомий метод навчання штучної нейронної мережі, розроблений для багатошарового перцептрону (рисунок 2.7). Backpropagation – ітеративний градієнтний алгоритм, що застосовується задля мінімізації помилки при роботі багатошарового перцептрону і отримання від нього бажаного виходу. Основна ідея в цьому методі полягає тому, що мережею поширюються сигналі помилки в напрямку від виходу мережі до її входу, тобто

в напрямку, протилежному прямому (штатному) поширенню сигналів у нормальному режимі роботи. Барц і Охонін одразу запропонували узагальнений метод (на «принципі подвійності»), який можна застосовувати до більш широкого класу систем: систем із запізненням, розподілених систем, та інших [28]. Для досягнення можливості застосування методу backpropagation функція активації усіх нейронів ШНМ повинна бути диференційованою (так як поширюється градієнт помилки, а сама помилка залежить від функції активації нейронів).

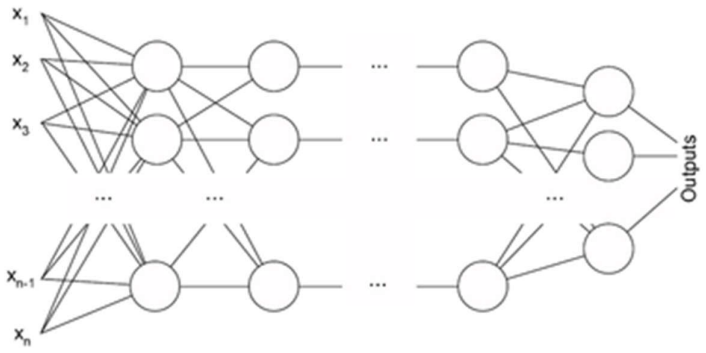

Рисунок 2.7 – Архітектура багатошарового перцептрону

Розглянемо приклад:

Алгоритм backpropagation застосовується для багатошарового перцептрону. Маємо множину $x_1, \dots, x_n$, – множину виходів мережі (*Outputs*) і деяку кількість внутрішніх вузлів. Пронумеруємо всі вузли (включно зі входами та виходами) числами від 1 до N (буде застосована наскрізна нумерація, топологія шарів несуттєва). Позначимо як w_{ij} вагу (множник) зв'язку, що з'єднує вузли i і j, а через o_i позначимо вихід вузла i. Якщо є деякий навчальний приклад (тобто правильні відповіді мережі, що позначимо як t_k, $k \in Outputs$), то тоді функцію помилки, отриману за методом найменших квадратів, можна визначити як (2.15):

$$E(\{w_{i,j}\}) = \frac{1}{2} \sum_{k \in Outputs} (t_k - o_k)^2 \qquad (2.15)$$

Знаючи помилку, можемо коригувати ваги. Це буде показано на прикладі стохастичного градієнтного спуску, тобто ваги будуть коригуватися після кожного прикладу навчальної вибірки і, таким чином, система буде «переміщуватися» в багатовимірному просторі ваг. Для «знаходження» точки мінімуму помилки, необхідно «рухатися» в бік, протилежний градієнту, тобто, на основі кожної групи правильних відповідей будемо додавати до кожної ваги w_{ij} коригуючий градієнт (2.16).

$$\Delta w_{i,j} = -\eta \frac{\partial E}{\partial w_{i,j}} \tag{2.16}$$

де $0 < \eta < 1$ – деякий множник, що регулює швидкість «руху» (навчання).

Похідна розраховується наступним чином: починаємо розраховувати зміни для нейронів $j \in Outputs$, тобто ваги, що нас цікавлять, входять у нейрон останнього (вихідного) рівня. Зазначимо, що вага w_{ij} має вплив на вихід мережі лише як частина суми (2.17), яка рахується по входах вузла j.

$$S_j = \sum_i w_{i,j} x_i \tag{2.17}$$

Тому (2.18)

$$\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial S_j} \frac{\partial S_j}{\partial w_{i,j}} = x_i \frac{\partial E}{\partial S_j} \tag{2.18}$$

Аналогічним чином S_j може впливати на загальну помилку НМ тільки в контексті виходу вузла o_j. Тоді (2.19)

$$\frac{\partial E}{\partial S_j} = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial S_j} = \left(\frac{\partial}{\partial o_j}\frac{1}{2}\sum_{k\in Outputs}(t_k - o_k)^2\right)\left(\frac{\partial \sigma(S_j)}{\partial S_j}\right) = \left(\frac{1}{2}\frac{\partial}{\partial o_j}(t_j - o_j)^2\right)(o_j(1-o_j)) = -o_j(1-o_j)(t_j - o_j) \quad (2.19)$$

де σ() – функція активації нейрона.

Якщо вузол *j* знаходиться не на вихідному шарі, то в нього є свої виходи; Їх позначимо як Children (j). Тоді (2.20)

$$\frac{\partial E}{\partial S_j} = \sum_{k\in Children(j)}\frac{\partial E}{\partial S_k}\frac{\partial S_k}{\partial S_j},$$

$$\frac{\partial S_k}{\partial S_j} = \frac{\partial S_k}{\partial o_j}\frac{\partial o_j}{\partial S_j} = w_{i,j}\frac{\partial o_j}{\partial S_j} = w_{i,j}o_j(1-o_j). \quad (2.20)$$

де $\delta E/\delta S_k$ – це така ж поправка, але вона обчислена для вузлів наступного рівня (позначимо її як δ_k – від Δ_k вона відрізнятиметься відсутністю множення на темп навчання $-\eta x_{i,j}$). Оскільки розраховано поправку для вузлів останнього рівня і виведено через поправки для вузлів нижчих рівнів через поправки вищих рівнів, можна додавати всі поправки до вагів і йти на нову ітерацію навчання. Саме через властивість рекурентного обчислення поправок нижчих рівнів від вищих цей алгоритм і називається алгоритмом зворотного поширення помилки.

Резюме вищесказаного:

– Для вузла вихідного шару (2.21)

$$\delta_j = -o_j(1-o_j)(t_j - o_j) \quad (2.21)$$

– Для внутрішніх вузлів мережі (2.22)

$$\delta_j = -o_j(1 - o_j) \sum_{k \in Outputs(j)} \delta_k w_{j,k} \tag{2.22}$$

– Після розрахунку (2.21-2.22) ваги вузлів коригуються(2.23)

$$\Delta w_{i,j} = -\eta \delta_j x_i \tag{2.23}$$

2.6.2. Стохастичний градієнтний спуск

Стохастичний градієнтний спуск (також відомий як Stochastic gradient descent чи SGD) – це модифікація алгоритму зворотнього поширення помилки. Застосовується для прискорення навчання вагів ШНМ шляхом використання під час навчання деякої підмножини тренувального набору, яка обирається випадкового на кожній ітерації навчання.

У недавній статті[29] розробка метода приписується Герберту Роббінсу та Саттону Монро, через їхнє описання цього методу в 1951 році у статті «Метод стохастичного наближення».[30]

Опис алгоритму:

Вхід:

– $\boldsymbol{X^l}$ – навчальна вибірка;
– $\boldsymbol{\eta}$ – темп навчання;
– $\boldsymbol{\lambda}$ – параметр згладжування функціоналу $\boldsymbol{Q}$ (оцінка помилки).

Вихід:

– Вектор ваг $\boldsymbol{w}$.

Тіло алгоритма:

I. Ініціалізація вагів w_j, $j = 0, \ldots, n$;
II. Ініціалізація поточної оцінки помилки (2.24):

$$Q := \sum_{i=1}^{l} L(a(x_i, w), y_i); \tag{2.24}$$

III. Ітеративне уточнення ваг до виконання умови п. IV:

i. Вибрати об’єкт x_i із навчальної вибірки $\boldsymbol{X^l}$ (випадковим чином);

ii. Обчислюється вихідне значення алгоритму $a(x_i, \boldsymbol{w})$ та помилку (2.25):

$$\varepsilon_i := L(a(x_i, w), y_i); \tag{2.25}$$

iii. Зробити крок градієнтного спуску (2.26):

$$w := w - \eta L'_a(a(x_i, w), y_i)\varphi'(< w, x_i >)x_i; \tag{2.26}$$

iv. Оцінити значення функціоналу (2.27):

$$Q := (1 - \lambda)Q + \lambda\varepsilon_i; \tag{2.27}$$

IV. Поки значення $\boldsymbol{Q}$ не стабілізується та/або ваги $\boldsymbol{w}$ не припинять змінюватись.

Порядок вибору об’єктів для навчання:

Для правильної роботи стохастичного градієнтного спуску слід обирати підмножини навчальної вибірки випадково. Однак існують підходи, направлені на пришвидшення збіжності алгоритму, шляхом дуяких модифікацій випадкового вибору:

– Перемішування – ідея в випадковому виборі об'єкти поперемінно з різних класів. Оскільки об’єкти належатимуть до різних класів, зміна вагів буде більш суттєвою, ніж при повністю випадковому виборі об’єктів.

– Відома модифікація алгоритму, у якій вибір підмножини об'єктів для навчання розподілений за таким законом, при якому ймовірність вибору конкретного об'єкта обернено пропорційна помилці ШНМ на ньому. Варто зазначити, що з цією модифікацією алгоритм набуває значної чутливості до шумів.

Способи ініціалізації ваг:

– Ініціалізувати вектор вагів нулями. Такий підхід застосовується в багатьох реалізаціях, але він не зажди дає швидку збіжність.

– $w_j \sim \text{rand}(-1 / n ; 1 / n)$, де n – розмірність вхідного вектора. Цей метод може давати кращі результати відносно попереднього, якщо вхідний вектор попередньо нормалізується.

– Ще один підхід полягає у вирішенні вихідної задачі оптимізації у випадку незалежності ознак (у сенсі теорії ймовірностей), лінійної функції активації та квадратичної функції помилки. Тоді ініціалізація вагів матиме наступний вигляд (2.28):

$$w_j := \frac{< y, f_j >}{< f_j, f_j >}. \tag{2.28}$$

Переваги:

1) метод пристосований для навчання в режиме online, тобто в умовах постійного надходження нових навчальних об’єктів із необхідністю швидколо оновлювати ваги;

2) за великого розміру навчальної вибірки алгоритм може бути досить ефективним, так як за рахунок випадковості підмножини, на якій проходитиме конкретна ітерація навчання, алгоритм може пристосувати мережу до узагальнення раніше, ніж пройде всю навчальну вибірку;

3) допускаються різні стратегії для навчання. Наприклад, за великого розміру навчальної вибірки або за умов динамічного навчання можливо не

навчатися повторно на об’єктах, які вже приймали участь у навчанні. При цьому за малого розміру навчальної вибірки можна використовувати об’єкти повторно.

Недоліки:

1) алгоритм може повільно збігатися або не збігатися взагалі;

2) оскільки градієнтний спуск ітеративно рухається вздовж простору розв’язків у напрямку, протилежному градієнту функціоналу Q, може статися так, що під час навчання буде досягнуто локальний оптимум. Для боротьби з цим слід вводити мутації, а також використовувати техніку різкої зміни вагів за ознак подібного явища;

3) за великої кількості ознак $\boldsymbol{n}$ та/або малого розміру навчальної вибірки $\boldsymbol{l}$ часто виникає явище перенавчання (коли класифікація нестійка, збільшується вірогідність помилки на тестовій вибірці). У боротьби з цим недоліком можна застосувати метод скорочення ваг. Його суть у полягає в обмеженні росту норми матриці ваг $\boldsymbol{w}$. Це досягається шляхом додавання до $\boldsymbol{Q(w)}$ штрафного доданку $\boldsymbol{Q_\tau(w)} = \boldsymbol{Q(w)} + (\tau/2)*\|\boldsymbol{w}\|^2$. Тоді ваги оновлюватимуться наступним чином (2.29):

$$w := w(1 - \eta\tau) - \eta\nabla Q(w). \tag{2.29}$$

4) за наявності горизонтальних асимптот у функції активації процес навчання може почати стагнувати. Це мотивується великим значенням скалярного добутку $< w_i, x_i >$, що веде до близьких до нуля значень $\varphi`$, тому ваги перестають істотно змінюватися. Для уникнення цього зазвичай застосовують попередню нормалізацію ознак (2.30):

$$x^j := \frac{x^j - x^j_{\min}}{x^j_{\max} - x^j_{\min}},\ j = 1, \ldots, n, \tag{2.30}$$

де x^j_{min}, x^j_{max} –мінімум та максимум відхилення ознаки *j*. А якщо до цього ваги визначили як $w_j \in [-1/n\ ;\ 1/n]$, тоді значення скалярного добутку $< w , x > \in [-1\ ;\ 1]$.

Слід зазначити, що регуляція (скорочення ваг) – теж спосіб уникнення стагнації навчання.

2.6.3. Адаптивний градієнтний спуск

Адаптивний градієнтний спуск (AdaGrad) – модифікація алгоритму стохастичного градієнтного спуску з індивідуальним розрахунком коефіцієнта навчання для кожного параметра, вперше опублікована в 2011 році.[31][32] Це дозволяє йому збільшити швидкість навчання на більш розріджених параметрах і зменшити швидкість навчання на менш розріджених. Ця стратегія часто покращує збіжність відносно стандартного стохастичного градієнтного спуску в умовах розрідженості даних і більш інформативних розріджених параметрів. Прикладами таких застосувань можуть слугувати задачі обробки природних мов або задачі обробки зображень. [31] У методі все ще є темп навчання $\boldsymbol{\eta}$, але він множиться на елементи вектора $\{ G_{j,j} \}$, що є діагональним елементом матриці (2.31).

$$G = \sum_{\tau=1}^{t} g_\tau g_\tau^{\mathrm{T}} \tag{2.31}$$

де g_τ – градієнт функціоналу $\boldsymbol{Q}$ на ітерації τ.

Діагональ розраховується як (2.32):

$$G_{j,j} = \sum_{\tau=1}^{t} g_{\tau,j}^2. \tag{2.32}$$

Вектор оновлюється на кожній ітерації. (2.33)

$$w := w - \eta \operatorname{diag}(G)^{-\frac{1}{2}} \circ g \tag{2.33}$$

У скалярній формі вектор оновлюється наступним чином (2.34)

$$w_j := w_j - \frac{\eta}{\sqrt{G_{j,j}}} g_j. \tag{2.34}$$

Кожен { $G_{j,j}$ } зменшує множник темпу навчання, що застосовується до параметра w_i. Оскільки знаменник темпу навчання є l_2 нормою попередніх похідних, сильні зміни параметрів затухають в той час, коли параметри з меншою зміною отримують вищі темпи навчання.

2.6.4. Поширення кореня середніх квадратів

Поширення кореня середніх квадратів (Root Mean Square Propagation, RMSProp) – ще одна модифікація методу градієнтного спуску з індивідуальним розрахунком темпу навчання параметрів. Головна ідея підходу полягає в діленні темпу навчання вагів на ковзне середнє величин нещодавніх градієнтів для цих вагів.[33] Ковзне середнє розраховується за ф-лою (2.35):

$$v(w,t) := \gamma v(w,t-1) + (1-\gamma)(\nabla Q_i(w))^2 \tag{2.35}$$

де γ – параметр забування.

Ваги оновлюються наступним чином (2.36):

$$w := w - \frac{\eta}{\sqrt{v(w,t)}} \nabla Q_i(w) \tag{2.36}$$

2.6.5. Адам

Адам (Adam, Adaptive Moment Estimation) – модифікація алгоритма RMSProp. У цьому алгоритмі використовується ковзне середнє градієнтів і других моментів градієнтів. При параметрах $w^{(t)}$ і функції втрат $L^{(t)}$, де t – поточна ітерація навчання, навчання параметрів алгоритмом реалізується як (2.37):

$$\begin{aligned}
m_w^{(t+1)} &\leftarrow \beta_1 m_w^{(t)} + (1-\beta_1)\nabla_w L^{(t)} \\
v_w^{(t+1)} &\leftarrow \beta_2 v_w^{(t)} + (1-\beta_2)(\nabla_w L^{(t)})^2 \\
\hat{m}_w &= \frac{m_w^{(t+1)}}{1-(\beta_1)^{t+1}} \\
\hat{v}_w &= \frac{v_w^{(t+1)}}{1-(\beta_2)^{t+1}} \\
w^{(t+1)} &\leftarrow w^{(t)} - \eta \frac{\hat{m}_w}{\sqrt{\hat{v}_w} + \epsilon}
\end{aligned} \tag{2.37}$$

Де ϵ – малий скаляр, що застосовується для уникнення ділення на нуль, β_1 та β_2 – параметри забування для градієнтів та других моментів градієнтів відповідно. Операції піднесення до квадрату та квадратного кореня виконуються поелементно.

2.6.6. Виключення з’єднань моделі

Перенавчання (overfitting) – одна з проблем штучних нейронних мереж, що полягає в наступному: модель якісно описує лише приклади з навчальної вибірки, адаптуючись до навчальних прикладів замість пристосування до класифікації прикладів, відсутніх у навчальній вибірці. Тобто модель втрачає можливість узагальнення. Для вирішення такої проблеми пропонується простий та надійний засіб – виключення з'єднань (dropout) (рисунок 2.8).

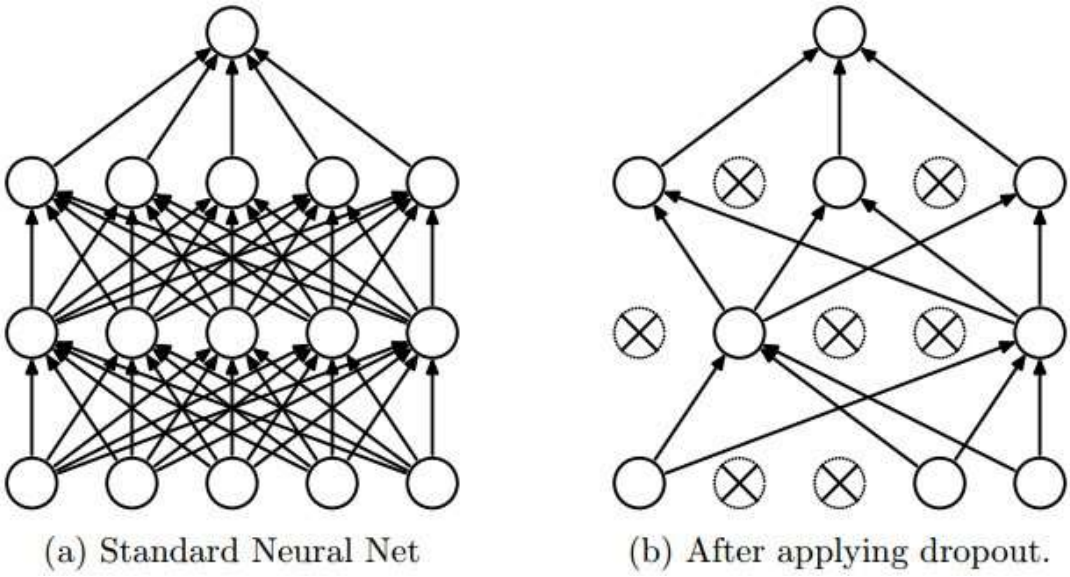

Рисунок 2.8 – Схематичне зображення Dropout

Головна ідея виключення з’єднань моделі полягає в тому, що замість одної нейронної мережі навчаються декілька, після чого отримані результати усереднюються.

Мережі для навчання будуються за допомогою виключення з мережі деяких структурних елементів з імовірністю *p*. Під виключенням мається на увазі, що за будь-яких вхідних параметрах він даватиме нульовий вихід.

Виключені нейрони не вносять свій вклад у процес навчання на жодному з етапів алгоритму зворотного поширення помилки, тому виключення хоча б одного з нейронів еквівалентне навчанню нової нейронної мережі. [35]

Механізм роботи Dropout:

При умовах, що

5) $h(x) = xW + b$ – лінійна проекція вхідного d_i-мірного вектора x на d_h-мірний простір вихідних значень;
6) $a(h)$ – функція активації,

Застосування виключення з'єднань моделі до даної проекції на етапі навчання можна представити як змінену функцію активації $f(h) = D \odot a(h)$, де $D = (X_1,\ldots,X_{dh})$ – d_h-мірний вектор випадкових величин X_i, розподілених за законом Бернуллі.

Застосування Dropout до нейрона і тоді можна визначити як (2.38):

$$O_i = X_i a(\sum_{k=1}^{d_i} w_k x_k + b) = \begin{cases} a(\sum_{k=1}^{d_i} w_k x_k + b), & \text{if} \quad X_i = 1 \\ 0, & \text{if} \quad X_i = 0 \end{cases}, \tag{2.38}$$

де $P(X_i = 0) = p$.

Так як на етапі навчання нейрон залишається в мережі, з імовірністю $1-p$, на етапі тестування необхідно симулювати поведінку ансамбля нейронних мереж, використаного під час навчання. Для цього пропонується на етапі тестування множити функцію тестування на $q=1-p$. Таким чином, на етапі навчання вихід нейрону i визначається як (2.39), а на етапі тестування він визначається як (2.40).

$$O_i = X_i a(\sum_{k=1}^{d_i} w_k x_k + b) \tag{2.39}$$

$$O_i = qa(\sum_{k=1}^{d_i} w_k x_k + b) \tag{2.40}$$

2.7. Порівняльний аналіз різних архітектур нейронних мереж

До порівняльного аналізу були включені наступні архітектури штучних нейронних мереж:

Мережа 1:

- один нейрон GRU;
- функція активації (softmax);

Мережа 2:

- 200 нейронів LSTM;
- повнозв'язний шар;
- функція активації (softmax);

Мережа 3:

- 1-вимірний згротковий шар;
- 200 нейронів GRU;
- повнозв’язний шар;
- функція активації (softmax);

Мережа 4:

- 2 шари по 200 LSTM нейронів;
- повнозв’язний шар;
- функція активації (softmax);

Мережа 5:

- 1-мірний згортковий шар;
- 3 шари по 200 LSTM-нейронів з нормалізацією виходу;
- повнозв’язний шар;
- функція активації (softmax).

У всіх моделях в якості функції активації нейронів була застосована сигмоїда, в якості алгоритму навчання був обраний алгоритм Adam із наступними параметрами:

– learning rate = 0.001;
– $\beta_1 = 0.9$;
– $\beta_2 = 0.999$;
– $\varepsilon = 10^{-8}$.

Кожна з моделей навчалася на одному наборі даних 20 епох, після чого проводився замір якостей моделей. Під час навчання проводилася мінімізація параметра ctc loss, що визначається з опису алгоритму connectionist temporal classification. [37]

Суть роботи алгоритму CTC подібна роботі EM-алгоритму, описаного в п.п. 2.2.1.

Якість моделі визначалася за середнім відсотком неправильно розпізнаних фраз (тобто якщо хоча б один символ розпізнаний невірно, уся фраза вважатиметься неправильною).

Результати порівняння наведені в таблиці 2.1 та рисунку 2.9.

Таблиця 2.1 – Порівняння моделей

MAPE	Модель				
Епоха	1	2	3	4	5
5	52.4%	47.5%	44.6%	48.6%	39.5%
10	22.1%	18.2%	20.3%	27.3%	20.3%
15	34.5%	32.6%	32.7%	14.8%	16.8%
20	22.5%	16.0%	19.6%	13.1%	7.6%

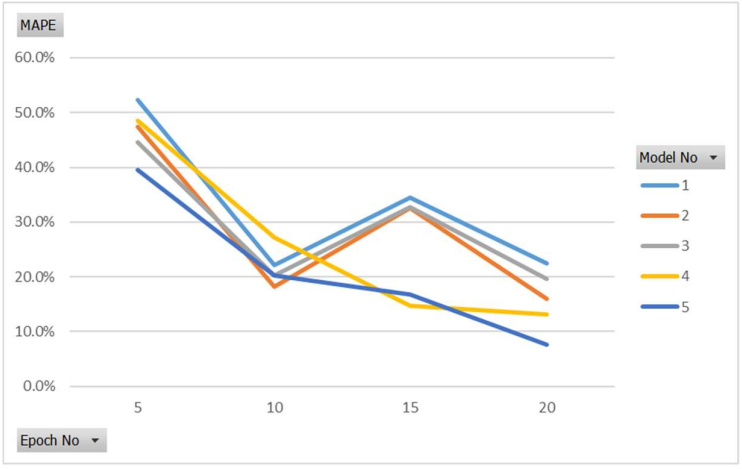

Рисунок 2.9 – динаміка навчання моделей.

Висновки до розділу

Було проведено дослідження відомих методів штучного інтелекту, що застосовуються в поставленій задачі. Серед них було обрано рекурентні нейронні мережі, а саме довгу короткочасну пам'ять через успішність її застосувань в останні роки.

Було досліджено різні алгоритми навчання нейронних мереж. Серед них було обрано алгоритм Адам через його кращу збіжність відносно інших альтернатив.

Було проведене порівняльне дослідження мережі «LSTM» у порівнянні з більшь простими архітектурами. Дослідження показало, що застосування більш складної мережі є цілком доцільним, так як навіть на обмеженому часі на навчання вона показує значно кращі результати.

Варто зазначити, що ще одним приводом до вибору «LSTM» стала її властивість нечутливості до затримок між важливими подіями, що дозволяє їй краще розпізнавати символи, беручи до уваги ті, що вже були розпізнані.

3 ОПИС АРХІТЕКТУРИ ПРОДУКТУ

Оскільки робота спрямована більше на дослідження застосування методів штучного інтелекту в поставленій задачі, ніж на розробку комерційного зразка, система має дещо «інженерний» вигляд і тому потребує окремого опису.

3.1. Концептуальна схема обробки звуку

Побудована модель складається з наступних елементів (вказані в порядку від входу до виходу):

- 1-мірний згортковий шар (функція активації – ReLU);
- 3 шари по 200 LSTM-нейронів з нормалізацією виходу (функція активації – ReLU);
- повнозв’язний шар (функція активації – ReLU);
- шар активації (softmax).

Етапи обробки вихідного файлу схематично зображені на рисунку 3.1.

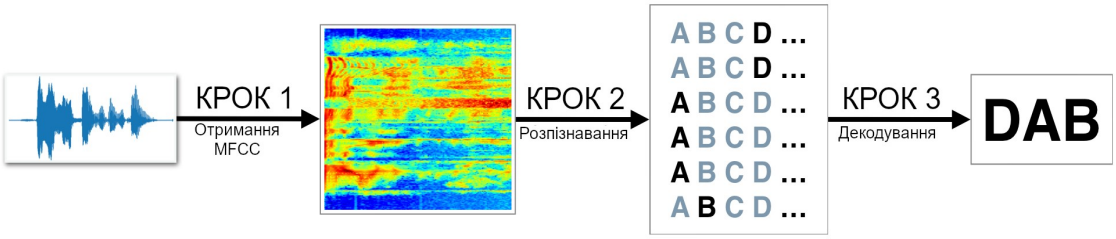

Рисунок 3.1 – Схема обробки звуку розробленої системи

Тобто на першому кроці обробки з ІКМ (для формату wav аудіофайл, формально, і є ІКМ, для MP3 – необхідне попереднє декодування для отримання ІКМ) вилучаються мел-кепстральні коефіцієнти розкладу вихідного сигналу. Застосовується ковзне вікно розміром 10 мс із кроком 5 мс.

Після отримання набору мел-кепстральних розкладів вони подаються на вхід моделі, яка для кожного з них генерує прогнозований індекс розпізнаної літери в алфавіті (у роботі розглядається російська абетка без спец. символів).

Оскільки в результаті розпізнавання буде значно більше літер, ніж у початковому звукозаписі (через застосування ковзного вікна), наступним етапом є застосування декодера на основі алгоритму CTC.

Після всіх цих операцій буде отриманий результат розпізнавання.

3.2. Попередні налаштування

Робота програмного продукту передбачає налаштоване середовище (інтерпретатор мови Python версії 3.6.7, список необхідних бібліотек можна знайти в дотатку з лістингом файлів проекту: файл «requirements.txt»). Оскільки проект використовує бібліотеку tensorflow для реалізації архітектури штучної нейронної мережі, для оптимізації часу навчання та швидкості розпізнавання рекомендується застосовувати його збірки під графічні процесори. Використані зовнішні набори для обчислень: NVidia CUDA [38] та cuDNN [39].

Для адекватної роботи проекту (з точки зору часу виконання операцій) рекомендується наступна конфігурація комп’ютера/сервера:

- центральний процесор з 4 фізичними ядрами та частотою не менше 3 ГГц;
- 16 ГБ оперативної пам’яті (на час навчання);
- 8 ГБ пам’яті на жорсткому диску (для проекту, навчальної вибірки та всіх бібліотек);
- Графічний прискорювач із підтримкою технології NVidia CUDA та об’ємом відеопам’яті не менше 6 ГБ.

3.3. Середовище Jupyter Notebook

Система є готовою для користування впевненого користувача середовища jupyter notebook, однак для можливості користування рядовому користувачеві необхідно або розробити окремий інтерфейс для роботи, або керуватися інструкціюєю по роботі з тим, що є. За час розробки було досягнуто лише першого з варіантів.

Jupyter Notebook – це графічна оболонка для IPython (інтерактивне середовище Python), що розширює ідею підходу до інтерактивних обчислень, закладену в інтерпретатор мови Python за замовчуванням.

Ключові можливості такого середовища:

– редагування Python коду у браузері, з підсвічуванням синтаксису, автоматичними відступами та автоматичним завершенням;

– запуск коду в браузері;

– відображення результатів обчислень з медіа представленням (схеми, графіки);

– робота з мовою розмітки Markdown і LaTeX.

Фрагменти файлу ipynb, що дозволяє працювати з моделлю:

– комірка 1 – підключення та ініціалізація основних бібліотек (рис. 3.2);

– комірка 2 – побудова архітектури моделі (рис 3.3)

– комірка 3 – навчання моделі (рис. 3.4);

– комірка 4 – вивід результатів розпізнавання для певного звукозапису навчальної чи перевірочної виборки (рис. 3.5);

– комірка 5 – вивід результатів розпізнавання для певного звукового файлу за його шляхом (рис. 3.6);

За бажанням можливо навчити і перевірити не фінальну архітектуру моделі, а будь-яку з модуля sample_models.py.

In [1]:
```
######################################################################
# RUN THIS CODE CELL IF YOU ARE RESUMING THE NOTEBOOK AFTER A BREAK #
######################################################################

# allocate 50% of GPU memory (if you like, feel free to change this)
from keras.backend.tensorflow_backend import set_session, get_session
import tensorflow as tf

config = tf.ConfigProto()
#config.gpu_options.per_process_gpu_memory_fraction = 0.95
#config.log_device_placement = True
set_session(tf.Session(config=config))

# watch for any changes in the sample_models module, and reload it automatically
% load_ext autoreload
% autoreload 2
# import NN architectures fo
# r speech recognition
from sample_models import *
# import function for training acoustic model
from train_utils import train_model
```

```
Using TensorFlow backend.
```

Рисунок 3.2 – підключення та ініціалізація ключових бібліотек

In [6]:
```
me = saloedov(filters=200,
              kernel_size=11,
              conv_stride=2,
              dropout_rate=.1,
              conv_border_mode='valid',
              units=150,
              lstm_layers=4, )
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
the_input (InputLayer)       (None, None, 13)          0
_________________________________________________________________
layer_1_conv (Conv1D)        (None, None, 200)         28800
_________________________________________________________________
conv_batch_norm (BatchNormal (None, None, 200)         800
_________________________________________________________________
bidirectional_1 (Bidirection (None, None, 300)         421200
_________________________________________________________________
bt_lstm_0 (BatchNormalizatio (None, None, 300)         1200
_________________________________________________________________
bidirectional_2 (Bidirection (None, None, 300)         541200
_________________________________________________________________
bt_lstm_1 (BatchNormalizatio (None, None, 300)         1200
_________________________________________________________________
bidirectional_3 (Bidirection (None, None, 300)         541200
_________________________________________________________________
bt_lstm_2 (BatchNormalizatio (None, None, 300)         1200
_________________________________________________________________
bidirectional_4 (Bidirection (None, None, 300)         541200
_________________________________________________________________
bt_final_lstm_layer (BatchNo (None, None, 300)         1200
_________________________________________________________________
time_distributed_2 (TimeDist (None, None, 36)          10836
_________________________________________________________________
softmax (Activation)         (None, None, 36)          0
=================================================================
Total params: 2,090,036
Trainable params: 2,087,236
Non-trainable params: 2,800
_________________________________________________________________
None
```

Рисунок 3.3 – ініціалізація моделі

```
from keras.optimizers import Adam

train_model(input_to_softmax=test,
            pickle_path='model_test.pickle',
            save_model_path='model_test.h5',
            epochs=20,
            minibatch_size=200,
            spectrogram=False,
            optimizer=Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-8),
            max_duration=10)
```

```
Epoch 1/20
17/17 [==============================] - 82s 5s/step - loss: 406.1687 - val_loss: 301.1680
Epoch 2/20
17/17 [==============================] - 66s 4s/step - loss: 257.5253 - val_loss: 253.3529
Epoch 3/20
17/17 [==============================] - 66s 4s/step - loss: 241.0051 - val_loss: 243.9929
Epoch 4/20
17/17 [==============================] - 66s 4s/step - loss: 235.5289 - val_loss: 239.6616
Epoch 5/20
17/17 [==============================] - 67s 4s/step - loss: 230.9776 - val_loss: 237.4719
Epoch 6/20
17/17 [==============================] - 67s 4s/step - loss: 226.8700 - val_loss: 236.9606
Epoch 7/20
17/17 [==============================] - 67s 4s/step - loss: 222.1577 - val_loss: 249.1466
Epoch 8/20
17/17 [==============================] - 68s 4s/step - loss: 214.3618 - val_loss: 283.3436
Epoch 9/20
17/17 [==============================] - 68s 4s/step - loss: 208.8090 - val_loss: 212.9035
Epoch 10/20
17/17 [==============================] - 68s 4s/step - loss: 202.3790 - val_loss: 203.7667
```

Рисунок 3.4 – Навчання моделі (перші 10 епох)

```
get_predictions(index=1,
                partition='train',
                input_to_softmax=test,
                model_path='results/model_test.h5')
```

```
--------------------------------------------------------------------------------
True transcription:

мы втянули его и он упал разбил земную кору отклонил полюсы
--------------------------------------------------------------------------------
Predicted transcription:

мы фтянули ево и он упал разбил зем ную кору отклонил полюс
--------------------------------------------------------------------------------
```

Рисунок 3.5 – перевірка результатів розпізнавання

Рисунок 3.6 – розпізнавання незалежного звукозапису

3.4. Навчання

Синтезована модель штучної нейронної мережі навчалася за допомогою навчального корпусу VoxForge [36]. Завантажена вибірка налічує близько 9000 звукозаписів та їх текстових форм.

З вибірки були обрані лише звукозаписи довжиною не більше 100 секунд, серед яких 20% (від початкової кількості) було вилучено на перевірочну вибірку для оцінювання якості навчання моделі.

Висновки до розділу

Розроблена система успішно виконує поставлену задачу. Точність розпізнавання є припустимою. Однак, вона має ряд способів покращення, які будуть описані у висновках по роботі.

РОЗДІЛ 4 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

У даному розділі буде розглянуто ключові особливості розробленої системи як стартап-проекту. Проект розглядатиметься як система для логування текстових записів нарад.

4.1 Опис ідеї проекту

Спочатку проаналізуємо та подамо у вигляді таблиці зміст ідеї стартап-проекту, можливі напрямки застосування та основні вигоди, які може отримати користувач товару. Ці характеристики стартап-проекту зображено в таблиці 4.1.

Таблиця 4.1 - Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Програмна онлайн-платформа для переводу звукових записів у їх текстову інтерпретацію.	1. Застосування для обробки звукозаписів нарад	Можливість швидко отримати текстову інтерпретацію звукозапису наради для подальшого аналізу й складання резюме наради і т. ін.
	2. Застосування для обробки звукозаписів контакт-центрів.	Можливість автоматично відстежувати застосування невідповідної лексики співробітниками контакт-центрів.

Тепер зробимо аналіз потенційних техніко-економічних переваг ідеї порівняно із пропозиціями конкурентів. Результати аналізу зображено в таблиці 4.2.

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

№ п/п	Техніко-економічні характеристики ідеї	Товари/концепції конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Конкурент 1	Конкурент 2			
1.	Ціна	2000$/рік	4000$/рік	5000$/рік			+
2.	Прибутки	30000$/рік	40000$/рік	20000$/рік		+	
3.	Контроль якості	Аналітики та програмісти	Аналітики, програмісти та деякі клієнти	Програмісти			+
4.	Динаміка галузі	Швидка	Повільна	Швидка		+	
5.	Постійні витрати	10000$/рік	20000$/рік	15000$/рік			+
6.	Змінні витрати	5000$ - 10000$/рік	1000$ - 2000$/рік	2000$ - 5000$/рік	+		
7.	Патенти на продукти	Немає	Патент на кожний проект	Декілька патентів на винахід	+		

Продовження таблиці 4.2

№ п/п	Техніко-економічні характеристики ідеї	Товари/концепції конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Конкурент 1	Конкурент 2			
8.	Гнучкі ціни	Ціна єдина	Ціна варіюється з року в рік	Ціна єдина		+	
9.	Законодавчі обмеження	Немає	Немає	Обмеження на кількість розробників			+

4.2 Технологічний аудит ідеї проекту

Визначимо технологічну здійсненність ідеї проекту за допомогою аналізу таких складових, як технології, за якою буде виготовлено товар згідно ідеї проекту, існування таких технологій, чи їх необхідно розробити / доробити, доступність таких технологій авторам проекту. Результати даного аналізу зображено в таблиці 4.3.

Таблиця 4.3 – Технологічна здійсненність ідеї проекту

Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
Програмна онлайн-платформа для переводу звукових записів у їх текстову інтерпритацію.	Технологія проектування та розробки нейронних мереж Scikit-learn.	Так	Технологія доступна, однак, її функціонал менший за відповідний у альтернативних технологій.
	Технологія проектування та розробки нейронних мереж tensorflow (без застосування додаткових модулів).	Так	Дані технології доступні, однак, штатного функціоналу все ще не вистачає для вирішення поставлених задач.
	Технологія проектування та розробки нейронних мереж tensorflow (із застосуванням Keras).	Так	Дані технології доступні.
Обрана технологія реалізації ідеї проекту: технологія проектування та розробки нейронних мереж tensorflow (із застосуванням Keras).			

4.3 Аналіз ринкових можливостей запуску стартап-проекту

Проведемо аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку. Результати даного аналізу зображено в таблиці 4.4.

Таблиця 4.4 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	*Показники стану ринку (найменування)*	*Характеристика*
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж, грн/ум.од	300 000
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Висока точність розпізнавання
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі (або по ринку), %	150

Таким чином, за попереднім оцінюванням, ринок є привабливим для входження.

Надалі визначимо потенційні групи клієнтів, їх характеристики, та сформуємо орієнтовний перелік вимог до товару для кожної групи. Ці дані зображено в таблиці 4.5.

Таблиця 4.5 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	*Потреба, що формує ринок*	*Цільова аудиторія (цільові сегменти ринку)*	*Відмінності у поведінці різних потенційних цільових груп клієнтів*	*Вимоги споживачів до товару*
1	Запис і обробка нарад з метою перетворити звукову доріжку на текстову інтерпретацію проведеної наради	Середній та великий бізнес	Великому бізнесу також необхідно мати можливість перекладати іншомовні записи	Зали для нарад споживача мають бути обладнані хоча б один (бажано по одному на члена наради) мікрофонами
2	Переклад іноземних мов на задану (у текстовій формі)	Великий бізнес	У разі нараз з участю іншомовних членів, які бувають у крупного бізнесу, може виникнути необхідність вести записи з інших мов	Компанія має мати міжнародні зв’язки або постійних іншомовних членів нарад

Після визначення потенційних груп клієнтів проведемо аналіз ринкового середовища: складемо таблиці факторів, що сприяють ринковому впровадженню проекту (таблиця 4.6), та факторів, що йому перешкоджають (таблиця 4.7).

Таблиця 4.6 – Фактори загроз

№ п/п	*Фактор*	*Зміст загрози*	*Можлива реакція компанії*
1	Відсутність попиту	Бізнес може не оцінити переваги продукту, або ж у цілому відмовитися від ведення нарад	Акцентувати увагу на клієнтах, що вже скористалися продуктом, якщо такі є, навести інфографіку результативності (очікувану), запропонувати знижку потенційному клієнту в рамках тендеру.
2	Неточне розпізнавання	Дефекти та особливості мовлення певних осіб можуть привести до неточностей у текстовій інтерпретації наради	Розробка і випуск оновлення мовленнєвого пакету, де виправлена ця проблема.

Таблиця 4.7 – Фактори можливостей

№ п/п	*Фактор*	*Зміст можливості*	*Можлива реакція компанії*
1	Кобрендінг	Пропозиція від певної компанії, що спеціалізується на системах корпоративного зв’язку, розробити спільний продукт	Виділення частини штату на реалізацію проекту, підготовка акційних пропозицій по переходу на новий продукт існуючим клієнтам.

Надалі проведемо аналіз пропозиції: визначимо загальні риси конкуренції на ринку. Результати даного аналізу зображені в таблиці 4.8.

Таблиця 4.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	*В чому проявляється дана характеристика*	*Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)*
1. Чиста конкуренція	Гравці ринку не мають явних переваг один над одним	Більш вигідні умови на тендерах, агресивний маркетинг
2. Регіональна конкуренція	Гравці ринку – інтернаціональні підприємства	Вихід на ті ринки, які ще не зайняті конкурентами
3. Внутрішньогалузева конкуренція	Гравці ринку знаходяться в одній галузі – розробці ПЗ	
4. Товарно-видова конкуренція	Усі продукти гравців ринку мають одне призначення	Розробка найбільш інтуїтивного інтерфейсу, розробка унікальних мовленнєвих пакетів, оптимізація алгоритмів (щоб аналіз проходив швидше, ніж у конкурентів)
5. Конкурентні переваги неціновi	Продукти відрізняються гнучкістю, функціоналом (незначно) і надійністю.	У маркетингу неявно порівнювати власний продукт з іншими, робити вигідні цінові пропозиції
6. Марочна конкуренція	Значна увага приділяється бренду, що розробив продукт	Кобрендінг

Після аналізу конкуренції проведемо більш детальний аналіз умов конкуренції в галузі програмних продуктів для проектування двигунів. Результати даного аналізу зображено в таблиці 4.9.

Тепер визначимо та обґрунтуємо фактори конкурентоспроможності, які зображені в таблиці 4.10.

Таблиця 4.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	*Фактор конкурентоспроможності*	*Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)*
1	Інтеграція	Продукт може бути використаний абсолютно в любому приміщенні за умови наявності одного або декількох мікрофонів. Продукт не потребує встановлення будь-якого іншого спеціалізованого ПЗ чи придбання спеціалізованого апаратного забезпечення.
2	Модульність	Кожен замовник індивідуально обирає мовленнєві пакети для себе, при цьому не втрачаючи можливість докупити інші пакети в майбутньому
3	Гнучкість	Кожен замовник має можливість замовити розширення функціоналу продукту під його конкретні задачі
4	Розділення осіб	Продукт автоматично розподіляє репліки між членами наради і у виводі позначатиме, хто саме, що і коли говорив

Таблиця 4.9 – Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Складові аналізу	Динаміка галузі, продуктова лінія, бар'єри проникнення	Наявність товарних знаків, доступ до ресурсів, патенти на продукти	Концентрація постачальників, диференціація витрат	Рівень чутливості до зміни цін, прибутки, контроль якості	Ціна, лояльність споживачів
Висновки:	Конкуренція не є інтенсивною, адже даний ринок ще ніким не зайнятий.	Для входу на ринок необхідно створити товарний знак та написати бета-версію програмного продукту. На даний момент потенційних конкурентів немає.	Постачальники не диктують умови роботи на ринку, бо програмному продукту не потрібно постачання.	Клієнти диктують умови роботи на ринку, бо вони є єдиним джерелом прибутку компанії.	При наявності товарів замінників необхідно буде зменшувати ціну програмного продукту чи створювати ПЗ для інших технічних систем.

За визначеними факторами конкурентоспроможності проведемо аналіз сильних та слабких сторін стартап-проекту. Результати даного аналізу зображено в таблиці 4.11.

Таблиця 4.11 – Порівняльний аналіз сильних та слабких сторін системи «MeetSpeech»

№ п/п	*Фактор конкурентоспроможності*	*Бали 1-20*	*Рейтинг товарів-конкурентів у порівнянні з* MeetSpeech						
			–3	*–2*	*–1*	*0*	*+1*	*+2*	*+3*
1	Інтеграція	15					*		
2	Модульність	16			*				
3	Гнучкість	18			*				
4	Розділення осіб	17			*				

Тепер проведемо SWOT-аналіз на основі виділених загроз і можливостей, та сильних і слабких сторін проекту. SWOT-матриця зображено в таблиці 4.12.

Таблиця 4.12 – SWOT-аналіз стартап-проекту

Сильні сторони: Розділення реплік між членами нарад, багатомовність, гнучкість	Слабкі сторони: Інтеграція обмежена необхідністю наявності ПК/Сервера з ОС «Windows».
Можливості: Кобрендінг	Загрози: Неточність розпізнавання, відсутність попиту

На основі SWOT-аналізу розробимо альтернативи ринкової поведінки для виведення стартап-проекту на ринок та орієнтований оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок. Дані альтернативи зображено в таблиці 4.13.

Таблиця 4.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	*Альтернатива (орієнтовний комплекс заходів) ринкової поведінки*	*Ймовірність отримання ресурсів*	*Строки реалізації*
1	Реалізація можливості використання системи не тільки в нарадах, а й телефонних дзвінках	Середня	18 місяців
2	Створення багатокомпонентного варіанту системи для переговорів між віддаленими офісами	Висока	24 місяці
3	Розробка MVP	Висока	12 місяців

Серед даних альтернатив було обрано третю альтернативу, адже строки її реалізації найменші та є ймовірність отримання ресурсів.

4.4 Розроблення ринкової стратегії проекту

Для розроблення ринкової стратегії першим кроком необхідно описати цільові груп потенційних споживачів, які можна побачити в таблиці 4.14.

Таблиця 4.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1.	Малий бізнес	Середня	5-10 підприємств в рік	Середня	Середня
2.	Середній бізнес	Готові	5-10 підприємств в рік	Слабка	Середня
3.	Великий бізнес	Готові	3-5 закладів в рік	Слабка	Складна
Було обрано цільову групу підприємств групи середнього бізнеса.					

Для роботи в обраних сегментах ринку необхідно сформувати базову стратегію розвитку, яку ображено в таблиці 4.15.

Таблиця 4.15 – Визначення базової стратегії розвитку

Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Концентрація на потребах одного цільового сегменту – нарадах.	Створений продукт є унікальним і не має аналогів.	Стратегія спеціалізації.

Наступним кроком є вибір стратегії конкурентної поведінки, яку зображено в таблиці 4.16.

Таблиця 4.16 – Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохід-цем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
Так.	Компанія буде шукати нових споживачів, але і буде намагатися забирати існуючих у конкурентів.	Компанія не буде копіювати характеристики конкурентів.	Стратегія заняття конкурентної ніші.

Тепер розробимо стратегія позиціонування, що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект. Її зображено в таблиці 4.17.

Таблиця 4.17 – Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспро-можні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
Обробка звукозапису має бути швидкою та отриманий текст має адекватно відповідати реплікам.	Проведен-ня крупних оновлень (оптимі-зація розрахун-ків), створення додатков-ого функціона-лу.	Товар є «першопрохід-цем» на ринку та проект не буде копією інших програм.	Перша в країні програма для розпізнавання звукозаписів нарад, одна програма - єдина база розпізнаних звукозаписів, програма працює в режимі онлайн.

4.5 Розроблення маркетингової програми стартап-проекту

Сформуємо маркетингову концепцію товару, який отримає споживач. В таблиці 4.18 зображено результати попереднього аналізу конкурентоспроможності товару.

Таблиця 4.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1.	Точне розпізнавання мовлення.	Точне розпізнавання усного мовлення.	Розпізнавання мовлення на багатьох мовах та з розділенням осіб, що говорять.

Надалі розробимо трирівневу маркетингову модель товару: уточнимо ідею продукту, його фізичні складові, особливості процесу його надання. Дана модель зображена в таблиці 4.19.

Таблиця 4.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
I. Товар за задумом	Програмний продукт для створення електричних двигунів, який дозволяє користувачу задавати власні обмеження на технічні та технологічні характеристики двигуна та критерій оптимальності, оптимум якого буде розраховано.
II. Товар у реальному виконанні	Властивості / характеристики: 1. Можливість задавати власні технічні характеристики двигуна 2. Можливість вибору критерію оптимальності серед запропонованих. 3. Зберігання отриманих результатів в базі даних користувача.
	Якість: програмний продукт пройшов всі етапі тестування та готовий до використання.
	Файл з розширенням “.py”, віртуальне середовище.
	Марка: назва організації-розробника «iEngine», назва товару «EngineOpt».

Продовження таблиці 4.19

Рівні товару	Сутність та складові
III. Товар із підкріпленням	Спеціаліст із впровадження встановлює ПЗ.
	Відділ розробки підтримує життєдіяльність ПЗ.
Захист програмного продукту буде організовано за допомогою ноу-хау.	

Тепер визначимо цінові межі, якими необхідно керуватись при встановленні ціни на потенційний товар, яке передбачає аналіз ціни на товари-аналоги або товари субститути, а також аналіз рівня доходів цільової групи споживачів. Аналіз проводився експертним методом і його результати зображено в таблиці 4.20.

Таблиця 4.20 – Визначення меж встановлення цін

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
30000-50000 $/рік	50000-60000 $/рік	300000-500000 $/рік	Нижня межа – 40000 $/рік, верхня межа - 50000 $/рік

Надалі визначимо оптимальну систему збуту, в межах якого приймається рішення. Дану систему зображено в таблиці 4.21.

Таблиця 4.21 – Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Клієнт виплачує гроші на рік, тоді до нього приходить спеціаліст із впровадження інформаційних систем і встановлює ПЗ на комп'ютер клієнта.	Встановити програмний продукт на комп'ютери клієнтів.	Один посередник – спеціаліст по впровадженню інформаційних систем.	Канал збуту одного рівня.

Тепер розробимо концепцію маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів. Дану концепцію зображено в таблиці 4.22.

Таблиця 4.22 – Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
Клієнт намагається знайти нові методи покращення створюваних ним двигунів.	E-mail, мережа Інтернет, соціальні мережі.	Унікальність ПЗ, можливість вибору власних критеріїв оптимальності двигуна.	Продемонструвати унікальність даного ПЗ та показати можливу економію щодо створення електричних двигунів при його використанні.	Показати можливість не за велику ціну збільшити свій прибуток за допомогою покращення двигунів.

Висновки до розділу

В даному розділі було повністю виконано перший етап розроблення стартап-проекту, а саме, виконано маркетинговий аналіз стартап-проекту.

За допомогою нього можна сказати, що існує можливість ринкової комерціалізації проекту, адже на ринку програм для підприємств з машинобудування наявний попит на програмні продукти, здатні оптимально проектувати машини, до того ж рентабельність роботи є досить високою.

З огляду на потенційну групу клієнтів, а саме, підприємства сфери машинобудування, та відсутності конкуренції є великі перспективи впровадження даного програмного забезпечення.

Для ринкової реалізації проекту доцільно обрати таку альтернативу впровадження: створення MVP та впровадження його в невелику кількість компаній машинобудівників.

ВИСНОВКИ ПО РОБОТІ

Під час роботи було проведено дослідження сучасних методів обробки та збереження звуку. Було проведено аналіз сучасних засобів інтелектуальної обробки звукових сигналів у задачі автоматичного розпізнавання мовлення.

У якості контейнера звукозаписів було обрано MP3 через його поширеність і відсутність ліцензійних зборів.

Серед методів обробки звуку як інформаційного сигналу було обрано MFCC, так як він дає кращі результати відносно фур'є-аналізу.

У якості засобу інтелектуального аналізу звуку було обрано рекурентну нейронну мережу «довга короткочасна пам'ять», так як вона показала чудові результати в порівняльному аналізі з розділу 2, нечутлива до затримок між важливою інформацією, а також широко застосовується провідними компаніями у задачі автоматичного розпізнавання мовлення і має значний успіх.

Кінцева система успішно розпізнає мовні символи, однак, у ній є що покращувати:

– додати модель мови для покращення якості тексту (у поточний момент подвійні та співзвучні символи можуть розпізнаватися з низькою точністю). У якості такої моделі може виступати прихована марковська модель;

– додати модель для розділення спікерів (класифікація реплік за особою, що їх говорить);

– додати можливість користувачу вносити корективи в уже розпізнаний текст для перенавчання моделі на нових даних;

– розробити веб-інтерфейс для так званого пересічного користувача;

– збільшити кількість шарів ШНМ та нейронів у них (можливо лише за наявності більш потужного апаратного забезпечення);

– розширити навчальну вибірку.

ПЕРЕЛІК ПОСИЛАНЬ

1. mp3 [Електронний ресурс] – Режим доступу: https://www.iis.fraunhofer.de/en/ff/amm/prod/audiocodec/audiocodecs/mp3.html

2. MPGE AUDIO FRAME HEADER [Електронний ресурс] – Режим доступу: http://mpgedit.org/mpgedit/mpeg_format/mpeghdr.htm

3. ДП «УкрНДНЦ» [Електронний ресурс] – Режим доступу: https://web.archive.org/web/20160306202509/http://www.ukrndnc.org.ua/index.php?id=22926&itemid=194&option=com_terminus&task=view

4. HMM-Based Audio Keyword Generation [Електронний ресурс] – Режим доступу:https://web.archive.org/web/20070510193153/http:/cemnet.ntu.edu.sg/home/asltchia/publication/AudioAnalysisUnderstanding/Conference/HMM-Based%20Audio%20Keyword%20Generation.pdf

5. Design, analysis and experimental evaluation of block based transformation in MFCC computation for speaker recognition [Електронний ресурс] – Режим доступу: https://www.sciencedirect.com/science/article/abs/pii/S0167639311001622

6. Power-Normalized Cepstral Coefficients (PNCC) for Robust Speech Recognition [Електронний ресурс] – Режим доступу: http://www.cs.cmu.edu/~robust/Papers/OnlinePNCC_V25.pdf

7. Stratonovich, R.L. Conditional Markov Processes / Stratonovich, R.L. // Theory of Probability and its Applications – 1960. – No. 5 (2). – pp. 156–178.

8. LeNet-5, convolutional neural networks [Електронний ресурс] – Режим доступу: http://yann.lecun.com/exdb/lenet/

9. Shift-invariant pattern recognition neural network and its optical architecture [Електронний ресурс] – Режим доступу: https://drive.google.com/file/d/0B65v6Wo67Tk5Zm03Tm1kaEdIYkE/view

10. Parallel distributed processing model with local space-invariant interconnections and its optical architecture [Електронний ресурс] – Режим доступу: https://drive.google.com/file/d/0B65v6Wo67Tk5ODRzZmhSR29VeDg/view

11. Subject independent facial expression recognition with robust face detection using a convolutional neural network [Електронний ресурс] – Режим доступу: http://www.iro.umontreal.ca/~pift6080/H09/documents/papers/sparse/matsugo_etal_face_expression_conv_nnet.pdf

12. Deep content-based music recommendation [Електронний ресурс] – Режим доступу: http://papers.nips.cc/paper/5004-deep-content-based-music-recommendation.pdf

13. Convolutional Neural Networks (LeNet) – DeepLearning 0.1 documentation [Електронний ресурс] – Режим доступу: http://deeplearning.net/tutorial/lenet.html

14. Guide to convolutional neural networks : a practical application to traffic-sign detection and classification [Електронний ресурс] – Режим доступу: https://www.worldcat.org/title/guide-to-convolutional-neural-networks-a-practical-application-to-traffic-sign-detection-and-classification/oclc/987790957

15. Flexible, High Performance Convolutional Neural Networks for Image Classification [Електронний ресурс] – Режим доступу: http://people.idsia.ch/~juergen/ijcai2011.pdf

16. ImageNet Classification with Deep Convolutional Neural Networks [Електронний ресурс] – Режим доступу: http://www.image-net.org/challenges/LSVRC/2012/supervision.pdf

17. Multi-column deep neural networks for image classification [Електронний ресурс] – Режим доступу: http://people.idsia.ch/~juergen/cvpr2012.pdf

18. LSTM: A Search Space Odyssey [Електронний ресурс] – Режим доступу: https://arxiv.org/abs/1503.04069

19. The Large Text Compression Benchmark [Електронний ресурс] – Режим доступу: http://www.mattmahoney.net/dc/text.html#1218

20. A Novel Connectionist System for Improved Unconstrained Handwriting Recognition [Електронний ресурс] – Режим доступу: http://people.idsia.ch/~juergen/tpami_2008.pdf

21. Speech Recognition with Deep Recurrent Neural Networks. Acoustics, Speech and Signal Processing (ICASSP) [Електронний ресурс] – Режим доступу: https://ieeexplore.ieee.org/document/6638947

22. With QuickType, Apple wants to do more than guess your next text. It wants to give you an AI [Електронний ресурс] – Режим доступу: https://www.wired.com/2016/06/apple-bringing-ai-revolution-iphone/

23. RECURRENT NEURAL NETWORKS - FEEDBACK NETWORKS - LSTM RECURRENT NETWORK - FEEDBACK NEURAL NETWORK - RECURRENT NETS - FEEDBACK NETWORK - RECURRENT NET - - FEEDBACK NET [Електронний ресурс] – Режим доступу: http://people.idsia.ch/~juergen/rnn.html

24. Learning to Forget: Continual Prediction with LSTM [Електронний ресурс] – Режим доступу: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.55.5709

25. LSTM Recurrent Networks Learn Simple Context Free and Context Sensitive Languages [Електронний ресурс] – Режим доступу: ftp://ftp.idsia.ch/pub/juergen/L-IEEE.pdf

26. Learning precise timing with LSTM recurrent networks [Електронний ресурс] – Режим доступу: http://www.jmlr.org/papers/volume3/gers02a/gers02a.pdf

27. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting [Електронний ресурс] – Режим доступу: https://arxiv.org/abs/1506.04214

28. Барцев С. И Принцип двойственности в организации адаптивных сетей обработки информации / Барцев С. И., Гилев С. Е., Охонин В. А. // Динамика химических и биологических систем. – 1989. – С. 6-55.

29. A mean field view of the landscape of two-layer neural networks. Proceedings of the National Academy of Sciences [Електронний ресурс] – Режим доступу: http://www.pnas.org/content/115/33/E7665.short

30. A stochastic approximation method [Електронний ресурс] – Режим доступу: https://www.jstor.org/stable/2236626

31. Adaptive subgradient methods for online learning and stochastic optimization [Електронний ресурс] – Режим доступу: http://jmlr.org/papers/volume12/duchi11a/duchi11a.pdf

32. Notes on AdaGrad [Електронний ресурс] – Режим доступу: https://web.archive.org/web/20150330033637/http:/seed.ucsd.edu/mediawiki/images/6/6a/Adagrad.pdf

33. rmsprop: Divide the gradient by a running average of its recent magnitude [Електронний ресурс] – Режим доступу: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

34. Improving neural networks by preventing co-adaptation of feature detectors [Електронний ресурс] – Режим доступу: https://arxiv.org/abs/1207.0580

35. Dropout – метод решения проблемы переобучения в нейронных сетях [Електронний ресурс] – Режим доступу: https://habr.com/company/wunderfund/blog/330814/

36. VoxForge Downloads [Електронний ресурс] – Режим доступу: http://www.voxforge.org/home/downloads

37. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks [Електронний ресурс] – Режим доступу: https://www.cs.toronto.edu/~graves/icml_2006.pdf

38. NVidia CUDA [Електронний ресурс] – Режим доступу: https://developer.nvidia.com/cuda-90-download-archive

39. NVidia cuDNN [електронний ресурс] – Режим доступу: https://developer.nvidia.com/rdp/cudnn-download

Додаток А. Лістинг файлів

Файл charmap.py:

```
"""
Defines two dictionaries for converting
between text and integer sequences.
"""

char_map_str = """
' 0
<ПРОБЕЛ> 1
а 2
б 3
в 4
г 5
д 6
е 7
ё 8
ж 9
з 10
и 11
й 12
к 13
л 14
м 15
н 16
о 17
п 18
р 19
```

с 20

т 21

у 22

ф 23

х 24

ц 25

ч 26

ш 27

щ 28

ъ 29

ы 30

ь 31

э 32

ю 33

я 34

"""

the "blank" character is mapped to 28

```
char_map = {}
index_map = {}
for line in char_map_str.strip().split('\n'):
    ch, index = line.split()
    char_map[ch] = int(index)
    index_map[int(index)+1] = ch
index_map[2] = ' '
```

Файл create_desc_json.py:

"""

Use this script to create JSON-Line description files that can be used to train deep-speech models through this library.

This works with data directories that are organized like LibriSpeech:

```
data_directory/group/speaker/[file_id1.wav, file_id2.wav, ...,
                        speaker.trans.txt]

Where speaker.trans.txt has in each line, file_id transcription
"""

from __future__ import absolute_import, division, print_function

import argparse
import json
import os
import wave
import re
import pandas as pd


def main(data_directory, output_file_train, output_file_test, test_pcn=.1):
    labels = []
    durations = []
    keys = []

    for speaker in os.listdir(data_directory):
        speaker_path = os.path.join(data_directory, speaker)
        labels_file = os.path.join(speaker_path, 'etc', 'prompts-original')
        with open(labels_file, 'rb') as labels_f:
            lines = labels_f.read().decode('utf8')
            for line in lines.split('\n'):
                split = line.strip().split()
                if len(split) < 2:
                    continue
```

```
                file_id = split[0]
                label = re.sub(r'[,.;?!:+]', '', ' '.join(split[1:]).lower()).replace(' - ', '
').replace('-', ' ')
                label = label.replace('a', 'a').replace('e', 'e').replace('i', 'и').replace('m',
'м')
                label = label.replace('18', 'восемнадцать').replace('1662', 'тысяча
шестьсот шестьдесят второй')
                label = label.replace('12', 'двенадцать').replace('2x',
'двух').replace('2 x', 'двух').replace('23',
                                                                    'двадцать
три')
                label = label.replace('326 го', 'триста двадцать шестого').replace('23', 'двадцать три')

                def remove_doubles(word):
                    res = ''
                    for c in word:
                        if res == '':
                            res += c
                            continue
                        if res[-1] != c:
                            res += c
                        else:
                            res += '''' + c
                    return res

                new_label = []
                for word in label.split(' '):
                    new_label.append(remove_doubles(word))
                label = ' '.join(new_label)
```

```
                if len(label) > 150:
                    continue
                audio_file = os.path.join(speaker_path, 'wav', file_id) + '.wav'
                audio = wave.open(audio_file)
                duration = float(audio.getnframes()) / audio.getframerate()
                audio.close()
                keys.append(audio_file)
                durations.append(duration)
                labels.append(label)

        test_size = int(test_pcn * len(keys))

        dataset = pd.DataFrame(
            [{'key': key, 'duration': duration, 'text': label} for key, duration, label in
zip(durations, keys, labels)])
        test_ix = dataset.sample(n=test_size).index.values

        del dataset

        with open(output_file_train, 'wb') as out_file:
            for i in range(len(keys)):
                if i not in test_ix:
                    line = json.dumps({
                        'key': keys[i],
                        'duration': durations[i],
                        'text': labels[i],
                    }, ensure_ascii=False)
                    out_file.write((line + '\n').encode('utf8'))
        with open(output_file_test, 'wb') as out_file:
            for i in test_ix:
```

```
            line = json.dumps({
                'key': keys[i],
                'duration': durations[i],
                'text': labels[i],
            }, ensure_ascii=False)
            out_file.write((line + '\n').encode('utf8'))


if __name__ == '__main__':
    # parser = argparse.ArgumentParser()
    # parser.add_argument('data_directory', type=str,
    #                     help='Path to data directory')
    # parser.add_argument('output_file', type=str,
    #                     help='Path to output file')
    # args = parser.parse_args()
    main(os.path.abspath('./Voxforge'),
         os.path.abspath('./train_corpus.json'),
         os.path.abspath('./valid_corpus.json'),
         0.5)
```

Файл data_generator.py

```
"""
Defines a class that is used to featurize audio clips, and provide
them to the network for training or testing.
"""

import json
import numpy as np
import random
from python_speech_features import mfcc
import librosa
```

```
        import scipy.io.wavfile as wav
        import matplotlib.pyplot as plt
        from mpl_toolkits.axes_grid1 import make_axes_locatable

        from utils import calc_feat_dim, spectrogram_from_file, text_to_int_sequence
        from utils import conv_output_length

        RNG_SEED = 123


        class AudioGenerator():
            def __init__(self, step=10, window=20, max_freq=8000, mfcc_dim=13,
                         minibatch_size=20, desc_file=None, spectrogram=True,
max_duration=10.0,
                         sort_by_duration=False):
                """
                Params:
                    step (int): Step size in milliseconds between windows (for spectrogram
ONLY)
                    window (int): FFT window size in milliseconds (for spectrogram
ONLY)
                    max_freq (int): Only FFT bins corresponding to frequencies between
                        [0, max_freq] are returned (for spectrogram ONLY)
                    desc_file (str, optional): Path to a JSON-line file that contains
                        labels and paths to the audio files. If this is None, then
                        load metadata right away
                """

                self.feat_dim = calc_feat_dim(window, max_freq)
                self.mfcc_dim = mfcc_dim
```

```
        self.feats_mean = np.zeros((self.feat_dim,))
        self.feats_std = np.ones((self.feat_dim,))
        self.rng = random.Random(RNG_SEED)
        if desc_file is not None:
            self.load_metadata_from_desc_file(desc_file)
        self.step = step
        self.window = window
        self.max_freq = max_freq
        self.cur_train_index = 0
        self.cur_valid_index = 0
        self.cur_test_index = 0
        self.max_duration = max_duration
        self.minibatch_size = minibatch_size
        self.spectrogram = spectrogram
        self.sort_by_duration = sort_by_duration

    def get_batch(self, partition):
        """ Obtain a batch of train, validation, or test data
        """
        if partition == 'train':
            audio_paths = self.train_audio_paths
            cur_index = self.cur_train_index
            texts = self.train_texts
        elif partition == 'valid':
            audio_paths = self.valid_audio_paths
            cur_index = self.cur_valid_index
            texts = self.valid_texts
        elif partition == 'test':
            audio_paths = self.test_audio_paths
            cur_index = self.test_valid_index
```

```
            texts = self.test_texts
        else:
            raise Exception("Invalid partition. "
                            "Must be train/validation")

        features = [self.normalize(self.featurize(a)) for a in
                    audio_paths[cur_index:cur_index + self.minibatch_size]]

        # calculate necessary sizes
        max_length = max([features[i].shape[0]
                          for i in range(0, self.minibatch_size)])
        max_string_length = max([len(texts[cur_index + i])
                                 for i in range(0, self.minibatch_size)])

        # initialize the arrays
        X_data = np.zeros([self.minibatch_size, max_length,
                           self.feat_dim * self.spectrogram + self.mfcc_dim * (not
self.spectrogram)])
        labels = np.ones([self.minibatch_size, max_string_length]) * 28
        input_length = np.zeros([self.minibatch_size, 1])
        label_length = np.zeros([self.minibatch_size, 1])

        for i in range(0, self.minibatch_size):
            # calculate X_data & input_length
            feat = features[i]
            input_length[i] = feat.shape[0]
            X_data[i, :feat.shape[0], :] = feat

            # calculate labels & label_length
            label = np.array(text_to_int_sequence(texts[cur_index + i]))
```

```
                labels[i, :len(label)] = label
                label_length[i] = len(label)

            # return the arrays
            outputs = {'ctc': np.zeros([self.minibatch_size])}
            inputs = {'the_input': X_data,
                      'the_labels': labels,
                      'input_length': input_length,
                      'label_length': label_length
                      }
            return (inputs, outputs)

        def shuffle_data_by_partition(self, partition):
            """ Shuffle the training or validation data
            """
            if partition == 'train':
                self.train_audio_paths, self.train_durations, self.train_texts = shuffle_data(
                    self.train_audio_paths, self.train_durations, self.train_texts)
            elif partition == 'valid':
                self.valid_audio_paths, self.valid_durations, self.valid_texts = shuffle_data(
                    self.valid_audio_paths, self.valid_durations, self.valid_texts)
            else:
                raise Exception("Invalid partition. "
                                "Must be train/validation")

        def sort_data_by_duration(self, partition):
            """ Sort the training or validation sets by (increasing) duration
            """
```

```
            if partition == 'train':
                self.train_audio_paths, self.train_durations, self.train_texts = sort_data(
                    self.train_audio_paths, self.train_durations, self.train_texts)
            elif partition == 'valid':
                self.valid_audio_paths, self.valid_durations, self.valid_texts = sort_data(
                    self.valid_audio_paths, self.valid_durations, self.valid_texts)
            else:
                raise Exception("Invalid partition. "
                                "Must be train/validation")

        def next_train(self):
            """ Obtain a batch of training data
            """
            while True:
                ret = self.get_batch('train')
                self.cur_train_index += self.minibatch_size
                if self.cur_train_index >= len(self.train_texts) - self.minibatch_size:
                    self.cur_train_index = 0
                    self.shuffle_data_by_partition('train')
                yield ret

        def next_valid(self):
            """ Obtain a batch of validation data
            """
            while True:
                ret = self.get_batch('valid')
                self.cur_valid_index += self.minibatch_size
                #print(ret)
                if self.cur_valid_index >= len(self.valid_texts) - self.minibatch_size:
```

```
            self.cur_valid_index = 0
            self.shuffle_data_by_partition('valid')
        yield ret


def next_test(self):
    """ Obtain a batch of test data
    """
    while True:
        ret = self.get_batch('test')
        self.cur_test_index += self.minibatch_size
        if self.cur_test_index >= len(self.test_texts) - self.minibatch_size:
            self.cur_test_index = 0
        yield ret


def load_train_data(self, desc_file='train_corpus.json'):
    self.load_metadata_from_desc_file(desc_file, 'train')
    self.fit_train()
    if self.sort_by_duration:
        self.sort_data_by_duration('train')


def load_validation_data(self, desc_file='valid_corpus.json'):
    self.load_metadata_from_desc_file(desc_file, 'validation')
    if self.sort_by_duration:
        self.sort_data_by_duration('valid')


def load_test_data(self, desc_file='test_corpus.json'):
    self.load_metadata_from_desc_file(desc_file, 'test')


def load_metadata_from_desc_file(self, desc_file, partition):
    """ Read metadata from a JSON-line file
```

```
    (possibly takes long, depending on the filesize)
Params:
    desc_file (str):  Path to a JSON-line file that contains labels and
        paths to the audio files
    partition (str): One of 'train', 'validation' or 'test'
"""
audio_paths, durations, texts = [], [], []
with open(desc_file, 'rb') as json_line_file_b:
    json_line_file = json_line_file_b.read().decode('utf8')
    for line_num, json_line in enumerate(json_line_file.split('\n')):
        if json_line == '':
            continue
        try:
            spec = json.loads(json_line, encoding='utf-8')
            if float(spec['duration']) > self.max_duration:
                continue
            audio_paths.append(spec['key'])
            durations.append(float(spec['duration']))
            texts.append(spec['text'])
        except Exception as e:
            # Change to (KeyError, ValueError) or
            # (KeyError,json.decoder.JSONDecodeError), depending on
            # json module version
            print('Error reading line #{}: {}'
                  .format(line_num, json_line))
if partition == 'train':
    self.train_audio_paths = audio_paths
    self.train_durations = durations
    self.train_texts = texts
elif partition == 'validation':
```

```
            self.valid_audio_paths = audio_paths
            self.valid_durations = durations
            self.valid_texts = texts
        elif partition == 'test':
            self.test_audio_paths = audio_paths
            self.test_durations = durations
            self.test_texts = texts
        else:
            raise Exception("Invalid partition to load metadata. "
                            "Must be train/validation/test")

    def fit_train(self, k_samples=100):
        """ Estimate the mean and std of the features from the training set
        Params:
            k_samples (int): Use this number of samples for estimation
        """
        k_samples = min(k_samples, len(self.train_audio_paths))
        samples = self.rng.sample(self.train_audio_paths, k_samples)
        feats = [self.featurize(s) for s in samples]
        feats = np.vstack(feats)
        self.feats_mean = np.mean(feats, axis=0)
        self.feats_std = np.std(feats, axis=0)

    def featurize(self, audio_clip):
        """ For a given audio clip, calculate the corresponding feature
        Params:
            audio_clip (str): Path to the audio clip
        """
        if self.spectrogram:
            return spectrogram_from_file(
```

```
                audio_clip, step=self.step, window=self.window,
                max_freq=self.max_freq)
        else:
            (rate, sig) = wav.read(audio_clip)
            return mfcc(sig, rate, numcep=self.mfcc_dim)

    def normalize(self, feature, eps=1e-14):
        """ Center a feature using the mean and std
        Params:
            feature (numpy.ndarray): Feature to normalize
        """
        return (feature - self.feats_mean) / (self.feats_std + eps)


def shuffle_data(audio_paths, durations, texts):
    """ Shuffle the data (called after making a complete pass through
        training or validation data during the training process)
    Params:
        audio_paths (list): Paths to audio clips
        durations (list): Durations of utterances for each audio clip
        texts (list): Sentences uttered in each audio clip
    """
    p = np.random.permutation(len(audio_paths))
    audio_paths = [audio_paths[i] for i in p]
    durations = [durations[i] for i in p]
    texts = [texts[i] for i in p]
    return audio_paths, durations, texts


def sort_data(audio_paths, durations, texts):
```

```
            """ Sort the data by duration
            Params:
                audio_paths (list): Paths to audio clips
                durations (list): Durations of utterances for each audio clip
                texts (list): Sentences uttered in each audio clip
            """
            p = np.argsort(durations).tolist()
            audio_paths = [audio_paths[i] for i in p]
            durations = [durations[i] for i in p]
            texts = [texts[i] for i in p]
            return audio_paths, durations, texts



        def vis_train_features(index=0):
            """ Visualizing the data point in the training set at the supplied index
            """
            # obtain spectrogram
            audio_gen = AudioGenerator(spectrogram=True)
            audio_gen.load_train_data()
            vis_audio_path = audio_gen.train_audio_paths[index]
            vis_spectrogram_feature =
audio_gen.normalize(audio_gen.featurize(vis_audio_path))
            # obtain mfcc
            audio_gen = AudioGenerator(spectrogram=False)
            audio_gen.load_train_data()
            vis_mfcc_feature =
audio_gen.normalize(audio_gen.featurize(vis_audio_path))
            # obtain text label
            vis_text = audio_gen.train_texts[index]
            # obtain raw audio
```

```
        vis_raw_audio, _ = librosa.load(vis_audio_path)
        # print total number of training examples
        print('There are %d total training examples.' % len(audio_gen.train_audio_paths))
        # return labels for plotting
        return vis_text, vis_raw_audio, vis_mfcc_feature, vis_spectrogram_feature, vis_audio_path


    def plot_raw_audio(vis_raw_audio):
        # plot the raw audio signal
        fig = plt.figure(figsize=(12, 3))
        ax = fig.add_subplot(111)
        steps = len(vis_raw_audio)
        ax.plot(np.linspace(1, steps, steps), vis_raw_audio)
        plt.title('Audio Signal')
        plt.xlabel('Time')
        plt.ylabel('Amplitude')
        plt.show()


    def plot_mfcc_feature(vis_mfcc_feature):
        # plot the MFCC feature
        fig = plt.figure(figsize=(12, 5))
        ax = fig.add_subplot(111)
        im = ax.imshow(vis_mfcc_feature, cmap=plt.cm.jet, aspect='auto')
        plt.title('Normalized MFCC')
        plt.ylabel('Time')
        plt.xlabel('MFCC Coefficient')
        divider = make_axes_locatable(ax)
```

```
    cax = divider.append_axes("right", size="5%", pad=0.05)
    plt.colorbar(im, cax=cax)
    ax.set_xticks(np.arange(0, 13, 2), minor=False);
    plt.show()


def plot_spectrogram_feature(vis_spectrogram_feature):
    # plot the normalized spectrogram
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_subplot(111)
    im = ax.imshow(vis_spectrogram_feature, cmap=plt.cm.jet, aspect='auto')
    plt.title('Normalized Spectrogram')
    plt.ylabel('Time')
    plt.xlabel('Frequency')
    divider = make_axes_locatable(ax)
    cax = divider.append_axes("right", size="5%", pad=0.05)
    plt.colorbar(im, cax=cax)
    plt.show()
```

Файл requirements.txt:

```
h5py==2.7.0
ipykernel==4.6.1
jupyter==1.0.0
keras==2.0.5
librosa==0.5.1
opencv-python==3.2.0.7
python-speech-features==0.5
seaborn==0.7.1
soundfile==0.9.0.post1
```

Файл train_utils.py:

```
"""
```

```
Defines a functions for training a NN.
"""

from data_generator import AudioGenerator
import _pickle as pickle

from keras import backend as K
from keras.models import Model
from keras.layers import (Input, Lambda, BatchNormalization)
from keras.optimizers import SGD, RMSprop
from keras.callbacks import ModelCheckpoint
import os

import tensorflow as tf

from tensorflow.python.ops import math_ops, array_ops


def ctc_lambda_func(args):

    y_pred, labels, input_length, label_length = args

    #return K.ctc_batch_cost(labels, y_pred, input_length, label_length)

    # test

    label_length = math_ops.to_int32(array_ops.squeeze(label_length, axis=-1))
    input_length = math_ops.to_int32(array_ops.squeeze(input_length, axis=-1))
    sparse_labels = math_ops.to_int32(
        K.ctc_label_dense_to_sparse(labels, label_length))
```

```
        y_pred = math_ops.log(array_ops.transpose(y_pred, perm=[1, 0, 2]) +
K.epsilon())

        return array_ops.expand_dims(
            tf.nn.ctc_loss(
                inputs=y_pred, labels=sparse_labels, sequence_length=input_length,
ignore_longer_outputs_than_inputs=True),
            1)


    def add_ctc_loss(input_to_softmax):
        the_labels = Input(name='the_labels', shape=(None,), dtype='float32')
        input_lengths = Input(name='input_length', shape=(1,), dtype='int64')
        label_lengths = Input(name='label_length', shape=(1,), dtype='int64')
        output_lengths = Lambda(input_to_softmax.output_length)(input_lengths)
        # output_length = BatchNormalization()(input_lengths)
        # CTC loss is implemented in a lambda layer
        loss_out = Lambda(ctc_lambda_func, output_shape=(1,), name='ctc')(
            [input_to_softmax.output, the_labels, output_lengths, label_lengths])
        model = Model(
            inputs=[input_to_softmax.input, the_labels, input_lengths, label_lengths],
            outputs=loss_out)
        return model


    def train_model(input_to_softmax,
                    pickle_path,
                    save_model_path,
                    train_json='train_corpus.json',
```

```
                valid_json='valid_corpus.json',
                minibatch_size=20,
                spectrogram=True,
                mfcc_dim=13,
                optimizer=SGD(lr=0.02, decay=1e-6, momentum=0.9,
nesterov=True, clipnorm=5),
                epochs=20,
                verbose=1,
                sort_by_duration=False,
                max_duration=10.0):
        # create a class instance for obtaining batches of data
        audio_gen = AudioGenerator(minibatch_size=minibatch_size,
                                   spectrogram=spectrogram, mfcc_dim=mfcc_dim,
max_duration=max_duration,
                                   sort_by_duration=sort_by_duration)
        # add the training data to the generator
        audio_gen.load_train_data(train_json)
        audio_gen.load_validation_data(valid_json)
        # calculate steps_per_epoch
        num_train_examples = len(audio_gen.train_audio_paths)
        steps_per_epoch = num_train_examples // minibatch_size
        # calculate validation_steps
        num_valid_samples = len(audio_gen.valid_audio_paths)
        validation_steps = num_valid_samples // minibatch_size

        # add CTC loss to the NN specified in input_to_softmax
        model = add_ctc_loss(input_to_softmax)
        # ctc_loss = tf.reduce_mean(tf.nn.ctc_loss(labels, logits, final_seq_lens))
        # CTC loss is implemented elsewhere, so use a dummy lambda function for
the loss
```

```
    model.compile(loss={'ctc': lambda y_true, y_pred: y_pred}, optimizer=optimizer)

    # make results/ directory, if necessary
    if not os.path.exists('results'):
        os.makedirs('results')#

    # add checkpointer
    checkpointer = ModelCheckpoint(filepath='results/' + save_model_path, verbose=0)

    # train the model

    hist = model.fit_generator(generator=audio_gen.next_train(), steps_per_epoch=steps_per_epoch,
                               epochs=epochs, validation_data=audio_gen.next_valid(), validation_steps=validation_steps,
                               callbacks=[checkpointer], verbose=verbose)

    # save model loss
    with open('results/' + pickle_path, 'wb') as f:
        pickle.dump(hist.history, f)
```

Файл utils.py:

```
"""
Defines various functions for processing the data.
"""
import numpy as np
import soundfile
from numpy.lib.stride_tricks import as_strided
from char_map import char_map, index_map
```

```
def calc_feat_dim(window, max_freq):
    return int(0.001 * window * max_freq) + 1

def conv_output_length(input_length, filter_size, border_mode, stride,
                       dilation=1):
    """ Compute the length of the output sequence after 1D convolution along
        time. Note that this function is in line with the function used in
        Convolution1D class from Keras.
    Params:
        input_length (int): Length of the input sequence.
        filter_size (int): Width of the convolution kernel.
        border_mode (str): Only support `same` or `valid`.
        stride (int): Stride size used in 1D convolution.
        dilation (int)
    """
    if input_length is None:
        return None
    assert border_mode in {'same', 'valid'}
    dilated_filter_size = filter_size + (filter_size - 1) * (dilation - 1)
    if border_mode == 'same':
        output_length = input_length
    elif border_mode == 'valid':
        output_length = input_length - dilated_filter_size + 1
    return (output_length + stride - 1) // stride


def spectrogram(samples, fft_length=256, sample_rate=2, hop_length=128):
    """
    Compute the spectrogram for a real signal.
```

```
The parameters follow the naming convention of
matplotlib.mlab.specgram

Args:
   samples (1D array): input audio signal
   fft_length (int): number of elements in fft window
   sample_rate (scalar): sample rate
   hop_length (int): hop length (relative offset between neighboring
      fft windows).

Returns:
   x (2D array): spectrogram [frequency x time]
   freq (1D array): frequency of each row in x

Note:
   This is a truncating computation e.g. if fft_length=10,
   hop_length=5 and the signal has 23 elements, then the
   last 3 elements will be truncated.
"""
assert not np.iscomplexobj(samples), "Must not pass in complex numbers"

window = np.hanning(fft_length)[:, None]
window_norm = np.sum(window**2)

# The scaling below follows the convention of
# matplotlib.mlab.specgram which is the same as
# matlabs specgram.
scale = window_norm * sample_rate

trunc = (len(samples) - fft_length) % hop_length
```

```
    x = samples[:len(samples) - trunc]

    # "stride trick" reshape to include overlap
    nshape = (fft_length, (len(x) - fft_length) // hop_length + 1)
    nstrides = (x.strides[0], x.strides[0] * hop_length)
    x = as_strided(x, shape=nshape, strides=nstrides)

    # window stride sanity check
    assert np.all(x[:, 1] == samples[hop_length:(hop_length + fft_length)])

    # broadcast window, compute fft over columns and square mod
    x = np.fft.rfft(x * window, axis=0)
    x = np.absolute(x)**2

    # scale, 2.0 for everything except dc and fft_length/2
    x[1:-1, :] *= (2.0 / scale)
    x[(0, -1), :] /= scale

    freqs = float(sample_rate) / fft_length * np.arange(x.shape[0])

    return x, freqs



def spectrogram_from_file(filename, step=10, window=20, max_freq=None,
                          eps=1e-14):
    """ Calculate the log of linear spectrogram from FFT energy
    Params:
        filename (str): Path to the audio file
        step (int): Step size in milliseconds between windows
        window (int): FFT window size in milliseconds
```

```
        max_freq (int): Only FFT bins corresponding to frequencies between
            [0, max_freq] are returned
        eps (float): Small value to ensure numerical stability (for ln(x))
    """
    with soundfile.SoundFile(filename) as sound_file:
        audio = sound_file.read(dtype='float32')
        sample_rate = sound_file.samplerate
        if audio.ndim >= 2:
            audio = np.mean(audio, 1)
        if max_freq is None:
            max_freq = sample_rate / 2
        if max_freq > sample_rate / 2:
            raise ValueError("max_freq must not be greater than half of "
                             " sample rate")
        if step > window:
            raise ValueError("step size must not be greater than window size")
        hop_length = int(0.001 * step * sample_rate)
        fft_length = int(0.001 * window * sample_rate)
        pxx, freqs = spectrogram(
            audio, fft_length=fft_length, sample_rate=sample_rate,
            hop_length=hop_length)
        ind = np.where(freqs <= max_freq)[0][-1] + 1
    return np.transpose(np.log(pxx[:ind, :] + eps))


def text_to_int_sequence(text):
    """ Convert text to an integer sequence """
    int_sequence = []
    for c in text:
        if c == ' ':
            ch = char_map['<ПРОБЕЛ>']
```

```
            else:
                ch = char_map[c]
            int_sequence.append(ch)
        return int_sequence


    def int_sequence_to_text(int_sequence):
        """ Convert an integer sequence to text """
        text = []
        for c in int_sequence:
            ch = index_map[c]
            text.append(ch)
        return text
```

Файл vui_notebook.ipynb:

```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 6,
   "metadata": {
    "collapsed": false
   },
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "The autoreload extension is already loaded. To reload it, use:\n
%reload_ext autoreload\n"
```

```
            ]
           }
          ],
          "source": [

"#####################################################################
#\n",
          "# RUN THIS CODE CELL IF YOU ARE RESUMING THE NOTEBOOK
AFTER A BREAK #\n",

"#####################################################################
#\n",
          "\n",
          "# allocate 50% of GPU memory (if you like, feel free to change this)\n",
          "from keras.backend.tensorflow_backend import set_session, get_session\n",
          "import tensorflow as tf\n",
          "\n",
          "config = tf.ConfigProto()\n",
          "#config.gpu_options.per_process_gpu_memory_fraction = 0.95\n",
          "#config.log_device_placement = True\n",
          "set_session(tf.Session(config=config))\n",
          "\n",
          "# watch for any changes in the sample_models module, and reload it
automatically\n",
          "% load_ext autoreload\n",
          "% autoreload 2\n",
          "# import NN architectures fo\n",
          "# r speech recognition\n",
          "from sample_models import *\n",
          "# import function for training acoustic model\n",
```

```
 "from train_utils import train_model\n"
 ]
},
{
 "cell_type": "code",
 "execution_count": 7,
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [

"_________________________________________________________________\nL
ayer (type)                 Output Shape              Param #
\n=================================================================
======\nthe_input (InputLayer)        (None, None, 13)          0
\n_________________________________________________________________\n
conv1d (Conv1D)              (None, None, 200)         28800
\n_________________________________________________________________\n
bn_conv_1d (BatchNormalizati (None, None, 200)         800
\n_________________________________________________________________\nr
nn (GRU)                     (None, None, 200)         240600
\n_________________________________________________________________\n
bn_rnn_1d (BatchNormalizatio (None, None, 200)         800
\n_________________________________________________________________\nt
ime_distributed_2 (TimeDist (None, None, 36)          7236
\n_________________________________________________________________\n
softmax (Activation)         (None, None, 36)          0
\n=================================================================
```

======\nTotal params: 278,236\nTrainable params: 277,436\nNon-trainable params:
800\n_____
_\nNone\n"
]
}
],
"source": [
"test = cnn_rnn_model(input_dim=13, filters=200, kernel_size=11, conv_stride=2,\n",
" " conv_border_mode='valid', units=200, output_dim=36)"
]
},
{
"cell_type": "code",
"execution_count": 8,
"metadata": {
"collapsed": false
},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"Epoch 1/20\n"
]
},
{
"name": "stdout",
"output_type": "stream",

"text": [

"\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\
b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\r17/17
[==============================] - 82s 5s/step - loss: 406.1687 - val_loss:
301.1680\n"
]
},
{
"name": "stdout",
"output_type": "stream",
"text": [
"Epoch 2/20\n"
]
},
{
"name": "stdout",
"output_type": "stream",
"text": [

"\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\
b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\r17/17
[==============================] - 66s 4s/step - loss: 257.5253 - val_loss:
253.3529\n"
]
},
{
"name": "stdout",
"output_type": "stream",
"text": [

"Epoch 3/20\n"
]
},
{
"name": "stdout",
"output_type": "stream",
"text": [

"\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\
b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\r17/17
[==============================] - 66s 4s/step - loss: 241.0051 - val_loss:
243.9929\n"
]
},
{
"name": "stdout",
"output_type": "stream",
"text": [
"Epoch 4/20\n"
]
},
{
"name": "stdout",
"output_type": "stream",
"text": [

"\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\
b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\r17/17
[==============================] - 66s 4s/step - loss: 235.5289 - val_loss:
239.6616\n"

]
},
{
"name": "stdout",
"output_type": "stream",
"text": [
"Epoch 5/20\n"
]
},
{
"name": "stdout",
"output_type": "stream",
"text": [

"\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\
b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\r17/17
[==============================] - 67s 4s/step - loss: 230.9776 - val_loss:
237.4719\n"
]
},
{
"name": "stdout",
"output_type": "stream",
"text": [
"Epoch 6/20\n"
]
},
{
"name": "stdout",
"output_type": "stream",

"text": [

"\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\
b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\r17/17
[==============================] - 67s 4s/step - loss: 226.8700 - val_loss:
236.9606\n"
]
},
{
"name": "stdout",
"output_type": "stream",
"text": [
"Epoch 7/20\n"
]
},
{
"name": "stdout",
"output_type": "stream",
"text": [

"\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\
b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\r17/17
[==============================] - 67s 4s/step - loss: 222.1577 - val_loss:
249.1466\n"
]
},
{
"name": "stdout",
"output_type": "stream",
"text": [

"Epoch 8/20\n"
]
},
{
"name": "stdout",
"output_type": "stream",
"text": [

"\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\
b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\r17/17
[==============================] - 68s 4s/step - loss: 214.3618 - val_loss: 283.3436\n"
]
},
{
"name": "stdout",
"output_type": "stream",
"text": [
"Epoch 9/20\n"
]
},
{
"name": "stdout",
"output_type": "stream",
"text": [

"\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\
b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\r17/17
[==============================] - 68s 4s/step - loss: 208.8090 - val_loss: 212.9035\n"

]
},
{
"name": "stdout",
"output_type": "stream",
"text": [
"Epoch 10/20\n"
]
},
{
"name": "stdout",
"output_type": "stream",
"text": [

"\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\
b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\r17/17
[==============================] - 68s 4s/step - loss: 202.3790 - val_loss:
203.7667\n"
]
},
{
"name": "stdout",
"output_type": "stream",
"text": [
"Epoch 11/20\n"
]
},
{
"name": "stdout",
"output_type": "stream",

"text": [
"\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\
b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\r17/17
[==============================] - 67s 4s/step - loss: 197.6383 - val_loss:
195.2024\n"
]
},
{
"name": "stdout",
"output_type": "stream",
"text": [
"Epoch 12/20\n"
]
},
{
"name": "stdout",
"output_type": "stream",
"text": [
"\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\
b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\r17/17
[==============================] - 67s 4s/step - loss: 191.1561 - val_loss:
190.6448\n"
]
},
{
"name": "stdout",
"output_type": "stream",
"text": [

"Epoch 13/20\n"
]
},{
"name": "stdout",
"output_type": "stream",
"text": [

"\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\
b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\r17/17
[==============================] - 67s 4s/step - loss: 183.1598 - val_loss: 186.7194\n"
]
},
{
"name": "stdout",
"output_type": "stream",
"text": [
"Epoch 14/20\n"
]
},{
"name": "stdout",
"output_type": "stream",
"text": [

"\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\
b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\r17/17
[==============================] - 67s 4s/step - loss: 174.8511 - val_loss: 179.9593\n"
]
},

{
"name": "stdout",
"output_type": "stream",
"text": [
"Epoch 15/20\n"
]
},{
"name": "stdout",
"output_type": "stream",
"text": [

"\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\
b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\r17/17
[==============================] - 67s 4s/step - loss: 166.7852 - val_loss: 168.4928\n"
]
},
{
"name": "stdout",
"output_type": "stream",
"text": [
"Epoch 16/20\n"
]
},{
"name": "stdout",
"output_type": "stream",
"text": [

"\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\
b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\r17/17

[==============================] - 66s 4s/step - loss: 161.1418 - val_loss: 182.6162\n"
]
},
{
"name": "stdout",
"output_type": "stream",
"text": [
"Epoch 17/20\n"
]
},{
"name": "stdout",
"output_type": "stream",
"text": [

"\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\
b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\r17/17
[==============================] - 66s 4s/step - loss: 155.3598 - val_loss: 164.6843\n"
]
},
{
"name": "stdout",
"output_type": "stream",
"text": [
"Epoch 18/20\n"
]
},{
"name": "stdout",
"output_type": "stream",

"text": [

"\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\
b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\r17/17
[==============================] - 67s 4s/step - loss: 149.4317 - val_loss:
153.5656\n"
]
},
{
"name": "stdout",
"output_type": "stream",
"text": [
"Epoch 19/20\n"
]
},{
"name": "stdout",
"output_type": "stream",
"text": [

"\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\
b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\r17/17
[==============================] - 67s 4s/step - loss: 143.0694 - val_loss:
151.3519\n"
]
},
{
"name": "stdout",
"output_type": "stream",
"text": [
"Epoch 20/20\n"

]
},{
"name": "stdout",
"output_type": "stream",
"text": [

"\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\
b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\r17/17
[==============================] - 66s 4s/step - loss: 138.1250 - val_loss:
142.6263\n"
]
}
],
"source": [
"from keras.optimizers import Adam\n",
"\n",
"train_model(input_to_softmax=test,\n",
"   pickle_path='model_test.pickle',\n",
"   save_model_path='model_test.h5',\n",
"   epochs=20,\n",
"   minibatch_size=200,\n",
"   spectrogram=False,\n",
"   optimizer=Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-
8),\n",
"   max_duration=10)\n"
]
},
{
"cell_type": "code",
"execution_count": 14,

```
"metadata": {
 "collapsed": true
},
"outputs": [
 {
  "name": "stdout",
  "output_type": "stream",
  "text": [

"_________________________________________________________________\nL
ayer (type)                 Output Shape              Param #
\n=================================================================
======\nthe_input (InputLayer)         (None, None, 13)          0
\n_________________________________________________________________\nl
ayer_1_conv (Conv1D)         (None, None, 200)         28800
\n_________________________________________________________________\n
conv_batch_norm (BatchNormal (None, None, 200)         800
\n_________________________________________________________________\n
bidirectional_1 (Bidirection (None, None, 300)         421200
\n_________________________________________________________________\n
bt_lstm_0 (BatchNormalizatio (None, None, 300)         1200
\n_________________________________________________________________\n
bidirectional_2 (Bidirection (None, None, 300)         541200
\n_________________________________________________________________\n
bt_lstm_1 (BatchNormalizatio (None, None, 300)         1200
\n_________________________________________________________________\n
bidirectional_3 (Bidirection (None, None, 300)         541200
\n_________________________________________________________________\n
bt_lstm_2 (BatchNormalizatio (None, None, 300)         1200
\n_________________________________________________________________\n
```

bidirectional_4 (Bidirection (None, None, 300) 541200
\n_____\n
bt_final_lstm_layer (BatchNo (None, None, 300) 1200
\n_____\nt
ime_distributed_2 (TimeDist (None, None, 36) 10836
\n_____\n
softmax (Activation) (None, None, 36) 0
\n==========
======\nTotal params: 2,090,036\nTrainable params: 2,087,236\nNon-trainable
params:
2,800\n_____
___\nNone\n"
]
}
],
"source": [
"me = saloedov(filters=200,\n",
" kernel_size=11,\n",
" conv_stride=2,\n",
" dropout_rate=.1,\n",
" conv_border_mode='valid',\n",
" units=150,\n",
" lstm_layers=4,)\n"
]
},
"source": [
"from keras.optimizers import Adam\n",
"\n",
"train_model(input_to_softmax=me,\n",
" pickle_path='model_my.pickle',\n",

```
"            save_model_path='model_my.h5',\n",
"            epochs=5,\n",
"            minibatch_size=2,\n",
"            spectrogram=False,\n",
"            #optimizer=Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-8),\n",
"            max_duration=10)\n"
]
},
{
"cell_type": "markdown",
"metadata": {},
"source": [
"Please execute the code cell below to train the neural network you specified in `input_to_softmax`.  After the model has finished training, the model is [saved](https://keras.io/getting-started/faq/#how-can-i-save-a-keras-model) in the HDF5 file `model_end.h5`. The loss history is [saved](https://wiki.python.org/moin/UsingPickle) in `model_end.pickle`.  You are welcome to tweak any of the optional parameters while calling the `train_model` function, but this is not required."
]
},
{
"cell_type": "heading",
"metadata": {
"collapsed": true
},
"level": 6,
"source": [
"#model_end = model_end(input_dim=13,\n",
```

```
            "#                          filters=200,\n",
            "#                          kernel_size=11,\n",
            "#                          conv_stride=2,\n",
            "#                          conv_border_mode='valid',\n",
            "#                          units=250,\n",
            "#                          activation='relu',\n",
            "#                          cell=GRU,\n",
            "#                          dropout_rate=1,\n",
            "#                          number_of_layers=2,\n",
            "#                          output_dim=36)\n",
            "\n",
            "train_model(input_to_softmax=model_4,\n",
            "           pickle_path='model_4.pickle',\n",
            "           save_model_path='model_4.h5',\n",
            "           minibatch_size=100,\n",
            "           spectrogram=False,\n",
            "           max_duration=1)  # change to False if you would like to use MFCC
features\n"
          ]
         },
         {
          "cell_type": "heading",
          "metadata": {
           "collapsed": true
          },
          "level": 6,
          "source": [
           "from keras.optimizers import Adam\n",
           "\n",
           "train_model(input_to_softmax=model_end,\n",
```

```
            "          pickle_path='model_end.pickle',\n",
            "          save_model_path='model_end.h5',\n",
            "          train_json='train_corpus.json',\n",
            "          valid_json='valid_corpus.json',\n",
            "          minibatch_size=10,\n",
            "          #optimizer=Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-
8),\n",
            "          spectrogram=False,\n",
            "          max_duration=1)\n"
           ]
          },
          {
           "cell_type": "markdown",
           "metadata": {},
           "source": [
            "<a id='step3'></a>\n",
            "## STEP 3: Obtain Predictions\n",
            "\n",
            "We have written a function for you to decode the predictions of your
acoustic model.  To use the function, please execute the code cell below."
           ]
          },
          {
           "cell_type": "code",
           "execution_count": 4,
           "metadata": {
            "collapsed": true
           },
           "outputs": [],
           "source": [
```

```
"import numpy as np\n",
"from data_generator import AudioGenerator\n",
"from keras import backend as K\n",
"from utils import int_sequence_to_text\n",
"from IPython.display import Audio\n",
"\n",
"def get_predictions(index, partition, input_to_softmax, model_path):\n",
"    \"\"\" Print a model's decoded predictions\n",
"    Params:\n",
"        index (int): The example you would like to visualize\n",
"        partition (str): One of 'train' or 'validation'\n",
"        input_to_softmax (Model): The acoustic model\n",
"        model_path (str): Path to saved acoustic model's weights\n",
"    \"\"\"\n",
"    # load the train and test data\n",
"    data_gen = AudioGenerator(spectrogram=False)\n",
"    data_gen.load_train_data()\n",
"    data_gen.load_validation_data()\n",
"    \n",
"    # obtain the true transcription and the audio features \n",
"    if partition == 'validation':\n",
"        transcr = data_gen.valid_texts[index]\n",
"        audio_path = data_gen.valid_audio_paths[index]\n",
"        data_point = data_gen.normalize(data_gen.featurize(audio_path))\n",
"    elif partition == 'train':\n",
"        transcr = data_gen.train_texts[index]\n",
"        audio_path = data_gen.train_audio_paths[index]\n",
"        data_point = data_gen.normalize(data_gen.featurize(audio_path))\n",
"    else:\n",
```

```
"                raise Exception('Invalid partition!  Must be \"train\" or \"validation\"')\n",
"      \n",
"    # obtain and decode the acoustic model's predictions\n",
"    input_to_softmax.load_weights(model_path)\n",
"        prediction = input_to_softmax.predict(np.expand_dims(data_point, axis=0))\n",
"     output_length = [input_to_softmax.output_length(data_point.shape[0])] \n",
"    pred_ints = (K.eval(K.ctc_decode(\n",
"                                                prediction, output_length, greedy=False)[0][0])+1).flatten().tolist()\n",
"    \n",
"    # play the audio file, and display the true and predicted transcriptions\n",
"    print('-'*80)\n",
"    Audio(audio_path)\n",
"    print('True transcription:\\n' + '\\n' + transcr)\n",
"    print('-'*80)\n",
"                        print('Predicted transcription:\\n' + '\\n' + ".join(int_sequence_to_text(pred_ints)))\n",
"    print('-'*80)"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "Use the code cell below to obtain the transcription predicted by your final model for the first example in the training dataset."
   ]
```

```
},
{
 "cell_type": "code",
 "execution_count": 5,
 "metadata": {
  "collapsed": true
 },
 "source": [
  "for i in range(10):\n",
  "    get_predictions(index=i, \n",
  "                partition='train',\n",
  "                input_to_softmax=test, \n",
  "                model_path='results/model_test.h5')"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "Use the next code cell to visualize the model's prediction for the first example in the validation dataset."
 ]
},
{
 "cell_type": "code",
 "execution_count": 12,
 "metadata": {
  "collapsed": true
 },
 "source": [
```

```
   "for i in range(1650, 1700):\n",
   "    get_predictions(index=i, \n",
   "                    partition='validation',\n",
   "                    input_to_softmax=test, \n",
   "                    model_path='results/model_test.h5')"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "One standard way to improve the results of the decoder is to incorporate a language model.  We won't pursue this in the notebook, but you are welcome to do so as an _optional extension_. \n",
   "\n",
   "If you are interested in creating models that provide improved transcriptions, you are encouraged to download [more data](http://www.openslr.org/12/) and train bigger, deeper models.  But beware - the model will likely take a long while to train.  For instance, training this [state-of-the-art](https://arxiv.org/pdf/1512.02595v1.pdf) model would take 3-6 weeks on a single GPU!"
  ]
 }
],
"metadata": {
 "anaconda-cloud": {},
 "kernelspec": {
  "display_name": "Python 3",
  "language": "python",
  "name": "python3"
```

```
 },
 "language_info": {
  "codemirror_mode": {
   "name": "ipython",
   "version": 3
  },
  "file_extension": ".py",
  "mimetype": "text/x-python",
  "name": "python",
  "nbconvert_exporter": "python",
  "pygments_lexer": "ipython3",
  "version": "3.5.3"
 }
},
"nbformat": 4,
"nbformat_minor": 2
}
```