НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації і управління

«До захисту допущено»
В.о. завідувача кафедри
_____ __*О.А.Павлов*__
(підпис) (ініціали, прізвище)
“___” _____ 2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки *6.050103 «Програмна інженерія»*

спеціальність *«Програмне забезпечення систем»*

на тему: *Програмний комплекс для розпізнавання промовленого тексту з відео по міміці людини*

Виконав: студент 4 курсу, групи ІП-52

Онбиш Олександр Олегович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *доцент, к.ф-м.н., доцент Гавриленко О.В* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант з графічної документації *доц. к.т.н. Ліщук К.І.* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент *доц. каф. ТК, к.т.н., доц. Пасько В.П.* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЕКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проект	2	
2	А4	КПІ.ІП-ХХХХ. ХХХХХХ.02.81	«Програмний комплекс для розпізнавання промовленого тексту по відео міміці людини»		
			Пояснювальна записка	62	
3	А4	КПІ.ІП-ХХХХ. ХХХХХХ.03.81	«Програмний комплекс для розпізнавання промовленого тексту по відео міміці людини».		
			Технічне завдання	10	
4	А4	КПІ.ІП-ХХХХ. ХХХХХХ.04.51	«Програмний комплекс для розпізнавання промовленого тексту по відео міміці людини»		
			Програма та методика тестування	14	

АНОТАЦІЯ

У даній бакалаврській роботі розглянуті питання розпізнавання промовленого тексту за відеорядом міміки людини за допомогою методів глибокого навчання. Досліджені різні способи розпізнавання промовленого тексту. Розроблений спосіб розпізнавання на базі згорткової нейронної мережі з рекурентною мережою.

Розроблена програма для розпізнавання промовленого тексту за відеорядом обличчя людини, досліджено роботу моделі на різних наборах даних.

Загальний обсяг роботи: 62 сторінки, 23 рисунків, 8 таблиць, 23 джерел.

Ключові слова: обробка зображень, комп'ютерний зір, машинне навчання, розпізнавання промовленого тексту.

ABSTRACT

In this work, the problem of recognition of spoken text by the video of human facial expressions through the methods of deep learning are studied. Different ways of recognition of spoken text have been investigated. The method of recognition on the basis of a convolutional neural network with a recurrent network is developed.

The program for recognition of the spoken text on the face of the person is developed, the model work is investigated on different data sets.

Total volume of work: 62 pages, 23 figures, 8 tables, 23 sources.

Keywords: image processing, computer vision, machine learning, recognition of spoken text.

Пояснювальна записка
до дипломного проекту

на тему: *Програмний комплекс для розпізнавання промовленого тексту з відео по міміці людини*

Київ – 2019 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Back propagation — алгоритм зворотнього поширення помилки.

CNN (convolutional neural network) - згорткова неи□ронна мережа.

ASR - Automatic speech recogintion – автоматичне розпізнавання мови.

image features – декстриптори зображень.

video features – декскриптор зображення.

optical flow – є зразком видимого руху об'єктів, поверхонь і країв на візуальній сцені, викликаних відносним рухом між спостерігачем і сценою.

CTC loss - є виходом нейронної мережі і пов'язаної з нею функцією оцінки, для навчання рекуррентних нейронних мереж.

ВСТУП

Читання по губам відіграє вирішальну роль у людському спілкуванні і розумінні мови, цеи□ факт підтверджується багатьма дослідженнями, одне з яких є дослідження МакГурка (McGurk & MacDonald, 1976). У своі□и□ роботі МакГурк вивчав ефект коли поверх відео с обличчям людини яка промовляє певні фонеми накладається звуковии□ ряд на якому звучать зовсім інші фонеми, таким чином людина сприи□має третіи□ вид фонеми яка відрізняється від тоі□ яка буда на відеоряді та аудіозаписі. Це дослідження підтверджує що людина сприи□має не лише звукові сигнали але и□ співставляє і□х с візуальною інформацією яка включає в себе міміку людини та і□і□ артикуляцію.

Читання по губам є важкою задачею для людеи□, особливо у відсутності контексту. Більшість положень облицця, губ, зубів та іноді язика є латентними таким чином утрорюючи складність для розпізнавання без додаткового контексту [1]. Наприклад Фішер (1968) дає 5 категоріи□ візуальних фонем (візем), зі списку 23 базових фонем, які зазвичаи□ плутаються людьми коли вони спостерігають за артикуляюею розмовника. Таким чином люди погано справляються з задачею чинання по губам.

Люди з вадами слуху досягають лише 17 ± 11% точності для обмедженноі□ підмножини з 30 односкладових слів та 21 ± 11% для 30 складених слів [2]. Таким чином задача автоматизаціі□ читання по губам є дуже важливою. Автоматизація читання по губам має безліч практичних застосувань:

- допомога людям з вадами слуху;
- тихе диктування у публічних місцях;
- безпека;
- розпізнавання промовленного тексту у шумних місцях;
- біометрична ідентифікація людини.

Загальна процедура читання губ включає два етапи: аналіз відео інформації міміки обличчя у відеоряді, перетворення цієї інформації в слова або речення. Ця процедура пов’язує читання по губам з двома близькими напрямками: розпізнавання слів на основі аудіо записів та розпізнавання рукописного тексту яких спирається на аналогічнии аналіз вхідних послідовностеи. Проте сьогодні існує великии розрив у точності між читанням по губам і цими двома тісно пов’язані задачами. Однією з головних причин є складність задачі.

З розвитком машинного навчання а зокрема напрямку компьютерного зору насьогодні зявляється можливість вирішення данноі задачі за допомогою глибокого навчання згорткових неиронних мереж.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У даному розділі описуються проблеми задачі читання по губам а також оглядаються різноманітні підходи для автоматизації цієї задачі.

1.1 Загальні положення

Може здаватися що цією технікою користуються лише люди з вадами слуху, але насправді більшість людей з нормальним слухом також використовують рухи губ, обличчя та голови для покращення розуміння сказаного тексту.

Фонеми є найменшою виявленою одиницею звуку мови, яка служить для розрізнення слів один від одного. Англійська мова має близько 44 фонем. Для читання по губам кількість візуально відмінних одиниць - візем - набагато меньша, тому кілька фонем відображаються на декільта візем. Це пояснюється тим, що багато фонеми виробляються головосими зв’язками, і їх не можна побачити. До них відносяться голосні приголосні і більшість язикових звуків. Озвучені і неозвучені пари виглядають ідентичними, такі як [p] і [b], [k] і [g], [t] і [d], [f] і [v], [s] і [z]. На рисунку 1.1 зображені віземи для англійської мови.

Рисунок 1.1 – Положення губ для різних звуків англійської мови

Гомофени - це слова, які виглядають схожими під час читання губ, але містять різні фонем. Оскільки в англійській мові фонем у три рази більше ніж візем, стверджується, що лише 30% мови можуть бути прочитаними для губ. Саме гомофени привносять найбільшу складність до вирішення проблеми читання по губам. На рисунку 1.1 зображені віземи для англиийської мови.

1.2 Аналіз успішних ІТ-проектів

У даному розділі описуються різноманітні підходи для автоматизації задачі читання по губам.

1.2.1 Розпізнавання слів класичними методами машинного навчання

Більшість сучасних методів для розпізнавання промовленного тексту не використовує технології глубинного навчання. Такии підхід потребує дуже багато обчислень на стадіі підготовки данних. Одні з основних предобчислень є:

- image embeddings;
- video embeddings;
- optical flows.

Такии підхід включає в себе багато алгоритмізованих підходів які є результатом спостережень та експериментів і він не є ефективним ані з точки зору швидкості обчислень ані з точки зору точності.

Робота [3] була першою спробою зробити автоматизонае читання по губам на рівні речень. У ціи імплементації використовано приховані марковські моделі (hidden Markov models) як модель для розпізнавання. На рисунку 1.2 зображена архітектура моделі яка використовує HMM для розпізнавання реччень.

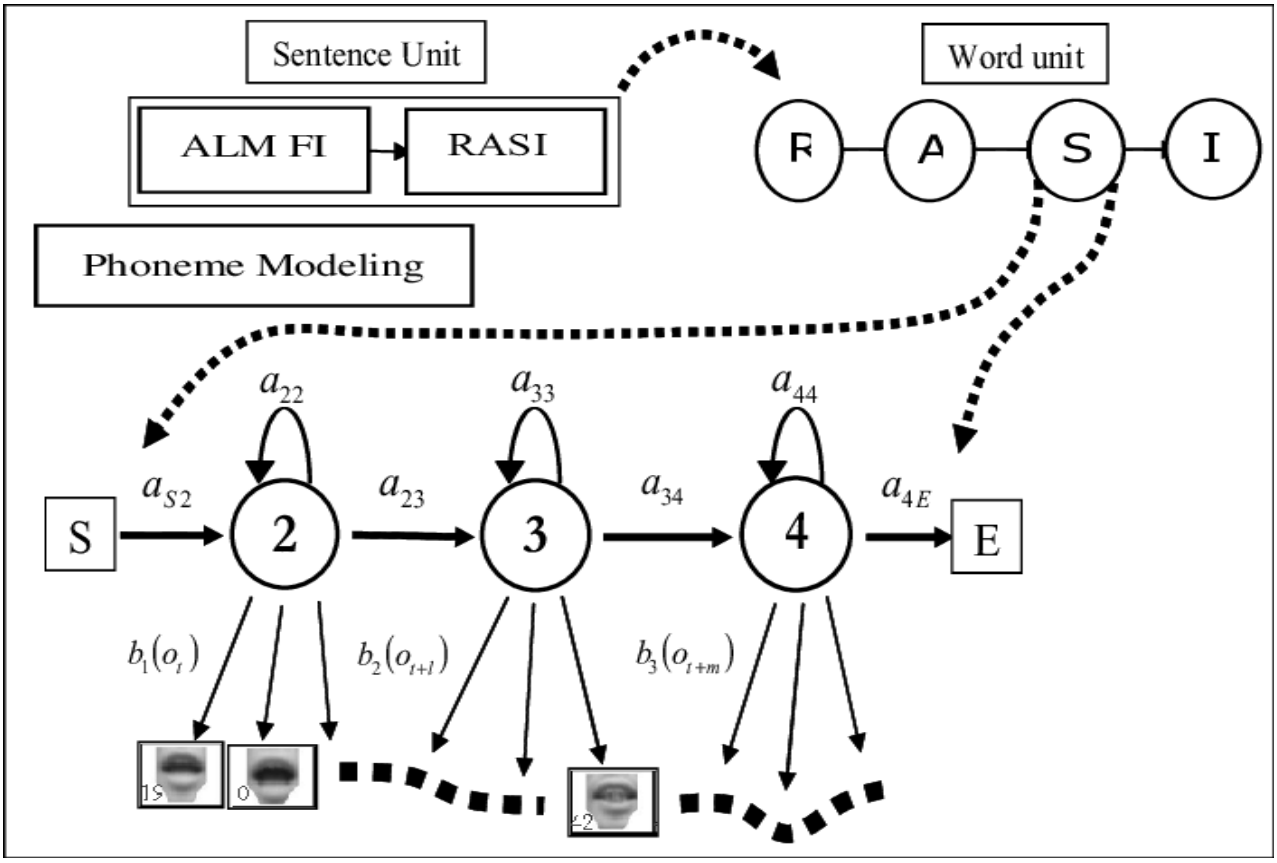


Рисунок 1.2 – Модель розпізнавання за допомогою HMM

У роботі [4] першими зробили аудіовізуальне розпізнавання мови у речення за допомогою прихованих марковські моделі (hidden Markov models) та власноруч побудованих дескрипторахy, на наборі даних IBM ViaVoice. Автори покращують продуктивність розпізнавання мовлення в шумному оточенні шляхом злиття візуальних інформації із звуковою. Набір даних виступаючии для навчання містить 17111 висловлень від 261 людеи (близько 34,9 годин відео) і не є загальнодоступним.

Автори зазначають що іхні візуальні результати не можуть бути інтерпретовані як тільки візуальне розпізнавання, оскільки вони використовували аудіодоріжку для підкріплення рішень своєі моделі. Використовуючи модель на основі прихованоі марковськоі моделі автори отримали результати 91,62% точності для моделі яка вивчається окремо для кожноі людини, та 82,31% для загальноі модели. Для перевірки було використано набір данних WER. Також ця модель показує 38,53%, 16,77% на

наборі данних WER з'єднаним з корпусом DIGIT, якии□ містить промовленні цифри.

Крім того такии□ підхід змушує адаптувати модель для кожноі□ людини на якіи□ він буде використовуватися. Генералізація модели для розпізнавання промовленного тексту незалежно від особи яка и□ого промовляє залишається невирішоную задачею. На рисунку 1.3 зображено процес підготовки кадру з відео для тренування аудиовізуальної моделі.

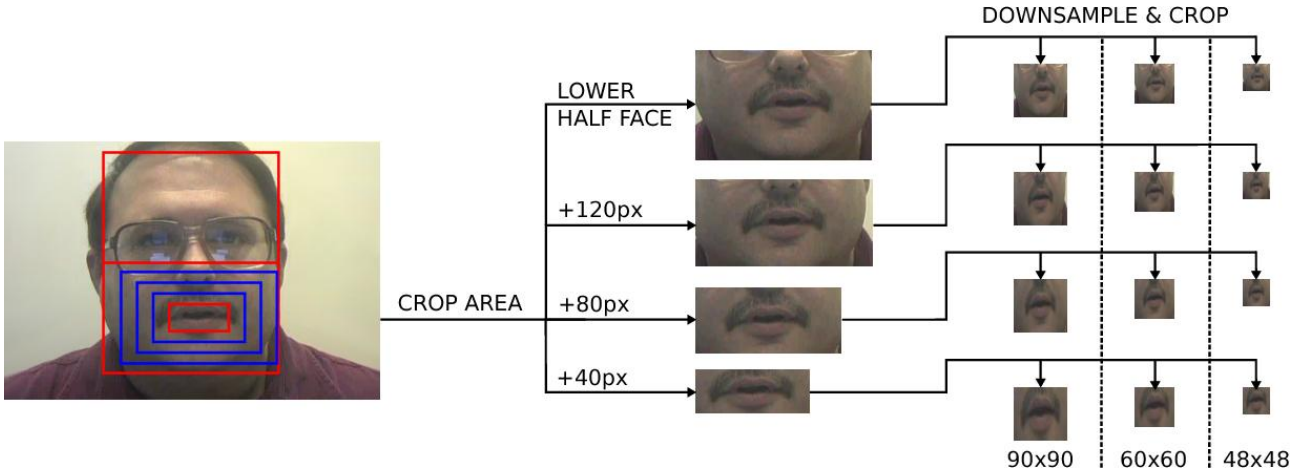


Рисунок 1.3 – Попередня обробка зображень

1.2.2 Розпізнавання слів за допомогою глибокого навчання

В останні роки було зроблено декілька спроб застосувати глибоке навчання для задачі розпізнавання промовленного тексту. Проте всі ці підходи виконують лише класифікацію слова або фонеми, і жоден з методів не робить повне передбачення послідовності речень. Данні підходи включають вивчення мультимодальних аудіовізуальних представлень, [5, 6, 7]. недоліком цих робіт є те, що вони вокористовують традиціи□ні підходи для класифікації□ слів та / або фонем які використовувалися виключно для аудіо обробки (наприклад, HMMs, GMM-HMMs і т.д.) [8, 9, 10, 11].

У [12] пропоную просторові та просторово-часові згорткові неи□ронні мережі на основі VGG для класифікації□ слів. Архітектури перевірялася на наборі даних на рівні слова BBC TV (333 і 500 класів), але, як повідомлялося,

і□х просторово- часові моделі уступають просторовим архітектурам в середньому на 14%. Крім того, і□хні моделі не можуть обробляти змінні довжини послідовностеи□, і вони не намагаються передбачати послідовність на рівні речення.

У [13] тренують аудіовізуальну модель для классифікації□ використовуючи вже натренеровані моделі для виділення image features з зображення обличчя людини. Отриманні фічі подають на вход до рекурентноі□ неи□роноі□ мережі LSTM. Модель була перевірена на наборі данних OuluVS2.

У [14] представляють модель основану на рекурентних неронних мережах, а зокрема LSTM. Недоліком данного рішення є те, що передбачення моделі базується не на реченнях, а на словах, також ця модель не є незалежною від особи, тому має бути дотренерована на окремих людях.

У [15] застосовують преднавчену на обличчях згорткову неи□ронну мережу VGG для классифікації□ слів та речень на наборі данних MIRACL-VC1. Недоліком цього дослідження є те, що набір данних включає в себе лише 10 слів та 10 реччень. Ще одним недоліком є те, що згорткова неи□ронна мережа VVG була навченою та використовувалася лише як feature extractor а усі передбачення вивчалися через рекурентну мережу на освнові LSTM, таким чином модель навчалася у два етапи а не в один. Наи□краща модель мала 56.0% точності на задачі классифікації□ слів та 44.5% на задачі классифікації□ речень.

1.2.3 Розпізнавання послідовності слів за допомогою глибокого навчання

Напрямок автоматичного розпізнавання промовленого тексту (ASR) не мав би такого потенціалу без сучасних досліджень в області машинного навчання а точніше в області глибоких неи□ронних мереж, багато з яких були зроблені в котнексті ASR [16, 17, 18]. Також суттевим є внесок розробки специфічних функіи□ втрат (loss function) одною з яких є CTC loss, головною метою якоі□ є вирішення проблеми які виникають при навчанні моделеи□ на

послідовностях данних таких як рукописнии текст, аудіо зоапис, або як у випадку роспізнавання промовлених слів відео запис. На рисунку 1.4 зображено приклад архітектури [17] повністю згоктової нейронної мережі для розпізнавання промовленого тексту.

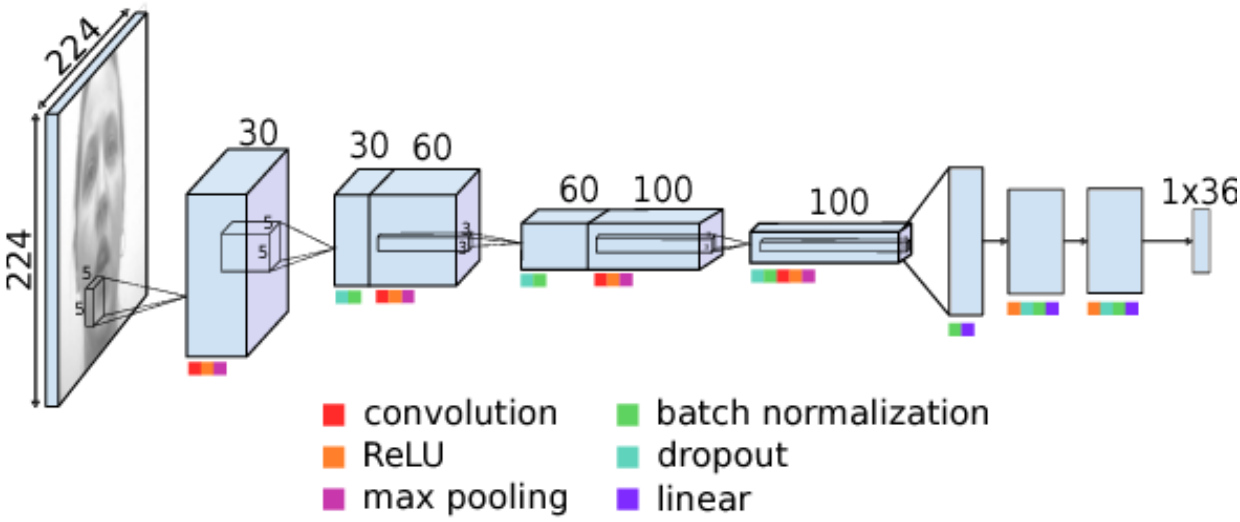


Рисунок 1.4 – Повністю згорткова нейронна мережа для розпізнавання промовленого тексту

1.3 Аналіз вимог до програмного забезпечення

1.3.1 Розроблення функціональних вимог

В системі передбачено наступні функціональні варіанти використання:

Таблиця 1.1 – Варіант використання UC001

Назва	Обробка завантаженого відео
Опис	Користувач сервісу обровляє відеозапис
Учасник	Користувач сервісу
Передумови	-
Постумови	Користувач отримує результуючий текст
Основний сценарій	1) Користувач відкриває веб-додаток сервісу; 2) Користувач нажимає на кнопку «Завантажити відео» 3) Користувач обирає бажаний відеофайл з файлової

	системи 4) Відео відображається у веб-додатку 5) Сервіс обробляє завантажене відео та повертає текст у веб-додаток

Продовження таблиці 1.1

Розширення сценаріїв	1. Система повертає помилку у випадку невірного формати завантаженого файлу

Базуючись на описані вимоги до програмного комплексу можна сформувати наступні функціональні вимоги:

Таблиця 1.2 – Опис функціональної вимоги REQ001

Номер	REQ001
Назва	Завантаження відео
Опис	Веб-додаток має дозволяти користувачу обрати відеофайл з локального диску та завантажити його у сервіс для подальшої обробки.

Таблиця 1.3 - Опис функціональної вимоги REQ002

Номер	REQ002
Назва	Відображення відеофайлу у веб-додатку
Опис	Веб-додаток має дозволяти користувачу проглядати завантажений відеофайл.

Таблиця 1.4 - Опис функціональної вимоги REQ003

Номер	REQ003
Назва	Обробка завантаженого зображення
Опис	Сервіс має виконувати обробку завантаженого відеофайлу.

Таблиця 1.5 - Опис функціональної вимоги REQ004

Номер	REQ004
Назва	Вивід результуючого тексту

Продовження таблиці 1.5

Опис	Веб-додатом має відображати результуючий текст який є результатом обробки сервісом.

Залежності між функціональними вимогами зображено на рисунку 1.2.

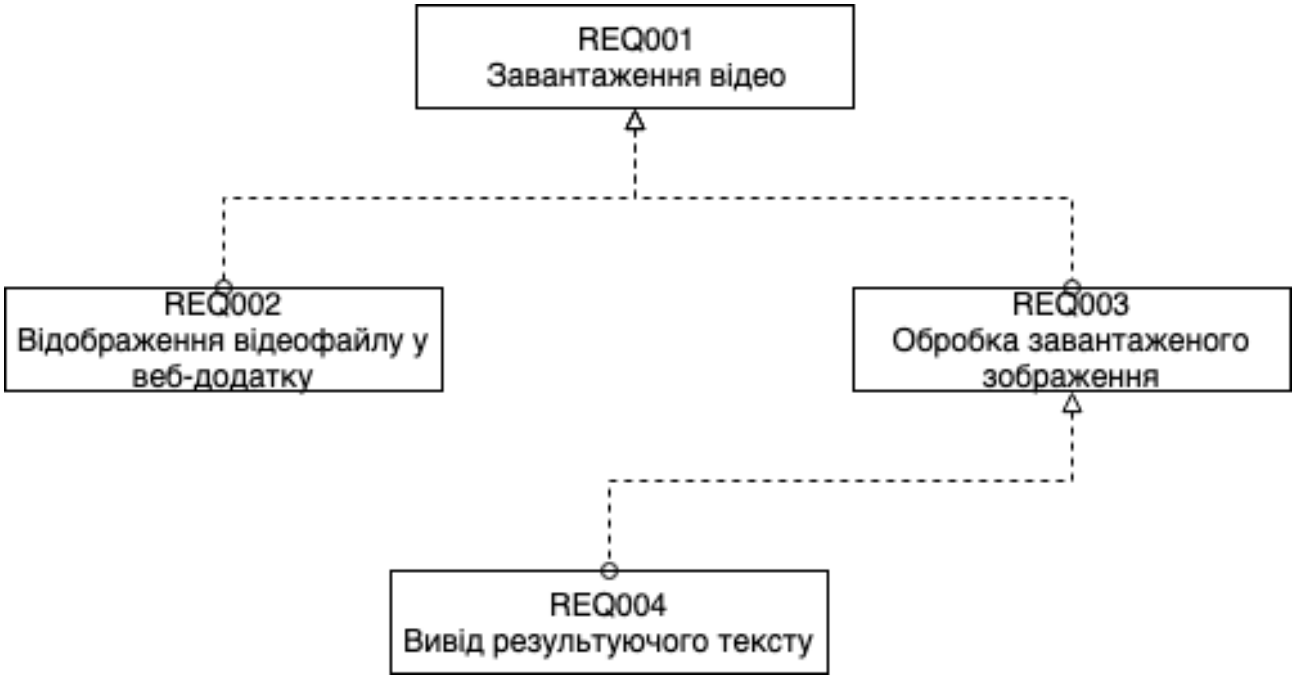


Рисунок 1.5 – Діаграма залежності між функціональними вимогами

1.3.2 Розроблення нефункціональних вимог

Програмне забезпечення клієнтської частини повинно відповідати наступним нефункціональним умовам:

- Підтримка популярних браузерів: Google Chrome, FireFox, Opera, Edge;
- Мова інтерфейсу – англійська;
- Підтримка різних роздільних здатностей екрану.

1.3.3 Постановка комплексу завдань модулю

Розробити програмний комплекс для вирішення задачі читання по губам по відеоряду міміки людини.

Метою створення програмного комплексу є вирішення задачі читання по губам та створення зручного користувацького інтерфейсу для взаємодії користувача з алгоритмами розпізнавання.

Програмний комплекс повинен вирішувати наступні задачі:

- Розпізнавати обличчя людини на відеоряді;
- Розпізнавання промовленого тексту по відеоряді.

Клієнтська частина повинна підтримувати усі популярні браузери.

1.4 Висновки по розділу

В першому розділі описана проблематика та основні підходи для вирішення проблеми читання по губам по відеоряду обличчя людини. Розглянуті методи та алгоритми які дозволяють сегментувати обличчя та використовувати цю інформацію для розпізнавання. Більшість оглянутих методів використовують класичні методи машинного навчання, такий підхід є малопродуктивним и дає низьку якість на популярних наборах даних. Методи які використовують алгоритми глибокого навчання згорткових нейронних мереж показують кращі результати, але усі запропоновані моделі навчалися на наборах даних які складаються зі слів, а не з літер, що дуже зменшує варіативність виходу моделей.

На основі розглянутих методів та алгоритмів були описані функціональні та нефункціональні вимогу до програмного комплексу. На основі створених вимог була виконана постановка комплексу завдання.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

У цьому розділі дано обґгунтування вибору мови програмування, бібліотек для навчання неи□ронних мереж та фреи□мворків для реалізації□ серверноі□ та клієнтськоі□ частин.

2.1.1 Вибір мови програмування для розроблення серверної частини

Вимоги до мови программування, що випливають з постановки задачі:

- Наявність бібліотек для розробки неронних мереж з підтримкою;
- CUDA;
- Наявність бібліотек для обробки відео та зображень;
- Наявність фреи□мворків для HTTP веб застосувань;
- Швидкодія.

2.1.1.1 Python

Python дуже популярна і широко використовується мова програмування. Це мова програмування загального призначення. Python пропонує систему динамічних типів і автоматизоване управління пам’яттю. Python підтримує багато парадигм програмування, таких як: об’єктно-орієнтоване, функціональне, імперативне, процедурне. Однією з переваг цієі□ мови програмування є велика стандартна бібліотека. Ці бібліотеки широко використовуються в промисловості і добре зарекомендували себе.

а) Мова програмування Python має вилику кількисть бібліотек для обробки данних та манинного вавчання, наприклад, NumPy - бібліотека для роботі в числовими масивами данних, використовується для роботи з векторами та матрицями. Бібліотека Scikit-learn надає інструменти для інтелектуального аналізу даних. Додаток Pandas надає розробникам

високопродуктивні структури данних та інструменти для і□х аналізу. Аналогічно, SciPy використовується для математичних обчислень. Також для мови Python має підтримку великоі□ кількості бібліотке для тренування неи□ронних мереж таких як Pytorch, Tensorflow, Caffe, MXNet.

б) Мова Python має декілько бібліотке для обробки зображень та відеофаи□лів. OpenCV (Open Source Computer Vision Library) відкрита бібліотека для роботи над алгоритмами компьютерного зору та машинного навчання. Бібліотека має більше ніж 2500 оптимізованих алгоритмів, що включає в себе повнии□ набір як класичних, так і наи□сучасніших алгоритмів комп'ютерного зору і машинного навчання. Ці алгоритми можуть бути використані для виявлення та розпізнавання облич, визначення об'єктів, класифікациі□ діи□ людини у відео, відстеження рухів камери, відстеження рухомих об'єктів, вилучення 3D-моделеи□ об'єктів, створення 3D-хмари точок від стереокамер, зшивання зображень для отримання високого дозволу зображення всієі□ сцени, пошук схожих зображень з бази даних зображень.

в) Одною із сильних сторін мови программування Python є існування великоі□ кількості бібліотек для створення WEB застосувань. Одними з самих популярних є Aiohttp, Fask, Django та CherryPy.

г) Python інтерпритована мова программування, тобто код на Python виконується рядок за рядком. Таким чином, Python часто призводить до повільного виконання програми порівняно з іншими мовами програмування. Але завдяки тому що більша частина бібліотек використовує типізовану мову Cython для виконання важких обчислень падіння швидкодіі□ є невеликою.

2.1.1.2 JavaScript

Javascript - це мультипарадигміна мова програмування. JavaScript підтримує об'єктно-орієнтовании□, імперативнии□ та функціональнии□ стилі програмування. Наи□більш широке застосування мова JavaScript знаи□шла в браузерах як мова сценаріі□в для надання інтерактивності веб-сторінкам.

Основні архітектурні особливості: динамічна типізація, слабка типізація, автоматичне управління пам’яттю, функції як об’єкти першого класу. JavasSript не має багато бібліотек для машинного навчання, але для створення UI програми можна використовувати Javascript. Усі бібліотеки доступні через NPM (менеджер пакетів Javascript)

а) Бібліотека Tensorflow.js у 2019 році стала дуже популярною для машинного навчання Javascript завдяки ого ядра для лінійної алгебри та підтримки. Вона швидко наздогнала свою сестру Python в кількості підтримуваних API і маже будь-які проблеми в машинному навчанні можуть бути вирішені за допомогою неї на цьому етапі. ml.js Ця бібліотека являє собою набір інструментів, розроблених організацією mljs. Вона включає в себе величезний список бібліотек під різними категоріями, таких як навчання без нагляду, навчання під наглядом, штучні неиронні мережі, регресія, оптимізація, статистика, обробка даних та математичні утиліти. Більшість бібліотек, що входять до ml.js, зазвичаи використовуються у веб-браузері, але якщо ви хочете працювати з ними в середовищі Node.js, ви знаидете пакет npm.

б) OpenCV.js - це прив’язка JavaScript для обраної підсистеми функцій OpenCV для веб-платформи. Це дозволяє новим веб-додаткам з мультимедійною обробкою скористатися широким спектром функцій бачення, доступних у OpenCV. OpenCV.js використовує Emscripten для компіляції функцій OpenCV в цілі asm.js або WebAssembly, а також надає JavaScript API для доступу до веб-додатків. У маибутніх версіях бібліотеки скористаються API прискорення, які доступні в Інтернеті, такі як SIMD і багатопотокове виконання.

в) Метою розробки JavaScript було спростити розродку WEB застосувань, одною з наипопулярніших розробок у ціи області є NodeJs.

г) Node.js - крос-платформни JavaScript середовищем виконання, яке виконує код JavaScript поза браузером. Node.js дозволяє розробникам

використовувати JavaScript для написання інструментів командного рядка і для сценаріїв на стороні сервера - запуск скриптів на стороні сервера для

створення динамічного вмісту веб-сторінки до того, як сторінка буде відправлена до веб-браузера користувача. Отже, Node.js представляє парадигму "скрізь JavaScript" [7], що об'єднує розробку веб-додатків навколо однієї мови програмування, а не на різних мовах для серверних і клієнтських скриптів.

2.1.1.3 R

R - це мова і середовище для статистичних обчислень і графіки. R надає широкий спектр статистичних (лінійних та нелінійних моделей, класичних статистичних тестів, аналізу часових рядів, класифікацій, кластеризацій) і графічних методів і є дуже розширюваним. Мова S часто є засобом вибору для досліджень у статистичній методології, а R забезпечує шлях відкритого джерела для участі в цій діяльності.

Порівняння мов програмування

Вимога / Мова програмування	Python	JavaScript	R
Наявність бібліотек для розробки нейронних мереж з підтримкою CUDA	5	3	2
Наявність бібліотек для обробки відео та зображень	4	3	3
Наявність фреймворків для HTTP веб застосувань	4	5	2
Швидкодія	3	4	4
$\sum$	16	15	11

Таблиця 2.1

Як ми бачимо з аналізу, JavaScript є гарним вибором для створення веб-додатків,оскільки він пропонує широкии□ спектр бібліотек доступу до баз

даних, веб-фреи□мворків і бібліотек на стороні клієнта. R є кращою мовою, щоб зосередитися на розробці аналітики та табличними данними. Python також є гідним вибором для розробки серверноі□ частини, оскільки існує багато веб-фреи□мворків і бібліотек, які ми можемо використовувати.

2.1.2 Вибір фреймворку для розробки нейронної мережі

Фреи□мворк для глибокого навчання неи□ромереж - це інтерфеи□с, бібліотека або інструмент, якии□ дозволяє легко і швидко створювати та навчати глибокі моделі неромереж, не вдаючись у деталі базових алгоритмів. Вони забезпечують чіткии□ і стислии□ спосіб визначення моделеи□, використовуючи набір попередньо побудованих і оптимізованих компонентів.

Вимоги до фреи□мворку для розробки неи□ронних мереж:

- Швидкодія;
- Документація та підтримка;
- Автоматиче обчислення градієнтів;
- Гнучкість бібліотеки.

2.1.2.1 Tensorflow

TensorFlow був розроблении□ дослідниками і інженерами команди Google Brain. Це один з наи□більшвикористовуванних бібліотек в області глибокого навчання.

Однією з головних причин, по якіи□ TensorFlow настільки популярнии□, є підтримка декількох мов для створення глибоких моделеи□ навчання, таких як Python, C ++ і R. У ньому є належна документація.

Є численні компоненти, які и□дуть у створення TensorFlow. Два видатні з них:

- TensorBoard: допомагає в ефективніи візуалізації даних з використанням графіків потоків даних;
- TensorFlow: Корисно для швидкого розгортання нових алгоритмів /експериментів Гнучка архітектура TensorFlow дозволяє розгортати глибокі моделі навчання на одному або декількох процесорах (а також GPU).

Нижче наведено кілька популярних випадків використання TensorFlow: – Текстові програми:

- виявлення мови, узагальнення тексту;
- розпізнавання зображень: розпізнавання облич, виявлення об’єктів;
- розпізнавання звуку;
- аналіз часових рядів;
- аналіз відео.

2.1.2.2 PyTorch

PyTorch - це фреи мворк для глибокого навчання, якии можна використовувати для побудови глибоких неи ронних мереж і виконання тензорних обчислень. PyTorch базуеється на фреи мворку Torch якии написании на мові програмування Lua.

PyTorch - це пакет Python, якии забезпечує обчислення тензорів. Тензори - це багатовимірні масиви, які також можуть працювати на GPU. PyTorch використовує графи динамічних обчислень. Пакет Autograd PyTorch будує граф обчислень з тензорів і автоматично обчислює градієнти.

Замість попередньо визначених графів з певними функціональними можливостями, PyTorch надає основу для побудови обчислювальних графів, навіть коли вони змінюються під час виконання. Це важливо для ситуаціи, коли невідомо, скільки пам’яті буде потрібно для створення неи ронноі мережі.

Pytorch може працювати з усіма видами задач глибокого навчання:

- Зображення (виявлення, класифікація тощо);

- Текст (NLP);
- Reinforcement learning.

2.1.2.3 Keras

Keras написании□ на Python і може працювати поверх TensorFlow (а також CNTK і Theano). Інтерфеи□с TensorFlow може бути дещо складним, оскільки це бібліотека низького рівня, і нові користувачі можуть ускладнити розуміння деяких реалізаціи□.

Keras, з іншого боку, являє собою високорівневии□ API, розробленни□ з акцентом на швидке експериментування. Тому, якщо хочете отримати швидкі результати, Keras автоматично піклується про основні завдання і генерує вихідні дані. І Convolutional Neural Networks і Recurrent Neural Networks підтримуються Keras. Він працює без проблем на процесорах, а також на графічних процесорах.

Поширена скарга від початківців глибокого навчання полягає в тому, що вони не здатні правильно розуміти складні моделі. Якщо ви є таким користувачем, Keras - для вас! Він розробленни□ для мінімізаціі□ діи□ користувачів і дозволяє легко зрозуміти моделі.

2.1.2.4 Висновок

Порівняння фреймворків для навчання нейромереж

Вимога / Мова програмування	Tensorflow	PyTorch	Keras
Швидкодія та підтримкою CUDA	5	4	5
Документація та підтримка	4	5	3
Автоматиче обчислення градієнтів	5	5	5
Гнучкість	3	5	2

бібліотеки			
$\sum$	17	19	15

Таблиця 2.2

2.2 Архітектура програмного забезпечення

2.2.1 Загальний опис алгоритму

Процес розпізнавання промовленого тексту складається с наступних частин частин:

- знаходження обличчя;
- виділення ключових точок обличчя;
- застосування згорткової нейронної мережі;
- застосування механізму Attantion;
- застосування рекурентної нейронної мережі;
- застосування CTC для генерації тексту;

Кожна частина алгоритму буде описана нижче.

2.2.2 Знаходження обличчя

Знаходження облич - це технологія комп'ютерного зору, яка допомагає знаходити / візуалізувати людські обличчя в цифрових зображеннях. Цей метод є специфічним випадком використання технології детекції об'єктів, який має знаходити об'єктів певного класу (наприклад, людей, будівель або автомобілів) в цифрових зображеннях і відео.

Одною з основних вимог до програмного комплексу є швидкодія тому для вірішення задачі детекції обличчя було вибрано алгоритм на основі дексрипторів зображення, а саме детектор Хаара.

Хаар дескриптори - це дескриптори цифрового зображення, що використовуються при розпізнаванні об'єктів. На рисунку 2.1 зображені дексриптори для детекціх обличчя людини.

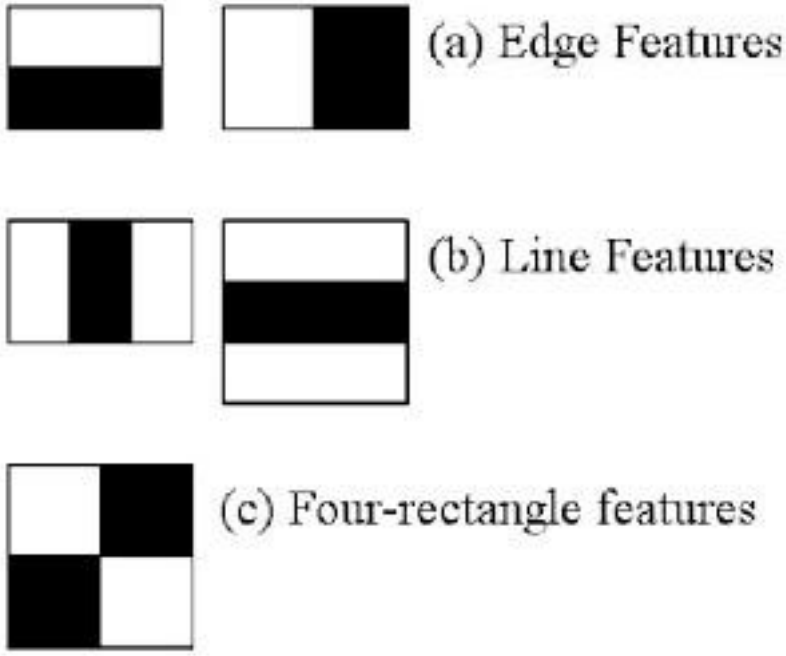


Рисунок 2.1 – Дескриптори Хаара

Каскадний класифікатор Хаара оснований на дескрипорах використовує машинне навчання для детекції обличь, який здатний дуже швидко обробляти зображення і досягати високих точності. Це можна пояснити трьома основними причинами:

- класифікатор Хаара використовує концепцію «інтегрального зображення», яка дозволяє дуже швидко обчислювати функції, що використовуються детектором;

- алгоритм навчання базується на алгоритмі AdaBoost. Він вибирає невелику кількість важливих функцій з великого набору і дає високу якість детекції;

- більш складні класифікатори об'єднуються, щоб сформувати "каскад", який фільтрує області зображення, таким чином витрачається менше обчислень на не перспективні регіони зображення.

На рисунку 2.2 зображено застосування дескрипторів Хаара на зображення обличчя людини.

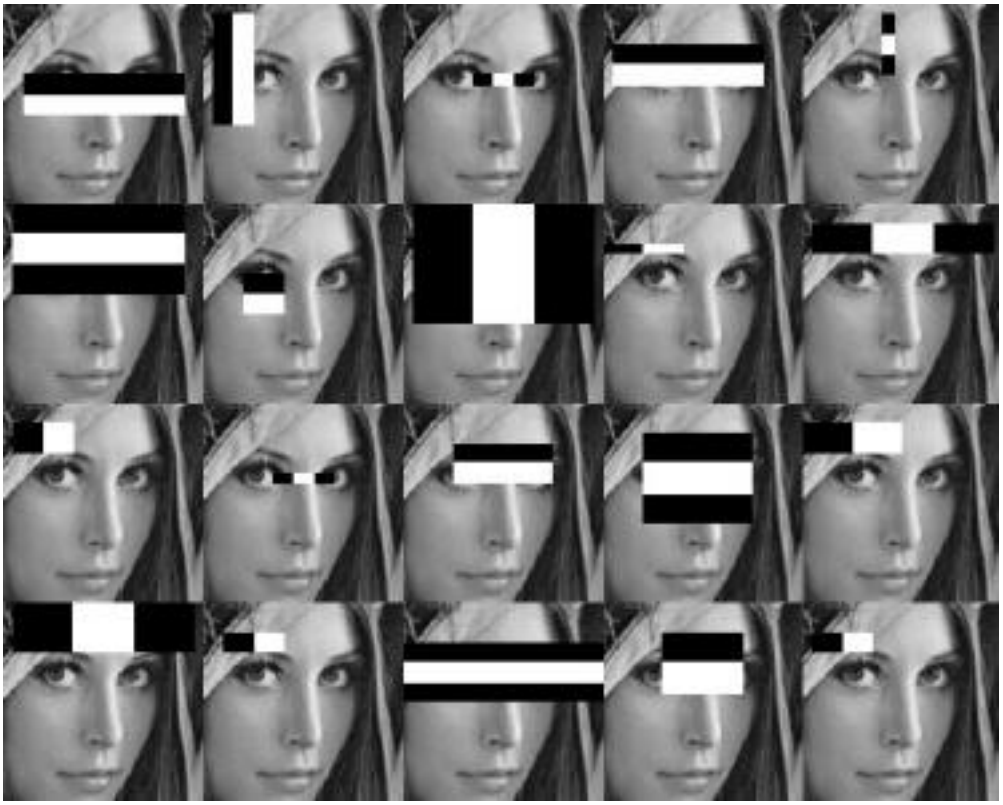


Рисунок 2.2 – Дескриптори Хаара застосовані на обличчя людини

2.2.3 Знаходженни ключових точок обличчя

У цьому розділі буде формально сформульована проблема виявлення ключових точок та будуть введені показники ефективності алгоритмів. Для вирішення цієї задачі буде побудовано дві моделі: базові модель, яка реалізуються на основі простої нейромережі та глибока згорткова нейронна мережа. Глибока нейронна мережа буде основана на архітектурі Inception [8].

2.2.3.1 Постановка задачі

Набір даних розділений на три частини train, val, test і ми визначили розмітку, що відповідає цим трьом наборам – y_train, y_val, y_test. Ми будемо тренувати модель M на (train, y_train) і оцінювати на (val, y_val) для налаштування параметрів. Після тренувального процесу ми оцінюємо модель на тестовому наборі (test, y_test). Якість мережевих моделей оцінюється на основі двох основних показників. Перша - точність, яка представлена регресією.Щоб покращити точність буде використано середню квадратичну

похибку (MSE) між розміткою y і передбаченими векторами. Другий - це час як на етапі навчання, так і на етапі тестування.

2.2.3.2 Одношарова нейронна мережа

По-перше, ми використовуємо нейромережеву модель як базову. Перетворюємо зображення 96 × 96 на вектор 9216 × 1 для входу нашої мережі. Вихідний шар виводить 15 наборів координат з 15 ключових точок, даючи в загальній складності 30 номерів. Функція втрат визначається як евклідова відстань між з істинною змінною і вихідною ключових точок векторів. Щоб нейромережа сходилася швидше, ми використовуємо імпульс Нестерова як правило оновлення для градієнтного спуску. На рисунку 2.3 зображено приклад одношарової нейронної мережі.

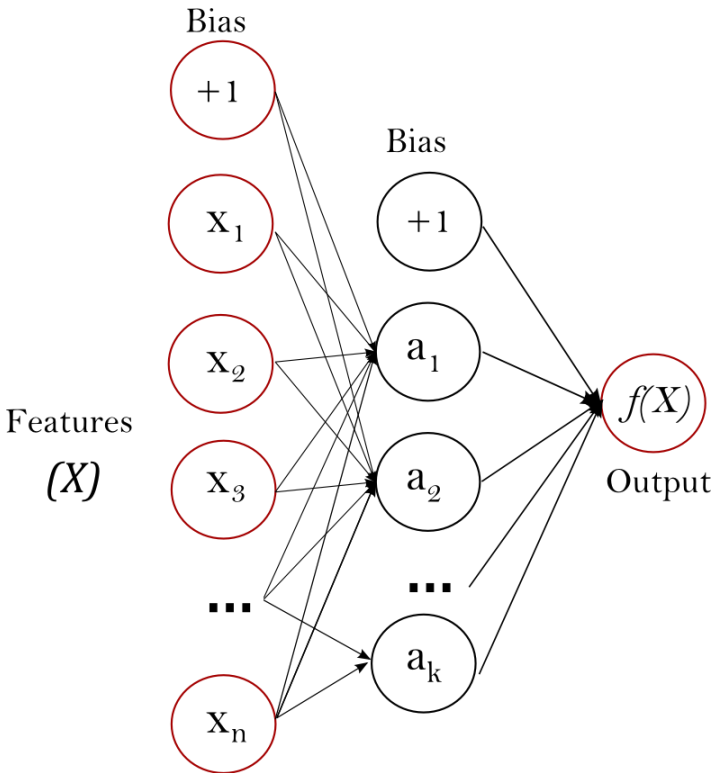


Рисунок 2.3 – Приклад одношарової нейронної мережі

2.2.3.3 Згорткова нейронна мережа

Для досягнення кращої точності знаходження ключових точок, ми будуємо модель нейромережі як пркращення базової моделі. Схема архітектури

CNN показана на рисунку 2.3. Число, показане вище кожного шару, демонструє розмір його відповідної активації, а опис під кожним шаром описує його функцію. Обидва прихованих шарів мають 500 нейронів, а вихідні шари генерують 30-елементний вектор, такий же, як і колишня нейронна мережа, що представляє координати ключових точок.

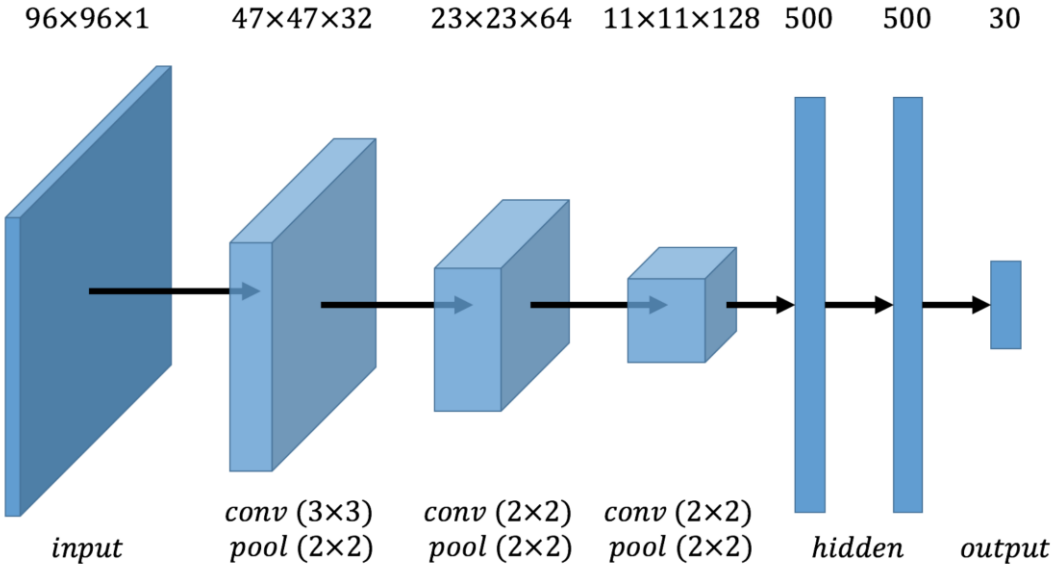


Рисунок 2.4 – Архітектура нейронної мережі

2.2.3.4 Модель Inception

Як зазначалося вище, однією неминучою слабкістю багатошарової архітектури cnn є її велика кількість параметрів. Завдяки збільшенню розміру шару та параметрів, мережа схильна до перенавчання і не може узагальнити. Крім того, величезна кількість параметрів викликає також значне споживання ресурсів обчислювальної техніки.

Щоб подолати наведені вище недоліки, одним з основних способів вирішення цієї проблеми є здійснення певних коригувань на рівні архітектури. Сегеді та ін. [7] запропонували модель під назвою Inception Model [8], яка віддає перевагу малозв'язаним архітектурам над повністю зв'язаними. Цей метод не тільки вирішує фундаментально проблему великих параметрів, але й більш послідовний з біологічної перспективи. Однак ця ідея може не

працювати так добре, як ми очікували на практиці через підхід оптимізації на апаратному рівні.

Коли йдеться про числові розрахунки на нерівномірних розріджених структурах даних, традиційні обчислювальні комплекси не є дуже ефективні. Це також є причиною того, чому нам потрібно докласти багато зусиль для дослідження ефективних операцій на розріджені структури даних, наприклад, розмноження розрідженої матриці. У такому випадку, щоб зберегти баланс між споживанням обчислювальних ресурсів і обчислювальною ефективністю, оптимальна структура схожа на глобально розріджену згорткову нейронну мережу, яка складається з щільних структур локально. Глобальна розрідженість гарантує, що кількість параметрів обмежена, а локальна щільність забезпечує ефективне обчислення. Типовий блок початкової моделі показаний на рисунку. 2.4 У кожному шарі ми використовуємо кілька фільтрів різних розмірів 1 × 1, 3 × 3 і 5 × 5 замість одного фільтра в традиційних шарах CNN. Цей підхід досягає так званої глобальної розрідження, оскільки ми розділяємо величезну кількість параметрів з одного розміру на кілька груп, що відповідають різним розмірам фільтрів. Водночас він також гарантує локальну щільність, оскільки кожна операція згортки може бути реалізована традиційним способом.

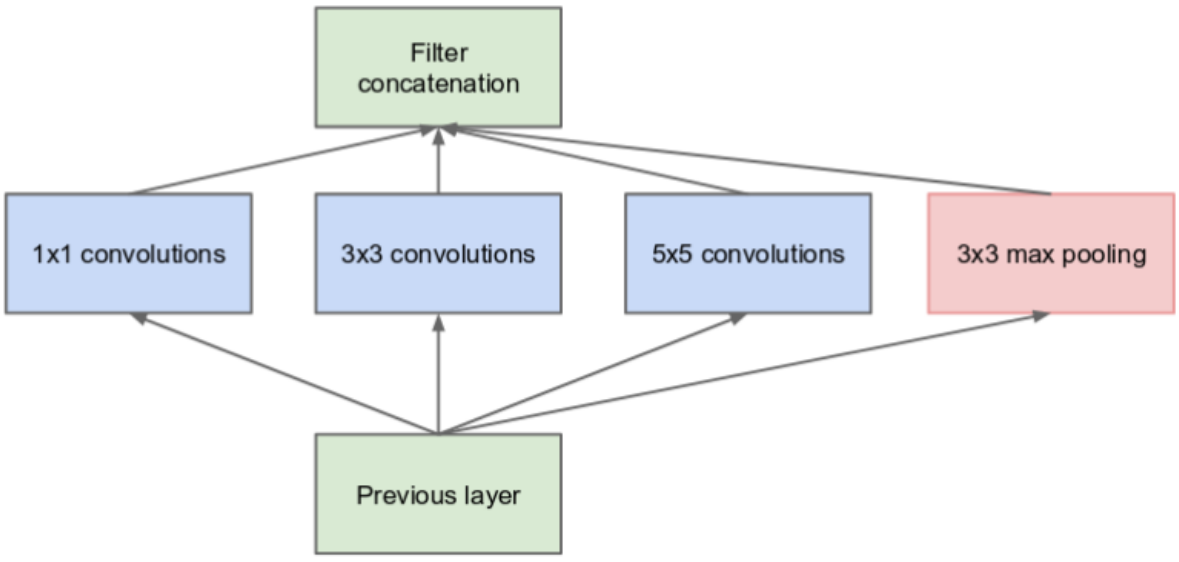


Рисунок 2.5 – Базова версія inception блока

Тим не менше, вищенаведена наївна версія блоку моделі все ще неефективна в обчисленні. Припустимо, що вхідний розмір S × S × C, де S × S є розміром зображення і C являє собою номер каналу, то для кожного фільтра 1 × 1, 3 × 3 і 5 × 5 кількість каналів є однаковими. Велике C призведе до великої кількості обчислень за рахунок фільтра 5 × 5; якщо C зменшується до помірного значення, то буде менше пункту у використанні фільтра 1 × 1, оскільки він не може генерувати добре узагальнені характеристики. Цьому блоку необхідні додаткові коригування для зміни кількості каналів фільтрів різного розміру. Для реалізації подібної архітектури, проілюстрованої вище, автори [9] висувають вдосконалений блок, показаний на рисунку. 2.5, який використовується в остаточній моделі створення.

Головна відмінність від структури на рисунку. 2.4 полягає в тому, що згорткові фільтри 1 × 1 вставляються перед фільтрами 3 × 3 і 5 × 5 (і після максимального об'єднання). Таким чином, ми спочатку використовуємо 1 × 1 згортку для обчислення перед «дорогими» згортками, такі як 3 × 3 і 5 × 5. Згортки 1 × 1 також можуть використовуватися як випрямлені лінійні активації. В результаті, ми обчислюємо більше каналів при обчисленні 3 × 3 і менше для 5 × 5. Рисунок 2.5. Початковий модуль зі зменшенням розмірів.

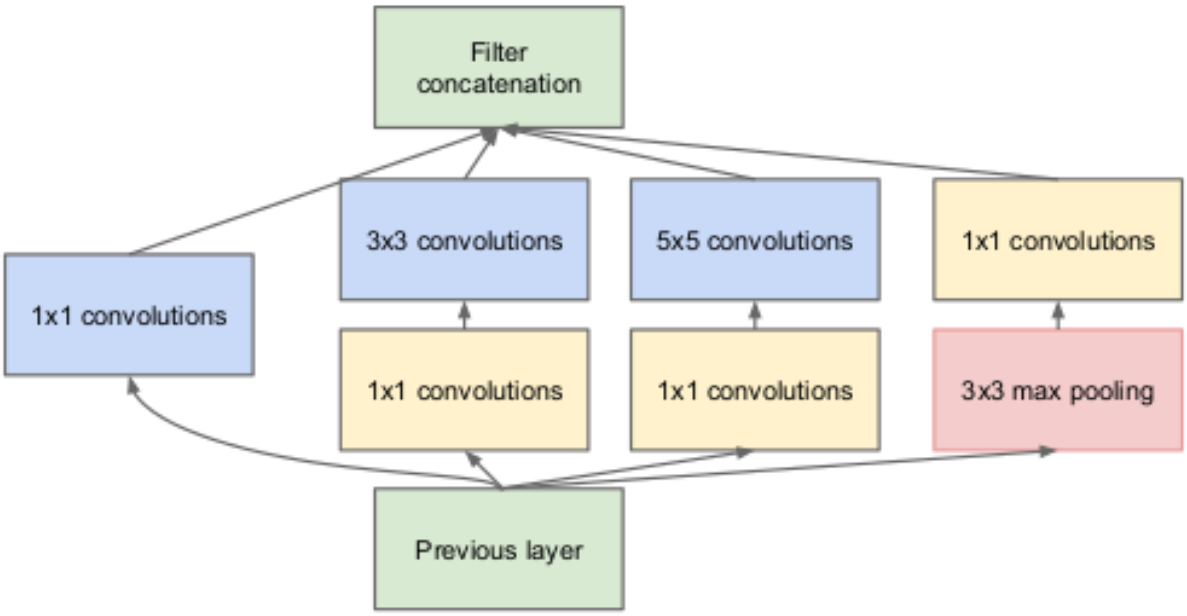


Рисунок 2.6 – Inception блок з згортками 1x1

У таблиці 2.3 відображена певна кількість глибин у кожному фільтрі. Зменшення 3 × 3 і 5 × 5 представляють дві згортки 1 × 1, вставлені перед ними. Вхідна глибина 128, яка зменшується до 96 і 16 перед фільтрами 3 × 3 і 5 × 5. Цифри, виділені жирним шрифтом, являють собою глибини згорткових виходів, які складаються до 256 (= 64 + 128 + 32 + 32).

Таблиця 2.3 Розмір зображення після різних операцій

Тип операції	Результуючий розмір
Згортка 1 х 1	64
Згортка 3 х 3 (зменшення)	96
Згортка 3 х 3	128
Згортка 5 х 5 (зменшення)	16
Згортка 5 х 5	32
Пулінг	32
Об’єднання	256

Іншою вражаючою перевагою початкової моделі є її гнучкість, яка пристосована до різних шарів. У нижніх шарах, ми більше зосереджуємося на локальних регіонах вхідного зображення, подібно до крайової інформації, більше точним - фільтри меншого розміру, такі як 1 × 1 і 3 × 3. З іншого боку, у вищих шарах, де функції як текстура більш схильні до захоплення, ми, ймовірно, повинні використовувати фільтри більшого розміру, такі як 5 × 5. Використовуючи такий блок моделі, ми можемо регулювати кількість глибин між різними фільтрами відповідно до наших вимог на різних рівнях шару.

2.2.3.5 Оцінка якості

Для експериментів, як наші метрики оцінки регресійних втрат, ми використовуємо середню квадратичну помилку (MSE) між вектором координат основної точки і передбаченою. Де y = {y1, · · ·, yi, · · ·, yn} – розмітка, та yˆ = {yˆ1, · · ·, yˆi, · · ·, yˆn} – передбачення моделі, втрата MSE визначається як середнє значення квадрат всієї помилки, яку можна представити як:

$$loss_{\text{MSE}} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad (2.1)$$

У нашому випадку координати ключових точок знаходяться в діапазоні [0, 95] × [0, 95]. Щоб краще відповідати нормальному розподілу ініціалізації (μ = 0) ваг в мережі, ми перемалюємоці координати до [−1, 1] × [−1, 1].

2.2.4 Просторово-часові згортки

Цей розділ спрямований на представлення моделей TempCNN. По-перше, коротко розглядається теорія нейронних мереж і CNN. Далі подається принцип просторово-часової згортки згортки. Нарешті, ми вводимо загальну форму архітектури TempCNN.

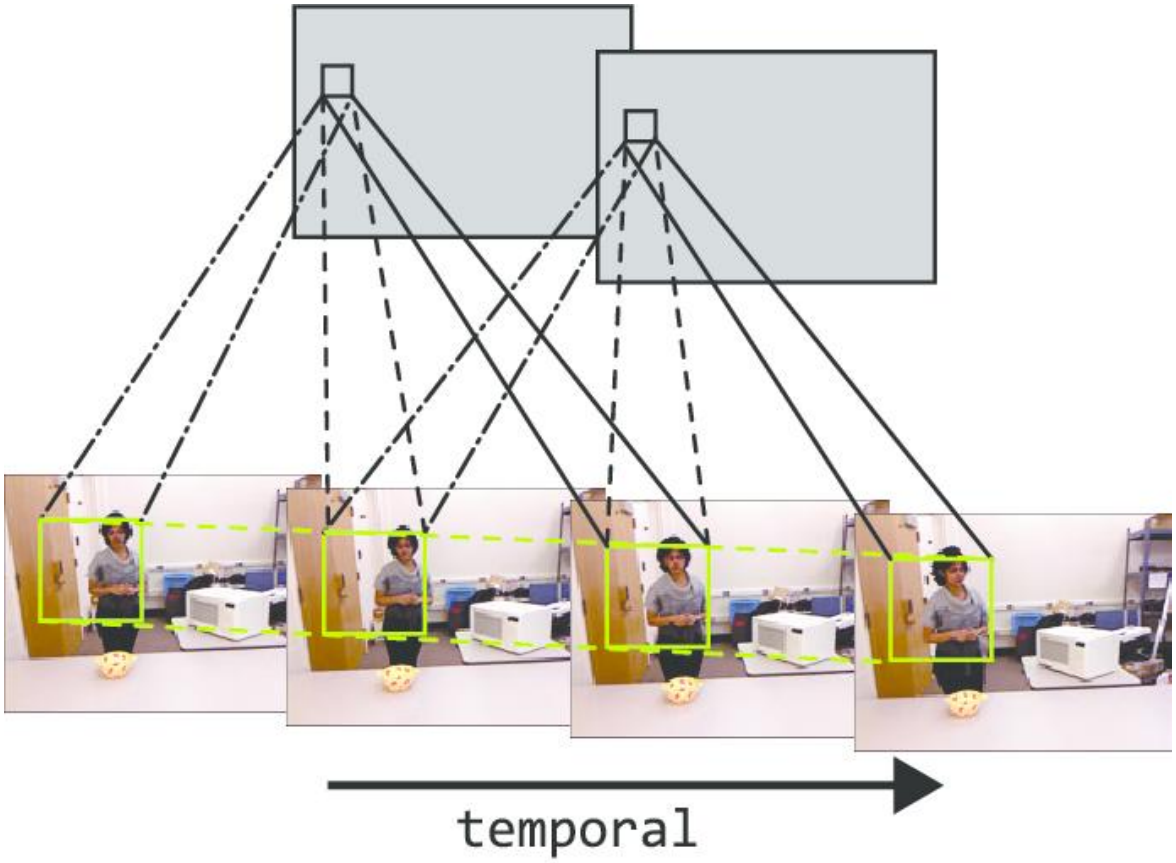


Рисунок 2.7 – Приклад просторово-часової згортки

Глибокі навчальні мережі засновані на конкатенації різних шарів, де кожен шар приймає виходи попереднього шару як входи. На малюнку 2.7 показаний приклад повністю пов'язаної мережі, де нейрони зеленого кольору являють собою вхід, нейрони синього кольору належать до прихованих шарів, а нейрони у червоному - виходи. Як зображено, кожен шар складається з певної кількості одиниць, а саме нейронів. Розмір вхідного шару залежить від розмірності екземплярів, тоді як вихідний шар складається з одиниць C для завдання класифікації класів C. Кількість прихованих шарів та їх кількість потрібно обирати практикуючим

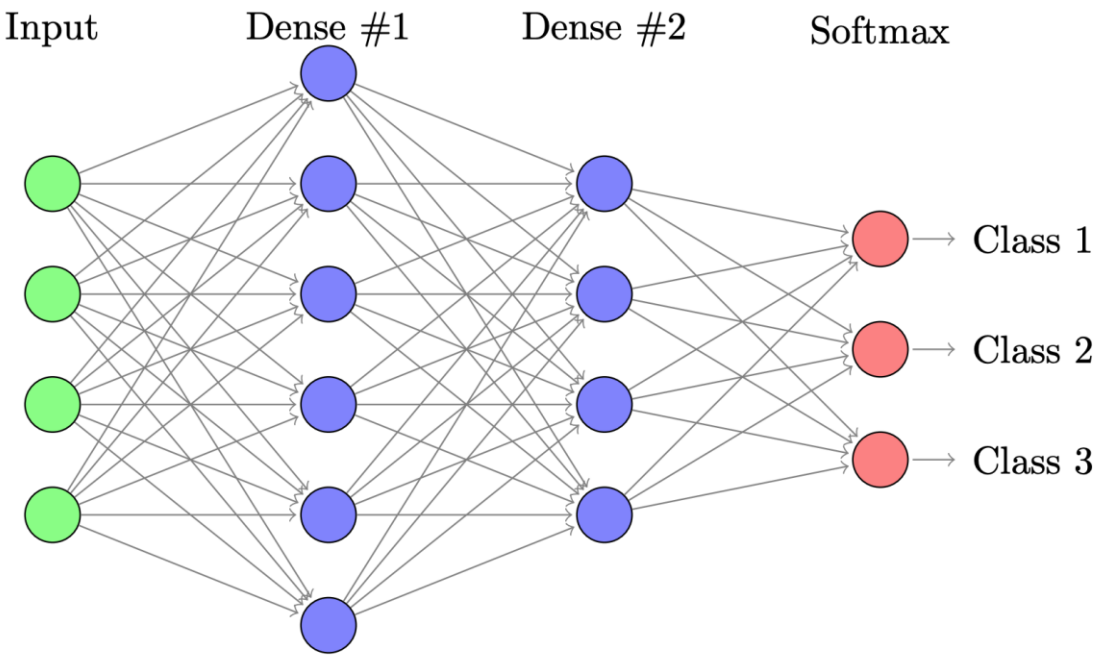


Рисунок 2.8 – Приклад повнозв’язної мережі

Були запропоновані згорткові шари для обмеження кількості ваг, які мережа повинна вивчати, намагаючись зробити більшість структурних розмірів у даних - напр. просторовий, часовий або спектральний [12]. Вони застосовують фільтр згортки до виходу попереднього шару. Навпаки, до щільного шару (тобто повністю пов'язаного шару), де вихід нейрона є єдиним числом, що відображає активації, вихідний сигнал згорткового шару є, таким чином, набором активацій. Наприклад, якщо вхід є одновимірним тимчасовим рядком, то висновок буде тимчасовим рядком, де кожна точка в серії є результатом фільтра згортки. На рис. 2.8 показано застосування градієнтного фільтра [−1 1 0 1 1] на часових рядах, зображених синім. Висновок зображений червоним кольором. Він приймає високі позитивні значення, де виявляється збільшення сигналу, і низькі негативні значення, де відбувається зменшення сигналу. Зауважимо, що так звана згортка технічно є взаємною кореляцією. У порівнянні з щільними шарами, які застосовують різні ваги до різних входів, згорткові шари відрізняються тим, що вони поділяють свої параметри: однакова лінійна комбінація застосовується шляхом зсуву її по всьому входу. Це різко зменшує кількість ваг у шарі, припускаючи, що однакова згортка може бути

корисною в різних частинах часових рядів. Тому кількість тренувальних параметрів залежить тільки від розміру фільтра згортки f і від кількості одиниць n, але не від розміру вхідного сигналу. І навпаки, розмір виходу буде залежати від розміру вхідного сигналу, а також від двох інших гіперпараметрів - кроку та частки. Крок являє собою інтервал між двома центрами згортки. Контроль заповнення для розміру після застосування згортки шляхом додавання значень (зазвичай нулів) на кордонах входу.

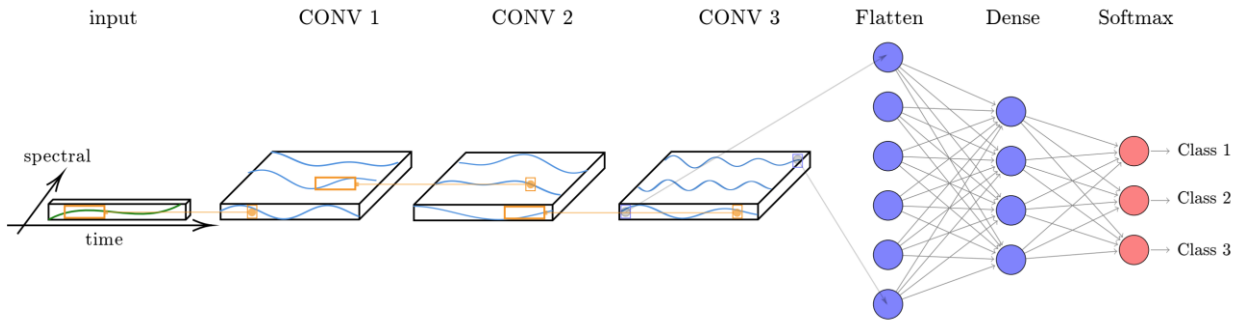


Рисунок 2.9 – мережа з просторово-часовими згортками

2.2.5 Рекурентра нейронна мережа

Крім того, ми навчимо і оцінюємо модель на основі RNN - рекурентна нейронна мережа - це штучна нейронна мережа, де зв'язки між одиницями формують спрямований граф уздовж послідовності.

Ідея RNN полягає в тому, щоб пристосувати нейронні мережі до послідовного характеру даних. У традиційній нейронній мережі передбачається, що входи і виходи є незалежними один від одного, але для багатьох завдань це не є реальним випадком. Якщо вам потрібно зробити прогнози для наступного слова в реченні, ви повинні знати, які слова прийшли до нього.

RNNs називаються рекуррентними, тому що вони виконують одні і тіж ваги для кожного елемента послідовності, при цьому вихідні дані залежать від попередніх обчислень.

Інший підхід до інтерпретації рекурентних нейронних мереж полягає в тому, що нейронна мережа має комірку пам'яті, яка захоплює контекстну інформацію. У теорії рекурентні нейронні мережі можуть захоплювати контекст послідовностей будь-якої довжини, але на практиці це не є правдою. Типова структура RNN має один прихований рекурентний блок, який відповідає за виконання математичних операцій на вхідних і вихідних матрицях. На наступному малюнку показана класична рекурентна архітектура нейронної мережі:

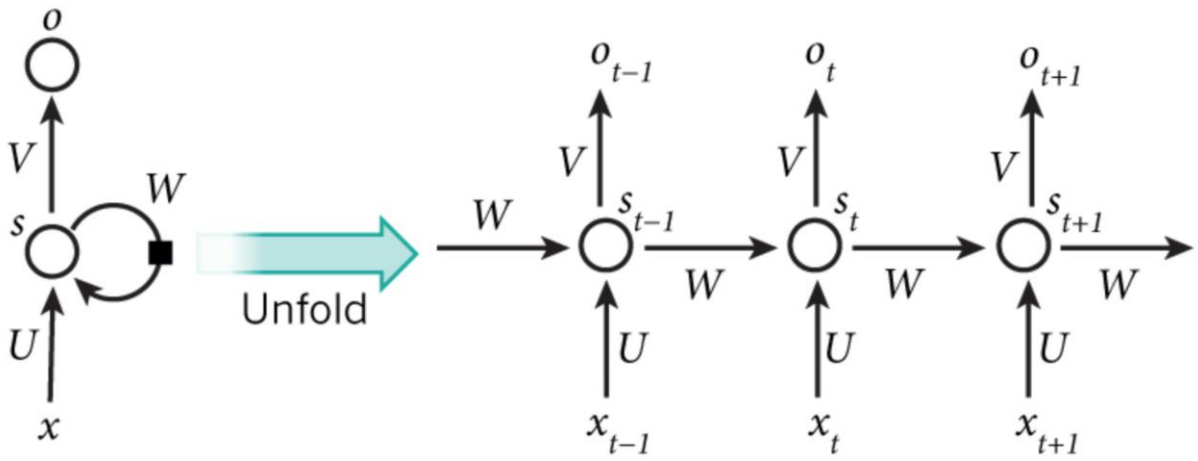


Рисунок 2.10 – Розгорнута схема рекурентної мережі

На малюнку показана розгорнута структура RNN, але насправді це одна клітина з розгорнутою відносно вхідної послідовності. Наприклад, якщо вхідна послідовність має довжину 5, ми повинні розгорнути нейронну мережу в 5-шарову мережу, по одній для кожного входу.

RNN показали високу продуктивність при обробці природної мови через послідовний характер даних. Однак класична RNN-архітектура зазвичай не використовується через проблему з випадаючими градієнтами і проблеми з довгостроковими залежностями. Замість цього використовується модель LSTM, яка вирішує задачі RNN.

2.2.6 LSTM

Термін «довга короткочасна пам'ять» (long-short term memory) походить від наступної міркувань. Прості рекурентні нейронні мережі мають довгострокову пам'ять у вигляді ваг. Ваги змінюються повільно під час навчання, кодуючи загальні знання про дані. Вони також мають короткочасну пам'ять у вигляді ефемерних активацій, які проходять від кожного вузла до послідовних вузлів.

Модель LSTM вводить проміжний тип зберігання через комірку пам'яті. Комірка пам'яті - це складова одиниця, побудована з більш простих вузлів у конкретній схемі зв'язку, з новим включенням мультиплікативних вузлів, представлених на діаграмах літерою P. Всі елементи комірки LSTM описані нижче. Зауважимо, що коли ми використовуємо векторні позначення, ми маємо на увазі значення вузлів у цілому шарі клітин. Наприклад, S являє собою вектор, що містить значення Sc на кожній осередку пам'яті c в шарі. Коли використовується індекс c, він призначений для індексування окремої комірки пам'яті.

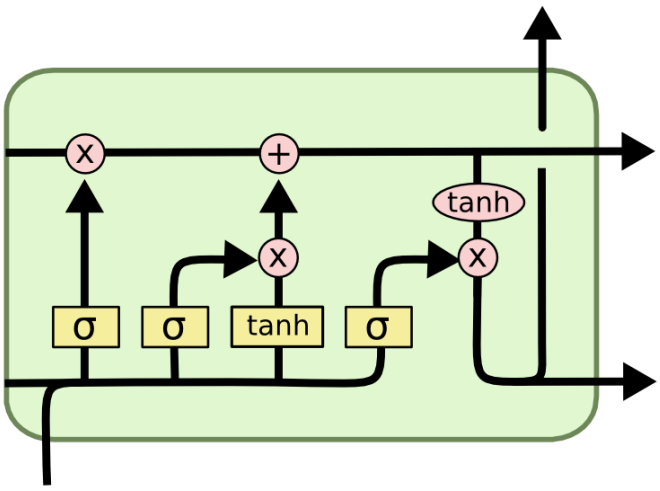


Рисунок 2.11 – Блок LSTM мережі

Вхідний вузол: цей блок, позначений Gc, є вузлом, який приймає активацію стандартним способом з вхідного шару x (t) на поточному етапі часу

і (уздовж повторюваних ребер) з прихованого шару на попередньому кроці h (t−1). Як правило, підсумкований зважений вхід виконується через активацію tanh Функція, хоча в оригінальній докладі LSTM, функція активації є сигмоподібною. На рисунку 2.11 зображений вхідний вузол LSTM.

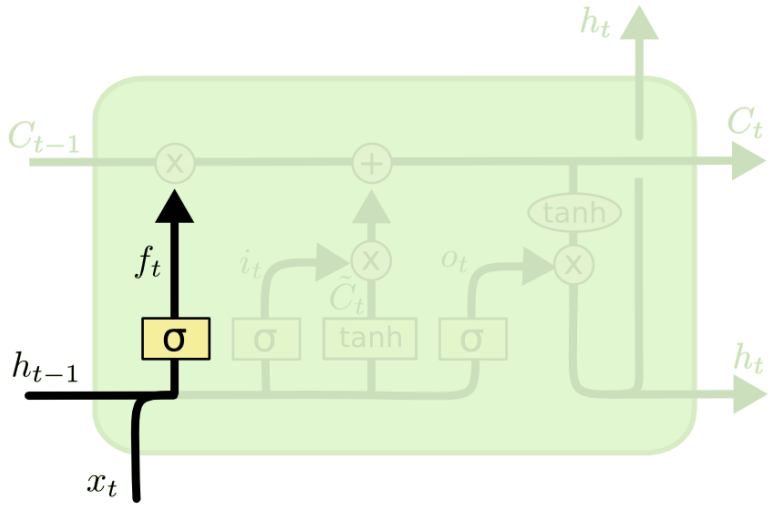


Рисунок 2.12 – Вхідний вузол LSTM

Вхідні ворота: вхідні ворота є відмінною рисою підходу LSTM. Ворота - це сигмоїдальна одиниця, яка, як і вхідний вузол, бере активацію з поточної точки x (t) даних, а також з прихованого шару на попередньому кроці часу. Ворота є так званим, тому що його значення використовується для множення значення іншого вузла. Це ворота в тому сенсі, що якщо його значення дорівнює нулю, то відтік з іншого вузла обрізається. Якщо значення ворота одиниця, весь потік проходить через них. Значення вхідного затвора Ic помножує значення вхідного вузла.

Внутрішній стан: в основі кожної комірки пам'яті лежить вузол Sc з лінійною активацією, який в оригіналі згадується як «внутрішній стан» клітини. Внутрішній стан sc має самоз'єднаний рекурентний край з фіксованою одиницею ваги. Оскільки цей край охоплює сусідні кроки часу з постійною вагою, помилка може протікати через кроки часу без зникнення або вибуху. Цей край часто називають змінна постійної помилки. У векторному позначенні

оновленням для внутрішнього стану є s (t) = g (t) * (t) + s (t − 1), де * матричне множення. На рисунку 2.12 зображено внутрішній стан нейронної мережі.

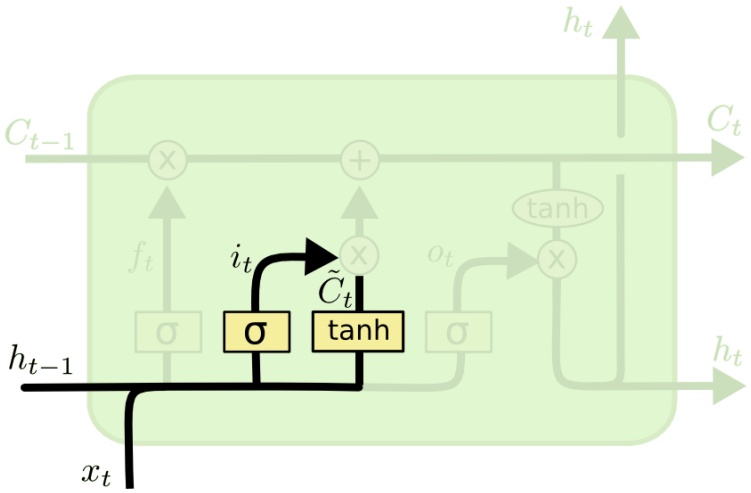


Рисунок 2.13 – Внутрішній стан LSTM

Ворота забування: ці ворота Fc були введені Gers et al. [15]. Вони забезпечують спосіб, за допомогою якого мережа може навчитися очищати вміст внутрішнього стану. Це особливо корисно у безперервно працюючих мережах. З воротами забування рівняння для обчислення внутрішнього стану на передньому проході є

$$s^{(t)} = g^{(t)} \odot i^{(t)} + f^{(t)} \odot s^{(t-1)} \qquad (2.2)$$

Вихідні ворота: Значення Vc, яке в кінцевому підсумку виробляється осередком пам'яті, є значенням внутрішнього стану Sc, помноженого на значення виходу Oc. Звичайно, що внутрішній стан спочатку виконується через функцію активації tanh, оскільки це дає виходу кожної комірки той же динамічний діапазон, що і звичайний прихований блок tanh. Проте в інших дослідженнях нейронних мереж легше піддаються випрямленим лінійним блокам, які мають більший динамічний діапазон. Таким чином, здається вірогідним, що нелінійна функція на внутрішньому стані може бути опущена.

2.3 Конструювання програмного забезпечення

У цьому розділі описується мережа яка навчає передбачати символи у реченнях, які говорять з відеоролика розмовляючої людини без звуку.

Ми моделюємо кожен символ Yi у вихідній послідовності символів $y = (y_1, y_2, \dots, y_l)$ як умовний розподіл попередніх символів y < i, вхідна послідовність зображення $x^v = (x_1^v, x_2^v, \dots, x_n^v)$ для читання губ. Отже, моделюємо розподіл ймовірності виходу як:

$$P(y|x^v, x^a) = \prod_i P(y_i|x^v, x^a, y_{<i}) \tag{2.3}$$

Кадри, вилучені з відеопослідовності, обробляються малими наборами в межах Convolutional Neural Network (CNN), [23], тоді як LSTM працює на виході CNN послідовно для створення вихідних символів. Точніше, послідовність з 10 кадрами групується разом у блоки (ширина x висота x 10), довжина послідовності може змінюватися, але послідовний характер цих кадрів створює просторово-часовий CNN.

Потім вихід цього LSTM, який називається Gated Recurrent Unit (GRU), [24] обробляється багатошаровим персептроном (MLP) для виведення значень для різних символів, отриманих з просторово-часової CNN. Далі CTC забезпечує остаточну обробку результатів послідовності, щоб зробити її більш зрозумілою з точки зору виходів, тобто слів і речень. Такий підхід дозволяє передавати інформацію через періоди часу, що містять як слова, так і, в кінцевому рахунку, речення, підвищуючи точність прогнозів мережі.

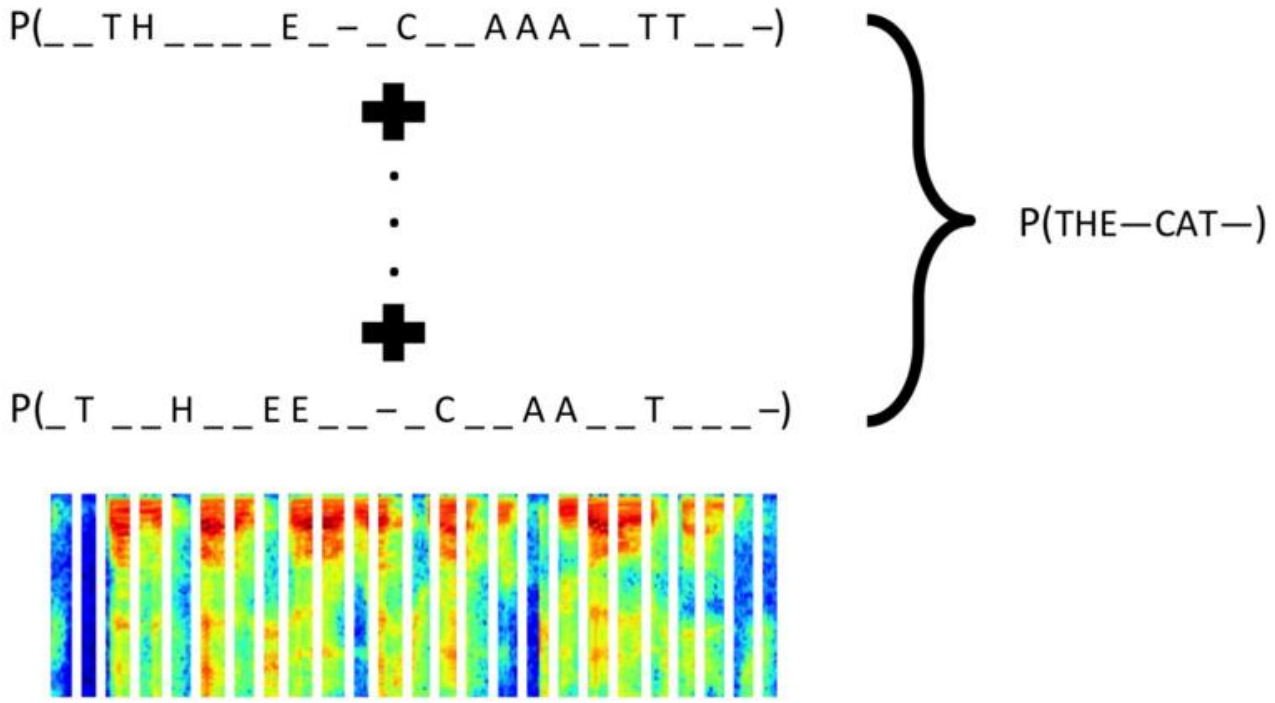


Рисунок 2.14 – Приклад CTC виходу

Прогнозуючи символи алфавіту та додатковий символ "_" (простір), можна створити прогноз слова, видаляючи повторювані букви і порожні простори, як це можна бачити на рисунку 2.14 для класифікації слова «the cat». Практично це означає, що витягнуті вимови, варіації акценту і часів, а також паузи між складами і словами все ще можуть виробляти послідовні прогнози, використовуючи CTC для виходів.

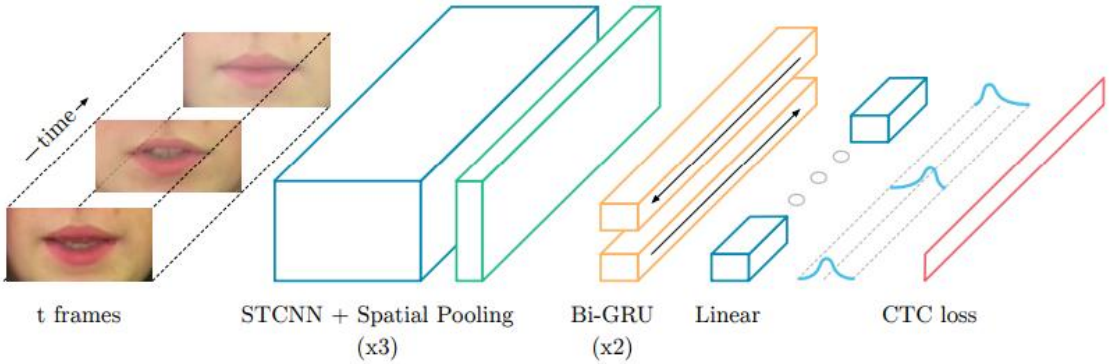
Рисунок 2.15 – Приклад активації CTC на обличчі людини

CTC є функцією для вирівнювання виводу і функцією корекції втрат, заснованої на цьому вирівнюванні, і не залежить від CNN і LSTM. Можна також сприймати CTC як софтмакс через перетворення вихідного сигналу мережі (у нашому випадку, символів) у очікуваний результат (наприклад,

розподіл ймовірностей або у цьому випадку слова та пропозиції) . CTC робить можливим підключення одного символу до рівня слова.

2.3.1 Архітектура нейронної мережі

Відмінною рисою цього методу є те, що вихідні мітки не обумовлені один на одного. Наприклад, буква "a" у "cat" не обумовлена на "c" або "t". Замість цього це відношення витягується трьома просторово-часовими згортками, за якими слідують два GRU, які обробляють задане число вхідних зображень. Вихідні дані для GRU потім проходять через MLP для обчислення втрати CTC.



Механізми Attention сприяли добре завдяки останнім успіхам у рамках глибокого навчання; завдяки більш ефективній та розумній обробці даних. Це також дозволяє цим моделям мати більшу інтерпретативність, тобто якщо запитати, чому мережа думає, що певне зображення є собакою, часто важко зрозуміти внутрішні стани мережі, щоб з'ясувати, чому. Attention дозволяє мережі виділяти виразні частини зображення, що використовуються при його прогнозуванні, наприклад морду і загострені вуха. Attention стала такою загальною методикою, що вона породила документи, такі як «увага - це все, що потрібно», яка відмовляється від методів згортки і повторення, повністю для проблеми машинного перекладу.

- Кодер зображення: приймає зображення та кодує їх у глибоке представлення, яке обробляється додатковими модулями.

- Декодер символів: Цей модуль містить інформацію з усіх попередніх модулів. Кожен вищезгаданий кодер перетворює їх відповідну вхідну послідовність в вектор фіксованого розміру стану і послідовності виходів кодера. Декодер символів, який є перетворювачем LSTM, потім зчитує фіксований стан і вектори Attention з обох кодерів і виробляє розподіл ймовірностей по вихідній послідовності символів. Нарешті, вектори уваги зливаються з вихідними станами для створення векторів контексту, які містять інформацію, необхідну для отримання наступного кроку виводу.
- Attention: Дотримуйтесь важливих для кожного конкретного вхідного сигналу / потоку, тобто відео. Без Attention модель отримує помилки в словах мойже у 100% і, здається, забуває вхідний сигнал. Це показує, що механізм подвійної уваги дійсно дозволив цю техніку працювати наскрізь.

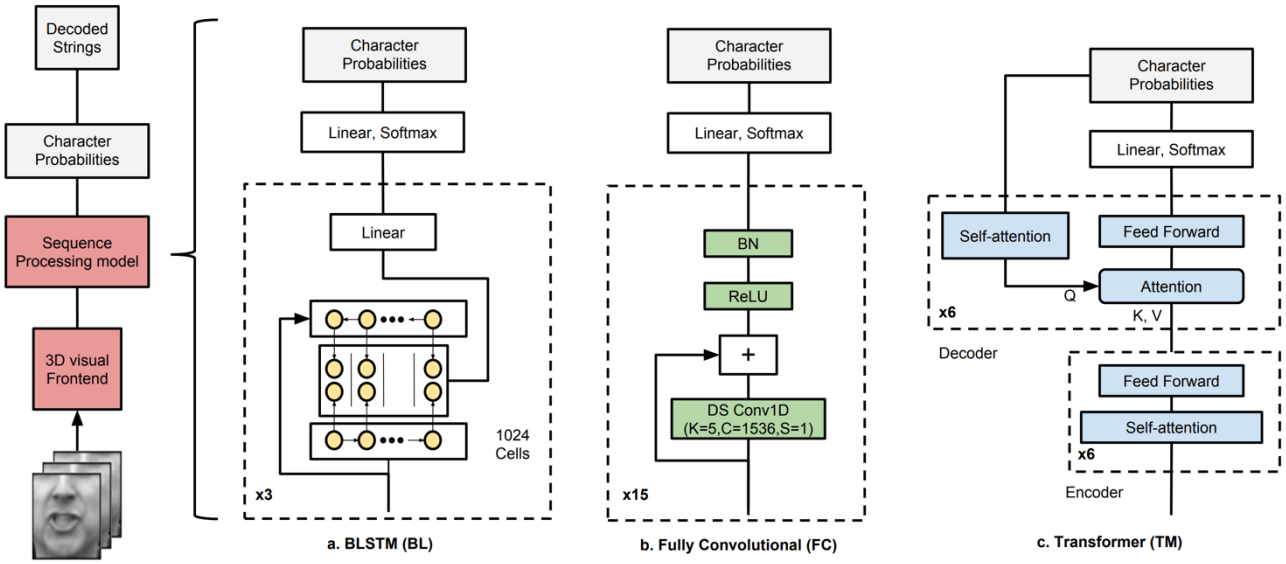
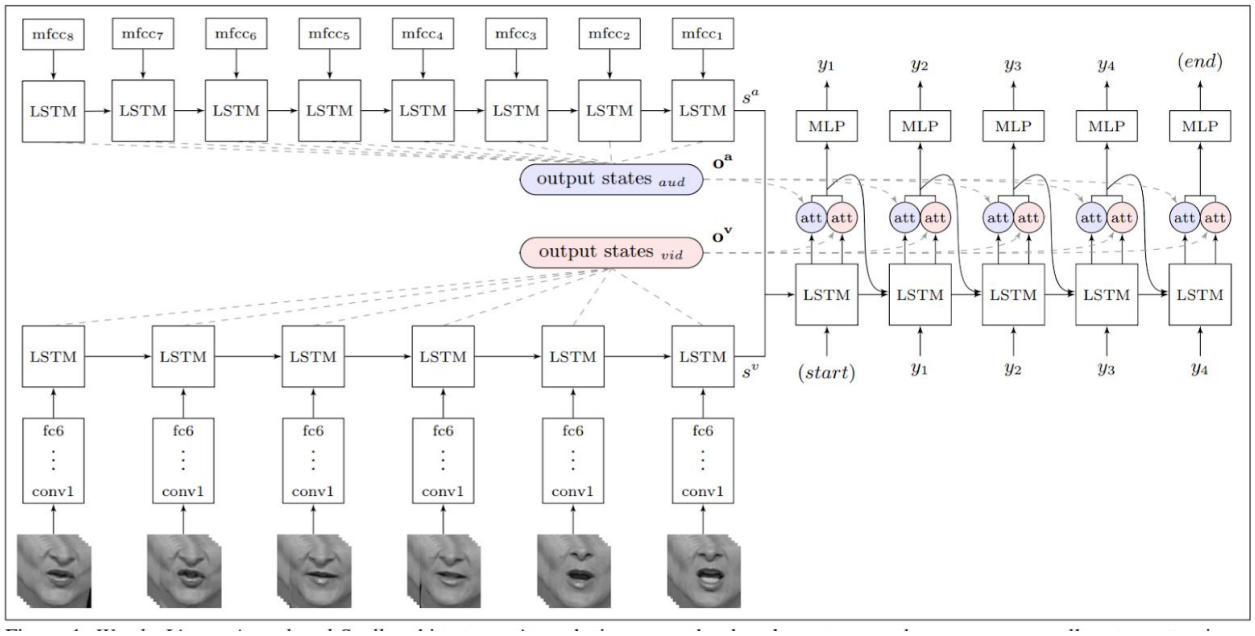


Рисунок 2.16

Кодером являє собою VGG-M, який витягує представлення кадрів, яке буде споживатися LSTM, який генерує вектор стану і вихідний сигнал. Модуль Кодера дивиться на кожен кадр у відео і витягує відповідні функції, які модуль

навчився шукати, тобто певні рухи / положення губ. Це робиться звичайним VGG-M CNN, який виводить представлення ознак для кожного кадру.

Ця послідовність ознак подається в звичайний LSTM, який генерує вектор стану (або стан клітини) і вихідний сигнал. З LSTMs та GRUs є вихід та "стан" введення до наступної LSTM. Вихідні дані - це передбачення символів (або розподіл ймовірностей прогнозованого характеру), а стан - це те, що кодує «минуле», тобто те, що LSTM обчислює / зберігає минуле, яке використовується для прогнозування наступного виходу.



2.3.2 Навчання моделі

Навчання триває три етапи: спочатку тренується візуальний модуль; по-друге, візуальні особливості генеруються для всіх навчальних даних за допомогою модуля Кодера; по-третє, навчається модуль обробки послідовностей.

Для першого етапу ми попередньо тренуємо візуальний модуль на наборі даних на рівні слів (LRW і MVLRS). Вхідні данні перетворюються у відтінки сірого, масштабуються та центрально обрізаються. Ми також виконуємо збільшення даних у вигляді горизонтального перегортання, видалення

випадкових кадрів і випадкових зсувів до ± 5 пікселів у просторовому вимірі та ± 2 кадру у часовому вимірі.

2.3.3 План навчання

Після попередньої підготовки візуального модуля, ми продовжуємо навчання мережі обробки послідовностей. Ми спочатку передаємо всі відео через заздалегідь підготовлений інтерфейс, щоб отримати візуальні особливості кадрів. Потім ми тренуємо моделі послідовностей безпосередньо на ознаках, використовуючи стратегію, яка починається з висловлювань 2-ох слів, потім 2- і 3 слів, потім {2, 3, 4} тощо. Вхідне відео відоме, ми можемо вибрати будь-який безперервний уривок, що міститься в наборі даних, обчислити відповідні індекси в послідовності візуальних ознак і завантажити функції, витягнуті з відеокадрів, що містять висловлювання. Такий підхід допомагає прискорити процедуру навчання. Спочатку ми тренуємо мережу на MV-LRS і частині LRS2 з попередньою підготовкою і, нарешті, точно налаштуємося на набір LRS2. Ми маємо справу з різницею у довжині висловлювання нульовим відступом до максимальної довжини послідовності, яку ми поступово збільшуємо разом з максимальною кількістю слів, що використовуються на кожному кроці навчального плану.

2.3.4 Деталі навчання

Декодер навчається з використанням техніки примусового навчання - ми подаємо розмітку попереднього етапу декодування як вхід до декодера, тоді як під час виведення ми робимо прогноз декодера. Мережа навчається з данних з ймовірністю 0.3 на входах і рекурентних одиницях шарів BLSTM. FC використовує вірогідність виброcу 0.8 після кожного шару нормування. Для BL архітектури ми використовуємо SGD з фіксованим імпульсом 0,9 і швидкістю навчання, починаючи з 0.001 і зменшуючи його кожного разу при плато, до 0.0001. Для Декодера та Енкодера ми використовуємо оптимізатор ADAM [21]

з параметрами за замовчуванням і початковою швидкістю навчання 0.001, зменшуючи його на плато до 0.0001. Всі моделі реалізовані в Pytorch і навчаються на одному GeForce GTX 1080 Ti GPU з 11 Гб пам'яті.

2.3.5 Веб-додаток

Опис процесу обробки відео:

- веб-додаток надсилає відео для обробки;
- серверна частина перевіряє відеофайд на відповідність;
- якщо завантажене відео не відповідає вимогам, то серверна частина повертає помилку;
- веб-додаток відображає відеофайл;
- серверна частина обробляє завантажений відеофайл;
- сервер повертає розпізнаний текст у веб-додаток.

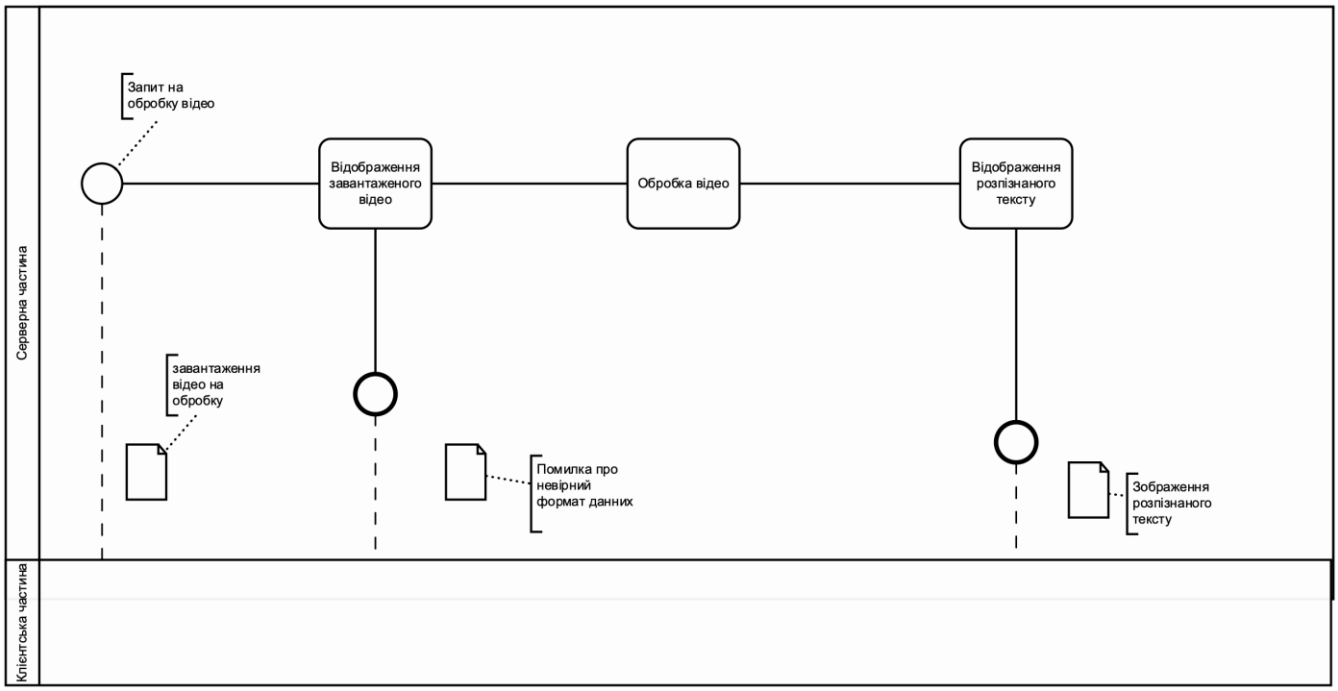


Рисунок 2.17 – Схема обробки відео

2.4 Висновки по розділу

У даному розділі було проаналізовано бізнес-процеси інформаці□ної□ системи та проілюстровано з використанням діаграм BP N. Було розроблено принципову схему клієнт-серверної□ моделі, описано і розроблено Р та описано переваги та недоліки обраного набору засобів розробки. Проаналізовано безпеку користувацьких даних в інформаці□ні системі.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Тестування програмного забезпечення є невід'ємною частиною циклу и□ого розробки, так як воно гарантує відповідність технічному завданню, коректність реалізації□, наявність всіх заявлених можливостеи□ та відсутність помилок.

Всі підходи до тестування можуть бути поділено на ручне або автоматичне. Автоматизоване тестування суттєво спрощує процес розробки для програміста, так як воно гарантує, що и□ого зміни в коді не привели до припинення роботи вже реалізованого функціоналу, дозволяючи впевнено та швидко вносити зміни в кодову базу.

При розробці алгоритму по розпізнаванню промовленого тексту по відеоряду обличчя людини, тестування є надзвичаи□но важливим, оскільки розробленни□ алгоритм являється комплексним програмним продуктом та вимагає перевірки усіх часткових випадків. Проте, оскільки розробленни□ алгоритм являється и□мовірнісним, що не можливо гарантувати 100 відмовну и□ого роботу. Проте навіть при таких жорстких обмеження на можливі набори тестів, які можуть бути використанні при тестуванні програмного додатку, и□ого тестування все ж таки не є неможливим. Перш за все існує можливість створення елементарних одиничних тестів, які будуть перевіряти лише певні компоненти програмного забезпечення.

Ключовими типами тестування даного програмного продукту є навантажувальне тестування. Це обумовлено високою обчислювальною складністю розробленого алгоритму деблюрінга.

Навантажувальне тестування в даному випадку дозволить перевірити не тільки швидкодію веб-додатку, а и□ інші компоненти розробленого

програмного забезпечення, що можуть бути чутливими до навантаження на систему.

Також будуть протестовані наступні компоненти розробленого програмного забезпечення:

- тестування часу відповіді на запити;
- тестування API сервера по обробці відео;
- тестування алгоритму обробки відео;
- тестування роботи інтерфеи□су користувача;
- навантажувальне тестування сервера по обробці відео.

Тестування цих функціи□ дозволить повністю перевірити отримании□ програмнии□ продукт на відповідність функціональним вимогам.

3.2 Опис процесів тестування

Тестування відбуватиметься у режимі чорноі□ коробки. Так як, у даному продукті дуже важлива и□ого невідновна робота для різних наборі вхідних даних, тому необхідно щоб розробленни□ функціонал був протестованни□ як при в позитивних, так і в негативних умовах. Так як велику частину функціоналу продукту складно перевірити через и□мовірнісну поведінку алгоритму та недоцільність и□ого ручного тестування через необхідність перевірки великоі□ кількості прикладів.

Будуть проводитись наступні типи тестів:

- димне тестування роботи API алгоритму;
- димне тестування роботи API саи□ту;
- інтеграціи□не тестування компонентів програмного додатку;
- навантажувальне тестування серверу обробки відео.

Навантажувальне тестування наи□критичнішим компонентом для даного продукту, тому и□ого важливо проводити під час як всіх етапі розробки програмного забезпечення, оскільки існують чіткі вимоги до швидкодіі□ розроблювального програмного продукту. Для проведення навантажувального

буду проведено генерація великого набору даних та апроксимація пікової та медіанної швидкості обробки на даному наборі даних. Даний підхід використовується по тій причині, що найбільш популярніший програмний додаток для проведення навантажувального тестування JMeter не підходить для проведення навантаження з великою кількістю різноманітних відео, які необхідно відправляти через мережу.

3.1 Опис контрольного прикладу

Повний опис процесу тестування та наявних сценаріїв для тестування вказано в додатку В Програма та методика тестування , а в даному розділі розглянуто проведення тестування на прикладі сценарію перевірки коректності розпізнавання промовленого тексту. Даний приклад наведено у таблиці 3.1.

Таблиця 3.1 – Опис контрольного прикладу

Мета тесту	Перевірка коректності розпізнавання промовленого тексту по відеоряду обличчя людини
Початковий стан	Система готова до обробки вхідних відео. Відео video.mp4 завантажено в файлову систему
Вхідні дані	Відео video.mp4
Опис процесу тестування	Запустити скрипт по обробці відео та обрати файл video.mp4 як вхідне зображення.
Кінцевий результат	Після проведення даного тесту стан системи не повинен змінитись
Очікуваний результата	У інтерфесу системи з'являється поле де виведено розпізнаний текст.

3.2 Висновок до розділу

В даному розділі було описано підходи та процеси тестування розробленого програмного забезпечення. Описано основні компоненти, які мають бути протестовані для впевнення, що фінальниии продукт відповідає усім вимогам якості.

Описано в розгорнутому вигляді контрольниии приклад для перевірки коректності роботи розробленого програмного забезпечення.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Для розгортання розробленого програмного забезпечення необхідно виконати наступні кроки:

1. Інсталювати Python 3.6.5 або вище в системі;
2. Інсталювати залежності виконавши команду «pip install -r requirements.txt»;
3. Запустити сервер виконавши команду «python3 app.py».

4.1 Робота з програмним забезпеченням

Розроблене програмне забезпечення передбачає наступнии□ сценаріи□ використання:

1. Відкриття додатку в браузері;
2. Завантаження відео на обробку програмним забезпеченням;
3. Натискання кнопки «Analyze»;
4. Отримати результуючий текст з веб-сторінки.

4.2 Висновок до розділу

У даному розділі продемонстровано схему розгортання системи та описано сценаріи□ використання розробленоі□ інформаціи□ноі□ системи.

ВИСНОВКИ

У ході дипломного проекту було проведено аналіз проблематики читання по губам за допомогою методів глибокого навчання: описано математичні моделі, проаналізовано існуючі підходи, описано і□хні переваги та недоліки.

Було спроектовано і розроблено новии□ підхід до розпізнавання промовленного тексту, що дозволило покращати уже існуючии□ алгоритмі. На основі розробленого алгоритму було розроблено веб-додаток для візуального представлення розробленого алгоритму.

Також було проведено аналіз якості інформаціи□ноі□ системи, проведено тестування з подальшим усуненням усіх знаи□дені недоліків.

Розроблено необхідну проектну документацію, схеми процесів застосунку, варіантів використань та керівництво для користувачів.

ПЕРЕЛІК ПОСИЛАНЬ

1) Ahmed Rekik, Achraf BenHamadou, and Walid Mahdi. A new visual speech recognition approach for RGB-D cameras. In Image Analysis and Recognition - 11th International Conference, ICIAR 2014, Vilamoura, Portugal, October 22- 24, 2014, Proceedings, Part II, pages 21–28, 2014.
2) Yannis M. Assael, Brendan Shillingford, Shimon White son, and Nando de Freitas. Lipnet: Sentence-level lipreading. CoRR, abs/1611.01599, 2016.
3) Yuru Pei, Tae-Kyun Kim, and Hongbin Zha. Unsupervised random forest manifold alignment for lipreading. In The IEEE International Conference on Computer Vision (ICCV), December 2013.
4) Rekik A., Ben-Hamadou A., Mahdi W. (2015) Human Machine Interaction via Visual Speech Spotting. In: Battiato S., Blanc-Talon J., Gallo G., Philips W., Popescu D., Scheunders P. (eds) Advanced Concepts for Intelligent Vision Systems. Lecture Notes in Computer Science, vol 9386. Springer, Cham
5) Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 25, pages 1097–1105. Curran Associates, Inc., 2012.
6) C. Sui, R. Togneri, and M. Bennamoun, “Extracting deep bottleneck features for visual speech recognition,” in ICASSP, 2015, pp. 1518–1522. [6] S. Petridis and M. Pantic, “Deep complementary bottleneck features for visual speech recognition,” in IEEE ICASSP, 2016, pp. 2304–2308.
7) Y. Li, Y. Takashima, T. Takiguchi, and Y. Ariki, “Lip reading using a dynamic feature of lip images and convolutional neural networks,” in IEEE/ACIS Intl. Conf. on Computer and Information Science, 2016, pp. 1–6.
8) K. Noda, Y. Yamaguchi, K. Nakadai, H. G. Okuno, and T. Ogata, “Audio-visual speech recognition using deep learning,” Applied Intelligence, vol. 42, no. 4, pp. 722– 737, 2015.

9) Ayaz A. Shaikh, Dinesh K. Kumar, Wai C. Yau, M. Z. Che Azemin, and Jayavardhana Gubbi. Lip reading using optical flow and support vector machines. 2010 3rd International Congress on Image and Signal Processing, 1:327–330, 2010.

10) Koller, O., Ney, H., Bowden, R.: Deep learning of mouth shapes for sign language. In: Proceedings of the IEEE International Conference on Computer Vision Workshops. pp. 85–91 (2015)

11) Graves, A., Fernandez, S., Gomez, F., and Schmidhuber, ´ J. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In ICML, Pittsburgh, USA, 2006.

12) M. Wand, J. Koutnik, and J. Schmidhuber. Lipreading with long short-term memory. In IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 6115–6119, 2016.

13) J. S. Chung and A. Zisserman. Lip reading in the wild. In Asian Conference on Computer Vision, 2016.

14) Ravi Garg, Vijay Kumar B. G, and Ian D. Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. CoRR, abs/1603.04992, 2016.

15) Zhou, Z., Hong, X., Zhao. A compact representation of visual speech data using latent variables. IEEE transactions on pattern analysis and machine intelligence 36(1), 1–1 (2014)

16) J. S. Chung and A. Zisserman, “Learning to lip read words by watching videos,” CVIU, 2018. 1

17) Y. M. Assael, B. Shillingford, S. Whiteson, and N. de Freitas, “Lipnet: Sentence-level lipreading,” arXiv preprint arXiv:1611.01599, 2016. 1, 2, 3

18) J. S. Chung and A. Zisserman, “Lip reading in profile,” in Proc. BMVC., 2017. 1, 3

19) T. Stafylakis and G. Tzimiropoulos, “Combining residual networks with lstms for lipreading,” in Interspeech, 2017. 1, 2, 3, 6

20) S. Petridis, T. Stafylakis, P. Ma, F. Cai, G. Tzimiropoulos, and M. Pantic, “End-to-end audiovisual speech recognition,” CoRR, vol. abs/1802.06424, 2018. 1

21) S. Bai, J. Z. Kolter, and V. Koltun, “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling,” arXiv preprint arXiv:1803.01271, 2018. 1 [19] J. Gehring, M. Auli, D. Grangier, and Y. Dauphin, “A convolutional encoder model for neural machine translation,” in ACL, 2017. 1

22) J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, “Convolutional sequence to sequence learning,” in ICML, 2017.

23) N. Kalchbrenner, L. Espeholt, K. Simonyan, A. van den Oord, A. Graves, and K. Kavukcuoglu, “Neural machine translation in linear time,” CoRR, vol. abs/1610.10099, 2016.