Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Факультет інформатики
Кафедра інформатики

Магістерська робота

Освітній ступінь: магістр

На тему: «ЗАСТОСУВАННЯ ГАУСОВОГО ПРОЦЕСУ В НЕЙРОННІЙ МЕРЕЖІ ДЛЯ РОЗПІЗНАВАННЯ МОВЛЕННЯ»

Виконала: студентка 2 року навчання
Спеціальності
122 Комп’ютерні науки
Пирогова Єлизавета Максимівна
Керівник Гороховський С.С.
кандидат фіз.-мат. наук

Рецензент М. М. Глибовець
Магістерська робота захищена
з оцінкою_____
Секретар ЕК С.А. Мелещенко
«___» _____2022

Київ 2022

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
Зав.кафедри інформатики,
к.ф.-м.н.
_____С. С. Гороховський
(підпис)
„____”_____2022 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на магістерську роботу

студентки 2 р.н. магістерської програми Комп’ютерні науки
Пирогової Єлизаветі Максимівні
Застосування Гаусового процесу в нейронній мережі для розпізнавання мовлення

Зміст текстової частини до магістерської роботи:

Зміст
Анотація
Вступ
1 Теоретична частина
2 Застосування Гаусового процесу в нейронній мережі для розпізнавання мовлення
3 Опис реалізації алгоритму та порівняння з іншими методами
Висновки
Список літератури
Додатки

Дата видачі „___” листопада 2021 р.

Керівник (підпис)

Завдання отримав (підпис)

Тема: “Застосування Гаусового процесу в нейронній мережі для розпізнавання мовлення”

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на дипломну роботу.	02.11.2021	
2.	Огляд технічної літератури за темою роботи.	25.11.2021	
3.	Виконати аналіз існуючих методів …	25.11.2021	
4.	Програмування розробленого алгоритму	15.01.2021	
5.	Застосування розробленого алгоритму в нейронної мережі для розпізнавання мовлення	15.02.2021	
6.	Виконання порівняльного аналізу результатів, отриманих за допомогою розробленого алгоритму на основі нейромережі, та результатів, отриманих за допомогою використання даного алгоритму.	15.04.2022	
7.	Написання пояснювальної роботи.	01.05.2022	
8.	Створення слайдів для доповіді та написання доповіді.	14.06.2011	
8.	Аналіз отриманих результатів з керівником, написання доповіді та попередній захист магістерської роботи.	15.06.2022	
10.	Корегування роботи за результатами попереднього захисту.	18.06.2022	
11.	Остаточне оформлення пояснювальної роботи та слайдів.	25.06.2022	
12.	Захист магістерської роботи (проекту)	06.07.2022	

Студентка **Пирогова Єлизавета Максимівна**

Керівник **Гороховський Семен Самуїлович**

“_____” _____

АНОТАЦІЯ

У нашому світі мова є природною формою спілкування, саме тому реалізація та розробка додатків на основі аналізу мовної інформації є перспективним напрямком розвитку у галузі інтелектуальних систем управління.

Сама задача розпізнавання мовлення є доволі комплексним завданням, адже чіпляє такі галузі науки: лінгвістику, розпізнавання образів та цифрову обробку сигналів.

Використання мовлення для спілкування з автоматизованими системами управління, комп'ютерами чи навіть роботами відкриває великі перспективи у майбутньому, такі як:

- забезпечення голосовим інтерфейсом людей із вадами опорно-рухового чи зорового апарату;
- простота зв'язку з системою;
- можливість користувача працювати в умовах перевантаженості тактильно-зорового каналу.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ГНМ – глибока нейронна мережа

ГП – Гаусовий процес

ГПНМ – нейронна мережа з Гаусовим процесом

ОПМ – обробка природнього мовлення

ПММ – прихована Марківська модель

МКК – мелчастотні кепстральні коефіцієнти

ГММ – модель суміші Гауса

АМО – алгоритм максимізації очікування

МОП – максимальна оцінка правдоподібності

АОК – алгоритм основного компонента

АГКЯ – аналіз головних компонентів ядра

ВСТУП

З кожною хвилиною відбувається стрімкий ріст технологій у світі. Все відбувається настільки швидко, що деякі методи та реалізації, які до сьогоднішнього моменту здавалися доволі прогресивними, вже завтра можуть вважатися застарілими. Вже нікого не здивуєш тим, що інформаційні технології та комп'ютерне забезпечення набирають колосальних обертів, навіть іноді виникає думка, що немає вже куди зростати. Але якщо повернутися у час виникнення перших обчислювальних машин, то вже тоді поставало запитання про розвиток взаємодії людини з комп'ютерною технікою. Довгий період часу це було доступно лише вузьким фахівцям, які мали змогу "спілкуватися" з комп'ютером через програміста. Це тривало доти, доки не з'явився діалоговий інтерфейс, після чого користувач вже міг особисто вводити команди через клавіатуру та отримувати змістовні відповіді. Поява графічних інтерфейсів усунула знання будь-яких команд, що призвело до популяризації персональних комп'ютерів.

Однак люди щосили намагалися знайти більш загальний та природній спосіб взаємодії з технікою. За часів перфокарт, у науково-фантастичних романах людина розмовляла з комп'ютером на рівних. Саме тоді відбулися відбулися перші кроки до реалізації мовного інтерфейсу.

Якщо взяти до уваги показники сучасних систем розпізнавання мовлення та порівняти їх з показниками систем ще на початку розвитку, то можна помітити, що великого прогресу за останні кілька десятиліть не

відбулося. Це розділило експертів на два табори: хтось ставить під сумнів у можливості реалізації мовного інтерфейсу у найближчому майбутньому; а хтось стверджує, що задача фактично вирішена. В більшості експерти сходяться на думці, що для розвитку розпізнавання мовлення потрібен деякий час.

На даному етапі одними із пріоритетних напрямків досліджень в інформаційних технологіях є зберігання, обробка та передача мультимедійних даних. Досі комп'ютер так і не може замінити людину у задачах аналізу мультимедійних даних. Саме основним завданням обробки даних є задача розпізнавання та аналізу мовлення людини.

Задача аналізу мовлення має доволі широкий список завдань, які можна розділити на три підгрупи: ідентифікація, класифікація та діагностика. Кожна з цих підгруп відповідає за вирішення певних питань, які постають перед експертами, проте за останні роки більша частина з цих питань досягнула значного прогресу. Наприклад, алгоритм ідентифікації доволі часто використовують у криміналістичних процедурах.

Завдання розпізнавання мовлення і досі зберігає свою актуальність, адже область рішень, які можна застосовувати досить обширна: автовідповідачі, голосове керування приладами, переведення людської мови в текст, системи передачі мовного сигналу, системи пошуку інформації та інше.

Більшість питань у житті людства вирішується спілкуванням, тобто мовою. Саме тому дослідження в галузі розпізнавання мовлення є перспективним розвитком інтелектуальних систем управління.

РОЗДІЛ 1. ТЕОРЕТИЧНА ЧАСТИНА

У першому розділі описується теоретична частина щодо розпізнавання мовлення. Розділ структурований наступним чином. У частині розділу 1.1 описується загальне поняття розпізнавання мовлення. В наступній частині описуються вже існуючі алгоритми для розпізнавання мовлення та їх недоліки.

1.1 Поняття розпізнавання мовлення

Коли ви телефонуєте у велику компанії, зазвичай чуєте записаний голос, який з великою ввічливістю просить вас зробити певні кроки аби встановити зв’язок з оператором компанії. Це штучний інтелект, що використовує автоматичну систему обробки виклика без необхідності в телефонному операторі. Штучний інтелект включає в себе дві основні ідеї. По-перше, це включає в себе вивчення людських розумових процесів. По-друге, він займається наданням цих процесів за допомогою створених машин (комп’ютерів, роботів та інше). Штучний інтелект – це поведінка машини, в разі якої, якщо б вона була людиною, то називалась би розумною. Це робить самі машини розумнішими та кориснішими, аніж природній інтелект.

Обробка природнього мовлення (ОПМ) відноситься до методів штучного інтелекту для спілкування з комп’ютерами на “материнській” мові. Основною ціллю програми ОПМ – це зрозуміти та ініціалізувати дію. Вхідні слова скануються та зіставляються з внутрішніми відомими словами. Визначення ключового слова приводить до певних дій. Таким чином, можна спілкуватися з комп’ютером на вашій рідній мові. Одним із основних застосувань штучного інтелекту є система розпізнавання мовлення, що дозволяє користувачу одночасно виконувати інші завдання.

Процес розпізнавання мовлення виконується програмним компонентом, відомим як механізм розпізнавання мовлення.

Система розпізнавання мовлення – це тип програмного забезпечення, що дозволяє користувачу перетворювати свої вимовлені слова у письмовий текст в додатку. Програмне забезпечення для розпізнавання мови має обмежений словниковий запас і може ідентифікувати слова та фрази лише тоді, коли вони вимовляються чітко. Більш складне програмне забезпечення може обробляти природнє мовлення, різні акценти та різні мови.

Розпізнавання мовлення використовує широкий спектр досліджень у галузі інформатики, лінгвістики та обчислювальної техніки. Багато сучасних пристроїв та текстових програм мають функції розпізнавання мовлення, щоб спростити використання мобільних пристроїв.

Розпізнавання мови та голосу – це дві різні технології та їх не слід плутати між собою:

- розпізнавання мови використовується для ідентифікації слів на розмовною мовою;
- розпізнавання голосу – це біометрична технологія ідентифікації голосу людини.

Системи розпізнавання мовлення використовують ком’ютерні алгоритми для обробки та інтерпретації вимовлених слів та перетворення їх у текст. Програма перетворює звук, записаний на мікрофон, на письмову мову, яку можуть зрозуміти комп’ютери і люди, виконавши наступні чотири кроки:

1) проаналізувати аудіо;
2) розбити його на частини;
3) оцифрувати його в комп’ютерно-читальний формат;
4) використати алгоритм, щоб зіставити його з найбільш відповідним текстовим представленням.

Програмне забезпечення для розпізнавання мовлення повинно адаптуватися до дуже мінливої та залежної від контексту природи людського мовлення. Програмні алгоритми, що обробляють та перетворюють аудіо в текст, навчаються на різних моделях мовлення, стилях мовлення, мовах, діалектах, наголосах та фразах. Саме програмне забезпечення також відокремлює розмовний звук від фонового шуму, який часто супроводжує сигнал. Щоб задовольнити ці вимоги, системи розпізнавання мовлення використовують два типи моделей:

1) акустичні моделі – представляють співвідношення між мовними одиницями та звуковими сигналами;
2) мовні моделі – звуки поєднуються з послідовністю слів, щоб розрізняти слова, які звучать подібно.

Системи розпізнавання мовлення мають доволі багато застосувань, наприклад:

- мобільні пристрої – смартфони використовують голосові команди для маршрутизації викликів, обробки мовлення в текст, голосового набору та голосового пошуку. На айфонах розпізнавання мовлення може бути забезпечене як з клавіатури, так і за допомогою віртуального помічника ”Siri”;
- освіта – під час навчання мови використовується програмне забезпечення для розпізнавання мовлення. Програмне забезпечення чує мову користувача та пропонує допомогу з вимовою;
- обслуговування клієнтів – автоматичні голосові помічники слухають запити клієнтів та надають корисні ресурси;
- охорона здоров’я – лікарі можуть використовувати програмне забезпечення для розпізнавання мовлення, щоб транскрибувати нотатки в режимі реального часу в медичні записи;

- допомога при інвалідності – використовуючи програмне забезпечення можна перекладати вимовлені слова в текст, використовуючи закриті субтитри, щоб людина з втратою слуху могла зрозуміти, що говорять інші. Розпізнавання мовлення може також дозволити людям з обмеженим використанням рук працювати за комп'ютером використовуючи голосові команди замість введення тексту;
- судова звітність – використовується програмне забезпечення задля стенограми судових засідань, виключаючи потребу в транскрібаторах;
- розпізнавання емоцій – дана технологія може аналізувати певні вокальні характеристики, щоб визначити які емоції відчуває оратор. У поєднанні з аналізом настроїв це може виявити як хтось ставиться до продукту чи послуги;
- спілкування без рук – водії мають змогу використовувати голосове керування для зв'язку без використання рук, керування телефонами, радіо та глобальними системами позиціонування.

Добре створені програми розпізнавання мовлення дозволяють користувачам налаштувати їх відповідно до своїх потреб. Функції, що дозволяють це зробити, включають:

1) Мовне зважування. Ця функція наказує алгоритму приділяти особливу увагу певним словам, тим які вимовляються часто або які є унікальними для розмови чи теми. Наприклад, застосунок можна навчити слухати посилання на конкретні продукти.
2) Акустична підготовка. Програмне забезпечення налаштовує навколишній шум, який заважає розмовному звуку. Застосунки з акустичним навчанням можуть розрізняти стиль мовлення, темп і гучність серед галасу багатьох людей, які розмовляють в громадських місцях.

3) Маркування динаміків. Дана можливість дозволяє програмам позначати окремих учасників і визначати їхній конкретний внесок у розмову.
4) Фільтрація нецензурної лекції. Програмне забезпечення відфільтровує небажані слова та мову.

1.2 Аналіз існуючих алгоритмів розпізнавання мовлення

Розпізнавання мовлення стає все більш популярним та охоплює вже більшу частину технологічної галузі. З появою та зростанням нових технологій, що використовуються за допомогою розпізнавання голосу (наприклад, “Google Home”, “Siri” та інше) не можна не помітити отриману вигоду від прогресу в даній області, але й не секрет що розробка програмного забезпечення нікуди не дінеться. Правда полягає в тому, що майже всі люди можуть говорити швидше, ніж вони можуть друкувати, що доволі зручно, а зручність є рушійною силою інновацій. Програмне забезпечення для розпізнавання мовлення не складно зрозуміти з точки зору користувача. Припустимо, що ви використовуєте передачу тексту на телефоні. Спочатку ви відкриваєте текстове повідомлення та натискаєте на мікрофон. Далі ви “говорите” зі своїм телефоном та можете спостерігати як відбувається введення слів на екрані. Однак, з цього виникає питання: “Як телефон дізнається які саме слова промовляють?”.

Коли телефон приймає мовлення на вхід, то спочатку переводить все в дані. Виконується дана операція, спираючись на хвилі голосу та рівномірно розташовуючи точки вздовж хвилі, які й будуть потім перетворені на дані, а це в свою чергу передається алгоритму.

1.2.1 Прихована модель Маркова

Існує кілька методів для запису мовлення та перетворення його в текст. Одним з найбільш ефективних методів є прихована модель Маркова з алгоритмом визначення кінцевої точки попередньої обробки для видалення небажаного шуму. ПММ потребує додавання інших інструментів для правильної інтерпретацій мовлення. Перед тим як метод ПММ буде використаний для пошуку мовлення, вибірки мовлення витягуються у функції (коефіцієнти) з використання мелчастотних кепстральних коефіцієнтів (МКК). Цей процес поділяє аудіозапис на невеличкі сегменти, які ПММ може використовувати в якості вхідних даних. Сама модель використовує словник вимовляння слів для визначення слова W, вибираючи максимальну ймовірність $P(W\backslash Y)$, де Y – записаний аудіозапис. Ця максимальна ймовірність вираховується з використанням ймовірності $P(W)$, що вираховується за допомогою використання мовної моделі. У словнику присутні ПММ для кожної основної моделі. $P(W\backslash Y)$ – це очікувані акустичні дані, коли W будується зі збережених ПММ у словнику. Ці дані використовуються для визначення максимальної ймовірності шляхом порівняння з $P(W)$, які були розраховані з використанням словника вимови. Використовуючи даний підхід, ПММ може створювати текст із заданої вимови.

Сучасні системи розпізнавання мовлення ґрунтуються на принципах розпізнавання статистичних образів. Вхідний мовний сигнал перетворюється вхідним сигнальним процесором у послідовність акустичних векторів: $Y = y_1, y_2, \dots, y_T$. Кожен із цих векторів є представлення короткострокового спектру мовлення, що охоплює певний період часу. База даних словника складається з послідовності слів: $W = w_1, w_2, \dots, w_n$. Система розпізнавання мови використовується для вибору

найкращої послідовності слів $\widehat{\boldsymbol{W}}$ даного акустичного сигналу $\boldsymbol{Y}$. Для реалізації цієї ідеї використовується правило Баєса:

$$\widehat{W} = \arg\max_W \frac{P(W)*P(Y|W)}{P(Y)} = \arg max_w P(W) * P(Y|W). \quad (1)$$

Згідно формули, для того аби знайти найкращу послідовність слів $\boldsymbol{W}$, що відповідає спостереження $\boldsymbol{Y}$, необхідно вибрати $\boldsymbol{W}$ так, щоб максимізувати аудіозапис $\boldsymbol{P(W)}$ та $\boldsymbol{P(W\backslash Y)}$, в цьому випадку $\boldsymbol{P(Y)}$ фіксоване. В Баєсовій філософії $\boldsymbol{P(W)}$ – це апріорна ймовірність спостереження $\boldsymbol{W}$, яка не залежить від сигналу $\boldsymbol{Y}$, ця ймовірність визначається мовною моделлю. $\boldsymbol{P(W\backslash Y)}$ – це ймовірність спостереження векторної послідовності $\boldsymbol{Y}$ при заданій послідовності слів $\boldsymbol{W}$, ця ймовірність визначається акустичною моделлю.

1.2.2 Модель суміші Гауса

Модель суміші Гауса (ГММ) є однією з найчастіше використовуваних функцій для ГММ і є параметричною функцією щільності ймовірності, представленою у вигляді виваженої суми щільності гаусівських компонентів. ГММ використовує параметри компонента Гауса як базовий класифікатор і отримує тимчасові варіації, в той час як ГММ фіксує особливі варіації. Це дозволяє йому ефективно та дієво обробляти тимчасові послідовності. По суті, послідовність ГММ використовується для аналізу вхідних даних ГММ, що надає йому чутливість до тимчасових змін. ГММ не відповідає вимогам у тому сенсі, що вона «не дозволяє ГММ повною мірою використовувати кореляцію, яка існує між кадрами фонетичного сегменту», оскільки передбачає умовну незалежність. Інший недолік полягає в тому, що оскільки він використовує ймовірність для визначення вхідних даних, він може неправильно вибрати більш поширений варіант.

ГММ широко використовуються як функції розподілу ймовірностей, таких як спектральні характеристики, пов'язані з голосовим трактом, в системах розпізнавання мовця або емоцій. Перевага ГММ полягає в тому, що вони більш підходять та ефективні для розпізнавання мовленнєвих емоцій з використанням спектральної ознаки мови.

ГММ це потужний метод оцінки щільності акустичного вектора ***Y***. Останнім часом у кластерному аналізі широко застосовується підхід, що ґрунтується на змішаній моделі. Ця модель більш гнучка, точна і легко вписується в даний багатовимірний набір даних. У обчисленнях це дуже легко обробляється. Умовна можливість ***P(W\Y)*** вимагає оцінної щільності ***Y*** для свого обчислення. Отже, ГММ може бути реалізований для пошуку найкращої послідовності слів ***W***, що відповідає даному акустичному сигналу ***Y***. Для послідовності $\boldsymbol{Y = y_1, y_2, \dots, y_T}$ кожен вектор

$$y_i = \begin{bmatrix} y_{i1} \\ \vdots \\ y_{id} \end{bmatrix}, (2)$$

є вектором довжини ***d***. На основі ***Y*** моделі акустичних сигналів обраховуються таким чином: багатовимірна функція щільності Гауса визначається виразом

$$f(y; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}} e^{-\frac{1}{2}(y-\mu)^T \Sigma^{-1} (y-\mu)}, (3)$$

де μ це вектор середнього, Σ – це коваріаційна матриця. З цього випливає, що щільність гаусової суміші є виваженою сумою гаусових щільностей $f(y; \mu_i, \Sigma_i)$,

$$f = (y, \mu_1, \dots, \mu_k, \Sigma_1, \dots, \Sigma_k) = \sum_{i=1}^{k} p_i f(y; \mu_i, \Sigma_i), (4)$$

де ваги $p_i \geq 0$ та $\sum_{i=1}^{k} p_i = 1$.

Параметри моделювання GMM задаються за допомогою

$$\lambda = (p_i, \mu_i, \Sigma_i)_{i=1,\dots,k}, (5)$$

де *k* - кількість компонентів суміші. Параметри ГММ можна ітеративно оцінити за допомогою алгоритму максимізації очікування. Практичною

проблемою тут є оцінка коваріаційної матриці. Теоретично коваріаційна матриця є позитивно визначеною і оборотною. Для даного набору даних деякі статистичні методи можуть забезпечити необоротну матричну оцінку.

1.2.3 Алгоритм кластеризації K-середніх

У моделі Гаусової суміші кількість компонентів суміші *k* має бути оцінена за допомогою методу оптимізації. Потім потрібно використати алгоритм кластеризації K-середніх, щоб розділити всі точки даних на *k* кластерів. Кожна точка даних належить лише одному кластеру з найближчим середнім значенням на евклідовій відстані. Логіка алгоритму виглядає таким чином:

1) Для кожної точки даних визначається найближчий центр кластера;

2) Кожен центр кластера замінюється середнім значенням усіх закритих для нього точок даних.

3) Кроки 1 і 2 чергуються до зближення. Збіжність алгоритму до локального мінімуму.

Тут необхідно розглянути два практичні питання. По-перше, як вибрати правильне число *k* кластерів для використання, часто дана дія може бути неочевидною. Хамерлі та Елкан представляють алгоритм вивчення *k* під час кластеризації. Одне просте практичне правило — використовувати квадратний корінь з половини розміру даних як число *k*. Існують деякі обговорення щодо оптимальної кластеризації, включаючи об’єднання кластерів і поділ кластера на два або більше кластерів. На практиці близько 10 компонентів суміші забезпечать хорошу роботу системи розпізнавання мовлення. Друга проблема - це *k* початкових

центрів кластерів. Як правило, на практиці k точки даних вибираються випадковим чином як припущення.

1.2.4 Алгоритм максимізації очікування

Алгоритм максимізації очікування (АМО) — це ітераційний метод для знаходження оцінок максимальної правдоподібності параметрів у статистичних моделях.

За допомогою цього алгоритму EM використовуються багатоваріантні дані, що відповідають моделям гаусової суміші. Цей алгоритм потужний і надійний. Однак це вимагає специфікації початкової оцінки цих невідомих параметрів щодо компонентів моделі суміші. Алгоритм для отримання початкової оцінки невідомих параметрів:

1) Використання алгоритму K-середніх для класифікації всіх точок даних на k кластерів;

2) Усередині кожного кластера, оцінка його середньої щільності та коваріаційної матриці з власних точок даних кластера за допомогою методів вибіркового середнього та вибіркового коваріаційного значення;

3) Використання пропорцій розміру даних кластера для оцінки пропорцій змішування моделі нормальної суміші.

Враховуючи спостережувані дані Y, набір неспостережуваних прихованих даних або відсутні значення Z та невідомий параметр $\theta = \mu, \Sigma$ з функцією правдоподібності $L\ \theta; Y, Z = p\ Y, Z, \theta$. Максимальна оцінка правдоподібності (МОП) невідомих параметрів визначається шляхом максимізації граничної ймовірність спостережуваних даних,

$$L(\theta; Y) = p(Y|\theta) = \int p(Y, Z|\theta) dZ. \quad (6)$$

АМО прагне знайти маргінала ймовірності МОП шляхом ітераційного застосування наступними двома кроками:

1) Крок очікування (крок E): визначити $Q(\theta|\theta^{(t)})$ як очікуване значення логарифмічної функції імовірності θ відносно поточного умовного розподілу Z, заданого Y, і поточних оцінок параметрів $\theta^{(t)}$:

$$Q\left(\theta\middle|\theta^{(t)}\right) = E_{Z|Y,\theta^{(t)}}[\log L(\theta; Y, z)]. \quad (7)$$

2) Крок максимізації (крок M): знайти параметри, які максимізують цю кількість:

$$\theta^{(t+1)} = arg\max_{\theta} Q(\theta|\theta^{(t)}). \quad (8)$$

При підході максимальної правдоподібності до оцінки θ, оцінка забезпечується відповідним коренем рівняння ймовірності,

$$\frac{\partial \log L(\theta;Y,Z)}{\partial \theta} = 0. \quad (9)$$

За таких умов ОМ збігається до максимуму $L(\theta; Y)$. Ключем до ефективної реалізації цього алгоритму є вибір спостережуваної/відсутньої структури даних, щоб кроки О та М мали прості вирази закритої форми.

Основними труднощами використання ОМ для підгонки моделі суміші є: її критична залежність від ініціалізації; можливість збіжності до точки на межі простору параметрів з необмеженою ймовірністю (тобто один з m параметрів наближається до нуля з відповідною коваріацією, яка стає довільно близькою до сингулярної). Якщо коваріація близька до сингулярної, для вирішення цієї популярної серйозної проблеми пропонується оцінка несингулярної робастної коваріаційної матриці.

“MIXFIT” — це програмне забезпечення для автоматичного підбору та тестування моделей Гаусса за допомогою алгоритму ОМ. Це дуже хороший варіант, особливо для практиків, які не мають досвіду статистики. Алгоритм ОМ дійсно забезпечує надійні оцінки коваріаційної матриці. Ці матриці повинні бути позитивно визначеними, а саме оберненими. У аналізі великих даних проблема необоротності матриці тут не існує. Теоретично існує більш ніж достатньо точок даних, щоб

гарантувати, що всі оцінки коваріаційної матриці є (досить близько) позитивно визначеними.

Інший основний варіант алгоритму “MIXFIT” дозволяє користувачеві автоматично проводити тест для найменшої кількості компонентів, сумісних з даними. Як описано раніше, сам алгоритм “MIXFIT” автоматично надає початкові значення для застосування алгоритм ОМ, якщо користувач не надає жодного, розглядаючи вибірку, отриману з трьох джерел: *(a)* випадкові запуски, *(b)* запуски на основі ієрархічної кластеризації та *(c)* запуски на основі *k*-середніх. Що стосується *(b)*, користувач має можливість використовувати в стандартизованій або нестандартизованій формі результати семи ієрархічних методів (найближчий сусід, найдальший сусід, середнє по групі, медіана, центроїд, гнучке сортування та метод Уорда). Існує кілька параметрів алгоритму, які користувач може вказати за бажанням; як альтернатива, використовуються значення за замовчуванням. Програма відповідає моделі нормальної суміші для кожної початкової групи, зазначеної з трьох джерел *(a)* до *(c)*.

Всі ці обчислення автоматично виконуються програмою. Користувачеві потрібно лише надати набір даних обмеження на коваріаційні матриці компонентів (рівні, нерівні чи діагональні), ступінь вибору початкових груп, які будуть використані для визначення початкових значень, і кількість компонентів, які будуть підігнаними. Підсумкова інформація автоматично надається як вихід для остаточної відповідності. Однак не пропонується, щоб кластеризація набору даних базувалась виключно на одному розв’язанні рівняння ймовірності, а скоріше на різних рішеннях, які розглядаються разом. За умовчанням остаточна підгонка приймається як відповідна найбільшому з локальних максимумів. Однак підсумкова інформація може бути відновлена для будь-якої чіткої відповідності».

1.2.5 Аналіз основних компонентів ядра

Спочатку потрібно розглянути стандартний алгоритм основного компонента (АОК). Це ефективний метод зменшення розмірності з додатками для стиснення зображень і розпізнавання обличчя. Він являє собою лінійне перетворення, де дані виражаються в новій координатній основі, що відповідає порядку пріоритету напрямку максимальної дисперсії. Це легко виконати, розв'язавши задачу на власні значення. АОК — це ортогональне перетворення системи координат, щоб була можливість найкраще представити набір даних. Нові значення координат називаються головними компонентами.

АОК можна розрахувати з коваріаційної матриці. Нехай $F = (f_1, \dots, f_N)$ — матриця даних $M \times N$, що містить усі дані. Мета полягає в тому, щоб знайти M ортогональних векторів u_j (основні компоненти), які найкраще описують мінливість даних. Потрібно визначити $u = (u_1, u_2, \dots, u_M)'$ та

$$\frac{1}{N}\sum_{i=1}^{N}\left(u_j * f_i\right)^2 = \cdots = u_j'\left(\frac{1}{N}F * F'\right)u_j. \quad (10)$$

Тут $\mathrm{C} = \frac{1}{N}F * F'$- це коваріаційна матриця. Це позитивно напіввизначений $x'Cx = \cdots = \frac{1}{N}\sum_{j=1}^{M}\sum_{i=1}^{n}(x_j F_{ji})^2 \geq 0$ і це означає, що власні значення невід'ємні. Обмежена оптимізаційна задача максимізації:

$$\sum_{i=1}^{N}(u_j * f_i)^2 \quad (11)$$

а вимога u_j розв'язується за допомогою множників Лагранжа на максимізацію:

$$u_j'Cu_j + \lambda(1 - u_j'u_2)^2. \quad (12)$$

Оптимальними векторами є власні вектори коваріаційної матриці

$$Cu_j = \lambda u_j. \quad (13)$$

Оскільки коваріаційна матриця симетрична, то власні вектори u_j ортогональні. Результат АОК такий

$$F_{M\times N} = U_{M\times M} * V_{M\times N}, \text{ (14)}$$

де $V_M = U'_{M\times M} * F_{M\times N}$. Набір даних можна відновити. Тому, якщо деякі основні компоненти вважаються неважливими, але є бажання відновити дані, використовувати потрібно лише $k < M$ основних компонентів,

$$F_{reconstructed} = U_{M\times k} * V_{k\times N}. \text{ (15)}$$

Часта дисперсії, що пояснює i^{th} основних компонентів це

$$\frac{\lambda_i}{\sum_{j=1}^{M} \lambda_j}. \text{ (16)}$$

Спочатку в 1901 році британський математик Карл Пірсон ввів основні компоненти для використання низько-вимірного підсумку для найкращого опису велико-вимірного набору даних. Останнім часом ця ідея широко застосовується в аналізі великих даних.

Однак в даному випадку більше цікавості викликає його застосування розпізнавання мови, а не зменшення розмірності даних. Аналіз головних компонентів ядра (АГКЯ) — це згенерований алгоритм АОК. Це ефективний метод вилучення ознак у системах розпізнавання мовлення. Техніка АГКЯ є результатом застосування функції ядра до АОК, щоб отримати представлення АОК у просторі вищого виміру, який, можливо, може генерувати більш помітні мовні характеристики.

1.2.6 Нейронна мережа

Нейронна мережа – це математична модель, яка використовується для виконання певної функції. Вона працює як людський мозок і передбачає використання різних функцій оцінки (функцій на основі ймовірностей) для обчислення ймовірних відповідей (оцінок), ці функції потім називають нейронами.

Нейронні мережі — це метод машин, які навчаються робити щось на основі навчальних прикладів. Наприклад, щоб дізнатися, як звучить слово «Привіт», машині будуть надані навчальні дані багатьох різних людей, які

використовують різні акценти, висоту звуку, різні рівні фонового шуму та інші змінні звуки/слова, які впливають, щоб допомогти машині екстраполювати основа слова.

Як вже описувалось вище, комп’ютери навчаються за допомогою алгоритмів машинного навчання виконувати завдання самостійно.

Мережі навчаються формувати зв'язок між вхідним і вихідним шарами з певними ваговими показниками. Ваговий показник навчає мережу класифікувати мовні моделі за категоріями фонем.

Мережі вивчають основні закономірності та узагальнюють дані навчання на нові приклади. Це важливо для розпізнавання мовлення. Мовлення — дуже нелінійний процес. Мережі можуть обчислювати будь-які нелінійні функції свого входу, що дозволяє їм виконувати як завгодно складні перетворення даних.

Мережі пропонують єдину обчислювальну парадигму, яка може легко інтегрувати обмеження з різних типів вхідних даних.

Це полегшує використання як базових, так і диференційних мовних вводів.

1.2.7 Глибока нейронна мережа (ГНМ)

За визначенням, глибоке навчання, яке іноді називають навчання репрезентації або неконтрольоване навчання функцій є новою областю машинного навчання. Глибинні нейронні мережі (ГНМ) походять від нещодавньої революції нейронних мереж, що використовуються в ігровій індустрії. «Складне зображення та швидкий темп сучасних відеоігор вимагають апаратного забезпечення, яке може встигати, і результатом став графічний процесор, який містить тисячі відносно простих процесорних ядер на одному чіпі», який було знайдено дуже схожим на нейронну мережу.

Системи розпізнавання мовлення можуть бути навчені автоматично та прості у використанні. Для порівняння, одним із головних недоліків ГММ є те, що вони статистично неефективні для моделювання даних.

Нейронні мережі, навчені похідними помилок зворотного поширення, стали привабливим підходом до акустичного моделювання для розпізнавання мовлення. Вони не роблять припущень щодо статистичних властивостей ознак. Коли нейронні мережі використовуються для оцінки ймовірностей сегмента ознак мовлення, вони дозволяють проводити дискримінаційне навчання природним і ефективним способом. Новий термін «глибоке» навчання походить від глибини мережевого рівня, що підтримується сучасними графічним процесором. Через ефективність глибокого навчання вона «стає основною технологією для розпізнавання мовлення» і стала заміною моделі суміші Гауса. Протягом багатьох років намагалися ефективно використати ГНМ, але нещодавно за допомогою кількох проривів це стало можливим. Зменшення витрат і збільшення обчислювальної потужності зробило розумним навчання цих ГНМ за допомогою грубої сили з використанням великих наборів даних. Іншою причиною збільшення популярності концепції ГНМ є те, що було зібрано набагато більше даних для навчання цих систем, а вдосконалені алгоритми, розроблені для великих обсягів даних, краще навчають ГНМ.

Висновки до розділу 1

Перший розділ був присвячений огляду задачі розпізнавання мовлення та алгоритмам її реалізації. В даному розділі розглянуто різні алгоритми реалізації до вирішення поставленої задачі.

РОЗДІЛ 2. ЗАСТОСУВАННЯ ГАУСОВОГО ПРОЦЕСУ В НЕЙРОННІЙ МЕРЕЖІ ДЛЯ РОЗПІЗНАВАННЯ МОВЛЕННЯ

У даному розділі описано як саме застосовується алгоритм Гаусового процесу в нейронній мережі, використовуючи переваги варіаційної апроксимації, що призводить до практичного алгоритму навчання, який в свою чергу сумісна з оригінальним механізмом навчання нейронних мереж. В даній нейронній мережі використовується рівень процесу Гауса саме як активація випадкового розширення характеристик узагальнених спектральних ядер на першому прихованому шарі. Таким чином, можна отримати нову форму архітектури стандартної моделі нейронної мережі, що продемонстровано на рис. 1.

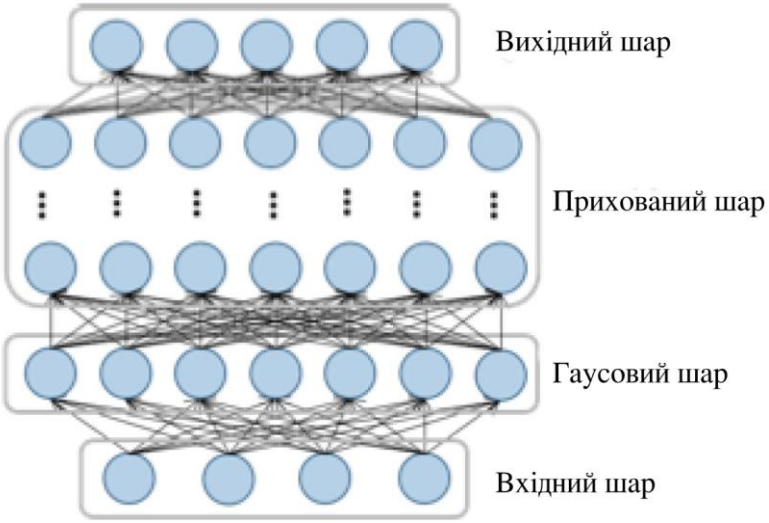


Рисунок 1. Архітектура запропонованого алгоритму Гаусового процесу в нейронній мережі

2.1 Стандартна форма глибоких нейронних мереж

Розпочнемо з визначення стандартної форми глибоких нейронних мереж як багатовимірного зображення:

$$f_{DNN}: \mathbb{R}^{D_{in}} \rightarrow \mathbb{R}^{D_{out}}, \quad (17)$$

де D_{in} та D_{out} позначають розмірність вхідних даних і розмірність вихідних даних відповідно. У глибоких нейронних мереж є три типи шарів:

1) вхідний шар $x \in \mathbb{R}^{D_{in}}$;

2) прихований шар $f_h: \mathbb{R}^{D_{h-1}} \to \mathbb{R}^{D_h}$ для $h = 1, \dots, H$ де $D_0 \triangleq D_{in}$;

3) вихідний шар $f_{out}: \mathbb{R}^{D_{in}} \to \mathbb{R}^{D_{out}}$.

Дотримуючись даних позначень можна вказати ширину шарів стандартної нейронної мережі просто за допомогою списку H+2 цілих чисел, тобто (D_{in}, D_1, …, D_H, D_{out}), де H зазвичай позначає кількість прихованих шарів. Щоб передати $\boldsymbol{x}$ через шари, тобто $f_{DNN}(x) = = f_{out}\big(\cdots f_2\big(f_1(x)\big)\cdots\big)$, потрібно рекурсивно обчислити для h = 1, …, H

$$z_h = W_h \cdot f_{(h-1)}(x) + b_h,\ f_h(x) = \sigma(z_h), \quad (18)$$

де · це позначення операції крапкового добутку, z_i^h - це i-ий вихід афінного відображення, застосованого до виходів попереднього шару з оптимізованими параметрами $W_h \in \mathbb{R}^{D_{h+1} \times D_h}$ та $b_h \in \mathbb{R}^{D_{h+1}}$, $f_0(x) \triangleq x$ – це ініціалізація рекурсії, а для $h = 1, \dots, H$, $f_h(x)$ виникає після проходження функції активації $\sigma(\cdot)$. Було помічено, що на практиці $\sigma(\cdot)$ зазвичай обирають із деяких загальних функцій, наприклад: *sigmoid, tanh* та інших, які як доведено є універсальними.

Після отримання $f_h(x)$, вихідні дані нейронної мережі можна обчислити, пройшовши нормалізоване афінне зображення без активації:

$$f_{DNN}(x) = \frac{1}{\sqrt{D_H}} W_{out} f_H(x) + b_{out}, \quad (19)$$

де $W_{out} \in \mathbb{R}^{D_{out} \times D_H}$ та $b_{out} \in \mathbb{R}^{D_{out}}$.

2.2 Гаусовий процес

Гаусівський процес (ГП) є випадковим процесом, таким чином, що будь-який скінченний набір випадкових величин у гаусовому процесі слідує багатовимірному розподілу Гаусса. Якщо функція відповідає процесу Гаусса, вона представляється як:

$$f(z) \sim \mathcal{G}\mathrm{P}\big(\mathrm{m}(\mathrm{z}), \mathrm{k}(\mathrm{z}, z')\big), \quad (20)$$

де $z, z' \in \mathbb{R}^{D_{in}}$ є довільні входи, $m(\cdot)$ – середнє значення функції та $k(\cdot,\cdot)$ є ядерною функцією. Насправді існує тісний зв'язок між глибокими нейронними мережами та Гаусовими процесами, якщо W_{ij}^{out} та b_i^{out} мають ідентичний і залежний розподіл в межах нескінченної ширини вихідного шару $D_H \to \infty$, то з центральної граничної теореми випливає, що $f_{DNN}(x)$ є нормалізованою сумою випадкових величин, яка прагне до багатовимірного розподілу Гаусса. Таким чином, будь-який скінченний набір $\{f_{DNN}(x_1), \dots, f_{DNN}(x_k)\}$, буде багатовимірним розподілом Гауса, який нагадує ГП. У регресії ГП перед навчанням необхідно вказати як середню функцію, так і ядерну функцію. На практиці середня функція зазвичай встановлюється рівною нулю, припускаючи, що розподіл виходів був добре перетворений. З іншого боку, ядерна функція була широко вивчена, оскільки вона контролює властивості кривизни її функції реалізації $f(\cdot)$, такі як ступінь гладкості та періодичності. Будь-яка функція, яка задовольняє позитивну визначеність $\sum_{i,j} c_i c_j k(z_i, z_j) > 0$, тобто для всіх $c_i c_j \in \mathbb{R}$ та $z_i, z_j \in \mathbb{R}^{D_{in}}$ є дійсною ядерною функцією. В даному випадку використання форми ядерних функцій, які узагальнюють усі нестаціонарні процеси, звані узагальненими спектральними ядрами, які приймають форму

$$k(z, z') = \phi(z)^T \phi(z'), \quad (21)$$

$$\phi(z) = \frac{1}{\sqrt{S}} \begin{bmatrix} \cos(w_1 \cdot z) + \cos(w_1' \cdot z) \\ \sin(w_1 \cdot z) + \sin(w_1' \cdot z) \\ \vdots \\ \cos(w_S \cdot z) + \cos(w_S' \cdot z) \\ \sin(w_S \cdot z) + \sin(w_S' \cdot z) \end{bmatrix}. \quad (22)$$

Потрібно зазначити, що S є заздалегідь заданим цілим числом, яке відноситься до кількості спектральних базисних функцій. Як виявляється, вищий S забезпечує кращу репрезентативну здатність ГП, але довший час навчання та більшу кількість параметрів. Тому на практиці потрібно

обирати максимально можливе S з огляду на доступні обчислювальні ресурси.

2.3 Огляд Гаусового процесу як шару в просторі

Одним з важливих питань, пов'язаних з глибокими нейронними мережами та й в цілому зі штучними нейронними мережами, є ручний вибір відповідних структур моделі, наприклад, форми функцій активації. Через відсутність методів автоматичного вибору моделі вибір функцій активації в основному базується на емпіричному досвіді. Щоб вирішити це питання, запропоновано послабити обмеження на використання детермінованої функції активації, накладаючи на дані функції "випадковість". За допомогою процес Гаусса для управління розподілом недетермінованих функцій активації дану задачу можна вирішити:

$$\sigma(z) \sim \mathcal{G}\mathrm{P}(0, \phi(z)^T, \phi(z')), \quad (23)$$

де $\mathbf{z}$ та $\boldsymbol{z}'$ — довільні вхідні дані для функцій активації, а $\boldsymbol{\phi}(\cdot)$ — це узагальнена форма функцій ядра. З точки зору простору ваги гауссового процесу формулу 20 можна переписати у вигляді $\sigma(z) = \alpha \cdot \phi(z)$, де $\alpha \sim N(0, \boldsymbol{I}_{2\mathrm{s}})$, в свою чергу $\boldsymbol{I}_{2\mathrm{s}}$ — це ідентична матриця з 2S ненульовими записами. Потрібно зазначити що α являє собою амплітуди різних спектральних базисних функцій. Теоретично можна використовувати функції активації на основі Гаусового процесу для всіх прихованих шарів глибокої нейронної мережі, але висновок про накопичені ГП є значно дорожчими, ніж висновок лише одного рівня ГП. Отже, у даній роботі, використовується шар ГП для заміни стандартного прихованого шару. Зокрема, шар ГП розміщується на першому шарі, щоб спростити реалізацію, а також уникнути проблеми з патологією. Математично, тепер вхід до GP є не що інше, як афінно перетворений вхід, як визначено в формулі 18, з чого випливає що

$$\sigma(z) = \alpha \cdot \phi(z) == \alpha \cdot \phi(W_{inx} + b_{in}), (24)$$

де $W_{in} \triangleq W_1$ та $b_{in} \triangleq b_1$. Оскільки $\phi(z)$ включає в себе добуток оптимізованих параметрів $\{w_1, w_1', \dots, w_s, w_s'\}$ можна скомбінувати $\{w_i, w_i'\}$ з параметром матриці W_{in}, щоб отримати

$$Z = \begin{bmatrix} w_1^T \\ \vdots \\ w_s^T \end{bmatrix} W_{in},\ Z' = \begin{bmatrix} w_1^T \\ \vdots \\ w_s^T \end{bmatrix} W_{in} \in \mathbb{R}^{S \times D_{in}}, (25)$$

і також векторний параметр b_{in} для отримання p та p'. Зараз можна спростити рівняння 21 до вигляду $\phi(\boldsymbol{z}) = \psi(Zx + p, Z'x + p')$, де

$$\psi(z, z') = \frac{1}{\sqrt{S}} \begin{bmatrix} \cos(z) + \cos(z') \\ \sin(z) + \sin(z') \end{bmatrix}. (26)$$

Оскільки в наступному шарі $\sigma(z)$ буде афінно перетворено за даними $\mathbf{W_2}$ та $\mathbf{b}_2$, оптимізація амплітуд $\boldsymbol{\alpha}$ подібна до оптимізації $\mathbf{W_2}$ у стандартних глибоких нейронних мережах, і, таким чином, має менший вплив на недетерміновані функції активації. Для підтримки випадковості функцій активації потрібно змоделювати розподіл спектру $p(\hat{Z}|X, Y)$ із $\hat{Z} = \begin{bmatrix} Z \\ Z' \end{bmatrix}$ замість моделювання розподілу амплітуд α.

2.2 Навчання нейронної мережі з Гаусовим процесом

Використаємо варіаційний розподіл для $\hat{Z}$ з варіаційними параметрами:

$$q(\hat{Z}_{ij}) = \mathcal{N}(m_{ij} s_{ij}^2). (27)$$

Для зручності використання θ позначає сукупність усіх θ_{ij} для всіх $i \in \{1, \dots, 2S\},\ j \in \{1, \dots, D_{in}\}$. Для того щоб зрозуміти певний висновок використання нейронної мережі з Гаусовим процесом потрібно знайти варіаційні параметри для представлення вихідного розподілу $p(\hat{Z}|X, Y)$. У варіантному висновку, θ можна безпосередньо дізнатися зі зворотного

поширення, диференціюючи нижню межу негативних доказів як цільову функцію мінімізації:

$$\mathcal{L}(\theta) = -\mathbb{E}_{q(\hat{\mathrm{E}};0)}\left[\log p\left(Y\middle|X,\hat{Z}\right)\right] + \mathcal{KL}(q(\hat{Z};\theta)||p\left(\hat{Z}\right)), \quad (28)$$

де $\mathcal{KL}(\cdot\,||\,\cdot)$ це дивергенція Кульбака-Лейблера (КЛ), яка вимірює відстань між розподілами. Використовуючи варіаційну оцінку Баєса стохастичного градієнта, потрібно застосувати трюк репараметризації для кожного запису випадкової матриці $\hat{Z}$ за допомогою диференційованої функції $g(\cdot)$:

$$\hat{Z}_{ij} = g\left(\epsilon_{ij}\right) = m_{ij} + s_{ij}\epsilon_{ij}, \epsilon_{ij} \sim \mathcal{N}(0,1) \quad (29)$$

Звідси випливає, що маємо

$$\mathbb{E}_{q(\hat{\mathrm{E}};0)}\left[\log p\left(Y\middle|X,\hat{Z}\right)\right] = \mathbb{E}_{\epsilon_{ij}\sim\mathcal{N}(0,1)}[\log p(Y|X,\epsilon)] \quad (30)$$

це означає, що джерело випадковості походить від ϵ_{ij}, який легко можна згенерувати за допомогою сучасних бібліотек програмування. Визначення $\log p(Y|X,\epsilon)$ залежить від того, яку саме задачу вирішує нейронна мережа з Гаусовим процесом.

У застосунку до задачі розпізнавання мовлення потрібно визначити:

$$\log p(Y|X,\epsilon) = \sum_{i=1}^{N} y_i \cdot \log softmax\left(f_{DNN}(x_i)\right), \quad (31)$$

де x_i, y_i – це і-та навчальна пара введення/виведення. Це важливо для того, щоб вище очікуваний термін нагадував типову кроссентропію, яка використовується в стандартній структурі глибокої нейронної мережі.

Більше того, дивергенція КЛ можна розглядати як термін регуляризації для рівня Гаусового процесу, який знову потрапляє в оригінальну структуру ГНМ. Це разом означає, що можна навчати нейронну мережу з Гаусовим процесом точно в тій самій структурі ГНМ, для якої доступний ряд інструментів програмування. Під час навчання можна виконати прямий перехід, а потім стандартний алгоритм зворотного поширення, щоб оновити параметри, як у стандартних ГНМ. Але під час тестування потрібно запустити алгоритм прямого проходу Т

разів і усереднювати всі T передбачення, щоб виключити всі можливі активації.

Висновки до розділу 2

Даний розділ був присвячений тому, як саме Гаусовий процес можна реалізувати у нейронних мережах та як буде виглядати архітектура зміненого алгоритму. З певних арифметичних дій, можна дійти до висновку, що є можливість навчати нейронну мережу з Гаусовим процесом точно в тій самій структурі ГНМ.

РОЗДІЛ 3. ОПИС РЕАЛІЗАЦІЇ АЛГОРИТМУ ТА ПОРІВНЯННЯ З ІНШИМИ МЕТОДАМИ

У рамках даної роботи був реалізований запропонований алгоритм, що демонструє свою перевагу над іншими алгоритмами при роботі з розпізнаванням мовлення. Ціль даної роботи – на базі розробленого додатку продемонструвати роботу покращеного алгоритма, який надалі можна використовувати при роботі з обробкою мовлення.

Дана частина роботи описує технічні деталі та особливості реалізації алгоритму в певному середовищі.

3.1 Реалізація нейронної мережі з Гаусовим процесом

Нейронна мережа з Гаусовим процесом відноситься до напрямку “машинного навчання”. На сьогодні машинне навчання вийшло далеко за рамки наукової фантастики. Чесно кажучи, людський мозок може аналізувати великі обсяги даних, але ця здатність обмежена обсягом даних, які він може поглинути в будь-який момент. Більш точні прогнози та ідеї, які надає AI, покращують ефективність бізнесу, знижують вартість виробництва та підвищують продуктивність. Саме тому проекти з

машинним навчанням відрізняються від традиційних програмних проектів. Відмінності полягають у наборі технологій, навичках, необхідних для проекту на основі штучного інтелекту, та необхідності глибокого дослідження. Щоб реалізувати свої прагнення до штучного інтелекту, потрібно використовувати мову програмування, яка є стабільною, гнучкою та має доступні інструменти. Python пропонує все це, тому саме ця мова використовується для реалізації нейронної мережі з Гаусовим процесом.

Отже, для реалізації даного алгоритму потрібно використати наступні технології:

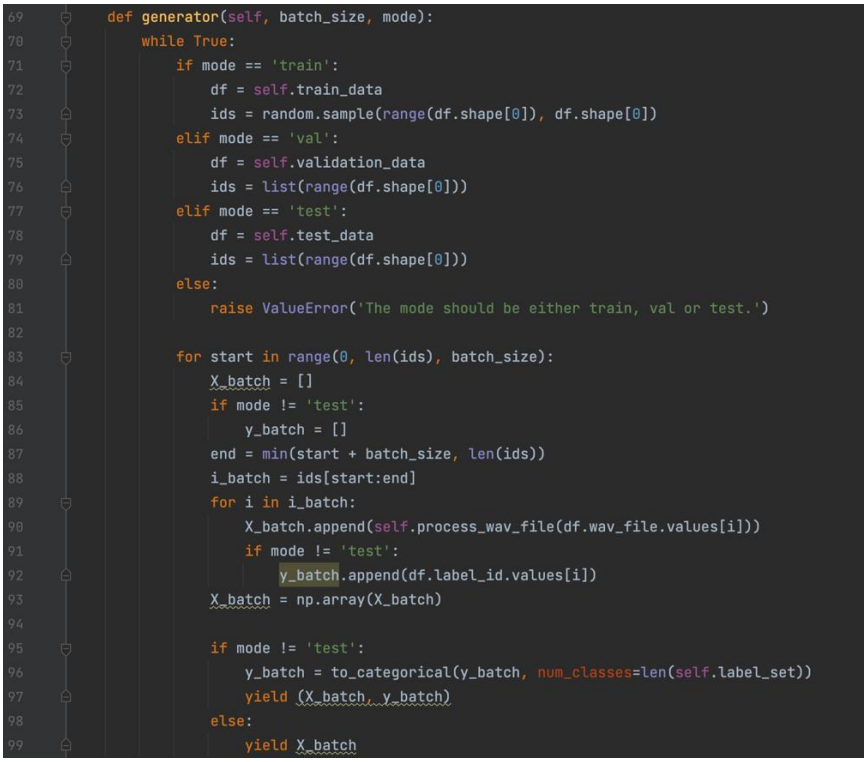
- Python 3.10 version
- TensorFlow 1.13.1 version

Опис основних файлів програми:

GPNN_model.py – файл за допомогою якого реалізується алгоритм нейронної мережі з Гаусовим процесом.

GPNN_test.py – файл за допомогою якого відбувається тестування даного алгоритму.

load.py – файл для завантаження та обробки аудіофайлів у “wav” форматі:



```
    def generator(self, batch_size, mode):
        while True:
            if mode == 'train':
                df = self.train_data
                ids = random.sample(range(df.shape[0]), df.shape[0])
            elif mode == 'val':
                df = self.validation_data
                ids = list(range(df.shape[0]))
            elif mode == 'test':
                df = self.test_data
                ids = list(range(df.shape[0]))
            else:
                raise ValueError('The mode should be either train, val or test.')

            for start in range(0, len(ids), batch_size):
                X_batch = []
                if mode != 'test':
                    y_batch = []
                end = min(start + batch_size, len(ids))
                i_batch = ids[start:end]
                for i in i_batch:
                    X_batch.append(self.process_wav_file(df.wav_file.values[i]))
                    if mode != 'test':
                        y_batch.append(df.label_id.values[i])
                X_batch = np.array(X_batch)

                if mode != 'test':
                    y_batch = to_categorical(y_batch, num_classes=len(self.label_set))
                    yield (X_batch, y_batch)
                else:
                    yield X_batch
```

Рис.1 Функція тренування обробленого датасету

view.py – файл за допомогою якого відбувається зчитування wav файлу у вигляді спектрограми.

tens.py – файл за допомогою відбувається вся робота з датасетом за допомогою нейронної мережі:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, Dense, Flatten, MaxPooling2D
NN = Sequential()
#--------------Then adding a convolution layer--------------------------------#
NN.add(Conv2D(filters=64, kernel_size=(3,3), activation='relu', input_shape=(28,28,1)))
#--------------Then adding a pooling layer-------------------------------------#
NN.add(MaxPooling2D(pool_size=(2,2)))
#--------------Then adding flattening the results-----------------------------#
NN.add(Flatten())
#-------Then adding a dense layer to reduce the number of features-------#
NN.add(Dense(units=128, activation='relu'))
#-------Then adding a dense layer to produce the final output-----------#
NN.add(Dense(units=10, activation='softmax'))
#-------------------Printing the Model's Summary---------------------------#
print(NN.summary()) # show all data conclusion
```

Рис.2 Створення нейронної мережі та додавання шарів до мережі

main.py – файл який відповідає за запуск програми:

```python
class dataloader():
    def __init__(self, batch, path):
        self.batch = batch
        self.path = path
        self.count = 0
        self.files = glob.iglob(os.path.join(self.path, "wav", "*.wav"))
        self.SPACE_TOKEN = '<space>'
        self.SPACE_INDEX = 0
        self.FIRST_INDEX = ord('a') - 1  # 0 is reserved to space
        self.char_list = [str(i) for i in range(10)]
        self.char_list_c = [chr(32) if i == 0 else chr(i + 96)
                            for i in range(27)]
        self.char_list_c.extend(self.char_list)
        self.char2int = OrderedDict((val, i)
                                    for i, val in enumerate(self.char_list_c))
        self.int2char = OrderedDict((i, val)
                                    for i, val in enumerate(self.char_list_c))

    def next(self):
        data = []
        length = []
        target = []
        original_text = []
        i = 0
        while i < self.batch:
            file = next(self.files)
            fs, audio = wav.read(file)
            print(file)
            if (audio.shape[0] > 0):

                filename = os.path.basename(file)
                file_name, ext = os.path.splitext(filename)
                try:
                    inputs = self.compute_linear_specgram(audio, fs)
                except ArithmeticError:
                    continue

                i += 1
                data.append(inputs)
                length.append(inputs.shape[0])
                with open(os.path.join(self.path, "txt", file_name + ".txt")) as f:
                    target_text = f.readline()
                    target_text = re.sub('[^a-zA-Z]+', ' ', target_text)
                    # target_text=re.sub("[".join(self.char_list),"",target_text)
                    original = ' '.join(target_text.strip().lower().split(' '))
                    # print("original:",original)
                    original_text.append(original)
                    targets = original.replace(' ', '  ')
                    targets = targets.split(' ')

                    # Adding blank label
                    targets = np.hstack(
                        [self.SPACE_TOKEN if x == '' else list(x) for x in targets])

                    # Transform char into index
                    targets = np.asarray(
                        [self.SPACE_INDEX if x == self.SPACE_TOKEN else self.char2int[x] for x in targets])

                    target.append(targets)
        max_len = np.max(length)
        for i in range(self.batch):
            if (data[i].shape[0] != max_len):
                data[i] = np.pad(
                    data[i], ((0, max_len - length[i]), (0, 0)), 'constant', constant_values=(0, 0))
                # length[i]=data[i].shape[0]

        data = np.asarray(data)
        train_inputs = (data - np.mean(data)) / np.std(data)
        train_inputs = np.expand_dims(train_inputs, axis=3)
        train_targets = self.sparse_tuple_from(target)
        train_seq_len = length

        return train_inputs, train_seq_len, train_targets, original_text

        # return list_files,list_data

    def __iter__(self):
        return self
```

Рис.3 Частини коду з файлу main.py

Результатом роботи даної програми став хвильовий сигнал і спектрограма зразка аудіофайлу:

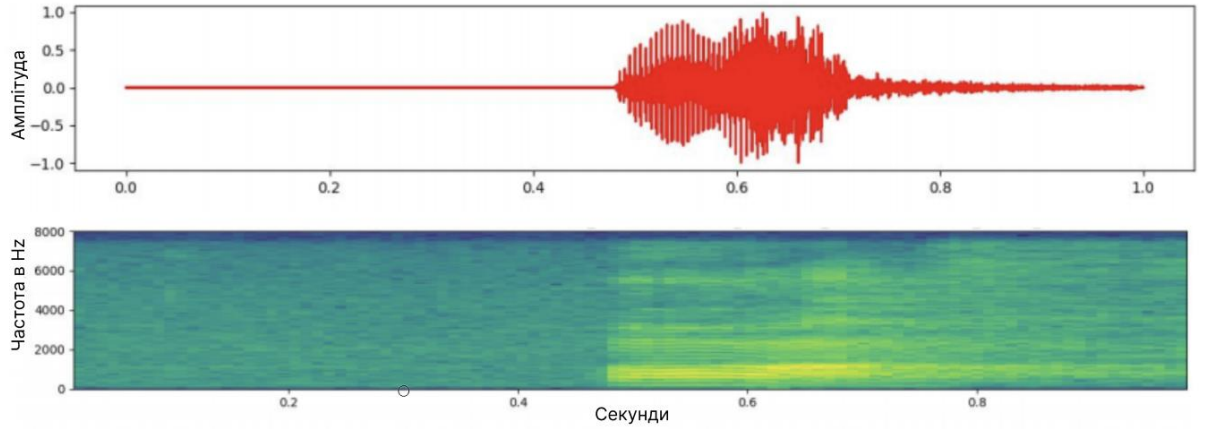


Рис.4 Результат роботи програми

Надалі потрібно зрозуміти чи дійсно цей алгоритм може працювати краще, аніж ті які вже існують. Порівняння результатів буде описано в наступному розділі.

3.2 Отримані результати та порівняння з іншими методами

Так як нейронна мережа з Гаусовим процесом наближена до нейронних мереж, тому було вирішено порівняти з глибокою нейронною мережею та порівняти результати роботи.

Експеримент проводився з набором даних RM. Вхідні функції складаються з 13 кепстральних коефіцієнтів частоти, об’єднаних з їхніми першим і другим тимчасовими диференціалами за допомогою контексту з 9 кадрів, що призводить до побудови 351-мірної акустичної вхідної характеристики на кожному рамка. Вихід складається з 689 зв'язаних станів трифона.

Базові гібридні системи були створені за допомогою набору інструментів з відкритим кодом у два кроки. На першому кроці проведено пошарове дискримінаційне попереднє навчання зі швидкістю навчання 0,001 за допомогою планувальника швидкості навчання, а потім налаштування вільних параметрів мережі зі швидкістю навчання 0,002 на основі планувальника швидкості навчання. Відбулась заміна першого прихованого рівня стандартної гібридної глибокої нейронної мережі на рівень активації Гаусового процесу, а решта архітектури моделі залишилася такою ж, як базова система глибокої нейронної мережі. Також був використаний алгоритм оптимізації Адама з швидкістю навчання 0,001 і стратегією ранньої зупинки. При навчанні обох мереж в якості функції втрат була обрана крос-ентропія. При декодуванні використовувалася триграмова модель мови, побудована з використанням усіх транскриптів акустичних даних RM для систем глибокої нейронної мережі так і для нейронної мережі з Гаусовим процесом.

Навчання мереж відбувалось з 5 прихованими шарами і зберігалась однакова кількість прихованих одиниць у кожному прихованому шарі. У нейронній мережі з Гаусовим процесом перший прихований рівень — це рівень Гаусового процесу, тоді як інші рівні залишаються такими ж, як

базові нейронній мережі. Зміна кількості вузлів у кожному прихованому шарі відповідає різним структурам. Оскільки мережа з Гаусовим процесом використовує більш узагальнені функції активації, вона може працювати більш надійно, ніж глибока нейронна мережа. Щоб продемонструвати стійкість до недообладнання та переобладнання, потрібно змінювати кількість прихованих вузлів D_h у всіх шарах (h = 1,...,H) від 50 до 4000.

		Глибока нейронна мережа					
S	D_h	N	Набір 1	Набір 2	Набір 3	Набір 4	Загальне
25	50	17600	11.49	10.39	17.27	12.30	12.95
60	125	44000	7.95	8.53	14.38	10.47	10.35
125	250	88000	7.15	7.13	12.66	8.98	9
250	500	176000	5.95	6.32	11.61	7.68	7.90
500	1000	352000	5.51	6.40	10.28	8.27	7.63
750	1500	528000	5.62	6.32	10.43	7.90	7.58
1000	2000	704000	5.47	6.08	9.57	8.23	7.36
2000	4000	14080000	6.56	6.88	10.86	8.35	8.17

Таблиця 1. Результати експерименту для глибокої нейронної мережі

		Нейронна мережа з Гаусовим процесом					
S	D_h	N	Набір 1	Набір 2	Набір 3	Набір 4	Загальне
25	50	17600	7.81	8.33	13.21	12.30	9.81
60	125	44000	7.54	7.93	13.38	10.47	9.57
125	250	88000	6.33	6.80	12.31	8.98	8.61
250	500	176000	5.58	6.44	10.39	7.68	7.54
500	1000	352000	5.58	5.88	9.69	8.27	7.18

750	1500	528000	5.62	5.64	9.42	7.90	7.02
1000	2000	704000	5.23	6.56	9.30	8.23	7.11
2000	4000	14080000	5.08	6.20	8.36	8.35	6.63

Таблиця 2. Результати експерименту для нейронної мережі з Гаусовим процесом

ГНМ	ГПНМ	Покращення
Загальне		
12.95	9.81	+24%
10.35	9.57	+7%
9	8.61	+4%
7.90	7.54	+4%
7.63	7.18	+5%
7.58	7.02	+7%
7.36	7.11	+3%
8.17	6.63	+18%

Таблиця 3. Результати покращення

Дані таблиці демонструють показники частоти помилок слів системи глибокої нейронної мережі та нейронної мережі з Гаусовим процесом із 50, 125, 500, 1000, 1500, 2000 і 4000 вузлів у кожному прихованому шарі. Як і очікувалося, ГНМ з меншою кількістю прихованих вузлів працювала гірше через недостатню здатність розуміти складні відносини, закодовані в мовних даних. З іншого боку, коли кількість прихованих вузлів занадто велика, існує більший ризик переобладнання в системі ГНМ.

Згідно з результатами, система ГПНМ досягла послідовних покращень продуктивності порівняно з базовими ГНМ. Особливо, коли кількість прихованих одиниць дорівнює 50 і 4000, де система ГПНМ

порівняно покращила 24,25% і 18,85% порівняно з базовою системою ГНМ відповідно. Це відповідає очікуванням від даної роботи – ГНМ лише з кількома прихованими вузлами не можуть охопити складне відображення між акустичними характеристиками та станами трифона, але ГПНМ здатні пом’якшити недостатнє пристосування, коли знаходяться під жорстким обмеженням складності моделі. Крім того, можна побачити, що ГНМ, як правило, перенавчається, коли використовується занадто багато прихованих вузлів, тоді як ГПНМ все ще зберігає хорошу продуктивність. Це наочно демонструє здатність ГПНМ запобігати переобладнанню при побудові великих мереж. На основі результатів експерименту можна зазначити, що ГПНМ можуть фіксувати більш складні відносини між акустичним входом і мітками стану цільового трифона для використання більш узагальнених форм функції активації. Враховуючи це, ми робимо висновок, що ГПНМ є більш надійними, ніж ГНМ.

У функціях активації Гаусового процесу кількість спектральних базових функцій відіграє життєво важливу роль у представленні можливих форм функцій активації. Тим не менш, ми змінювали кількість спектральних базисних функцій у шарі ГП, фіксуючи інші приховані шари.

S	Набір 1	Набір 2	Набір 3	Набір 4	Загальне
125	6.17	6.64	10.47	7.75	7.77
250	5.58	6.44	10.39	7.71	7.54
1000	5.54	6.16	9.34	7.04	7.03

Таблиця 4. Продуктивність системи під час роботи нейронної мережі з Гаусовим процесом

Таблиця 4 демонструє продуктивність систем ГПНМ, що містять 125, 250, 1000 спектральних базових функцій у наборі даних, при цьому фіксуються 500 прихованих вузлів у всіх прихованих шарах. В результаті

у всіх чотирьох тестових наборах були зменшені з більшою кількістю спектральних базисних функцій. Це означає, що шар Гаусового процесу отримав більшу потужність узагальнення через наявність більшої кількості спектральних базових функцій. Завдяки такому ефекту, обнадійливо використовувати більшу кількість спектральних базових функцій у ГПНМ.

Висновки до розділу 3

У даному розділі було повність описано реалізації алгоритму. Описано які технології використовуються та чому. Також були проведені експериментальні дослідження, які допомогли продемонструвати потужність та переваги роботи даного алгоритму.

ВИСНОВКИ

В даній дипломній роботі було проаналізовано та реалізовано нову структуру нейронної мережі під назвою нейронні мережі Гаусса (ГПНМ) для розпізнавання мовлення з використанням непараметричних функцій активації на основі ГП. Використовуючи переваги рівня ГП, вирішується проблема структурної та параметричної невизначеності моделі. Змінюючи кількість прихованих вузлів і отримали послідовні покращення ГПНМ порівняно з базовим ГНМ. З цих експериментів ми робимо висновок, що ГПНМ більш надійні, ніж стандартні ГНМ. Було виявлено, що ГНМ з меншою або більшою кількістю прихованих вузлів схильні до недообладнання або переобладнання, тоді як ГПНМ все ще можуть забезпечувати високі стандарти продуктивності. Крім того, проаналізувавши вплив кількості спектральних базисних функцій S на продуктивність ГПНМ, результати показують, що є змога підвищити продуктивність ГПНМ, постійно збільшуючи S. У майбутньому є можливість застосувати ГПНМ до великих завдань розпізнавання словника.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. S. M. Mon and H. M. Tun, “Speech-to-text conversion (STT) system using hidden Markov model (HMM),” International Journal of Scientific & Technology Research, vol. 4, no. 6, Jun. 2015.
2. F. Fauziya and G. Nijhawan, “A comparative study of phoneme recognition using GMM-HMM and ANN based acoustic modeling,” International Journal of Computer Applications, vol. 98, no. 6, July 2014.
3. A. S. Utane, “Emotion recognition through speech using gaussian mixture model and hidden Markov model,” International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, no. 4, April 2013
4. A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the EM algorithm (with discussion),” Journal of the Royal Statistical Society, 1997.
5. J. Wang, and C. Liu, “Generating multivariate mixture of normal distributions using a modified cholesky decomposition,” in Proc. the 2006 Winter Simulation Conference, 2006.
6. G. Hamerly and C. Elkan, “ Learning the K in K-Means,” in Proc. NIPS'03 the 16th International Conference on Neural Information Processing System, Whistler, British Columbia, Canada, December 09-11, 2003.
7. Wikipedia. Expectation–Maximization Algorithm. [Online]. Available: https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm
8. G. McLachlan and T. Krishnan. The EM Algorithm and Extensions, John Wiley & Sons, New York, 1997.
9. M. Wang, R. Martin, and G. Mao, “Nonsingular robust covariance estimation in multivariate outlier detection,” in Proc. International Conference on Scientific Computing, 2015.
10. G. J. McLachlan, “MIXFIT: an algorithm for the automatic fitting and testing of normal mixture models,” in Proc. the Fourteenth International Conference on Pattern Recognition, 1998, pp. 553-557.
11. A. Geitgey, “Machine learning is fun! Part 2,” Medium, Feb. 13, 2018.
12. L. Hardesty, “Explained: Neural networks,” MIT News, Feb. 13, 2018.
13. D. Dhanashri and S. B. Dhonde, “Speech recognition using neural networks: A review,” International Journal of Multidisciplinary Research and Development, June 2015.
14. 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, Jun. 2015.

15. G.Hinton,L.Deng,D.Yu,G.E.Dahl,A.-r.Mohamed,N.Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, 2012.
16. K. J. Han, S. Hahm, B.-H. Kim, J. Kim, and I. Lane, "Deep learning-based telephony speech recognition in the wild," in *Proc. Interspeech*, 2017.
17. A. H. Abdelaziz, S. Watanabe, J. R. Hershey, E. Vincent, and D. Kolossa, "Uncertainty propagation through deep neural net- works," in *Interspeech 2015*, 2015.
18. R. M. Neal, "Priors for infinite networks," in *Bayesian Learning for Neural Networks*. Springer, 1996.
19. J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein, "Deep neural networks as gaussian processes," 2017
20. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.
21. G.PundakandT.N.Sainath,"Highway-lstmandrecurrenthigh- way networks for speech recognition," in *Proceedings of Inter- speech*, 2017.
22. A. Damianou and N. Lawrence, "Deep gaussian processes," in *Artificial Intelligence and Statistics*, 2013.
23. C.E.RasmussenandC.K.Williams,*Gaussianprocessesforma- chine learning*. MIT press Cambridge, 2006.
24. D.Duvenaud,"Automaticmodelconstructionwithgaussianpro- cesses," Ph.D. dissertation, University of Cambridge, 2014.
25. J.La`zaro-Gredilla,C.E.Rasmussen,A.R.Figueiras-Vidal*etal.*, "Sparse spectrum gaussian process regression," *Journal of Ma- chine Learning Research*, 2010.
26. K. Cutajar, E. V. Bonilla, P. Michiardi, and M. Filippone, "Ran- dom feature expansions for deep gaussian processes," 2016.
27. Y. Gal, R. T. McAllister, and C. E. Rasmussen, "Improving pilco with bayesian neural network dynamics models," in *Data-Efficient Machine Learning workshop*, 2016.
28. A.Paszke,S.Gross,S.Chintala,G.Chanan,E.Yang,Z.DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differ- entiation in pytorch," 2017.
29. D. P. Kingma and J. Ba, "Adam: A method for stochastic opti- mization," 2014.
30. Funahashi, K. On the Approximate Realization of Continuous Mappings by Neural Networks / K. Funahashi // Neural Networks, 1989.
31. Driedger. J. & Müller. M. & Disch, S. (2014). Extending Harmonic-Percussive Separation of Audio Signals.

32. J. Martens, “Deep Learning via Hessian-free Optimization,” in *Proceedings of the 27th International Conference on Machine learning*, 2010.
33. N Morgan, “Deep and wide: Multiple layers in automatic speech recognition,” *IEEE Transactions on Audio, Speech, and Language Processing,* vol. 20, no. 1, jan. 2012.
34. N. Jaitly, P. Nguyen, A. Senior, and V. Vanhoucke, “An Application of Pretrained Deep Neural Networks To Large Vocabulary Conversational Speech Recognition,” Tech. Rep. 001, Department of Computer Science, University of Toronto, 2012.
35. Y. Bengio, R. De Mori, G. Flammia, and F. Kompe, “Global Optimization of a Neural Network - Hidden Markov Model Hybrid,” in *Proceedings of EuroSpeech,* 1991.