НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

«На правах рукопису»

УДК 518.977

До захисту допущено:

Завідувач кафедри

_____ О.Л. ТИМОЩУК

«___»_____20__ р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Системний аналіз фінансового ринку»

спеціальності 124 «Системний аналіз»

на тему: «Застосування нейронних мереж для вирішення задачі розпізнавання емоційного забарвлення тексту»

Виконав:
студент ІІ курсу, групи КА-02мп
Зайвелєв Юрій Ігорович _____

Керівник:
доцент кафедри ММСА,
к.ф.-м. н., доц., Яковлева А.П _____

Рецензент:
в.о. завідувач кафедри системного проектування
КПІ ім. Ігоря Сікорського, д. т. н., проф., Мухін В.Є. _____

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

Київ
2021

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

Рівень вищої освіти – другий (магістерський)

Спеціальність – 124 «Системний аналіз»

Освітньо-професійна програма «Системний аналіз фінансового ринку»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ О.Л. ТИМОЩУК

«___»_____20__ р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Зайвелєву Юрію Ірогоровичу

1. Тема дисертації: «Застосування нейронних мереж для вирішення задачі розпізнавання емоційного забарвлення тексту», науковий керівник дисертації Яковлева А.П., кандидат ф.-м. наук, доцент, затверджені наказом по університету від «_02_»_листопада_ 20_21_ р. №_3651-с__

2. Термін подання студентом роботи: 12 грудня 2021 року
3. Об’єкт дослідження: CNN мережі, LSTM мережі, наївний баєсівський класифікатор.
4. Предмет дослідження: класифікація текстів за емоційним забарвленням на класи – позитивні та негативні.
5. Перелік завдань, які потрібно розробити:
6. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо):
7. Дата видачі завдання: 5 вересня 2021

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання магістерської дисертації	Примітка
1	Отримання завдання	01.09.2021 – 05.09.2021	Виконано
2	Збір інформації	06.09.2021 – 12.09.2021	Виконано
3	Ознайомлення з літературою і підготовка теоретичної частини магістерської дисертації	13.09.2021 – 26.09.2021	Виконано
4	Аналіз вимог завдання, вибір методів і засобів розв'язання поставленої задачі	27.09.2021 – 3.10.2021	Виконано
5	Розробка програмного продукту	4.10.2021 – 31.10.2021	Виконано
6	Проведення аналізу ринкових можливостей стартап-проекту	1.11.2021 – 7.11.2021	Виконано
7	Отримання висновків після тестування	8.11.2021 – 15.11.2021	Виконано
8	Оформлення магістерської дисертації	15.11.2021 – 12.12.2021	Виконано
9	Отримання допуску до захисту та подача роботи до ДЕК	13.12.2021 – 14.12.2021	Виконано

Студент Ю.І.Зайвелєв

Науковий керівник дисертації А.П.Яковлева

РЕФЕРАТ

Магістерська дисертація: 108 с., 50 рис., 18 табл., 1 додаток, 7 джерел.

Тема магістерської дисертації «Застосування нейронних мереж для вирішення задачі розпізнавання емоційного забарвлення тексту».

Актуальність даної магістерської дисертації обумовлена тим що об’єм текстової інформації котрий потрібно оброблювати тільки збільшується тому є потреба в системі аналізу тексту такого напрямку.

Об’єктом дослідження є набір текстів, частини штучних нейронних мереж ,рекурентні нейронні мережі, конволюційні нейронні мережі, а також наївний беєсівський класифікатор.

Предметом дослідження є класифікація текстів за емоційним забарвленням на класи – позитивні та негативні.

Метою даної магістерської дисертації є аналіз та підвищення якості роботи та вибір найкращого рішення серед обраних кандидатів для вирішення поставленої задачі.

Для досягнення поставленої мети було виконано наступні задачі:

- Проведено ознайомлення з предметною областю та аналіз роботи нейронних мереж різних видів, а також архітектури.
- Розроблена модель для вирішення поставленої задачі – класифікації тексту за його тональністю.
- Виконані практичні експеременти та порівняння отриманих результатів з подальшим вибором найкращої мережі.

РЕКУРЕНТНІ НЕЙРОННІ МЕРЕЖІ, CNN, АНАЛІЗ ТЕКСТУ, КЛАСИФІКАЦІЯ, МАШИННЕ НАВЧАННЯ, MNB.

ABSTRACT

Master's dissertation: 108 pages, 50 figures, 18 tables, 1 appendix, 7 sources.

Theme of the master's dissertation "Application of neural networks to solve problems of recognition of emotional coloring of the text".

The relevance of this master's thesis is due to the fact that the volume of textual information that needs to be processed only increase, so there is a need for a system of text analysis in this area.

The object of research is a set of texts, parts of artificial neural networks, recurrent neural networks, convolutional neural networks, as well as a naive Bees classifier.

The subject of the study is the classification of texts by emotional color into classes - positive and negative.

The purpose of this master's dissertation is to analyze and improve the quality of work and choose the best solution among the selected candidates to solve the problem.

To achieve this goal, the following tasks were performed:

- Familiarization with the subject area and analysis of neural networks of different types, as well as architecture.
- Developed a model for solving the problem of the text in its key.
- Practical experiments and comparison of the obtained results with the subsequent selection of the best network.

RECURRENT NEURAL NETWORKS, CNN, TEXT ANALYSIS, CLASSIFICATION, MACHINE LEARNING, MNB.

ЗМІСТ

ВСТУП

Кількість нової інформації росте з дуже великою швидкістю що вимушує людину навчатися сприймати та аналізувати її швидше. За для вирішення цієї проблеми люди розробляють різні системи та методи корті дозволили вирішити задачу аналізу інформації. В наш час людство має достатньо знань для створення таких складних систем як нейронні мережі, котрі дають змогу спростити обробку інформації. На даному етапі розвитку цієї технології вони мають змогу аналізувати – текст, зображення та інше.

В наш час складно знайти – університет, установу чи школу котрі б не мали канал розповсюдження інформації – відео канал на “YouTube”, сторінка в “ Twitter” чи просто лендинг сторінка. Це дозволяє їм вести певний діалог зі споживачем, збирати відгуки про свою роботу – послуги, продукт та інше, а також зразу вносити корективи. Особливо цікавими для установ є відгуки, адже вони дають змогу корегувати свою роботу постійно та своєчасно, наприклад покращити обслуговування або змінити формат курсів врахувавши недоліки про котрі дізнались з відгуків. Але тут постає проблема аналізу великого об’єму інформації за емоційним забарвленням, адже потрібно не просто знайти відгук серед тисяч інших, але й оцінити його долю серед всіх інших. Вирішити цю проблему дають змогу нейронні мережі для аналізу емоційної тональності тексту, бо достатньо лишень двох категорій щоб оцінити те як Ваша аудиторія ставиться до Вашого продукту чи контенту.

Також така нейронна мережа дає змогу покращити роботу інформаційних систем, адже для навчання потрібна лише вибірка із коментарів певного ресурсу та різного емоційного забарвлення. Ця система дасть змогу пришвидшити час аналізу тексту, а також час прийняття рішення що є критичним для більшості установ. Також це дасть поштовх у якості та швидкості перекладу тексту та вилучення його сенсу.

У процесі виконання даної роботи цілю є створення декількох бінарних класифікаторів котрі будуть вирішувати задачу аналізу емоційного забарвлення тексту незалежно один від одного. Було визначено два класи до котрих нейронна мережа має віднести текст – позитивний та негативний.

Результатом будуть моделі нейронних мереж котрі можуть з деякою точністю визначити емоційну тональність тексту, котрі у результаті можна буде порівняти для визначення найкращого кандидата.

1 ОСНОВНІ СКЛАДОВІ ТА АЛГОРИТМИ НЕЙРОННИХ МЕРЕЖ

1.1 Основні складові нейронних мереж

Як відомо біологічні нейронні мережі - наш мозок наприклад стали прототипом для створення штучної нейронної мережі, котра за сенсом та складовими має у якомусь сенсі ті ж самі складові. Адже як біологічна нейронна мережа існує для вирішення наших фізіологічних задач, так і штучна виконує певні більш роздільні але складні задачі. Тому елементарною часткою у цій системі є як ми знаємо є нейрон (рисунок 1.1).

«Нейрон» у нейронній мережі — це математична функція, яка збирає та класифікує інформацію відповідно до певної архітектури. Мережа дуже схожа на статистичні методи, такі як підгонка кривої та регресійний аналіз.

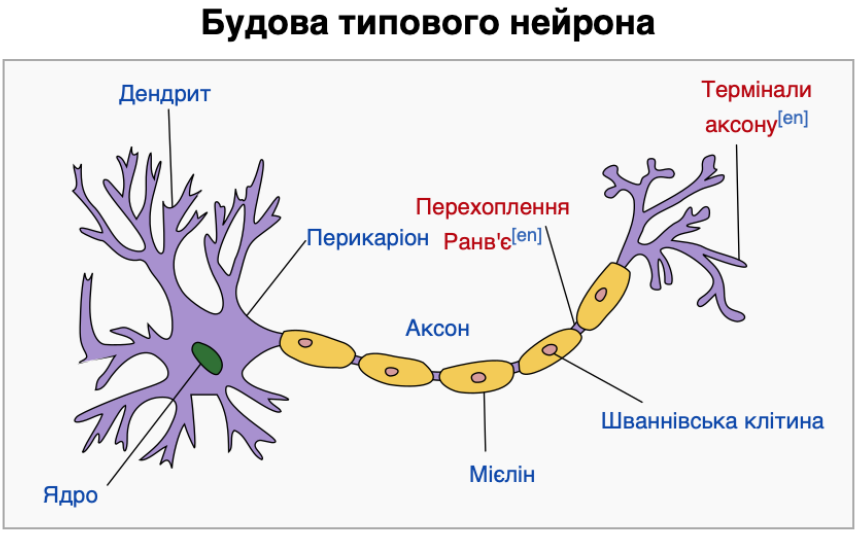


Рисунок 1.1 – будова нейрона

Нейрон у біологічній нейронній мережі працює таким самим чином – але у розрізі організму людини. Інформація отримує нейрон в синапсах на його дендрити. Кожен синапс являє собою з'єднання вхідного аксона від іншого нейрона з дендритом нейрона, представленого на малюнку (рисунок 1.1).

Відбувається електрохімічна передача в синапсі котра дозволяє передавати інформацію від одного нейрона до наступного. Інформація потім передається вздовж дендритів, поки не досягне тіла клітини, де підсумовування електричних імпульсів надходить до тіла, і виконується деяка функція цієї суми. Якщо ця функція перевищує певний поріг, нейрон повторно відправить сигнал (у вигляді хвилі іонізації) вздовж його аксона для зв'язку з іншими нейрони. Таким чином інформація передається від однієї частини мережі нейронів до іншої.

Коли ми вчимося, електричні сигнали посилаються між нейронами нашого мозку. Найбільше енергії потрібно під час першого проходження синапсу. Щоразу після цього з’єднання вимагає менше енергії. Ось як синапси ефективно полегшують як вивчення чогось нового, так і запам’ятовування того, що ми навчилися. Штучний синапс, на відміну від більшості інших версій мозкових обчислень, також виконує ці два завдання одночасно і робить це зі значною економією енергії.

Коротко опишемо біологічну будову синапсу. Синапс виникає між двома нейронами: пресинаптичним нейроном і постсинаптичним нейроном. Ви можете вважати пресинаптичний нейрон відправником, а постсинаптичний нейрон приймачем. Нижче наведено основні частини синапсу:

- Термінал аксона: кінець аксона пресинаптичного нейрона.
- Синаптична щілина: синаптична щілина - це проміжок між пресинаптичним і постсинаптичним нейроном, який може бути переповнений нейромедіатором.
- Дендритний остист: невеликий виступ із постсинаптичного дендрита, який зустрічається з пресинаптичним аксоном. Дендритні шипи пластичні — динамічно змінюються форми і розміри, з’являються і зникають. Вважається, що шипи є невід’ємною частиною навчання та пам’яті.
- Рецептори: білки, з якими зв’язується нейромедіатор.

Також існує штучний синапс заснований на конструкції батареї. Він складається з двох тонких, гнучких плівок з трьома клемами, з'єднаних електролітом солоної води. Пристрій працює як транзистор, причому один з клем контролює потік електрики між двома іншими.

Подібно до нейронного шляху в мозку, який підкріплюється навчанням, дослідники програмують штучний синапс, багаторазово розряджаючи та перезаряджаючи його. Завдяки цьому навчанню вони змогли передбачити в межах 1 відсотка невизначеної напруги, яка буде потрібна, щоб привести синапс у певний електричний стан, і, потрапивши там, він залишається в цьому стані. Іншими словами, на відміну від звичайного комп’ютера, де ви зберігаєте свою роботу на жорсткому диску, перш ніж вимкнути його, штучний синапс може відкликати своє програмування без будь-яких додаткових дій або частин.

Синапс нейронної мережі – це такий собі зв'язок між вузлами або нейронами в штучній нейронній мережі (ANN). Подібно до біологічного мозку, зв’язок контролюється силою або амплітудою зв’язку між обома вузлами, яка також називається синаптичною вагою. Кілька синапсів можуть з’єднувати одні й ті ж нейрони, причому кожен синапс має різний рівень впливу (тригера) на те, чи цей нейрон «запускається» і чи активує наступний нейрон.

У ШНМ [2] кожен нейрон визначається через його вхід, функцію активації та вихід. У термінології машинного навчання синапс часто називають вузлом. Однак, на відміну від біологічних нейронів, ці штучні нейрони не запускаються, а виводять значення з безперервної функції. Вибір функції активації буде контролювати, наскільки цей вихід поводиться як біологічний нейрон. Математично синапс представляється у вигляді вагового вектора. Вагові коефіцієнти контролюють, як вихідні дані попереднього шару подаються через функцію активації для кожного вузла. Використовуючи зручну масштабованість лінійної алгебри, усі вагові вектори синапсів для всього вузлового шару можна представити як одну вагову матрицю. Отриманий результат усіх цих вузлів використовується як вхід для наступного шару вузлів і так далі по всьому «мозку» нейронної мережі, поки не буде досягнуто кінцевого вихідного шару.

Синапс визначається цими ваговими коефіцієнтами та виражається алгебраїчно (для лінійного синапсу).

Зазвичай додають зміщення, а вихідні значення будуть передані через вибір функції активації перед використанням як вхідні дані в наступний шар.

Функція активації вирішує, чи слід активувати нейрон чи ні. Це означає, що він вирішуватиме, чи важливий вхід нейрона в мережу чи ні в процесі передбачення за допомогою простіших математичних операцій.

Роль функції активації полягає в отриманні виводу з набору вхідних значень, що подаються до вузла (або шару). Основна роль функції активації полягає в перетворенні підсумованого зваженого входу з вузла у вихідне значення, яке буде передано на наступний прихований шар або як вихід. Функції активації вводять додатковий крок на кожному рівні під час прямого поширення, але його обчислення того варте. Ось чому—припустимо, у нас нейронна мережа працює без функцій активації.

У цьому випадку кожен нейрон буде виконувати лише лінійне перетворення на входах, використовуючи вагові коефіцієнти та зміщення. Це тому, що не має значення, скільки прихованих шарів ми приєднаємо до нейронної мережі; всі шари будуть вести себе однаково, оскільки композиція двох лінійних функцій сама є лінійною функцією. Хоча нейронна мережа стає простішою, вивчення будь-якої складної задачі неможливо, і наша модель буде просто моделлю лінійної регресії.

Нижче наведені три основні функції активації.

Сигмоїдальна активаційна функція (рисунок 1.2,1.3). Функція є однією з найвикористовувагіших серед інших функцій активації, це обумолвено деякими її характеристиками. Вона зазвичай використовується для моделей, де ми повинні передбачити ймовірність як результат. Оскільки ймовірність чого-небудь існує лише в діапазоні від 0 до 1, сигмоїдальна функція є правильним вибором через її діапазон. Функція диференційована і забезпечує плавний градієнт, тобто запобігає стрибкам вихідних значень. Це обумовлено S-подібною формою сигмовидної функції активації. Математично її можна представити

формулої нижче, а також видобразити у вигляді графіку(рисунок 2) та (рисунок 1.3):

$$Sigmoid\ /\ Logistic$$

$$f(x) = \frac{1}{1 + e^{-x}}$$

Рисунок 1.2 – функція сигмоїдальна активаційна функція

Рисунок 1.3 – графік сигмоїдальної функції

Гіперболічний ReLU означає Rectified Linear Unit. Незважаючи на те, що він створює враження лінійної функції, ReLU від неї можна взяти похідну функцію, а також вона дозволяє здійснювати зворотне поширення, водночас роблячи його обчислювально ефективним. Головна проблема у цієї функції полягає в тому, що функція ReLU не може активувати всі нейрони одночасно. Нейрони можуть бути дезактивовані лише за тієї умови що вихідний результат функції (лінійного перетворення) менше від нуля. Математично її можна представити формулої нижче, а також видобразити у вигляді графіку(рисунок 1.4) та (рисунок 1.5):

$$ReLU$$

$$f(x) = max\ (0, x)$$

Рисунок 1.4 – функція ReLu активаційна функція

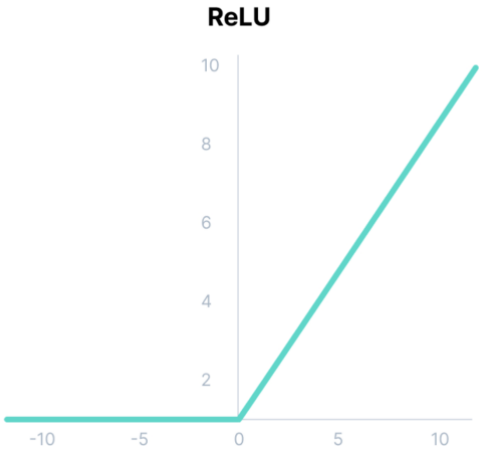


Рисунок 1.5 – функція ReLu активаційна функція

Переваги використання ReLU у якості функції активації полягають у наступному. Оскільки активується лише певна кількість нейронів, функція ReLU є набагато більш ефективною в обчислювальному відношенні порівняно з функціями сигмовидної і тангенсу.

ReLU прискорює зближення градієнтного спуску до глобального мінімуму функції втрат завдяки своїй лінійній властивості, що не має насичення.

Обмеження, з якими стикається ця функція звучать наступним чином. Від’ємна сторона графіка приводить до нульового значення градієнта. З цієї причини під час проходження процесу зворотного поширення ваги - зміщення для деяких нейронів не оновлюються. Це може створити мертві нейрони, корті у результати ніколи не активуються.

Розглянемо наступну функції активації.

Гіперболічний тангенс - Tanh. Нижче бачимо вираз котрим можна презентувати цю функцію, а також її графік(рисунок 1.6,1.7). Функція Tanh досить схожа на функції сигмовидної або логістичної активації та навіть має таку ж саму S-подібну форму з різницею лише в діапазоні значень даних від -1 до 1. У Tanh, чим більше вхід (більше позитивне), тим ближче вихідне значення буде до одениці, тоді як чим менше вхід (білье негативне), тим ближче вихід буде до мінус одного. Вихід функції активації Tanh має центр в нулі, отже, ми можемо легко представити вихідні значення у трьох варінатах - як сильно негативні, нейтральні або ж сильно позитивні. Дуже часто ця функція використовується в прихованих шарах нейронної мережі, оскільки значення гіперболічного тангенсу лежать від -1. Виходячи з цього ми маємо що - середнє значення для прихованого шару виявляється рівним 0 або дуже близьким до нього. Це допомагає центрувати дані та значно полегшує навчання для наступного рівня.

$$Tanh$$

$$f(x) = \frac{\left(e^{x} - e^{-x}\right)}{\left(e^{x} + e^{-x}\right)}$$

Рисунок 1.6 – функція гіперболіний тангенс

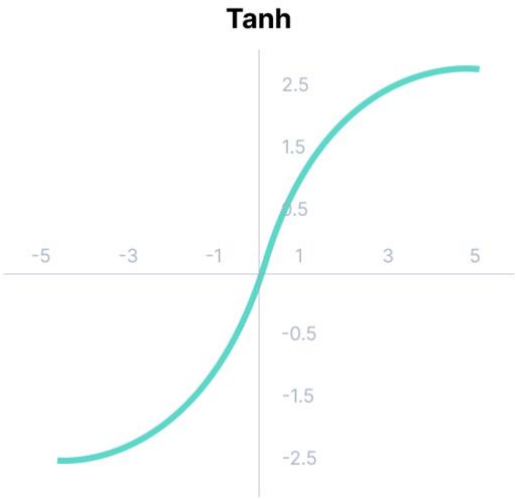


Рисунок 1.7 – графік гіперболічного тангенсу

Функція одиночний стрибок є кусково-лінійною фукцією. Функція бінарного кроку залежить від порогового значення, яке визначає, чи слід активувати нейрон чи ні. Вхід, що подається на функцію активації, порівнюється з деяким пороговим значенням, якщо вхідне більше, ніж це, то нейрон буде активуватися, а в іншому випадку він деактивується, тобто його вихідне значення не передається до наступного прихованого шару. Як результат якщо значення на вході менше ніж задане порогове значення, то функція приймає своє мінімальне значення, у іншому випадку максимальне(рисунок 1.8,1.9).

Binary step

$$f(x) = \begin{cases} 0 & for\ x < 0 \\ 1 & for\ x \geqslant 0 \end{cases}$$

Рисунок 1.8 – функція одиночний стрибок

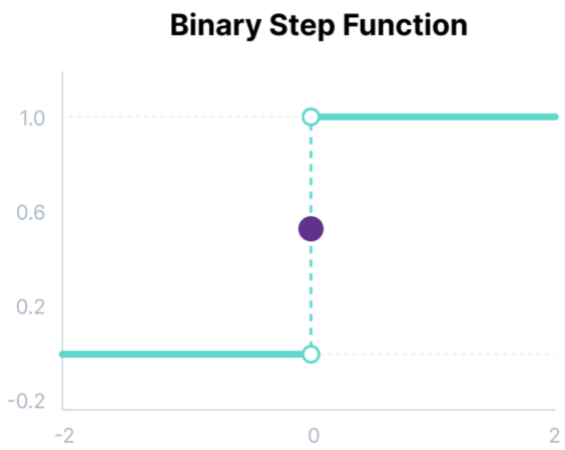


Рисунок 1.9 – графік функції одиночний стрибок

Персептрон - це тип одного з алгоритмів контрольованого навчання бінарних класифікаторів. Варто ввести поняття контрольованого навчання - це тип машинного навчання, який використовується для вивчення моделей на основі розмічених навчальних даних. Це дозволяє прогнозувати вихід для майбутніх або невизначених даних. Цей метод був винайдений вченим Френком Розенблатом у 1957 році. Даний алгоритм дає змогу обробляти та навчати нейрони та елементи з в навчальному набори по одному. Розенблат запропонував правило навчання персептрону котре засноване на оригінальному нейроні.

Існує загалом два типи персептронів – це одношарові та багатошарові. Одношарові персептрони мають можливість вивчати тільки лінійно роздільні шаблони. На відміну від першого типу, багатошарові персептрони або так звані багатошарові нейронні мережі – котрі мають два або більше шарів, ці мережі мають більшу обчислювальну потужність.

Правило навчання Персептрону гарантує, що алгоритм автоматично продукувати оптимальні вагові коефіцієнти. Потім вхідні характеристики перемножуються на ці ваги, щоб визначити, спрацьовує нейрон чи ні. Персептрон отримує на свій вхідний шар кілька вхідних сигналів, і у тому випадку коли сума вхідних сигналів перевищує певний поріг, він або гененерує сигнал, або не повертає вихід. У контексті контрольованого навчання та класифікації це потім можна використовувати наприклад для прогнозування класу вибірки.

Якщо розглядати роботу персептрону більш детально то спочатку подивимось на просту схему поетапної роботи персептрона. Персептрон орімує на свій вхідний шар значення $x1$, $x2$ ... і повертає бінарний результат – 1 або 0. Розенблат у своєму правилі навчання запропонував ввести так званні ваги – це звичайні числа, що відображають велечину вкладу кожного вхідного значення в кінцевий результат(0 або 1). Зважена сума (або ваги) порівнюється із завчасно заданим граничним значенням, котре теж є одним із параметрів нейрона, та виходячи з результату порівняння визначається, буде повернуто 0 або 1 на вихід. На зображені 1.10 можемо бачити схему одношарового персептрону(рисунок 1.10).

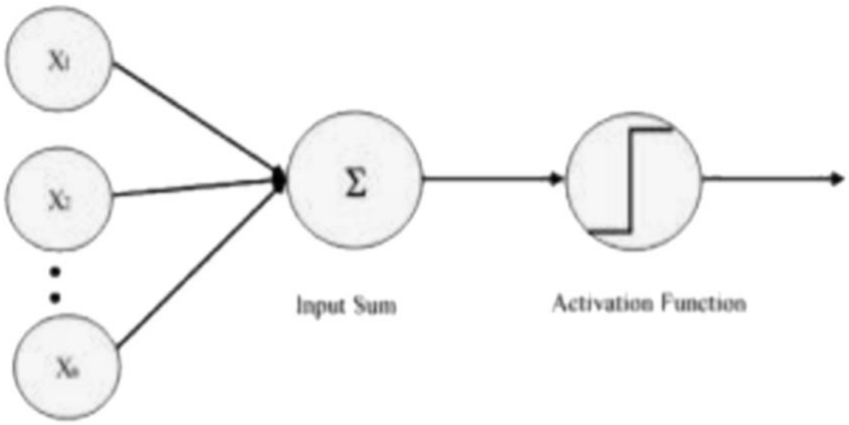


Рисунок 1.10 – одношаровий персептрон

Персептрон приймає дані котрі приходять на вхід, та потім модерує їх із певними значеннями ваги, та потім застосовує до них функцію перетворення для отримання фінального результату. На рисунку 1.11 зображено персептрон котрий надає булевий вивід.

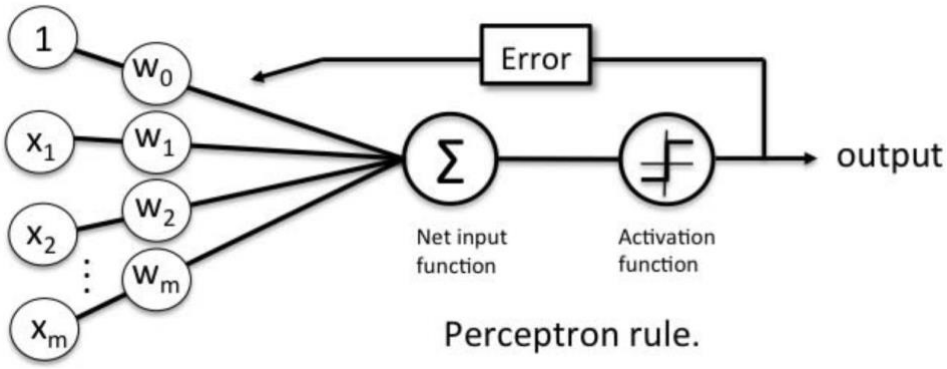


Рисунок 1.11 – персептрон з булевим виводом

Ваги або так звані параметри – це значення котрі представляють силу взаємодії між елементами. Тож якщо вага від вузла 1 до другого вузла має більшу величину, це значить що перший нейрон має більший вплив на нейрон два. Вага зменшує важливість вхідного значення. Вага, близька до нуля, означає, що зміна цього входу не змінить вихід. Від'ємні ваги означають, що збільшення цього входу відповідно зменшить вихід. Таким чином можемо зробити висновок що вага визначає, який вплив матиме вхід на вихід.

Булевий вихід базовий на таких вхідних даних, такими можуть бути такі дані як – вік, кредитний рейтинг, сімейний статус або заробітна плата і тому подібні параметри. Він має лише два значення так і ні або ж 0 чи 1. Таким чином функції сумації множить всі уведені значення x на значення ваг, а потім додає результат множення між собою.

Розенблат свого часу вигадав використовувати ваги – числа як ми визначили раніше, що визначають важливість вкладу кожного значення входу до кінцевого результату. Так звану зважену суму або ж "NET" (або так званні ваги) у ході навчання порівнюються з граничним значенням (T - threshold), і за результатами вище визначеного порівняння вирішуєтся, чи буде виданий нуль або один (рисунок 1.12).

$$OUT = \begin{cases} 1, NET \geq T \\ 0, NET < T \end{cases},$$

Рисунок 1.12 – принцип використання ваг

Є певні параметри за котрими можна класифікувати персептрон. Такими параметрами є активаційна функція, те як розповсюджується сигнал та за кількістю прихованих шарів.

Саме персептрон може бути класифікований за такими значеннями вище наведених параметрів:

- За так званою граничною функцією активації.
- Персептрон з прямим типом розповсюдження сигналу.
- Персептрон котрий має лише єдиний прихований шар.

Сам процес навчання персептрону є в тому щоб змінювати значення ваг у процесі навчання. Існують наступні види персептронів наведені нижче.

Одношаровий персептрон влаштований таким чином що має елементи на вході котрі пов'язані з вихідними вище визначеною системою ваг. Він за своєю архітектурою являеєтся мережею прямого поширення, такі мережі відрізняються тим що не маюють зворотнього зв’язку через те що сигнал поширюється тільку у напряму від вхідного шару до вихідного з якого ми вже отримуємо саме результат обробки сигналу вагами.

Багатошаровий персептрон, котрий визначають як багатошаровий персептрон по Розенблату, його особливість в тому що він має додаткові приховані шари.

Багатошаровий персептрон, але вже по Румельхарту також маж як і по Розенблату додаткові приховані шари, але навчання проводиться з використанням методу що називається зворотнє поширення помилки. Під час навчання нейронної мережі методом градієнтного спуску кожного разу мі обчислюємо функцію втрат, котра показує, наскільки прогнози мережі розходяться з істинними данними. Зворотне поширення помилки дозволяє

обчислювати градієнт функції втрат по відношенню до кожного з ваг мережі. Це дозволяє корегувати кожну вагу окремо від інших, це потрібно для того щоб поступово зменшувати функцію втрат протягом багатьох тренувань.

Останнім типом є - персептрон котрий включає в себе лишень один прихований шар.

З математичного уявлення можна сказати, що логіка порогового значення, яку використовує персептрон, дуже жорстка. Давайте подивимося на сувору логіку встановлення порогів на прикладі. Розглянемо процес прийняття рішення людиною, чи хоче він/вона придбати автомобіль чи ні, виходячи лише з одного входу X1—Заробітна плата та встановлюючи поріг $b(W_o) = -10$ і вагу $W_1 = 0,2$. Вихід з моделі персептрона буде виглядати так, як показано на рисунку 1.13 нижче.

Salary (in thousands)	Can buy a car?
80	1
20	0
65	1
15	0
30	0
49	0
51	1
87	1

Рисунок 1.13 – принцип використання ваг

Нульові точки вказують на те, що людина не купує автомобіль, а одиничні — на те, що людина хотіла б купити автомобіль. Чи не дивно, що людина з 50,1 тис. купить машину, а хтось із 49,9 тис. не купить машину? Невелика зміна вхідного сигналу до персептрона іноді може призвести до повного зміни вихідного сигналу, скажімо, від 0 до 1. Така поведінка не є характеристикою конкретної проблеми, яку ми вибираємо, або питомої ваги та порогу, який ми вибираємо. Це характеристика самого нейрона персептрона, який поводиться як

ступінчаста функція. Ми можемо подолати цю проблему, представивши новий тип штучного нейрона, який називається сигмовидним нейроном.

Представляємо сигмовидні нейрони, де вихідна функція набагато плавніша, ніж крокова функція. У сигмовидному нейроні невелика зміна входу викликає лише невелику зміну на виході на відміну від ступінчастого виходу. Існує багато функцій з характеристикою «S»-подібної кривої, відомих як сигмовидні функції. Найпоширенішою функцією є логістична функція.

Ми більше не бачимо різкого переходу на порозі b. Вихід сигмовидного нейрона не дорівнює 0 чи 1. Натомість це реальне значення між 0–1, яке можна інтерпретувати як ймовірність.

Цей факт дозволяє мережі що скомпонована з сигмоїдальних нейронів навчатися. На вхід сигмоїдальна нейрона подаються будь-які значення котрі лежать у проміжку від 0 до 1. На виході також видається значення у діапазоні від 0 до 1, так як в якості активаційної функції береться сигмоїда, котра у свою чергу є нелінійною.

Чим більше параметр β (параметр котрий відображає нахил сигмоїдальної функції активації), тим більш крутим є графік сигмоїдальної функції. При умові $\beta \to \infty$ сигмоїда прямує до функції Ховісайда.

Досить важливою властивістю сигмоїдальної функції є її диверенційовність. Використання функції котра є неперервною у якості функції активації дозволяє використовувати у процесі навчання градієнтні методи.

Нейрони мережі модна певним чином розділити на групи за їх функціональним використанням, або якщо простіше за їх розміщенням в мережі:

Першим типом є – вхідні нейрони, це нейрони котрі першими отримують вхідний вектор даних;

Другим типом є – проміжні нейрони, це нейрони котрі виконують основну ідеологічну функцію нейронної мережі, а саме обчислювальні операції;

Третім типом є – вихідні нейрони, це нейрони котрі віддають результат обчислень нейронної мережі, фактично вони виконують обернену функцію першого типу нейронів.

Четвертим типом є нейрон зміщення або так званий bias нейрон про котрий теж варто сказати, адже він виконує досить важливу функцію і застосовується у великій кількості нейронних мереж. Якщо коротко описувати місце цього нейрону у мережі то він не має входу, тож у нього відсутні згадувані раніше вхідні синапси, а також його вхід та вихід за будь-яких умов буде дорівнювати одиниці. Також нейрон зміщення має одне обмеження, котре варто знати при побудові власної нейронної мережі – ваша мережа може мати один bias нейрон або на кожному шарі, або не мати таких нейронів взагалі, тож не можна розраховувати на часткове розміщення нейронів такого типу в шарах нейронної мережі. (червоним кольором на зображенні виділено ті зв'язки та позиції нейронів котрих бути не може) (рисунок 1.14). Для того щоб зрозуміти роль цього нейрону розглянемо коротко як працює мережа та функція активації. Функція активації в нейронних мережах приймає вхідне значення «x», помножене на вагу «w». Зміщення котре нам дає bias нейрон дозволяє зміщувати функцію активації, додаючи константу (тобто задане зміщення) до входу. Зміщення в нейронних мережах можна розглядати як аналогічну роллю константи в лінійній функції, коли лінія ефективно транспонується значенням константи. На рисунку 1.15 можемо побачити як зміниться функція активації при додаванні bias нейрону.

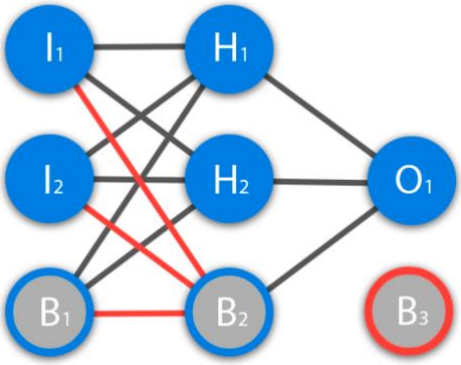


Рисунок 1.14 – функції активації з bias

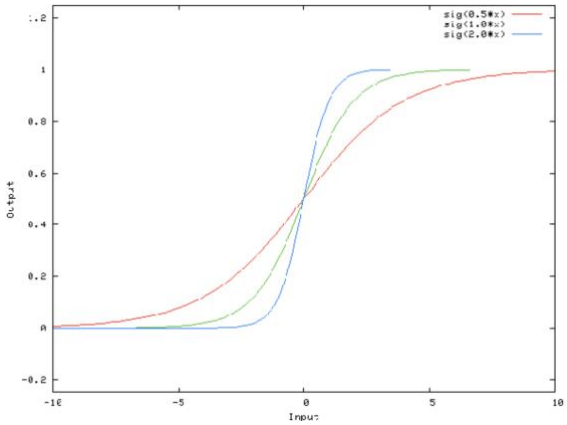


Рисунок 1.15 – схема нейронної мережі

Також варто зазначити що введені нами вище нейрони зміщення дають змогу вийти з ситуації коли коли всі вхідні нейрони на вхід отримують нуль і не дивлячись на те які у є них ваги, вони все таки віддають на наступний шар теж нуль, але в тому випадку коли ми маємо в шарі нейрон зміщення бо тоді ми додамо значення нейрону зміщення і отримаємо прийнятний результат. Оскільки нейронна мережа має свої гіперпараметри, визначення дамо далі, то те чи є в нас нейрон зміщення, чи він відсутній є саме одним з таких гіперпараметрів мережі. Фактично ми самі при навчанні нашої мережі маємо прийняти рішення – використовувати або ні нейрон зміщення в шарах мережі, найкращим рішенням буде провести навчання мережі з нейроном зміщення а потім без, та порівняти отримані результати для того щоб визначитись з остаточною архітектурою мережі. Оскільки фактично його вихід дорівнює одиниці то можемо просто уявляти що ми маємо додатковий синапс з вагою та потім додавати його вагу до зваженої суми котру отримуємо на виході шару, не згадуючи про самий нейрон зміщенняю.

Частою практикою є не зображати нейрон зміщення на схемі, та просто враховувати його при обчисленнях вхідного значення шару, як на прикладі (рисунок 1.16):

$$input = H1*w1+H2*w2+b3$$

$$b3 = bias*w3$$

Рисунок 1.16 – приклад використання нейрону зміщення

1.2 Архитектура нейронних мереж

Самим примітивним елементом штучної нейронної мережі є саме штучний нейрон котрий ми розглядали раніше. Саме ці елементарні одиниці штучної нейронної мережі пов’язані зв’язками та у результаті утворюють нейронну мережу. А вже сама штучна нейронна мережа дає змогу нам дуже ефективно обробляти інформацію та пристосовуватись до змін умов в котрі її ставлять – наприклад зміна вхідних даних або очікуваний результат. В момент роботи штучної нейронної мережі відбуваються певні перетворення вхідних даних, або як ми визначили раніше вхідного вектора та видавання результату або так званного вихідного сигналу. Сам процес обчислень визначається та повністю залежить від вхідних даних та очікуваного результату від мережи що в свою чергу визначає архітектуру мережі, її струтуру і метод обраний для її тренування та валідації, ці параметри в свою чергу тягнуть за собою визначення з характеристиками нейронів мережі. Тож визначимо основні характеристики нейронної мережі.

Базована на типах зв’язку між нейронами – прямого поширення при якому нейронна мережа не має циклів і сигнал поширюється тільки вперед, та рекурентні – в яких є певний зворотній зв'язок для того щоб корегувати ваги нейронів.

Базована на кількості шарів прихованого шару котрі є в штучній нейронній мережі. Тут класифікують два типи мереж – з одним прихованим шаром та з декількома прихованими шарами, одношаровий да багатошаровий персептрон відповідно.

Базовані на різній природі ваг – так ми маємо мережі з постійними значеннями ваг та штучній нейронні мережі в котрих ваги змінюються при тренуванні мережі

Базовані на елементі пам’яті – маємо статичні та динамічні, такими є мережі з прямим та оберненим поширенням помилки відповідно.

Також нейронні мережі відрізняються їх парадигмою котра визначає спосіб використання та спосіб навчання нейронної мережі – наразі відомо три основні парадигми навчання нейронних мереж – це навчання з вчителем, навчання без вчителя, а також навчання з підкріпленням.

Також нейронні мережі розділяють за їх структурою – тим як нейрони пов’язані між собою та взаємодіють.

Так само нейронні мережі градують по їх архітектурі. Поняття архітектура нейронної мережі пов’язує в собі декілька характеристик мережі котру можна створити – це типи нейронів за їх характеристиками, та те як вони взаємодіють у цій нейронній мережі. Треба прийняти до уваги один важливий факт, що на одній і тій самій архітектурі нейронної мережі можуть бути використані різні парадигми, і навпаки.

Виділяють наступні основні архітектурні рішення нейронних мереж[6]:

- Персептрон, будова та теорію котрого ми розглядали раніше.
- Нейронні мережі прямого поширення – всі нейрони повністю пов’язані між собою, мережа не має циклів, а також вона має лише один прихований шар.

Глибокі мережі прямого поширення – це тип мереж аналогічний другому, але має більше прихованих шарів. Раніше було неможливо навчати таку мережу, адже час навчання ріс експоненційно при збільшенні кількості прихованих

шарів. Але зараз за допомогою сучасних підходів до навчання є можливість закладати більше шарів та навчати такі мережі і отримувати результати кращі ніж у мереж з одним шаром.

Рекурентні нейронні мережі представляють різні типи клітин - повторювані клітини. Першою мережею такого типу була так звана мережа Йордана, коли кожна з прихованих клітин отримувала свій вихід з фіксованою затримкою — одну або кілька ітерацій. Крім того, це було як звичайний FNN. Цей тип NN в основному використовується тоді, коли важливий контекст — коли рішення з минулих ітерацій або вибірок можуть вплинути на поточні.

Загалом існує ще багато архітектур нейронних мереж, але наведені вище це основні, серед не названих є – CNN, LSTM, мережі Хопфілда, Кохонена та інші. Схеми основних мереж зображено схемі (рисунок 1.17)

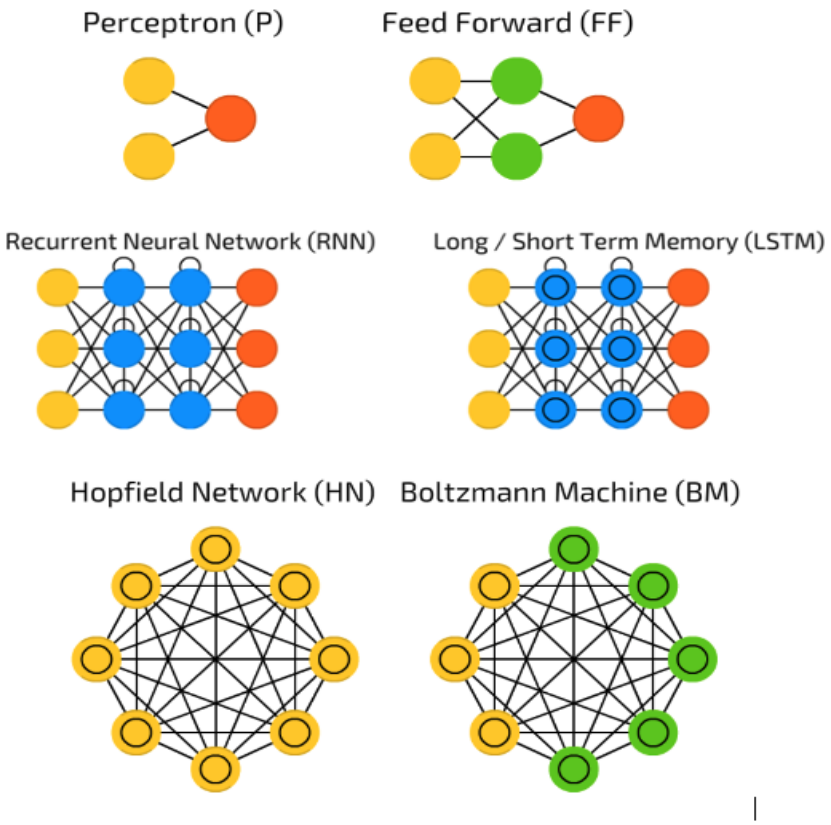


Рисунок 1.17 – Типи нейронних мереж

Для прикладу та більшого розуміння пропоную розглянути процес навчання нейронної мережі з учителем у вигляді задачі[7]. Нам на вході дано досить багато прикладів для тренування мережі X з лейблами або так званими мітками Y. Нейронні мережі визначають нелінійну гіпотезу hW, b (x) з параметрами W і b (рисунок 1.18).

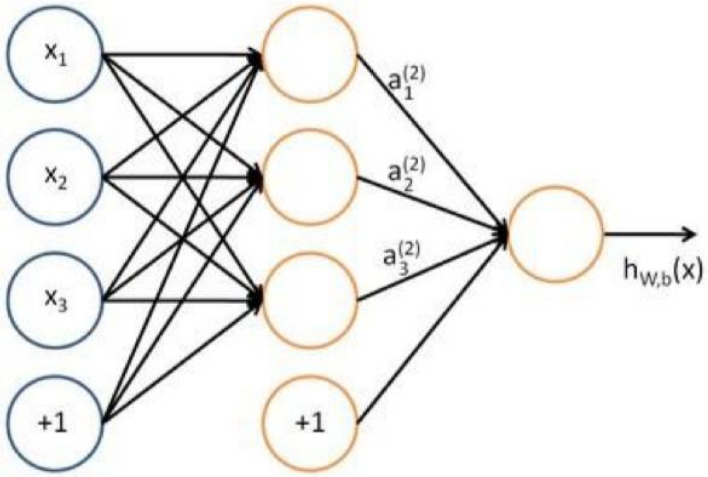


Рисунок 1.18 – приклад мережі для задачі

Як ми визначили раніше перший щар нейронної мережі називається вхідним та приймає вектор тренувальних, а далі і тестових даних. Останій шар називають вихідним, адже він повертає нам результати обчислень проведених нейронною мережею. Шар розміщений посередині є прихованим шаром котрий ми також визначили раніше, називається так через те, що його значення не спостерігаються під час навчання мережі. Таким чином в данній мережі ми має таку архітектуру - це шар входу, нейрони прихованого , а також нейрони вихідного шару. Покладемо що nl – це значення котре містить сумарну кількість шарів даної мережі, їх три – один вхідний, прихований та вихідний. З параметрів мережі маємо (W, b) = (W (1), b (1), W (2), b (2)). Будемо позначати результуюче значення використання активаційної функції як аі для і-ого елемента. Таким чином маємо наступні результати:

$$a_1^{(2)} = f\left(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_{(1)}^{1}\right) \quad (1.1)$$

$$a_2^{(2)} = f\left(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_{(1)}^{2}\right) \quad (1.2)$$

$$a_3^{(2)} = f\left(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_{(1)}^{3}\right) \quad (1.3)$$

де $a_i^{(j)}$ – значення кожного виходу мережі;

W – значення ваг кожного нейрону;

b – значення bias нейрону (нейрону зміщення).

З цього прикладу добре бачимо що нейроні мережі прямого поширення носять таку назву бо вони використовують для кожного входу наступного шару вихід попереднього.

1.3 Навчання нейронної мережі

Визначимо спеціальні поняття котрі фігурують у навчанні нейронних мереж.

- Епоха – так називають прямий і зворотний прохід усім тренувальним даним котрі обрані для навчання мережі.
- Розмір серії (англійською batch) – фіксована кількість визначених для тренування мережі наборів даних для однієї ітерації - прямого і зворотного проходів.
- Кількість ітерацій – це та сама кількість проходів: кожен прохід використовує тренувальні дані для мережі. Один прохід дорівнює прямому проходу та плюс зворотній прохід.

Тож наприклад маючи дві тисячі прикладів, batch буде рівним 1000, що у результаті приводить до того що нам буде потрібно дві ітерації, щоб завершити одну епоху.

Навчання штучних нейронних мереж це багато параметрична задача нелінійної оптимізації якщо подивитися з точки зору саме математики.

Розглянемо досить важливий алгоритм у навчанні нейроних мереж – алгоритм зворотного поширення помилки.

Поняття алгоритму зворотного поширення помилку був вперше введений у 1970-х роках, але на той час він не отримав широкого поширення та уваги. Але у 1986 році завдяки статті Девіда Румельхарта, Джефрі Хінтона та Рональда Вільямса у котрій було описано декілька нейронних мереж де зворотнє поширення помилки поширення помилки працює суттєво швидше ніж поширені в той час підходи, що дає змогу вирішувати більшу множину задач, алгоритм став більш поширеним.

Алгоритм зворотнього поширення помилки використовує математичний метод градієнту що дає змогу корегувати стратегію підбору ваг для мережі. Ефективність такого підходу обумовлена тим що функція котра є цільовою для методу зазвичай є неперервною оскільки вона представляє собою квадратичну різницю між фактичним і очікуваним значенням мережі. Для отримання результату при використанні даного методу ми маємо обчислити вектор градієнта що кожного з параметрів для всіх шарів мережі.

Загальний вигляд алгоритму:

- Ми повинні задати початкові значення ваг випадковими значеннями котрі є досить близькими до нуля.
- Передати на перший(вхідний шар) мережі вектор тренувальніх даних для навчання мережі
- Після проведення обчислень мережею, отримати вихідні значення.
- Обчислити значення різниці між реальним значенням та тим що ми отримали на вихідному шарі мережі.

- Ввести зміни у ваги мережі для того щоб відкорегувати результат на наступній ітерації.
- Всі наведені вище кроки потрібно повторювати у такому самому порядку до моменту поки не отримуємо прийнятне значення помилки.

Середня квадратична помилка – це сума квадратів різниць між прогнозованим і істинним значенням. І вихід є одним числом, що представляє вартість

Значення квадратичної помилки для цільової функції (squared-error cost function) для одного поданого на вхід нейронної мережі вхідного прикладу буде визначатись наступним чином:

$$J(W, b, x, y) = ½ \|h_{w,b}(x) - y\|^2 \tag{1.4}$$

де h – значення вихідного шару;

y – значення лейблів прикладів.

Якщо ж розглянути більш загальний та складний випадок то для m поданих на вхід прикладів, наша цільова функція буде обчислюватись наступним чином:

$$J(W, b) = \left[\frac{1}{m}\sum_{i=1}^{m} J(W, b, x_i, y_i)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}(W_{ij}^{l})^2 \tag{1.5}$$

де h – значення вихідного шару;

y – значення лейблів прикладів.

Очевидно, що перший доданок формули (1.5) це сума квадратів помилок прикладів, а другий доданок це так званий член регуляції котрий дає змогу нам зменшити значення ваг і таким чином завадати появі перенавчання. Введений вище параметр регуляризації ваг котрий позначається як λ зазвичай використовують для перевірки відносної значущості двох частин даного виразу.

При вирішення типопої задачі бінарної класифікації як очевидно у представлений нуль або одни, адже розглянута нами раніше сигмоїдальна функція активації можем приймати значення в межах [0; 1], але варто зазначити що при застосуванні функції гіперболічного тангенса нашими маркерами класів були б -1 і 1. Цілю цієї задачі є мінімізувати значення J (W, b). При навчанні штучної нейронної мережі потрібно задати кожен з параметрів W(l) і b(l) досить малими та випадковими величинами котрі як ми встановили раніше - близькі до нуля, а потім застосувати алгоритм оптимізації.

За кожної ітерації методу градієнтного спуску параметри оновлюватимуться таким чином як вказано на рисунку (рисунок 1.19):

$$\mathrm{W}_{\mathrm{ij}}^{(\mathrm{l})} = \mathrm{W}_{\mathrm{ij}}^{(\mathrm{l})} - \mathrm{a}\frac{\partial}{\partial \mathrm{W}_{\mathrm{ij}}^{(\mathrm{l})}}\mathrm{J}(\mathrm{W},\mathrm{b}) \tag{1.6}$$

$$\mathrm{b}_{\mathrm{i}}^{(\mathrm{l})} = \mathrm{b}_{\mathrm{i}}^{(\mathrm{l})} - \mathrm{a}\frac{\partial}{\partial \mathrm{b}_{\mathrm{i}}^{(\mathrm{l})}}\mathrm{J}(\mathrm{W},\mathrm{b}) \tag{1.7}$$

де α – параметр швидкісті навчання;

b – параметр зміщення;

J – цільова функція .

$$\frac{\partial}{\partial W_{ij}^{(l)}}J(W,b) = \left[\frac{1}{m}\sum_{i=1}^{m}\frac{\partial}{\partial W_{ij}^{(l)}}J\left(W,b,\ x^{i},\ y^{i}\right)\right] + \lambda W_{ij}^{(l)}$$

$$\frac{\partial}{\partial b_{i}^{(l)}}J(W,b) = \frac{1}{m}\sum_{i=1}^{m}\frac{\partial}{\partial b_{i}^{(l)}}J\left(W,b,\ x^{i},\ y^{i}\right)$$

Рисунок 1.19 – знаходження похідних

Функція перехресної ентропії зазвичай використовують у якості функції втрат: y'- передбачені мережею значення, yi – очікувані еталоні значення для перевірки:

$$L(x,y) = -\frac{1}{n}\sum_{i=1}^{n} y^i \log a\left(x^i\right) + \left(1 - y^i\right) \log 1 - \ a\left(x^i\right) \tag{1.8}$$

де a – отриманий вихід шару;

y – перевірочні потрібні значення;

n – загальна кількість ваг.

L1-регуляризація: при цьому відбувається зміна нерегуляризованої цільової функції способом додавання суми абсолютних значень ваг:

$$J = J_0 + \frac{\lambda}{n}\sum_W |W| \tag{1.9}$$

де n – загальна кількість ваг;

λ - встановлений параметр регуляції;

W – ваги.

У процесі використання L1-регуляризації буде відбуватися стягування одного або більше вагових коефіцієнтів до нуля, тому введена функція більше не потрібна. Наведений вище ефект називають селекцією функцій (feature selection);

L2-регуляризація (також встановлена як weight decay) На відміну від L1-регуляризації, в L2 ваги будуть зменшуватись на величину котра у свою чергу пропорційну кількості ваг:

$$J = J_0 + \frac{\lambda}{2n}\sum_W W^2 \tag{1.10}$$

де n – загальна кількість ваг;

λ - встановлений параметр регуляції;

W – ваги.

Введемо поняття dropout в навчанні нейронної мережі. Справа в тому що досить великі нейронні мережі котрі навчають на недостатньо великих наборах тренувальних даних можуть перейти у стан котрий називають перенавчанням. Це призводить до того зо мережа починає вивчати статистичний шум в даних котрі їй надали що призводить до досить поганої продуктивності такої мережі коли модель працює з новими даними. Тому використовують dropout –це метод регуляризації, який апроксимує паралельне навчання великої кількості нейронних мереж з різними архітектурами. В період навчання певна кількість вихідних даних шарів випадковим чином ігнорується або «випадає». Це призводить до того, що шар виглядає ніби з іншою кількістю вузлів і підключенням до попереднього шару. Фактично, кожне оновлення шару під час навчання виконується з іншим «виглядом» налаштованого шару.

Dropout не може впливати на значення цільової функції: змінюється тільки структура мережі, а точніше нейронів в ній. Кожен нейрон в мережі має імовірність p на те щоб бути видаленим. З результуючою проріджeною мережею проводиться зворотне поширення помилки, а для інших ваг проводиться градієнтний крок. В кінці цього процесу прибрані з мережі нейрони повертають. У процесі навчання нейронної мережі вихід кожного нейрона помножується на значення рівне (1-p). Таким чином можна отримати матсподівання відповіді мережі по її 2N архітектурам де N визначено як кількість нейронів в архітектурі нейронної мережі. Навчена на тренувальних даних нейромережа являється результатом усереднення 2N нейромереж. Нейронна мережа котра була навчена за допомогою ранньої зупинки, має досить велику помилку у порівнянні з мережею котра навчена з усередненням декількох нейронних мереж що призводить до істотного зниження помилки;

Пакетний градієнтний спуск - це метод котрий дозволяє знайти за допомогою руху уздовж градієнта локальний мінімуму або максимуму цільової функції. Пропонує поглянути графік функції на котрому по осі x відкладені значення ваг нейрона (w), а по осі y – помилка котру ставимо у відповніть цій вазі (e) (рисунок 1.20).

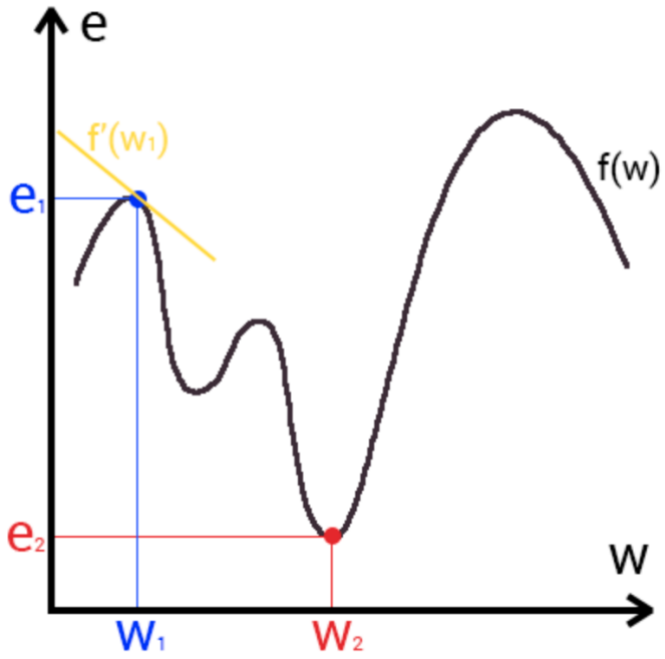


Рисунок 1.20 – графік ваги нейрона та помилки

Можемо помітити те що функція представлена на графіку є залежністю помилки від обраної чи обрахованої ваги. Точка графіку W2 є глобальним мінімумом, саме в цій точці ми отримуємр найменшу помилку і як результат, ми будемо мати найкращий вихідний вектор значень. Виявити цю точку нам дає змогу метод градієнтного спуску (жовтим на графіку позначений градієнт). Таким чином у кожної ваги в нейронній мережі буде власний графік функції та градієнт, і у кожної за методом потрібно знайти глобальний мінімум.

Градієнт – це просто вектор який відображає крутизну схилу, а також вказує його напрямок щодо будь-якої з точок на графіку. Щоб знайти градієнт

даної функції потрібно отримати похідну від графіка по даній точці (як це і показано на графіку). Рухаючись у напрямку котрий вказує нам градієнт, ми будемо плавно спускатися до низини, а саме до точки котра буде точною мінімуму.

Варто зазначити певні недоліки методу.

Головною проблемою при навчанні штучних нейронних мереж є в методах виходу з локальних мінімумів. Недоліками градієнтного спуску при навчанні мережі є:

- Так званий параліч мережі.

Значення ваг нейронів мережі в результаті корекції при навчанні можуть досягти дуже великих величин. Процес навчання може майже спинитися, це обумовлено тим що помилка котра буде посилатись назад при навчанні пропорційна до похідної на стискає функцію. Можна зменшувати крок η для того щоб запобігти проблемі наведеній вище, але це сповільнить процес навчання.

- Розмір кроку η.

За умови якщо значення кроку не змінювати, і воно буде лишатись досить малим, то метод може сходитися дуже повільно. Якщо лишати крок занадто великим, то у такому випадку ми можемо отримати проблему наведену вище. Необхідно змінювати значення кроку: збільшувати до того моменту, поки не зупиниться поліпшення оцінки у напрямку антиградієнту або зменшувати, у випадку коли оцінка не поліпшується.

1.4 Висновки

У цьому розділі ми досить детально розглянули теорію нейронних мереж починаючи з біологічних прототипів, адже це дуже важливо для подальшого

розуміння роботи штучний нейронних мереж. Також було введено базові але дуже важливі поняття при побудові нейронної мережі. Крім того ми ознайомились з проблемами котрі вирішують при навчанні цих систем та те чому це важливо робити в цьому процесі для досягнення найкращого результату. Виходячи з визе наведеного ми достатньо підготувались для того щоб розглянути складні нейронні мережі котрі будемо використовувати для вирішення поставленої задачі.

2 АРІТЕКТУРА МЕРЕЖ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ КЛАСИФІКАЦІЇ ТЕКСТУ

2.1 Глибокі нейронні мережі

Нейронні мережі котрі мають у своєму складі більше одного прихованого шару прийнято називати – глибокими нейронними мережами. Така нейрона мережа може виконувати більш складні обчислення у певному сенсі, адже вона проводить обчислення наступного шару на основі попереднього, тож вона має більше репрезентативну потужність та може оброблювати більш складні функції ніж мережа з одним прикованим шаром. Така мережа має головну умову – при її навчанні необхідно використовувати нелінійну активаційну функцію у всіх прихованих шарах.

Головний виграш таких нейронних мереж перед іншими мережами є можливість представлення досить великої кількості функцій досить зжато. Можна представити, що існують функції, котрі k-шарова мережа має змогу представляти стисло, а (k-1) шарова мережа нездатна цього зробити, якщо вона не має експоненційно великої кількості елементів у шарі. Приклад такої мережі наведено на рисунку (рисунок 2.1).

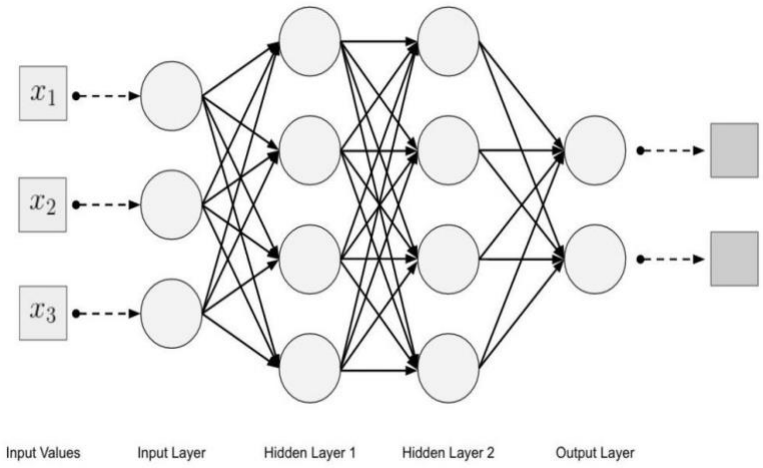
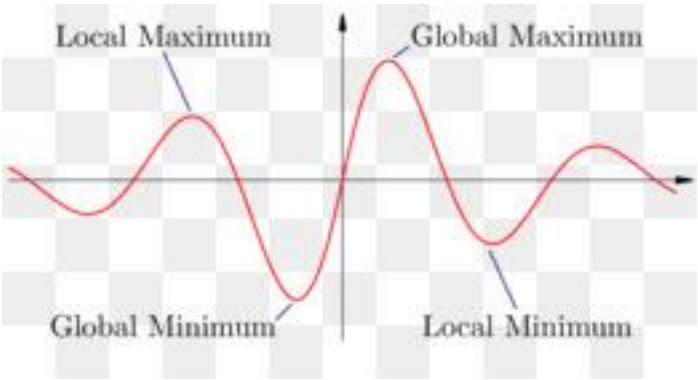


Рисунок 2.1 – приклад багатошарової мережі

2.2 Проблема локального оптимуму глибоких нейронних мереж

Процес навчання мережі з одним прихованим шаром із застосуванням так званого контрольованого навчання дуже часто дає результат з наближенням параметрів до відповідних перевірочних значень. Але у випадку навчання глибокої нейронної мережі, це спрацьовує в рази рідше. Варто виокремтити, що навчання нейронної мережі з застосуванням методу навчання з учителем може включати в себе вирішення відомої нам проблеми як – проблема неопукої оптмізації.

У багатошарової нейронної мережі можуть з’являтись локальні оптимуми, до того ж у великій кількості, при такій проблемі навчання з використанням методу градієнтного спуску може перестати працювати. Приклад такої ситуації бачимо (рисунок 2.2).



Русунок 2.2 – приклад проблемної для навчання функції

2.3 Згорткові нейронні мережі

Декілька останніх років використання нейронних мереж для вирішення певних задач таких як – розпізнавання зображень, голосу або тексту стало досить популярним заняттям, що в свою чергу дало великий поштовх для того щоб розвивати цей напрямок та призвало як результат до дуже потужних результатів у цій галузі. Одним з таких проривів у цій сфері можна без сумнівів назвати розробку та використання нейронних мереж котрі отримали назву – згорткові нейронні мереж або ж CNN. Якщо одним реченням описувати згорткову нейронну мережу то можна сказати що це мережа котра використовує той самий набір нейронів безліч разів що дає змогу зберігати стабільну кількість параметрів при роботі з досить великими моделями.

Мережа не просто так називається згортковою, а саме через операцію згортки котру можна застосувати наприклад до двох послідовностей f та g для породження третьої послідовності.

$$(f * g)(c_1, c_2) = \sum f(a_1, a_2)\, g(c_1 - a_1, c_2 - a_2) \qquad (2.1)$$

де f – довільна фукція;

g – довільна функція;

a1, a2, c1, c2 – певі коефіцієнти.

Загалом нейронна мережа з такою архітектурою[1] була створена для вирішення конкретної задачі, а саме для аналізу та розпізнавання зображень, для чого використовували мережу з будовою – згорткові шари по черзі, далі шор RELU, шари об’єднання та як у всіх штучних нейронних мереж вихідний шар. Приклад будови такої мережі можна побачити на рисунку (рисунок 2.3).

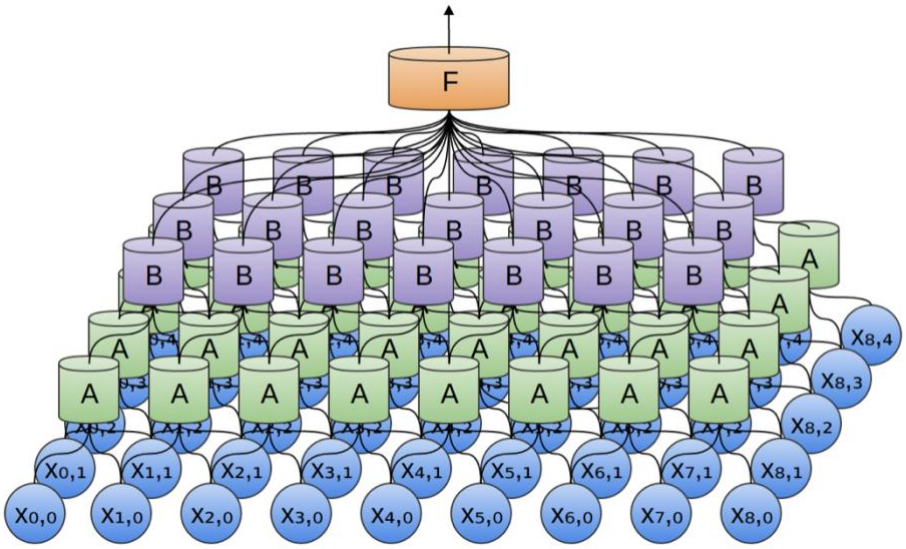


Рисунок 2.3 – 2D зображення CNN

Нейрони CNN мережі розміщенні фактично в 3-х вимірах, так як це зображено на одному з шарів. В прикладі котрий наведено на рисунку (рисунок 2.4) вхідний шар виділено червоним кольором, він має розмір котрий відповідає розміру вхідного зображення, але глибина дорівнює три. Повну схему можна побачити на рисунку (рисунок 2.4).

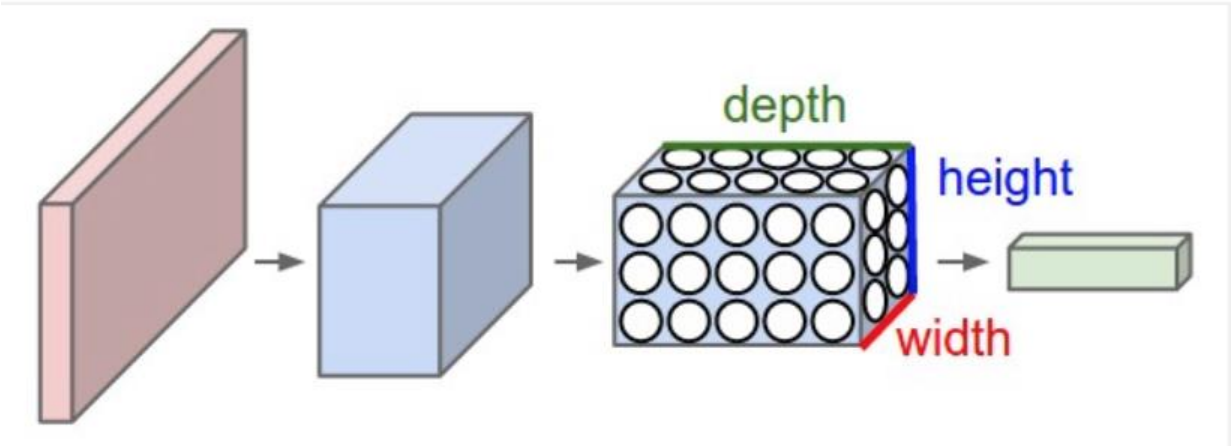


Рисунок 2.4 – 3D згортка

Оскільки схема для розуміння досить складна до пропоную розглянути іншу, спрощену схему в одному вимірі. На ній зображено одновимірний згортковий шар з n входами на позначеними виходами як F Тоді матимемо наступну схему зображену на (рисунок 2.5).

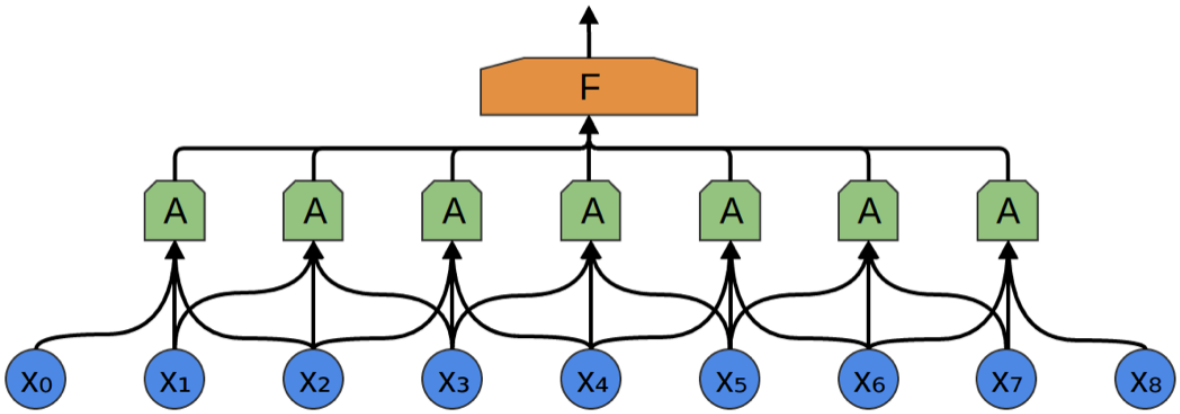


Рисунок 2.5 – Одновимірний вид шару

Вище ми вказали, що згортовкий шар містить в собі безліч копій одного нейрона тому на виході можемо мати однакові ваги для певних у

$$Y1 = f(W0x0+W1x1+W2x2+b) \quad (2.2)$$

$$Y2 = f(W1x1+W2x2+W3x3+b) \quad (2.3)$$

де W – певні ваги;

x – надані вхідні дані.

Згортковий[3] шар є основним будівельним блоком CNN, і саме в ньому відбувається більшість обчислень та є особливість мережі. Для таких обчислень потрібно кілька компонентів, а саме вхідні дані, фільтр і карта об’єктів. Покладемо, що у якості вхідних даних в нас буде випадкове кольорове зображення, котре відповідно складається з матриці пікселів у 3D форматі. Це нам говорить про те, що вхідні дані будуть у трьох вимірному формат, та будуть

мати - висоту, ширину та глибину - які відповідають формату RGB зображення. Також у цій схемі ми маємо так званий детектор ознак, також відомий як ядро або фільтр, який буде рухатися по частинах зображення, перевіряючи, чи є ознака котру можна виділити. Процес котрий ми тільки що описали і є згорткою.

Детектор ознак - це такий собі двовимірний (2-D) масив ваг, який може покрити частину зображення. Цю матрицю називають kernel(ядро), також для її характеристики використовують термін kernel size(розмірність ядра) – він характеризує розмірність матриці(ядра). Хоча він може відрізнятись за розміром, але розмір фільтра зазвичай є матрицею 3x3 та може бути наприклад 2x2; це також визначає розмір рецептивного поля. Далі фільтр застосовується до області зображення, і між вхідними пікселями та фільтром обчислюється крапковий добуток. Цей точковий добуток потім подається у вихідний масив. Для обчислення цього добутку кожен фрагмент на котрий накладається фільтр множиться на матрицю згортки по кожному елементу, а далі ці отримані значення додаються та записуються у відповідну позицію. Після цього фільтр зміщується на один крок, повторюючи процес, поки ядро не охопить усе зображення. Остаточний вихід із ряду точкових добутків із входу та фільтра відомий як карта ознак, карта активації або згорнута функція і так формується певна їх кількість, як результат ми маємо n кількість карт ознак на одному згортковому шарі, таким чином нейронна мережа стає багатоканальною.

Варто[4] зазначити що є три гіперпараметри, які можуть досить сильно впливати на розмір виводу, які необхідно встановити перед початком навчання нейронної мережі. До таких належать:

- Тож кількість фільтрів впливає на глибину виводу. Наприклад, три різні фільтри дадуть три різні карти об’єктів, створюючи глибину три.
- Stride – це крок, відстань або кількість пікселів, на які ядро(фільтр) переміщається по вхідній матриці(зображеню). Хоч і значення кроку два чи більше зустрічається досить рідко, більший крок дає не такий гарний результат.

- Zero-padding зазвичай використовується, коли фільтри не підходять по розміру до вхідного зображення. Це встановлює всі елементи, які виходять за межі вхідної матриці, на нуль, створюючи вихід більшого або такого ж розміру. Існує три типи підстановки:
- Дійсний відступ: це також відоме як відсутність заповнення. У цьому випадку остання згортка скидається, якщо розміри не вирівнюються.
- Аналогічне заповнення: це заповнення гарантує, що вихідний шар має той самий розмір, що і вхідний шар
- Повне заповнення: цей тип заповнення збільшує розмір виводу, додаючи нулі до межі вхідних даних.

Після кожної операції згортки конволюційна нейронна мережа застосовує перетворення Rectified Linear Unit (ReLU) до отриманої карти ознак, вносячи не лінійність у модель. Як ми вказували раніше, інший шар згортки може слідувати за початковим шаром згортки. Коли це станеться, структура конволюційної нейронної мережі може стати ієрархічною, оскільки пізні рівні можуть бачити пікселі в сприйнятливих полях попередніх шарів.

Об’єднання шарів, також називається як зниження дискретизації, ця операція зменшує розмірність, способом зменшення кількості параметрів у вхідних даних після конволюційних шарів. Подібно до згорткового шару, операція об’єднання, поєднує фільтр по всьому входу, але відрізняється ти що цей фільтр не має ніяких ваг. Замість цього ядро застосовує функцію агрегації до значень у прийнятній області, заповнюючи результуючий масив. Варт зазначити що існує два основних типи об’єднання:

- Максимальне об’єднання - це коли під час переміщення фільтра по входу він зберігає значення з максимальним значенням для відправки на вихідний масив. Крім того, цей підхід, часто використовується частіше в порівнянні зі середнім об’єднанням.

- Середнє об’єднання - це коли фільтр переміщується по входу, тим самим він обчислює середнє значення в області обробки для подальшого надсилання до вихідного масиву.

На перший погляд може здатись що цей шар робить точність гірше, адже він втрачає багато інформації на рівні об’єднання, але це також має ряд переваг для CNN. Цей шар допомагає зменшити складність, підвищити ефективність та обмежити ризик перенавчання нейронної мережі, тож цей шар вирішує певні проблеми мережі та оптимізує її роботу та навчання. Приклад роботи шару наведено на рисунку (рисунок 2.6).

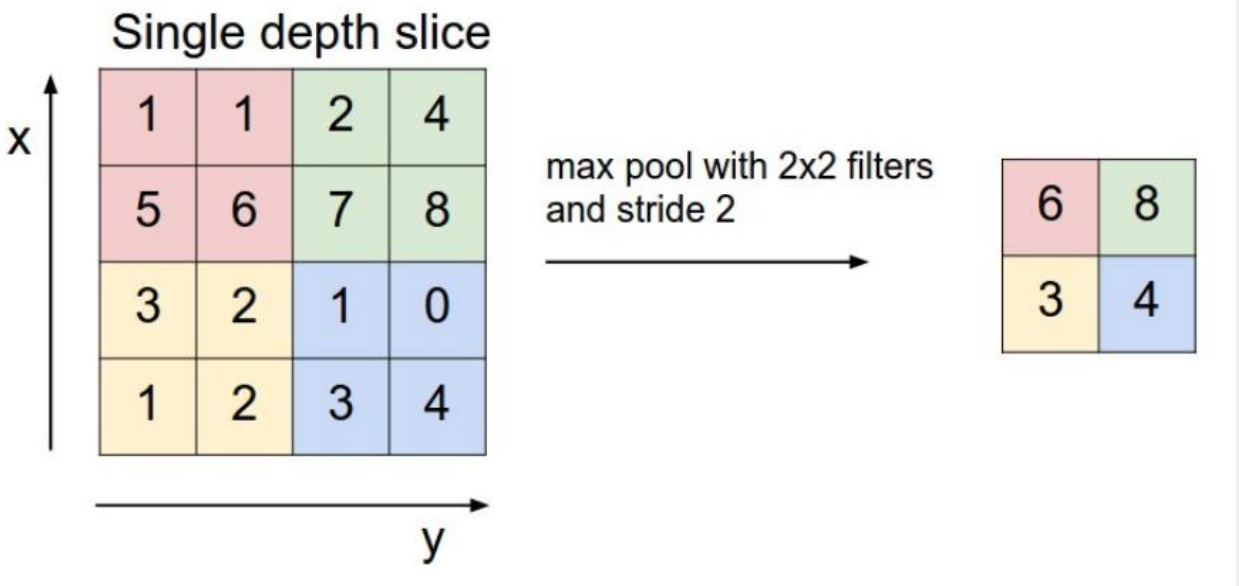


Рисунок 2.6 – приклад роботи pooling шару

Далі після проходження картами певної кількості шарів вона попросту кажучи витягується у вектор або навіть скаляр, але в той самий час таких карт стає досить багато – залежно від гіперпараметрів мережі, тому далі їх передають на повнозв’язні шари нейронної мережі наприклад персептрону. Цей шар виконує основне завдання такого типу мереж - завдання класифікації на основі ознак котрі раніше були витягнуті через попередні шари та їх різні фільтри. У той час як рівні згортки та об’єднання, як правило, використовують функції ReLu, повнозв’язні шари зазвичай використовують функцію активації такі як

softmax наприклад для вірної класифікації вхідних даних, повертаючи ймовірність від 0 до 1.

Пропоную розглянути те як застосовують згорткові нейронні мереж при вирішенні задачі розпізнаванні емоційної тональності тексту.

Перед початком роботи з нейронною мережею потрібно виконати підготовку даних котра для цієї задачі є досить важливою операцією, адже при невірній підготовці ми можемо отримати на виході некоректні результати, саме тому важливо виконати коректні маніпуляції з даними. Для цього потрібно провести ініціалізацію embedding шару котрий виконає потрібні маніпуляції. Якщо описувати більш детально то цей шар потрібен щоб поставити у відповідність слова цілі числа і як результат отримати певний словник, тому перед цим потрібно привести дані до одного вигляду – більш детально опишемо цей процес при розробці та навчанні нейронної мережі, але варто зазначити що досить популярним та влучним рішенням для цієї задачі є використання Word2vec,a Glove та FastText, таким чином буде використано один з цих методів.

У якості вхідних даних для нейронної мережі ми маємо певну матрицю із зафіксованою висотою, нехай це буде n, кожен рядок даної матриці це векторне представлення слова, але саме висота матриць може бути різною, це обумовлено тим що вона відображає довжину речення котре ми передаємо на вхід, тож висоту задають таку щоб вона покривала всі, або майже всі речення по довжині, якщо вона не покриває якесь слово у реченні то воно відпадає та не приймає участі у навчанні мережі. Для сигнатурного зготкового шару обирають вище введені нами фільтри, але для кращого покриття використовують наступний підхід. Обираємо фільтри різної розмірності а для кожного з розмірів вводять декілька шарів у мережу. В якості функції активації обирають досить часто функцію активації ReLu, її було було обрано і для практичної частини цієї роботи. Для зменшення розмірності карт на наступному кроці використовують шари котрі ми описали раніше – шари субдескритизації до матриць ознак котрі ми отримали, таким чином ми виділяємо основні ознаки та відкидаємо зайве, цей процес можна побачити на зображені (рисунок 2.7).

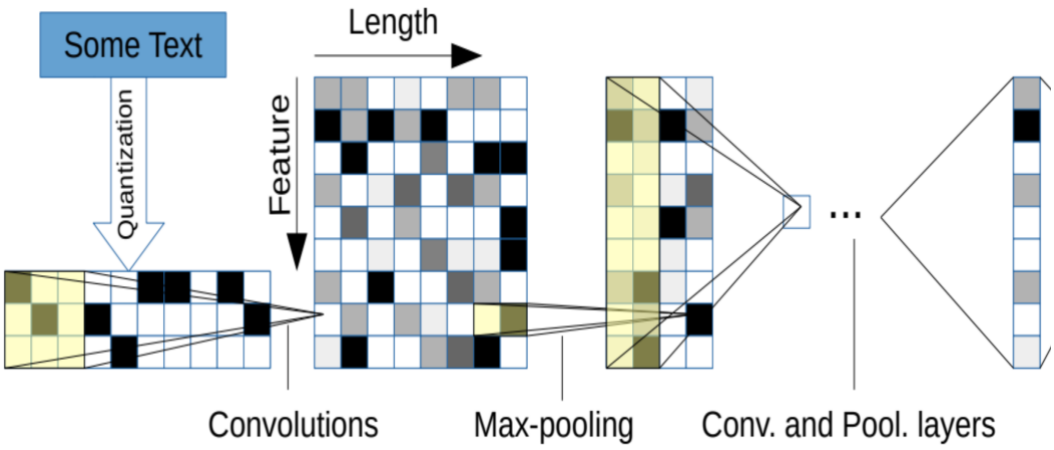


Рисунок 2.7 – Схема частини мережі з підготовкою тексту та згортковими шарами і субдекскритизації

Наступним потрібним для навчання етапом є поєднання карт в один загальний вектор ознак, цю маніпуляцію. Проводять за допомогою шару котрий носить назву шар поєднання. Після нього вектор ознак потрапляє на певний повнозв’язний шар котрий визначає результат мережі, після чого результат йде на вихідний шар мережі з деякою функцією активації котрий ми описували раніше. У попередніх розділах було зазначено що існує таке явище як те що нейронні мережі можуть не тільки погано навчитися, але й можуть перенавчитися, таким чином конволюційні нейронні мережі не вийняток тому для запобігання такому результату використовують dropout з певною заданою ймовірністю це зменшує ймовірність цього та допомагає запобігти перенавчанню, цю регуляцію додають перед прихованим повнозв’язним шаром (рисунок 2.8).

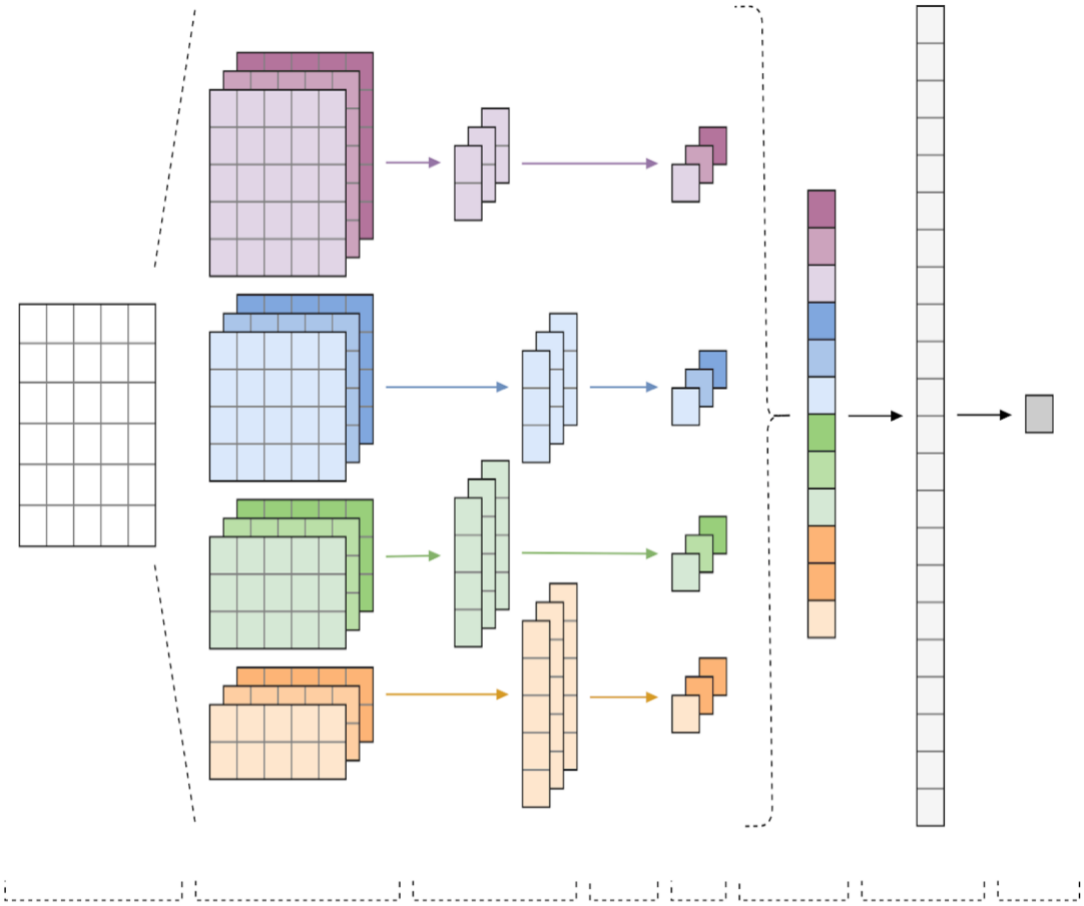


Рисунок 2.8 – Приклад побудованої нейромережі CNN для роботи з текстом

Навчання нейронної мережі неможливе без визначення якості роботи цієї нейронної мережі, адже без якихось параметрів неможливо визначити правильність отриманих результатів, та для цього використовують певні метрики. Вони дозволяють зрозуміти оптимальність роботи мережі та навчання. Далі перерахуємо та опишемо ці метрики.

Однією з таких метрик є точність(або англійською precision), вона використовується зазвичай як незалежна метрика для оцінки алгоритмів так і для знаходження інших метрик котрі використовують для цієї задачі. Загалом ця метрика відображає відношення кількості елементів що належать до якогось з визначених класів до кількості елементів котрі мережа віднесла до цього класу. У розрізі роботи з аналізом тональності тексту то елементами є текст котрий

мережа віднесла то категорії котрі ми для визначили для неї. Пропоную навести приклад коли ми маємо що мережа з десяти речень віднесла, що два з них негативні за своїм емоційним забарвленням, але насправді одне з цих двох речень є дійсно позитивним за своїм емоційним забарвленням, тоді точність мережі ½ (одне з обраних речень є дійсно негативним, а всього мережа визначила два речення як негативні). Для математичної інтерпретації потрібно ввести певні умовні позначення:

N_1 – Це значення котре відображає кількість елементів котрі система віднесла до правильного класу.

N_2 – Це значення котре відображає кількість елементів котрі система віднесла до класу

Тоді точність можна задати наступною формулою:

$$\text{Precision} = \frac{\mathrm{N}_1}{\mathrm{N}_2} \tag{2.4}$$

де N_1 – Це значення котре відображає кількість вірних елементів;

N_2 – Це значення котре відображає все що віднесено мережею до класу.

Другою з таких метрик є метрика котра носить назву повнота(або англійською reccal), вона також може існувати та бути використана як самостійна метрика, або ж бути використана підрахунку більш складних та зіставних метрик таких як F-міра чи R-точність. Вона відображає відношення кількості визначених нашою нейронною мережею текстів одного класу до загальної кількості текстів класу. Якщо знову розглянути приклад з десятьма реченнями, полкладемо що з них 4 це негативні, тому повнота дорівнюватиме значенню ¼ . Для математичної інтерпретації потрібно ввести певні умовні позначення:

N_1 – Це значення котре відображає кількість елементів котрі система віднесла до правильного класу

N_3 – Це значення котре відображає кількість елементів у класі

$$\text{Reccal} = \frac{N_1}{N_3} \tag{2.5}$$

де N_1 – Це значення котре відображає кількість вірних елементів;

N_3 – Це значення котре відображає все що у класі.

Третім параметром є F- міра, вона інтерпретує гармонічне середнє значення між повнотою та точністю, також варто зазнчити що вона теж прямує до нуля у разі коли ті два параметри теж прямують до нуля. Може виникнути логічне питання – навіщо ми вводимо ще одну метрику якщо дві попередні ніби то покривають потребу в аналізі якості роботи мережі, але справа в тому що при навчанні мережі та за реальних даних максимально велика повнота не може бути досягнута в той самий час як буде досягнута максимальна точність і саме через це треба знайти певний компроміс, через це була введена нова метрика котра як вище описано поєднує два попередні параметри котрі ми розглянули. Використовуючи цю метрику дуже часто приймають рішення про якість роботи нейронної мережі, та який варіант прийняти для реалізації. Наведемо формулу при котрій пріоритет між повнотою та точністю є рівним та дає змогу прийняти рішення:

$$\text{F} = 2 * \frac{\text{Precision} * \text{Reccal}}{\text{Precision} + \text{Recca}} \tag{2.6}$$

де Precision – метрика введена раніше точність;

Reccal – метрика введена раніше повнота.

Якщо виникає потреба у тому щоб віддати перевагу одній з наведених метрик то потрібно ввести новий параметр, котрий позначають β. У тому випадку коли він лежить від нуля до одного то пріоритет віддають точності, але якщо параметр заданий як більше одного то пріоритет віддано повноті.

$$F = (\beta + 1) * \frac{\text{Precision} * \text{Reccal}}{\beta^2 * \text{Precision} + \text{Recca}} \quad (2.7)$$

де Precision – метрика введена раніше точність;

Reccal – метрика введена раніше повнота.

Формально кажучи цей параметр є досить влучним рішенням для оцінки якості роботи штучної нейронної мережі, тому що він дає змогу не просто врахувати два параметри для відображення якості її роботи та спростити порівняння, але й віддати перевагу одному з них, що досить вадливо та спрощує процес прийняття рішення.

Можливо після наведеної теорії такий тип нейронних мереж як CNN не виглядає як влучний варіант для роспізнавання тексту та його емоційного забарвлення, бо коли ми аналізуємо зображення то фільтр виділяє ознаки з пікселів котрі в свою чергу є частиною одного зображення та мають певний зв’язок, а текст це набір слів котрий може бути частиною фраз речення розбитий на частки іншими словами. Але після виконання досліджень ми побачимо на практиці що такий тип нейронних мереж показує досить високий рівень у порівнянні з іншми.

Гіперпараметри CNN мережі та її архітектура наведені нижче.

Першим та основним гіперпараметром мережі є її глибина, це значення є еквівалентним до заданої кількості фільтрів котрі ми будемо використовувати при тренуванні та роботі нашої мережі. Наприклад у разі вирішення задачі розпізнаванні зображень RGB ми будемо мати три так званих канали зображення, якщо дивись у розрізі задачі аналізу тональності тексту то в обробці тексту це можуть бути різні представлення слів, наприклад перефразовані речення, інша мова і тому подібне.

Наступним гіперпараметром є так зване доповнення нулями, що дає нам змогу управляти просторовими розмірами вхідних даних мережі, а саме розмір цього доповнення буде заданим гіперпараметром. Найчастіше це використовують для того щоб зробити значення висоти та ширини однаковими

для вхідних даних, або для того щоб не отримати вузьку згортку то можна використати доповнення нулями.

Наступним гіперпараметром буде розмір кроку зсуву фільтру котрий ми описували раніше, якщо ми задали крон рівним одиниці то у випадку з вирішенням задачі аналізу зображень ми будемо зсувати фільтр на один піксель при русі по вхідним даним, коли значення задане як – два то відповідно ми будемо рухатись на два пікселі в бік при переміщенні фільтру, але варто зазначити що найбільш часто використовують крок один чи два. Також варто зазначити, що більший крок ми визначимо, тим меншим буде розмір вихідної матриці згортки, цікаво те що якщо взяти великий крок то поведінка мережі може нагадувати поведінку рекурсивної мережі.

Таку мережу можна описати як однонапрямлена мережа для навчання якої зазвичай беруть метод оберненого поширення помилки(або певні модифікації типу Adam, Adadelta, rmsPRoP), також помітимо що їй характерна велика кількість шарів. Приклад на (рисунок 2.8).

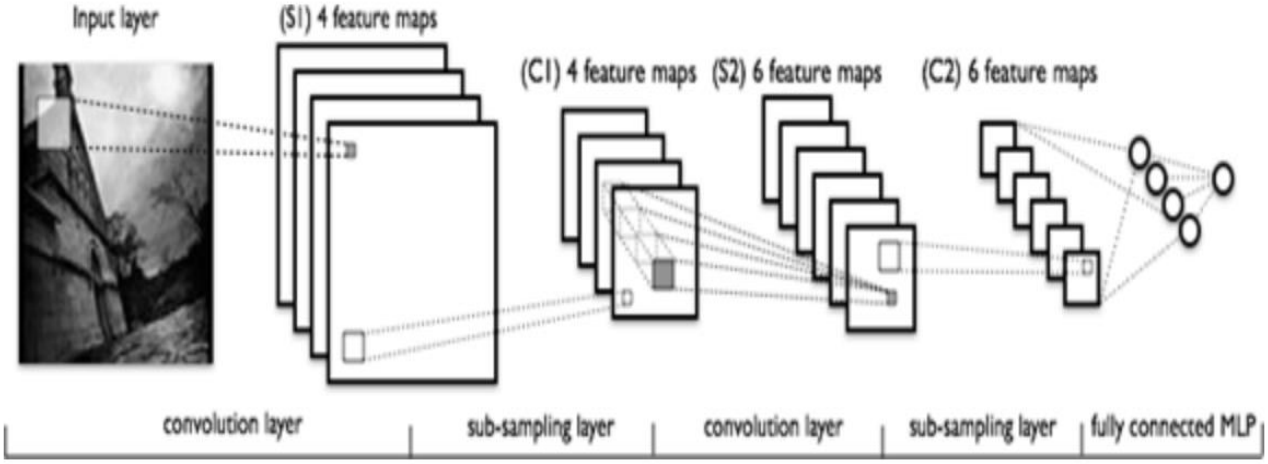


Рисунок 2.8 – архітектура мережі

2.4 Метод Adam

Для того щоб якісно виконати роботу по навчанню нейронної мережі потрібно спочатку прийняти досить вадливе рішення, а саме обрати метод оптимізації системи котру ми будуємо. Адже від цього рішення залежить якість навчання та час навчання котрий теж є досить важливим, бо надивлячись на великі сучасні ресурсі комп’ютерів ми не можемо дозволити собі навчати мережу надто довго.

Метод Adam - це альтернативний алгоритм оптимізації, який забезпечує отримання більш ефективних ваг штучної нейронної мережі, виконуючи повторювані цикли «оцінки адаптивного моменту». Адам побудований на стохастичному градієнтному спускі, щоб швидше вирішувати неопуклі задачі, використовуючи менше ресурсів, ніж багато інших методів оптимізації. Він найефективніший у надзвичайно великих наборах даних.

Давайте наведемо чому саме обирають цей алгоритм, для цього опишемо його переваги котрі надали автори методу:

- Метод добре працює із задачами котрі використовують великі об’єми даних.
- Є досить простим та не привередливим до оточення на котрому проводять навчання, хоча б тому що не потребує багато пам’яті, зараз це може здатись не досить важливим, але від цього прямо залежить час навчання, а також вартість обчислювальних потужностей котрі доведеться закласти на навчання та використання мережі.
- Гіперпараметри цього методу являються досить простими у плані розуміння та часто не потребують важкого та довгого налаштування.
- Відносно інших методів та модифікацій є досить простим для реалізації з більшістю мов програмування та не потребує довгого написання коду.

- Має досить високу обчислювальну ефективність.

Адам поєднує переваги двох інших методів стохастичного градієнта, адаптивних градієнтів і середньоквадратичного поширення, щоб створити новий підхід до навчання та оптимізації різноманітних нейронних мереж.

Порівняємо метод з класичним методом стохастичного градієнта. Зі стохастичним градієнтним спуском (SGD) для всіх оновлень ваги використовується єдина швидкість навчання, так звана альфа. Крім того, швидкість навчання для кожного параметра мережі, а саме ваги не змінюється під час навчання. Тепер SGD окремо розраховує індивідуальні швидкості адаптивного навчання для різних ваг для подальшого людського аналізу, взявши оцінки першого і другого моментів градієнтів. Адам, з іншого боку, адаптує швидкість навчання ваг у режимі реального часу на основі середнього значення першого та другого моментів. Зокрема, шляхом розрахунку експоненційного ковзного середнього градієнта, а також квадрата градієнта. Тоді параметри beta1 і beta2 можуть керувати швидкістю спаду обох ковзних середніх. Нарешті, оцінки корекції зміщення виконуються перед оновленням параметрів навчання.

Для того щоб продемонструвати можливості цього методу було проведено навчання тестової мережі, а саме багатошарового персептрона і порівняно результати(training cost) з іншими методами. Приклад на (рисунок 2.9).

Наступним кроком розглянемо параметри даного методу котрі до речі мають певні стандратні значення у деяких бібліотеках мов програмування, наприклад в Python - Tensorflow, Keras(надбудова Tensorflow), Blocks.Параметр α - це значення задає швидкість навчання, або розмір кроку з котрим оновлюються ваги:

- $\beta 1$ – це так званий показник експоненційного затухання першого моменту.
- $\beta 2$ – цей параметр має такий самий сенс як і попередній але для другого моменту, це значення може бути близьким до 1 або навпаки дуже малим значенням.

Всі ці параметри мають певні стандартні та рекомендовані значення у вище наведених бібліотеках для спрощення роботи з ними.

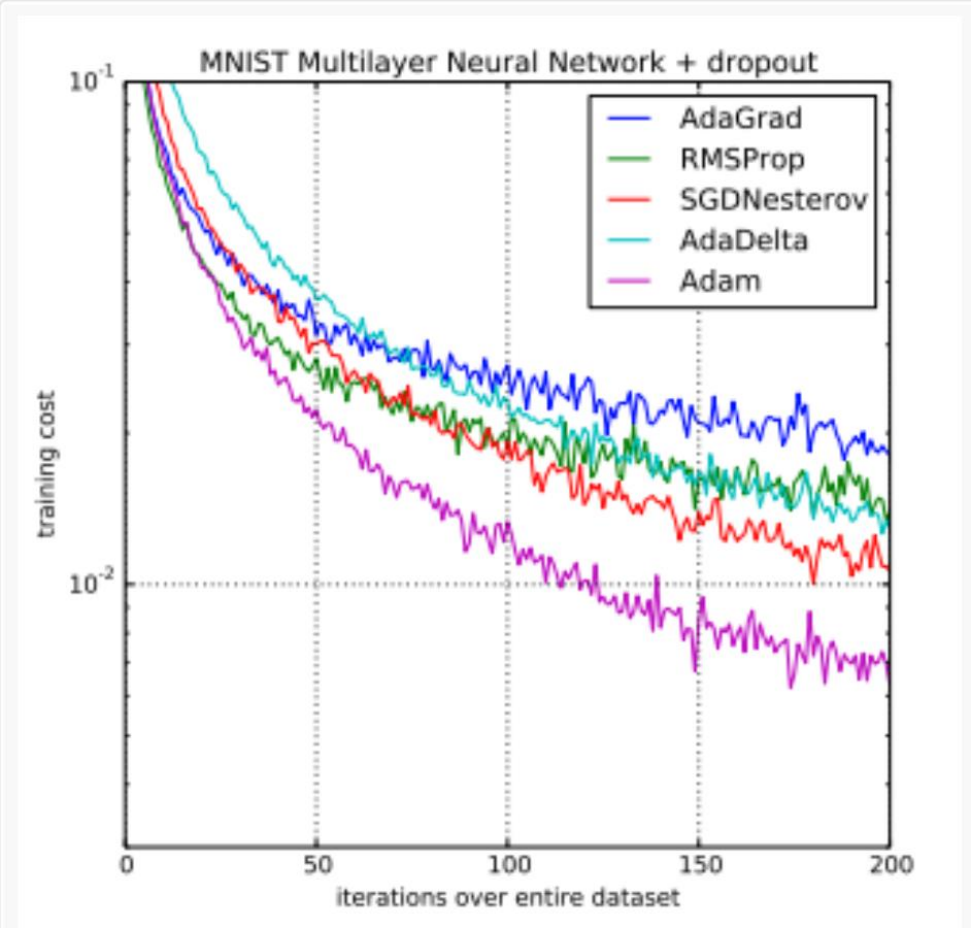


Рисунок 2.9 – Графік порівняння методу Адам з іншими

2.5 Наївний баєсівський класифікатор

Для вирішення задач аналізу емоційного забарвлення тексту, як базове стандартне рішення використовують метод під назвою - наївний баєсівський класифікатор. Головною[5] ідеєю роботи даного алгоритму є класифікація текст за імовірністю класів котрі визначені для текстів, використовуючи залежні ймовірності класів та самих слів. Тож пропоную розглянути приклад де довільне речення “T”, нехай це буде коментар у “Instagram” та маємо певну множину

класів позначену як “K”, то ми маємо знайти такий клас “k” щоб ймовірність входження речення “T” в цей клас була найбільшою серед всіх в “K”. Нижче наведена формальна інтерпретація котра б відображала наведений приклад:

$$k = \mathrm{argmax}P(K/T) \tag{2.8}$$

де P(K/T) – це певна відносна ймовірність.

Оскільки знайти потрібну ймовірність K відносно T не надто проста задача, то для її вирішення використовують відому всім теорему Баєса:

$$P(K/T) = \frac{P(T/K) * P(K)}{P(T)} \tag{2.9}$$

де P(T) – це константа за умови що значення змінних ознак відомі;

P(K) – це деяка йморівність.

Враховуючи сказане можна перевизначити формулу наступним чином:

$$P(K/T) = \frac{P(t1, t2, \ldots/K) * P(K)}{P(t1, t2, \ldots)} \tag{2.10}$$

де P(t) – це константа за умови що значення змінних ознак відомі;

P(K) – це деяка йморівність.

Далі ми маємо зробити наївне припущення що T залежить тільки від K, а також те що вони не впливають один на одного, це припущення спрошує обрахунки, але враховуючи що це базове рішення та те що зазвичай воно працює то можемо це робити. Тому кінцевий результат обчислень буде виглядати наступним чином:

$$k = \mathrm{argmax}P(k)\prod P(t_i/k) \tag{2.11}$$

де $P(t)$ – це константа за умови що значення змінних ознак відомі;

$P(t_i/k)$ – відносі ймовірності.

Як результат маємо те що потрібно знайти всього два параметри у формулі, та саме в цьому і є тренування даного класифікатору.

2.6 Нейронні мережі з довгою короткостроковою пам’яттю

Мережі довгострокової пам’яті, котрі зазвичай просто називають «LSTM» – це частний випадок мережу типу RNN, здатний вивчати довгострокові залежності. Вони були винайдені у 1997, та були непогано вдосконалені та популяризовані багатьма людьми за наступний період часу. Ці мережі надзвичайно добре вирішують дуже широкий спектр задач і в наш час досить широко використовуються.

Досить очевидно що LSTM розроблені, для того щоб уникнути довгострокової проблеми залежності. Під котрою розуміють запам’ятовування інформації протягом тривалого періоду часу, але в цих типах мереж це практично їхня поведінка за замовчуванням, а не проблема при навчанні. Цей тип мереж фактично має вигляд ланцюга з повторюваними модулями нейронної мережі, у простих RNN мережах цей модуль може мати просто просто функцію гіперболічного тангенсу. LSTM мають особливість в тому що їх повторюваний модуль має іншу структуру, замість одного шару вона має чотири котрі взаємодіють певним особливим чином.

На схемі зображено загальну будову модуля LSTM мережі котрий є повторюваним (рисунок 2.10).

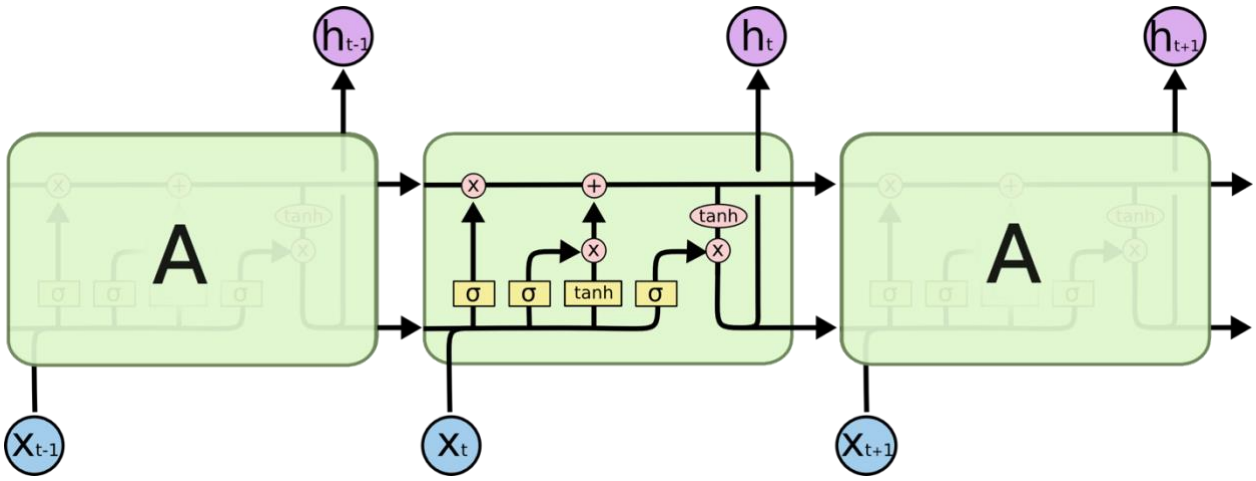


Рисунок 2.10 – Графік порівняння методу Адам з іншими

Також варто пояснити умовні позначення на діаграмі вище, адже це варто розуміти для розбору роботи даного типу нейронних мереж (рисунок 2.11).

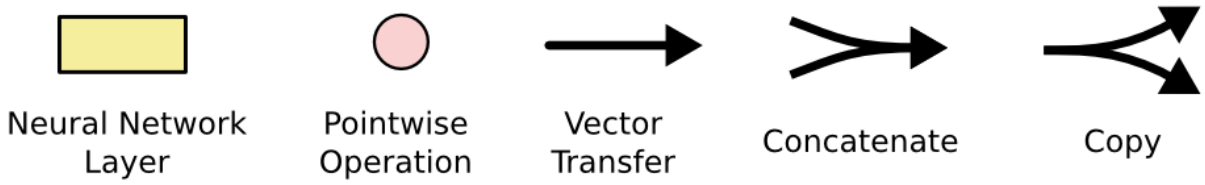


Рисунок 2.11 – Умовні позначення

Далі розглянемо що саме за операції зображені на схемі. Лініями позначено переміщення векторів даних від одного вузла до наступного. У кругах можемо помітити деякі математичні операції котрі проводять над векторами, наприклад – складання, різниця і т.д. . У прямокутниках позначенні шари нейронної мережі котрі є попередньо навченими з довгою короткостроковою пам’яттю. Сходженням або розходженням стрілок позначені операції злиття чи копіювання інформації, що вказані з вказуванням напрямку. У цій мережі є певне поняття стану котре виглядає як єдина нерозривна лінія, котра проходить у свою чергу через всі модулі мережі але варто зазначити що приймає участь не у всіх операціях. З цього поняття і виходить основна ідея даного типу мереж, адже в ній є передавання та збереження саме цього стану клітин(рисунок 2.12).

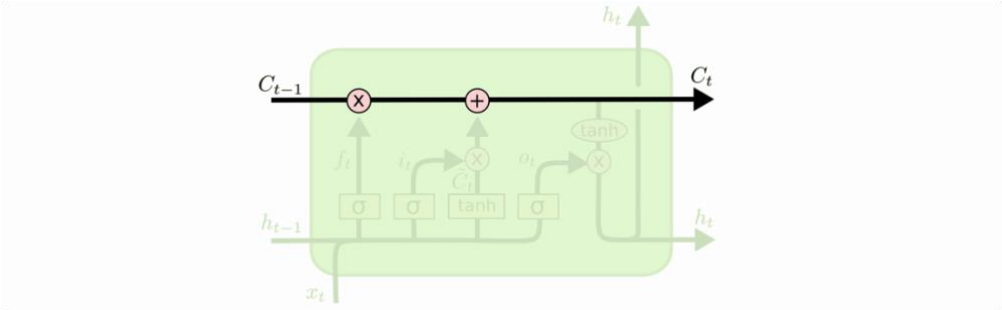


Рисунок 2.12 – Передавання стану клітини

Цю інформацію котра проходить через модуль мережа має можливість змінювати – викидати інформацію або додавати нову, цей процес досить ретельно регулюється такою частиною модуля котра носить назву – ворота (англійською gate). Ця структура фактично представляє собою сигмоїдну нейронну мережу та операції поточкового множення. Цей процес відбувається на етапі так званого фільтра забування котрий на свій вхід отримує інформацію з попередньої клітини та новий вхідний вектор для поточної клітини. Результатом роботи цього фільтру є число від нуля до одного, цей процес проходить для кожного значення зі стану клітини. У результаті ми отримаємо те що нуль означає що інформацію потрібно відкинути, а одиниця відповідно те що її можна лишити. Візуалізувати це можна наступним чином, а також задати формулу(рисунок 2.13):

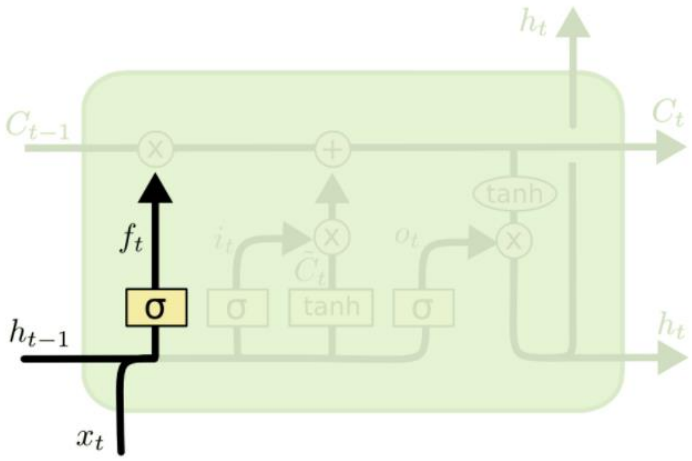


Рисунок 2.13 – Схема фільтру забування

$$f(t) = \sigma(w_f \cdot \left(\bar{x}(t), \bar{h}(t-1)\right) + b_f) \tag{2.12}$$

де $f(t) \in [0,1]$ – результуючий сигнал фільтру забування;

w_f – значення вагів даного фільтру;

b_f – зміщення фільтру.

Далі відповідно до попередньої операції визначають яка інформація має лишитись – використовує знову вхідний фільтр, котрий формульно можна описати наступним чином:

$$j(t) = \sigma(w_j \cdot \left(x(t), h(t-1)\right) + b_j) \tag{2.13}$$

Де будо використано ті самі позначення що і в попередній формулі окрім індексів. Після того як тимчасовий шар виконав свою функцію то шар тангенсу повертає нам значення вже нових кандидатів на те щоб поповнити дані клітини. Це можна описати формулою нижче:

$$\tilde{S}(t) = tanh(w_S \cdot \left(x(t), h(t-1)\right) + b_S) \tag{2.14}$$

Де задано ті ж самі позначення, що і в попередніх формулах, за винятком індексів, котрі в цьому випадку визначають проміжний стан клітини.

Наступним кроком буде заміна стану клітини, а саме $S(t-1)$ на новий стан. У цьому етапі використовують фільтри котрі було визначено раніше – забуваючого та вхідного відповідно. Тому поставлену задачу формалізують так:

$$S(t) = f(t) \cdot S(t-1) + j(t) \cdot \tilde{S}(t) \tag{2.15}$$

Далі ми маємо прийняти рішення про вихідний сигнал з модуля. Дані котрі вийдуть будуть засновані на новому стані клітини, але повинні бути опрацьованими фільтрами. На цьому етапі застосовують новий для нас тип фільтрів, а саме вихідний. Спочатку він застосовує так званий сигмоїдальний шар котрий після своєї роботи наддасть нам інформацію яку інформацію потрібно вивести із стану. Цей етап можна задати формулою наступним чином:

$$g(t) = \sigma(w_g \cdot \big(x(t), h(t-1)\big) + b_g) \tag{2.16}$$

Після чого значення поточного стану проходить через тангенс шар котрий на виході наддасть нам значення від нуля до одиниці. Котрі в свою чергу перемножують з даними котрі ми отримали на попередньому кроці із сигмоїдального шару. Це дозволить відсіяти непотрібну інформацію та вивести необхідну. Формалізуємо це наступним чином:

$$h(t) = g(t) \cdot \tanh\,(S(t)) \tag{2.17}$$

Після цього відбувається наступна ітерація. Такий принцип роботи приймається і в роботі з текстом, навіть підготовка даніх проводиться таким

самим чином з фільтрацією та використанням Word2vec наприклад. Таким чином ми розглянули роботу ще одного.

2.3 Висновки

Таким чином ми розглянули роботу, архітектуру та особливості кожної нейронної мережі чи математичного методу котрі будуть використані при роботі з практичною частиною. Ми досить детально ознайомились з роботою утворюючих шарів наших мереж та з тим в чому їх особливість роботи та налаштування для отримання найкращого результату. Ми виявили всі гіперпараметри мережі котрі потрібно досить уважно налаштувати та задали параметри за котрими можна визначити якість роботи мережі. Таким чином ми повністю готові до побудови нейронних мереж, їх тестування та навчання.

3 ОПИС МЕРЕЖ ТА ПОРІВНЯННЯ І АНАЛІЗ РЕЗУЛЬТАТІВ

3.1 Вступ

Результатом цього розділу буде те що ми розглянемо будову та гіперпараметри наших нейронних мереж, а також підхід до їх навчання. Також досить ретельно опишемо процес підготовки даних, опишемо навчальну вибірку та розіб'ємо її на тестову на навчальну для перевірки результатів роботи мережі, та проведемо очистку даних від зайвих елементів таких як посилання, никнейми, знаки пунктуації та різного регістру. Також оцінимо якість роботи та навчання мережі, використовуючи метрики описані у попередньому розділі. Як результат ми будемо мати можливість порівняти отримані результати роботи мереж та виділити фаворита для вирішення даної задачі розпізнавання емоційного забарвлення тексту.

Також досить ретельно розглянемо інструменти обрані для реалізації та побудови нейронних мереж, такі як – Python3, його бібліотеки, та оцінимо переваги та недоліки використання саме цього інструменту.

3.2 Вимоги до технічних та програмних інструментів

Написання програмного коду, його теоретична розробка, планування архітектури та тестування потребують завчасного прийняття рішення про вибір спеціального програмного забезпечення, а також бібліотек котрі в наш час є базовим рішенням. Таким чином для реалізації систем було обрано наступні програмні інструменти котрі є лідерами для реалізації рішень для подібних задач:

- Мова програмування Python 3.9.5.
- Операційна система MacOS 12.0.1.
- У якості середовища для розробки взято Jupyter Notebook, а також для перевірки роботи частин коду було обрано Visual Studio Code.
- Процесор Intel Core i5 2 GHz, графічний процесор Intel Iris Plus Graphics 1536Mb.

Використовуючи таку конфігурацію, зазвичай навчання нейронних мереж займає надто багато часу, а також сприяє зменшенню ресурсу роботи комп'ютера у якості робочої станції. Тому для економії часу при навчанні та розробці рекомендую використовувати хмарні сервіси котрі надають такі компанії як – Google, Amazon, Microsoft та Alibaba. Такі сервіси надають певним чином підготовлені сервери та сервіси параметри яких по потужності можна задати перед їх створенням, та доконфігорувати потрібним програмним забезпеченням. Таке рішення дасть виграш у часі при навчанні та тестуванні системи.

Також для експериментів варто використати безкоштовні потужності ресурсу у відкритому доступі Kaggle котрий надає можливість будувати, тренувати та тестувати роботу нейронних мереж з використанням їх середовища розробки Jupyter Notebook котрий розміщений на власних серверах Kaggle. Наприклад якщо розглянути навчанная мережі CNN то вона для цього використовує GPU, то як висновок це оптимальний варіант для проведення навчального процесу та тестування мережі та для фінального порівняння з іншими.

Також як ми визначили раніше для навчання та тестування нейронних мереж буде використано низка бібліотек котрі мають досить велику популярність у світі для вирішення багатьох задач пов'язаних з нейронними мережами. Зазвичай використання бібліотек пришвидшує розробку нейронної мережі, покращує вигляд коду, а також дає змогу сконцентруватись на налаштуванні мережі, а не виправлені помилок у синтаксисі та логіці коду. Тому пропоную розглянути деякі основні бібліотеки котрі буде використано:

Keras – ця бібліотека котра побудована на основі так званої бібліотекиTensorFlow, котра містить в собі методи котрі дозволяють нам будувати мережу блоками та задавати гіперпараметри мережі у явному вигляді, що спрощує розробку та робить код більш прийнятним для розуміння та пошуку помилок.

Re – ця бібліотека застосовується для написання регулярних виразів прямо у програмі на мові Python, вона має досить високий спектр використання, але ми її застосуємо для підготовки даних, що більш детально опишемо далі.

Sclearn – бібліотека дає змогу вирішувати певні математичні задачі, у нашому випадку ми використаємо її того щоб поділити датасет на потрібні нам частини у випадковому порядку.

Sqlite3 – бібліотека для роботи з БД sqlite, в неї ми будемо записувати дані використовуючи цю бібліотеку.

Logging – зазвичай бібліотека потрібна для логування виводу та помилок програми у нашому випадку ми будемо використовувати її для подібних маніпуляцій.

Multiprocessing – бібліотека виконує досить важливу задачу, адже дає змогу виконувати програму у багатьох потоках що підвищує швидкість роботи програми та як результат навчання.

Gensim – бібліотека для роботи з текстом, як результат дає змогу працювати з моделями такими як Word2vec, FastText, котрі ми будемо використовувати при підготовці даних.

3.3 Підготовка даних

Перед початком навчання нейронних мереж потрібно сформувати та підготувати навчальну вибірку, а також дані для тестуванні роботи на якості тренування нейронної мережі.

Так як в наші часи є досить популярним ділитись своїми думками, враженнями та відгуками через соціальні мережі, а задачею роботи є – показати що нейронні мережі досить вправно справляються з аналізом тональності тексту, у тому числі коментарі то не будемо відходити від поставленої задачі та генерувати певним чином текст, адже можем призвати до не дуже якісних результатів навчання мережі, а також недостовірних результатів роботи мережі. Тому було прийнято рішення скористатись для навчання мереж, зібрану базу даних котра збережена у досить популярному для цього форматі CSV, котра має в собі так звані пости із соціальної мережі Twitter. Ця вибірка містить понад 227 000 розмічених лейблами текстів котрі підходять для навчання та тестування нейронної мережі. Тоді ми маємо 115 000 і 112 000 позитивних та негативних текстів відповідно, розмічені вони значеннями 0 або 1 у залежності від емоційного забарвлення коментаря. Дана вибірка поділена мною у відношенні чотири до одного що дає змогу тренувати нейронну мережу та перевіряти її роботу, тут навчальна вибірка більша від тестової. Також варто уточнити, що в нашій вибірці відсутні речення котрі є малоінформативними – до сорока символів довжиною, а також немає речень котрі можна було віднести до емоційно-нейтральних.

Наступним кроком після пошуку вибірки потрібно навчити модель Word2Vec котра можем визначати певний симантичний зв’язок між словами котрі їй передані, вона це робить виходячи з контексту у котрому вона їх зустрічає. Для навчання було взято вибірку із 17,5 млн текстів котрі призначені саме для навчання подібних моделей. Цю модель далі буде використано при конфігурації embedding шару для наших нейронних мереж. Але перед навчанням

моделі потрібно досить ретельно підготувати дані, адже напряму впливає на якість роботи embedding шару у майбутньому. Для цього виконаємо наступні операції:

- Проведемо видалення та заміну всіх входжень посилань котрі зустрічаються у вибірці на токен URL, це пояснюється тим що мережа не може класифікувати зв'язок посилання з іншими словами адже воно не несе якогось смислового значення котре можна оцінити навіть людиною якщо по ньому не перейти, тож ця заміна зроблена для того щоб покращити результати навчання.
- Виконуємо заміну всіх букв верхнього регістру на нижній.
- Всюди те зустрічається тег користувача - ім'я користувача у соуіальній мережі Twitter, замінюємо його на токен USER по тій самій причині що з URL.
- Наступним кроком видаляємо всі знаки пунктуації котрі є у текстах.
- Всі зміни провередно потужностями мови програмування Python. Нижче наведено приклад підготованого речення до роботи з мережею відповідно до заданих нами стандартів(рисунок 3.1):

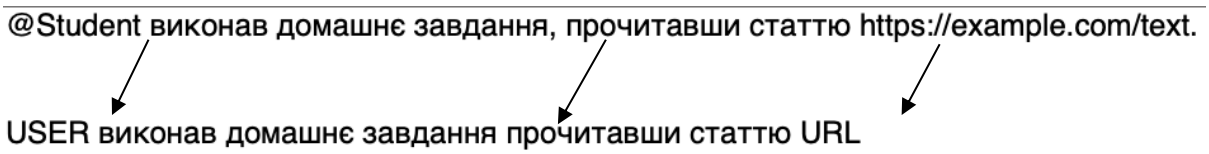


Рисунок 3.1 – приклад обробки тексту

3.4 Побудова та навчання нейронної мережі CNN та аналіз отриманих результатів

Будувати мережу будемо на основі архітектури котра була задана в другому розділі цієї роботи. Таким чином наша мережа CNN має наступну будову:

- Embedding шар котрий відповідає за то щоб дати векторне представлення слова та далі надати результат на вхід нейронної мережі.
- Згорткові шари нейронної мережі котрі більш детально були описані у другому розділі.
- 1-max-pool.
- Шар поєднання у один вектор ознак.
- Прихований шар.
- Вихідний шар мережі.

Першим кроком ми провели навчання моделі Word2Vec на навчальній вибірці із 17,5 млн текстів, ця модель потрібна для embedding шару, котрий виконує векторне представлення слів, варто зазначити що він враховує семантичне значення слів максимально можливо, для передавання їх на вхід мережі. Також досить важливим є то, що ми маємо можливість явно задати параметри мережі завдяки бібліотекам котрі ми описали у минулих підрозділах, тому ми ініціалізували параметр котрий дає змогу відсіяти всі слова з текстів котрі зустрічаються загалом менше 3 разів, адже за такої частоти не можна співставити повного семантичного їх значення. Після навчання моделі ми її завантажили, та заповнили embedding шар нулями, а далі заповнили певною кількістю котра зустрічається найчастіше.

Наступним кроком потрібно визначити введені нами раніше гіперпараметри мережі котрі ми описали раніше. А саме це кількість шарів, розмір фільтрів. Мною було вирішено взяти розміри фільтрів зі значеннями – 2,3,4,5 по десять шарів для кожної з наведених висот фільтрів. У якості активаційної функції було обрано ReLu, адже вона є досить популярним та перевіреним рішенням для цих потреб, але аналогічним рішенням також буває сигмоїда. Наступним кроком має бути шар субдескритизації для виділення основних ознак, а також варто зазначити що ми використаємо Dropout регуляцію для того щоб не отримати у результаті перенавчену мережу, тому було обрано імовірність 0,2, перед прихованим повнозв’язним шаром. У прихованому шарі

буде використано також функцію активації ReLu. Метод Adam котрий ми описали в минулому розділі також буде задіяний як метод оптимізації мережі.

Сам процес навчання CNN мережі ми розбили на два послідовні етапи, такі як – навчання за перших десяти епох, та вже другі п’ять епох.

Після навчання за перших десяти епох, ми отримали досить непогані результати, цей висновок було отримано з метрик оцінки якості роботи мережі котрі ми розглядали раніше - точності, повноти, f-міри та функції loss, та їх графіків. На них маємо значення метрик для кожної епохи на тренувальній та валідаційній вибірці (рисунок 3.2 – 3.5):

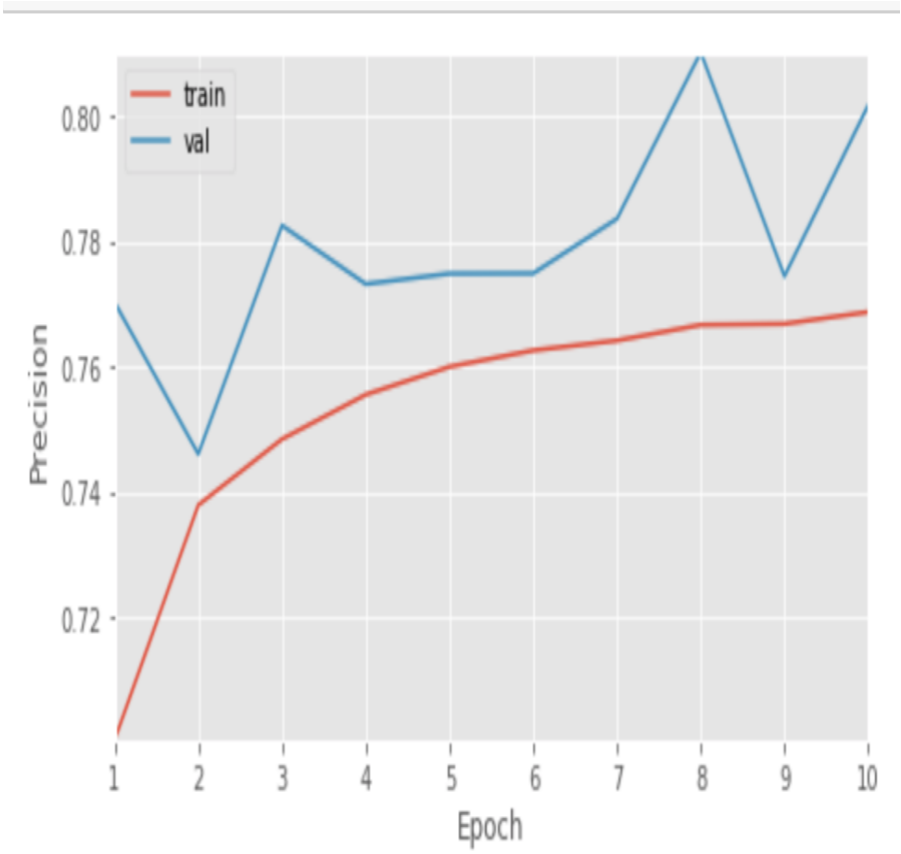


Рисунок 3.2 – Точність на кожній з епох

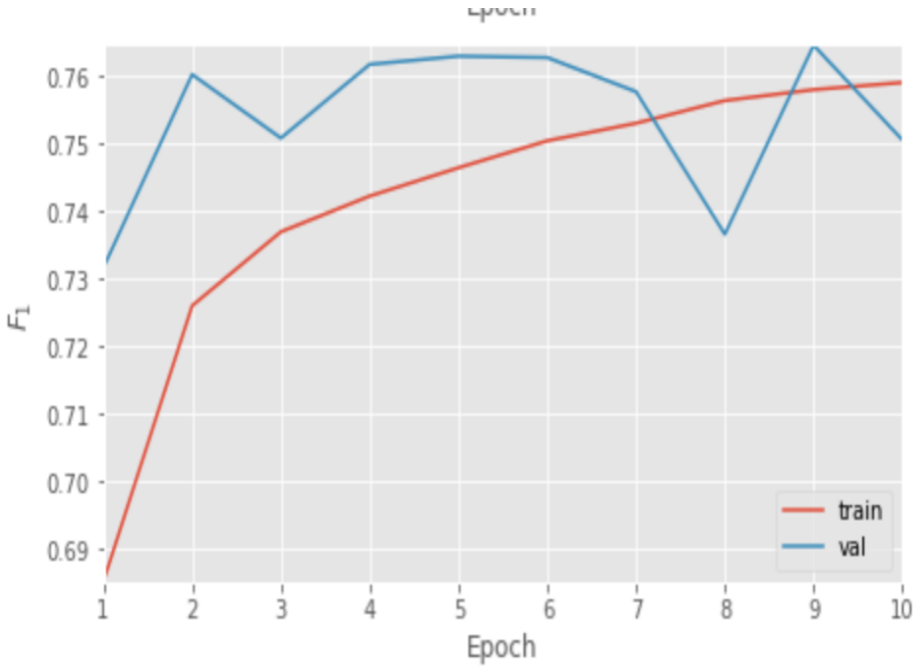


Рисунок 3.3 – F-міра на кожній з епох

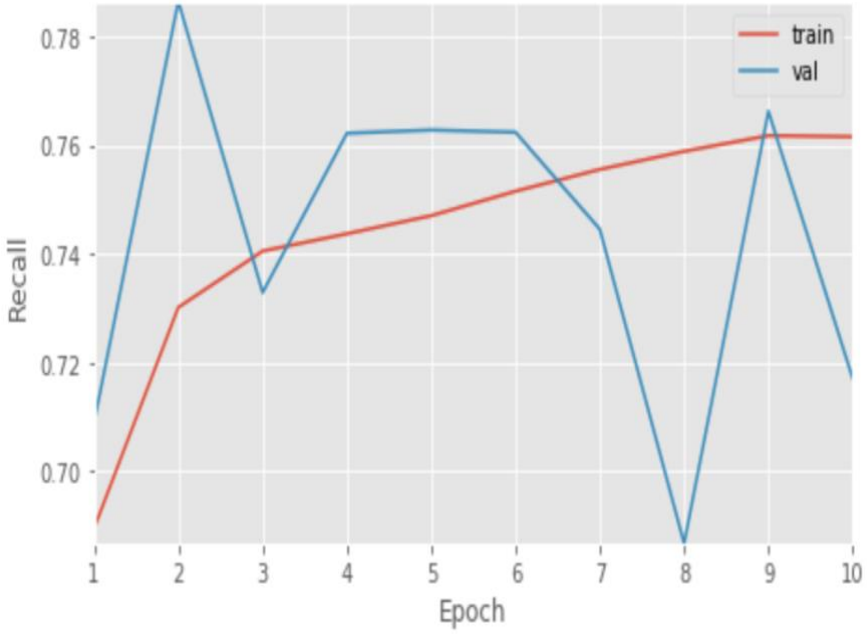


Рисунок 3.4 – повнота на кожній з епох

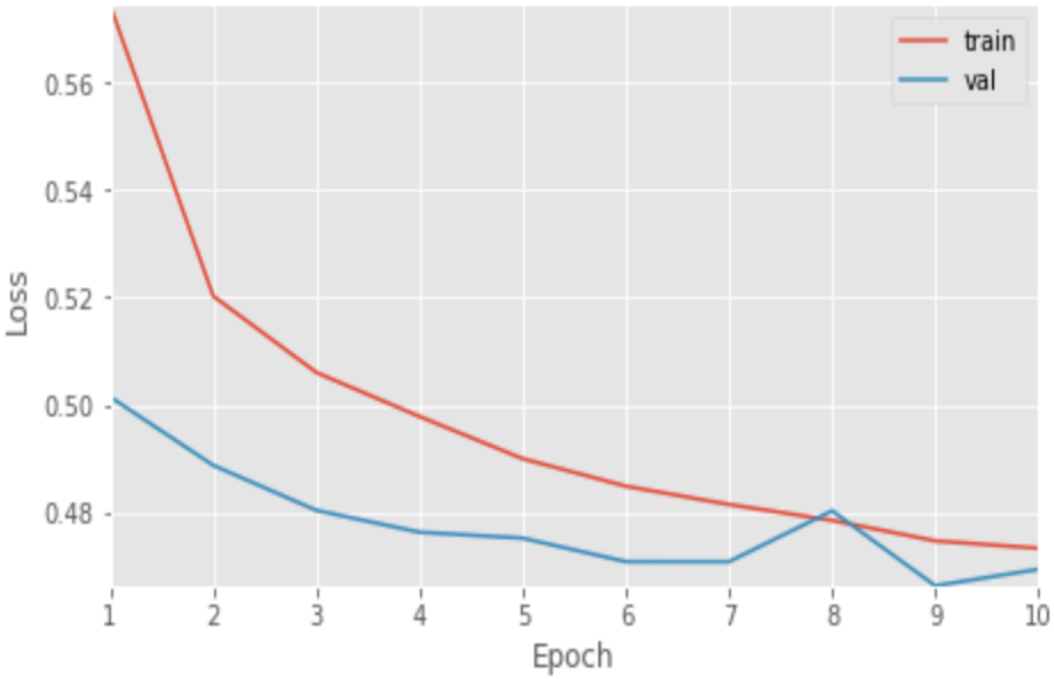


Рисунок 3.5 – loss на кожній з епох

Наступним кроком було перевірено що ми отримали на тестовій вибірці на даному етапі навчання мережі, (рисунок 3.6):

	precision	recall	f1-score	support
0	0.76608	0.77713	0.77156	22457
1	0.77239	0.76117	0.76674	22313

Рисунок 3.6 – параметри на тестовій вибірці

Бачимо що результати навіть на даному етапі досить непогані, але поглянемо на результати після повного навчання мережі, нагадаю що лишилось останні п’ять епох.

Як ми можемо побачити по графікам, мережа навчилась досить добре, F – міра зараз має краще значення, а loss суттєво зменшилась, (рисунок 3.7 – 3.10). Наступним кроком мережі було передано тестову вибірку, та ось яких результатів ми дійшли, (рисунок 3.11):

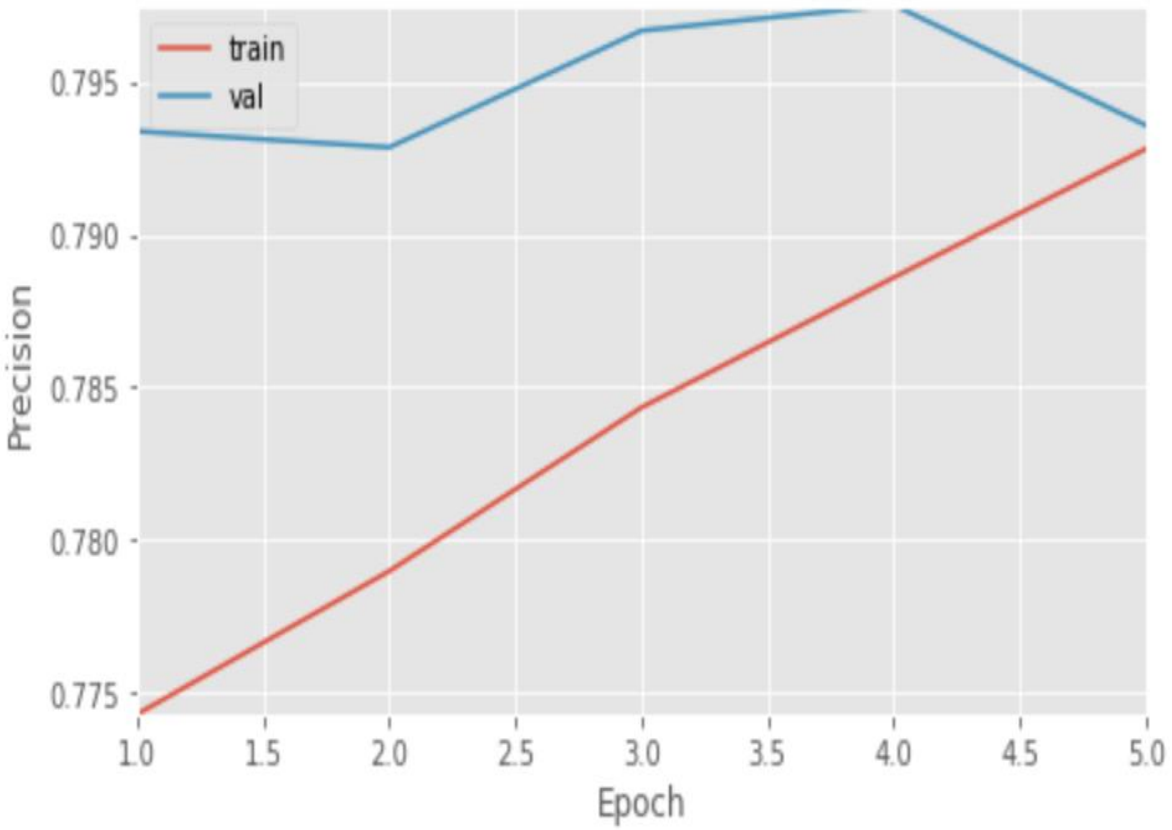


Рисунок 3.7 – точність на кожній з епох

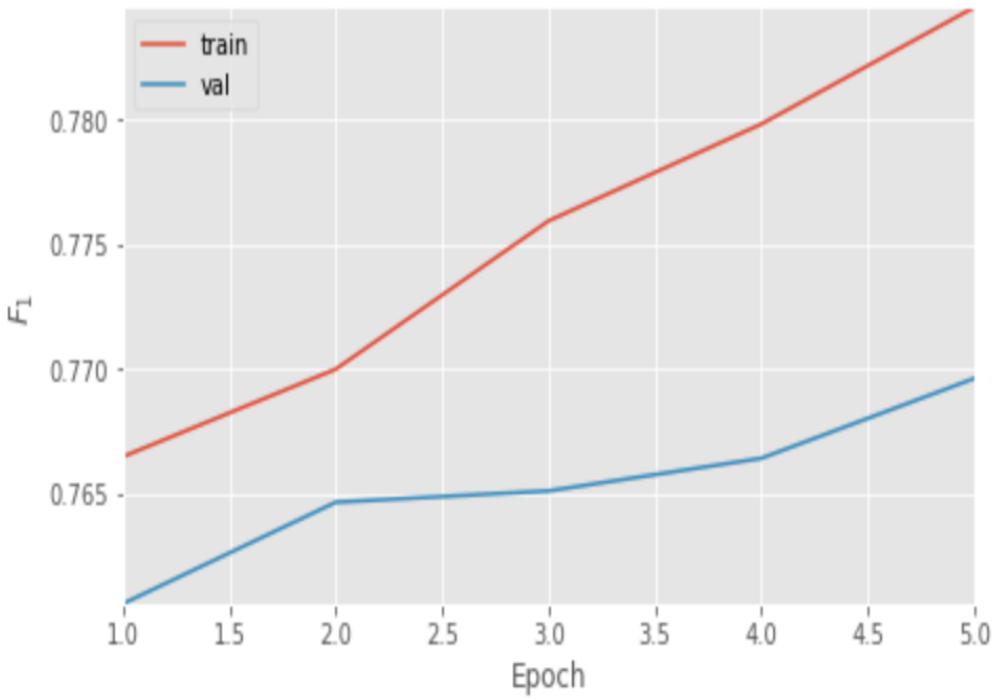


Рисунок 3.8 – F-міра на кожній з епох

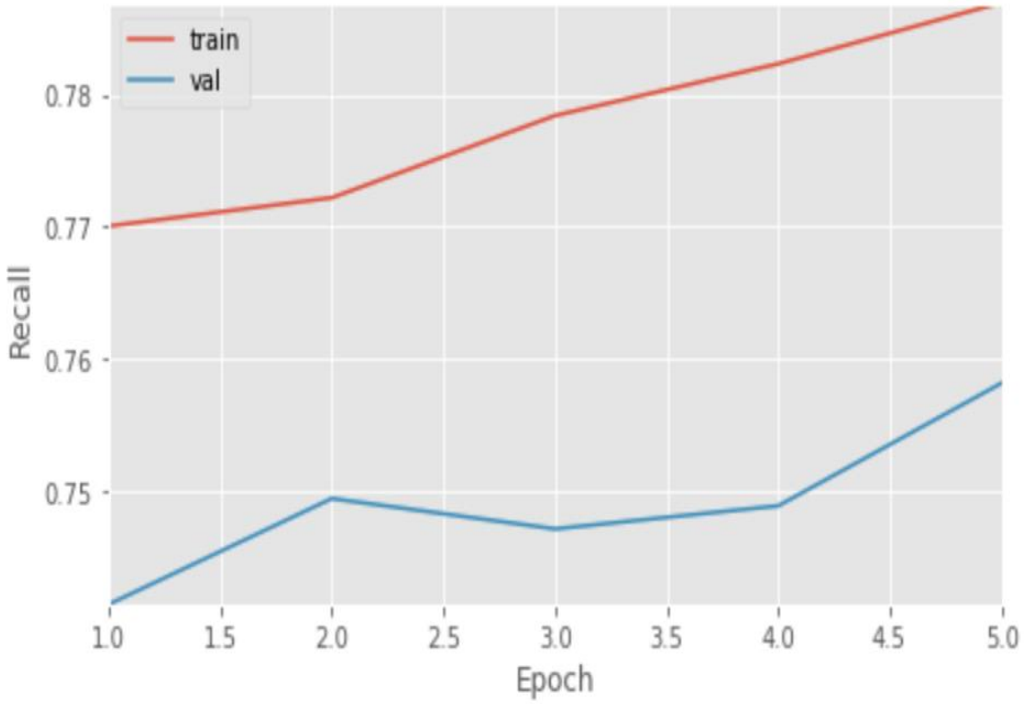


Рисунок 3.9 – повнота на кожній з епох

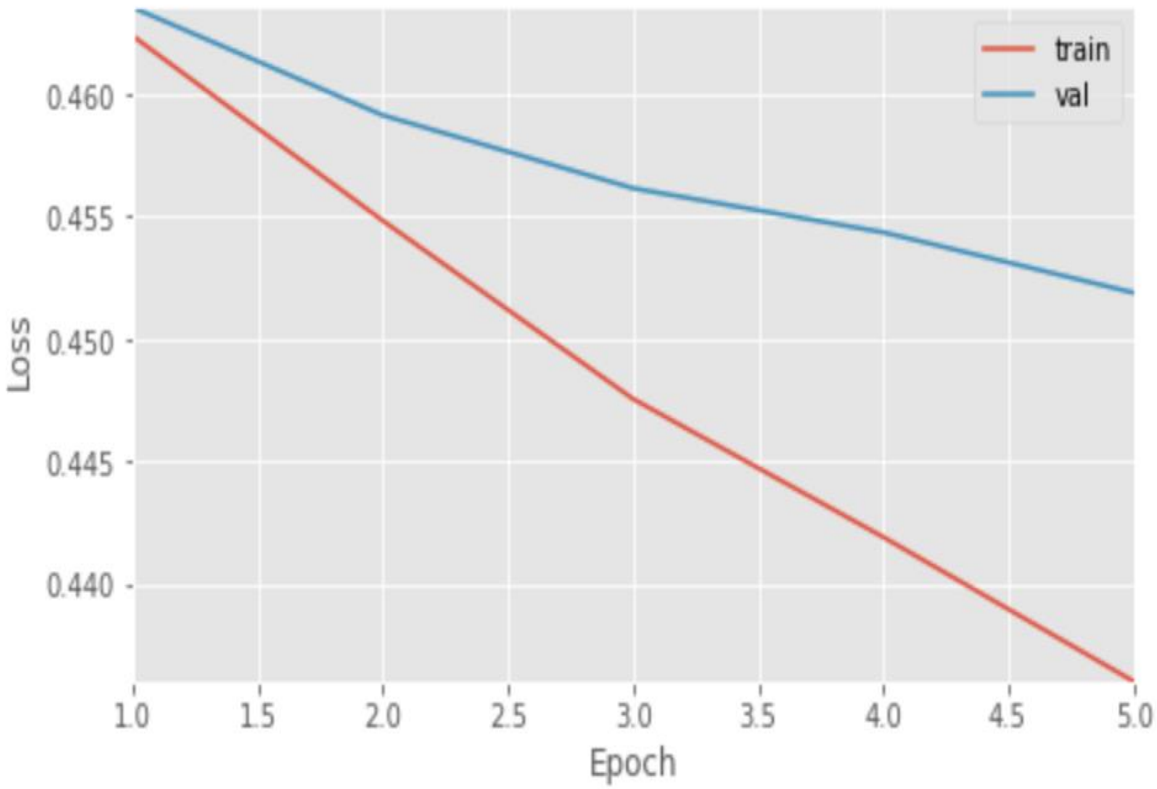


Рисунок 3.10 – loss на кожній з епох

	precision	recall	f1-score
0	0.76814	0.80371	0.78552
1	0.79279	0.75584	0.77387

Рисунок 3.11 – результат тестування мережі

З таблиці результатів та ключових метрик можемо побачити, що результати стали суттєво кращими - приблизно на 1 відсоток, на перший погляд це не великий прогрес, але варто взяти до уваги що на великій вибірці це охоплює досить багато додаткових текстів котрі були класифіковані вірно. На наступному кроці ми навчимо наївний баєсівський класифікатор та протестуємо якість його роботи.

3.4 Побудова та навчання наївного баєсівського класифікатора та аналіз отриманих результатів

Очевидно що для навчання наївного баєсівського класифікатора було використано туж саму тренувальну та тестувальну вибірку. Дані були підготовані таким самим чином як і для CNN. Навчання класифікатора відбувалось використовуючи описану вище бібліотеку sclearn. Як результат було отримано наступні дані по якості роботи моделі, наведені нижче (рисунок 3.12):

	precision	recall	f1-score
0	0.7397	0.7941	0.7659
1	0.7759	0.7183	0.7460

Рисунок 3.12 – результати тестування наївного баєсівського класифікатора

3.4 Побудова і навчання LSTM та підведення підсумків

Для навчання мережі LSTM було використано той самий набір даних що і для двох попередніх рішень, також варто зазначити що підготовка даних була така ж сама. Для навчання мережі було обрано наступну архітектуру на основі теорії котру ми описували у попередньому розділі. Тож наша мережа матиме наступні шари:

- Embedding шар.
- LSTM шар.
- Dropout.
- Повнозв’язний шар.
- Вихідний шар мережі.

Варто зазначити що більшість з шарів ми описали раніше у CNN мережі, але додам що ми маємо дропаут шар котрий запобігає перенавчанню та працює з імовірністю 0,2, а також зазначимо що у цьому випадку будо використано сигмоїдну активаційну функцію. Фактично архітектура нашої мережі представлена на наступній схемі(рисунок 3.13)

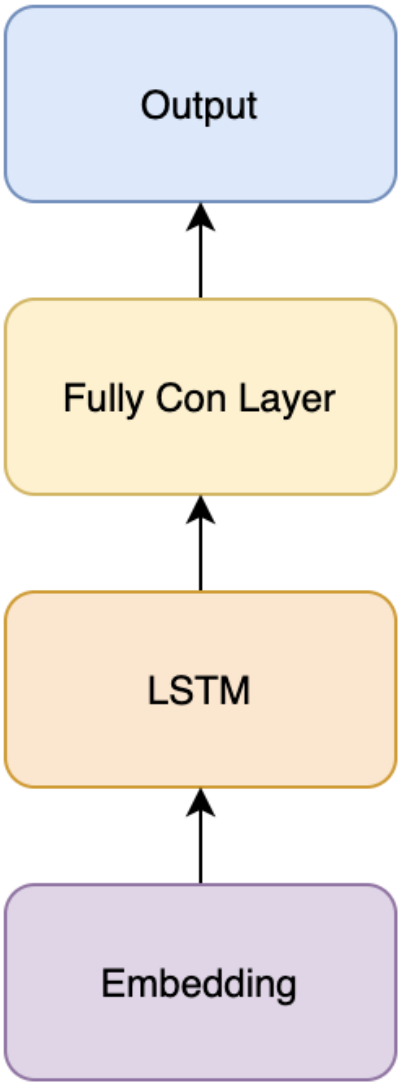


Рисунок 3.13 – архітектура мережі LSTM

Також варто зазначити що імовірність Dropout 0,2 для того щоб уникнути перенавчання мережі. Перший шар моделі – це шар embedding , який використовує вектор довжини 32, наступним шаром є LSTM, який має 100 нейронів, які будуть працювати як одиниця пам’яті моделі. Після LSTM йде саме прихований шар, який є вихідним шаром із сигмоїдною функцією. Результати навчання за 15 епох можемо бачити на рисунках(рисунок 3.14 – 3.15).

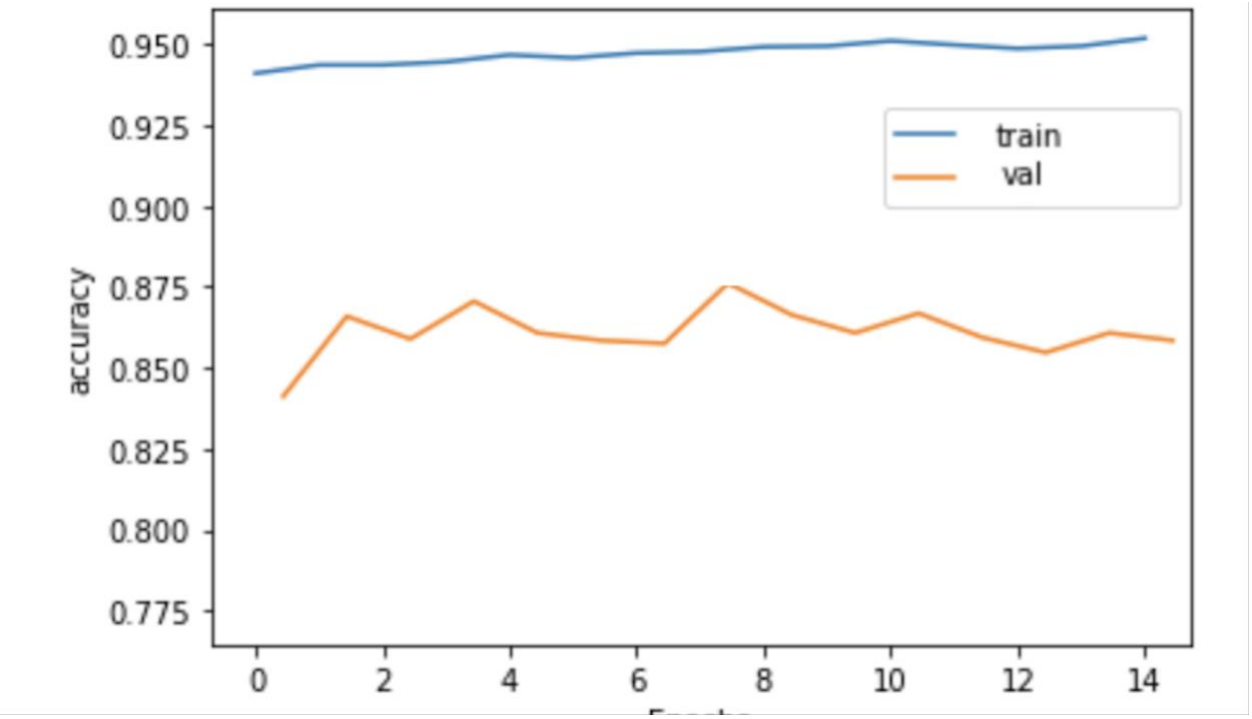


Рисунок 3.14 – точність на кожній з епох

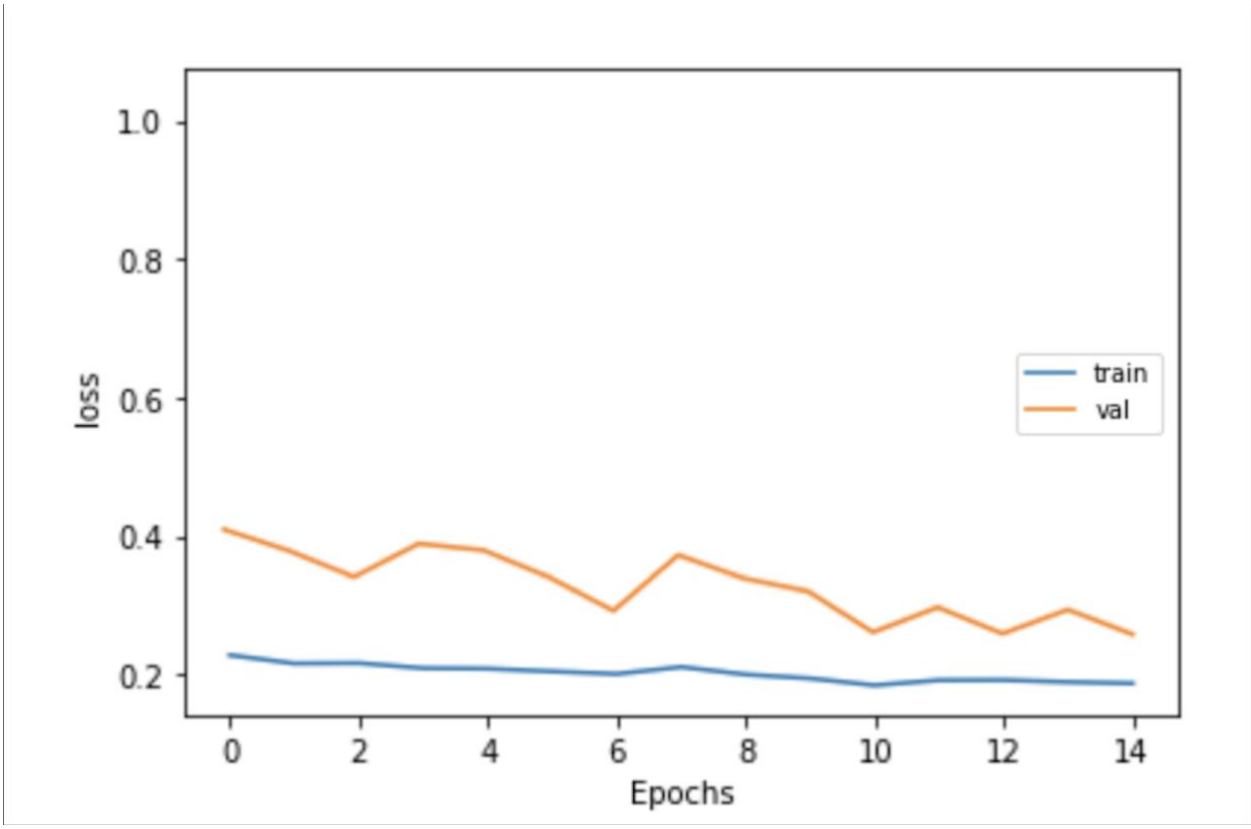


Рисунок 3.15 – loss на кожній з епох

Тож маємо наступні результати навчання та тестування мережі LSTM для поставленої задачі(рисунок 3.16).

	precision	recall	f1-score
0	0.92	0.81	0.86
1	0.86	0.79	0.82

Рисунок 3.16 – результати тестування LSTM

Порівнюючи результати отримані з використанням CNN, результати баєсівського наївного класифікатора та LSTM можемо бачити, що у всіх мережах f-міра має краще значення при визначені негативних речень, це може бути обумовлено неоднорідністю набору даних, з відхиленням в сторону речень з негативним забарвленням, зазвичай цю проблему вирішують прирівнюванням вибірок щоб отримати однакову кількість даних, але у нашому випадку це не мало досить великого впливу щоб виконувати таку операцію. Також можемо помітити що найкращий результат отримано за допомогою мережі LSTM, далі йде CNN і вже потім базове рішення MNB(рисунок 3.17).

Класифікатор	Precision	Recall	F1
CNN	0,7804	0,7798	0,7796
LSTM	0,89	0,8	0,84
MNB	0,7577	0,7564	0,76

Рисунок 3.17 – Порівняльна таблиця

Виходячи з того що можна спробувати різні варіанти навчання мереж то на мою думку результати для кожної мережі можна зробити кращими. Також варто спробувати протестувати різні варіанти будови цих мереж, зменшити чи збільшити значення гіперпараметрів наприклад. Також можна спробувати використати іншу модель для векторного представлення слів, але на мою думку Word2Vec дуже добре виконує поставлену задачу.

3.5 Висновки

У цьому розділі ми розглянули способи та методи для підготовки даних для навчання нейронних мереж для вирішення задачі класифікації тексту за емоційним забарвленням. Та надання їх нейронній мережі, та прийоми які для цього використовуються. Було розглянуто та використано один із методів векторного представлення слів, а також навчання моделі Word2Vec. Побудували основні нейронні мережі для цієї задачі із визначеними нами параметрами та провели процес навчання використовуючи навчальну вибірку, а також процес тестування на тестовій. Як ми визначили – мережа LSTM показала найкращі результати, це можна аргументувати ти що вона має пам’ять, що допомагає їй у визначенні класів речень, чого не може собі дозволити CNN чи баєсівский класифікатор. Для подальшого розвитку мережі можна спробувати поєднати сгорткові шари та LSTM мережу для покращення виокремлення ознак.

4 СТАРТАП АНАЛІЗ ПРОЕКТУ

4.1 Вступ та постановка задачі стартап проекту

В наш час досить актуальною є проблема оцінки якості послуг, товару, контенту за коментарями, у тому числі ця проблема досить вагома для надавачів цих послуг. Одним із способів аналізу коментарів та відгуків є система аналізу емоційної тональності тексту. Робота довзоляє обрати одну з нейронних мереж котра покаже найкращі результати та побудувати на її основі систему для аналізу відгуків у форматі Web додатку.

Для розробки стартап проекту та виведення його на ринок необхідно провести детальне дослідження, яке передбачає виконання наведених нижче чотирьох кроків.

Здійснити маркетинговий аналіз стартап-проекту, в рамках якого:

- розробимо опис ідеї проекту, визначимо основні напрямки використання товару чи послуги та сформулювати основні відмінності від товарів/послуг конкурентів;
- проаналізувати ринкові можливості для його реалізації;
- розробити стратегію виведення товару на ринок базуючись на аналізі ринкового середовища та потреб потенційних користувачів.

Організація стартап-проекту, яка включає такі кроки:

- скласти календарний план реалізації та запуску стартап-проекту;
- визначити плановий обсяг виробництва(поширення) потенційного товару та на його основі розрахувати потребу у матеріальних ресурсах і персоналі для обслуговування;
- розрахувати витрати, необхідні для реалізації проекту, та витрати на запуск проекту.

Виконати фінансово-економічний аналіз та оцінити ризики стартап проекту, в межах якого:

- визначити обсяг інвестиційних витрат;

- розрахувати основні фінансово-економічні показники проекту (собівартість, ціну продукту/послуги, податковий збір та чистий прибуток) та визначити показники інвестиційної привабливості проекту (рентабельність продажів, період окупності проекту);
- визначити основні ризики проекту та способи для їх запобігання.

Розробити заходи з комерціалізації проекту. Цей етап націлений на пошук фінансування проекту та просування інвестиційної пропозиції. Для його досягнення необхідно:

- визначити цільову групу інвесторів та описати їх бізнес інтереси;
- скласти інвестиційну пропозицію: стислий опис проекту для ознайомлення інвестора із стартап-проектом;
- визначити основні канали та заходи для просування оферти інвесторам.

В розділах нижче наведено результати проведених кроків.

4.2 Карта стартап проекту

Стартап проект полягає у створенні системи для аналізу коментарів у різних сервісах де можна виражати власну думку. Система дає змогу за посиланням на товар, ресур, послугу і тд отримати відксоток позетивних та негативних відгуків. Що буде корисно не тільки користувачам, але і тим хто веде свої сторінки в соціальних мережах для того щоб робити свій контент краще.

В таблиці 4.1 представлено основна інформація проекту, розкрито основну ідею та рішення поставленої проблеми.

Таблиця 4.1 – інформаційна карта проекту

1. Назва проекту	Система аналізу тональності коментарів у мережі та парсингу цих коментарів
2. Автори проекту	Зайвелєв Юрій
3. Коротка анотація	Система надає користувачу результат аналізу його відгуків таким чином пряме розуміння що можна покращити у продукті.
4.Термін реалізації проекту	*6 місяців*
5. Необхідні ресурси	Обладнання – комп'ютер. Програмне забезпечення, операційна система, антивірусне обладнання. Електрика, водопостачання, швидкісний інтернет. Фінансові ресурси – заробітна плата працівникам на 6 місяців роботи, гроші на на оплату комунальних послуг, оренди, реклами тощо. Приміщення з усіма необхідними комунікаціями та умовами - коворкінг.
6. Опис проблеми, які вирішує проект	Дана комплексна система дозволяє зменшити затрати часу на ручний аналіз великого об'єму даних, оскільки зараз дуже важливо отримати швидкий результат та думку про свої послуги, контент чи товар. Тому цей продукт дасть змогу досить швидко та точно виявити загальний відгук про товар та виправити його по негативним коментарям або лишити як є.

Продовження таблиці 4.1

7. Головні цілі та завдання проекту	Основна мета проекту – доведення ефективності обраного методу дослідження. Додаткові завдання – новий досвід у розробці стартап проекту, розробка нової комплексної системи, робота із реальними даними та створення комерційно успішного продукту, а також перспектива його розвитку.
8. Очікувані результати	Повністю автоматична система аналізу інформації котра могла б її отримувати за посиланням, а також до навчатися та покращувати результати роботи.

4.3 Технологічний аудит ідеї проекту

Далі була виділено основу ідею стартапу та зведено її в наступній таблиці 4.2.

Таблиця 4.2 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Дана комплексна система дозволяє зменшити затрати часу на ручний аналіз великого об’єму даних, оскільки зараз дуже важливо отримати швидкий результат та думку про свої послуги, контент чи товар.	1. Отримання відгуку про свій продукт чи послугу, або контент.	Система дозволяє досить швидко виконати багато ручної роботи та виключає людський фактор та помилки для виготовника товару.
	2. Отримання інформації про загальний відгук, товар чи контент для потенційного покупця.	Система дозволяє досить швидко виконати багато ручної роботи та виключає людський фактор та помилки для виготовника товару.

Далі було визначено конкурентів на ринку і зроблено порівняльний аналіз програмних продуктів конкурентів, виявлено їх переваги та недоліки. Також було представлено перелік переваг над існуючими програмними рішеннями (таблиця 4.3, 4.4). Варто відмітити відсутність конкурентів на ринку України тому порівняння було зроблене з закордонними конкурентами.

Таблиця 4.3 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	(Потенційні) товари/концепції конкурентів				W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	YouScan	VidIOVision	SeeZislab			
1.	Точність аналізу	Висока	Середня	Середня	Низька			+

Продовження таблиці 4.3

2.	Простота роботи	Висока	Висока	Середня	Висока			+
3.	Ризики невірного аналізу	Середня	Середня	Низька	Низька		+	
4.	Доступність	Висока	Висока	Низька	Низька		+	

Таблиця 4.4 – Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1.	Створення системи надання математичного прогнозу стосовно цін акцій	Використання мови програмування Java	Потрібні опрацювання	Доступні
2.		Використання мови програмування C++	Потрібні опрацювання	Не доступні
3.		Використання мови програмування Python 3.8	Є	Достуані
Обрана технологія реалізації ідеї проекту: мова програмування Python 3.8				

4.4 Аналіз ринкових можливостей запуску стартап-проекту

Наступним кроком необхідно охарактеризувати основні групи потенційних користувачів (таблиця 4.5).

Таблиця 4.5 – Попередня характеристика потенційного ринку стартап-проекту.

№п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	0
2	Загальний обсяг продаж	5 млн $
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі (або по ринку), %	10%

У результаті аналізу ринку можна зробити висновок, що він є достатньо сприятливим для створення програмного продукту та його представлення, оскільки динаміка ринку позитивна, а конкуренти відсутні.Наступним кроком необхідно охарактеризувати основні групи потенційних користувачів продукту і скласти опис вимог кожної такої групи (таблиця 4.6).

Таблиця 4.6 – Характеристика потенційних клієнтів стартап-проекту.

№ п/п	Потреба, що формує ринок	Цільова аудиторія (Цільові сегменти ринку)	Відмінності у поведінці різних цільових груп клієнтів	Вимоги споживачів до товару
1	Необхідність аналізу коментарів власного продукту	Громадяни України в віці від 18 до 45 років	Незначні об’єми аналізу	Простота використання.

Продовження таблиці 4.6

2	Необхідність аналізу коментарів власного контенту	Громадяни України в віці від 14 до 45 років	Великі об’єми аналізу	Простота використання, та точність
3	Необхідність аналізу коментарів контенту чи продукту користувачам	Громадяни України в віці від 14 до 45 років	Великі об’єми аналізу	Простота використання, та точність

Далі необхідно проаналізувати можливі загрози, що можуть виникнути на етапі виведення продукту. Результати представлені у таблиці 4.7.

Таблиця 4.7 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкуренція	Доволі незначний рівень виходу на ринок. рівень виходу на ринок, можливий	Пришвидшити вихід оновлень програмних продуктів
2	Розповсюдження	Ускладнення збуту через недовіру користувачів	Розміщення нових рекламних відео в інтернеті, розширена наукова база, відкритий код.

Також необхідно розглянути можливі фактори, що навпаки сприятимуть запуску проекту. Результати представлені у таблиці 4.8.

Таблиця 4.8 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Відсутність конкуренції	Бути першим на ринку з таким продуктом	Розширення можливостей продукту та якості
2	Створення позитивного іміджу компанії та продукту.	Надання послуг на найвищому рівні, забезпечення задоволення клієнтів та безпеку.	Створення якісної рекламної кампанії, збір відгуків та виправлення помилок.

Наступним кроком необхідно охарактеризувати конкурентне середовище, а саме визначити тип та рівень конкуренції. Результати аналізу наведено у таблиці 4.9.

Таблиця 4.9 – Ступеневий аналіз конкуренції на ринку

Особливості середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспромож но)
1. Вказати тип конкуренції - Досконала конкуренція	Багато систем та команда	Розробити зручний та якісний продукт.
2. За рівнем конкурентної боротьби: міжнародний	Є системи котрі розроблені за кордоном.	Розширення мов котрі можуть опрацьовуватись
3. За галузевою ознакою - внутрішньогалузева	____	____

Продовження таблиці 4.9

4. Конкуренція за видами товарів: товарно-родова	Конкуренція між системами аналізу мови та команд розробки.	Збільшення точності та швидкості обробки даних
5. За характером конкурентних переваг: Нецінова	Різні методи с різною точністю проводять аналіз	Навчання більш точних алгоритмів
6. За інтенсивністю: марочна	Популярний бренд дає досить багато великих переваг	Багато уваги надати розвитку бренду

Далі необхідно виконати детальний аналіз конкуренції за моделлю 5 сил конкуренції Майкла Портера, яка використовується для розуміння структури галузі, аналізу її привабливості з точки зору отримання прибутку, оцінки конкуренції і розробки стратегії бізнесу. Результати аналізу зведено в таблицю 4.10.

Таблиця 4.10 – Аналіз конкуренції в галузі за М. Портером.

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	YouScan	Вже існують	Немає	Якість	Немає
Висновки	Немає прямих конкурентів	Легкий вихід на вітчезняний ринок	-	Якість ПЗ	Випустити якісне ПЗ

За результатами конкурентного середови можемо підтвердити, що на ринку сприятлива ситуація для розробки і запуску наведеного стартап-проекту. Основуючись на проведеному аналізі конкуренції (таблиця 4.10), а також враховуючи характеристики ідей стартап-проекту (таблиця 4.5), характеристики

потенційних клієнтів і їх вимоги до продукту (таблиця 4.6) та факторів ринкового середовища (таблиці 4.7 та 4.8) як результат було сформульовано та обґрунтовано перелік факторів конкурентоспроможності. Аналіз оформлено у таблицю 4.11.

Таблиця 4.11 – Обґрунтування факторів конкурентоспроможності

Фактор конкурентноспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
Низька конкуренція	В межах нашої країни на момент розробки стартапу не було виявлено конкурентів
Доступність програмного продукту	Розроблений продукт є легкодоступним. Для доступу необхідне підключення до мережі Інтернет та браузер.
Зручність використання	Інтерфейс досить простий та інтуітивно зрозумілий, буде створена документація по роботі з сервісом.

Після проведення аналізу можна виділити сильні та слабкі сторони продукту (таблиця 4.12).

Таблиця 4.12 – Порівняльний аналіз сильних та слабких сторін системи

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів						
			–3	–2	–1	0	+1	+2	+3
1	Низька конкуренція	20				+			
2	Відкритість програмного продукту	20			+				

Продовження таблиці 4.12

3	Простота та результативність	19			+				

Фінальним етапом аналізу ринкових можливостей для запуску проекту є складання SWOT аналізу. Це аналіз дозволяє оцінити можливості та загрози бізнесу, а також сильні і слабкі сторони продукту. Результати наведені у таблиці 4.13.

Таблиця 4.13 – SWOT аналіз стартап-проекту

Сильні сторони: відсутність конкурентів; простий інтерфейс; не вимагає спеціальної підготови для використання; висока точність обчислень.	Слабкі сторони: Немає підтримки декількох мов
Можливості: розширення списку мов для аналізу; збільшення розпарсених сторінок; інтеграція з іншими програмними системами.	Загрози: поява конкурентів; розповсюдження .

На основі SWOT-аналізу було спроектовано альтернативну ринкову поведінку для випуску стартап-проекту на ринок та приблизний період реалізації програмного продукту, з урахуванням майбутніх проектів, що можуть бути введені на ринок (таблиця 4.14).

Таблиця 4.14 – Альтернативи ринкового впровадження стартап проекту

№п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Швидкий вихід на ринок з протестованим MVP продуктом, можливі проблеми із точністю аналізу та Інтерфейсом користувача	15%	4 місяці
2	Поступовий вихід з повністю готовим продуктом. Висока точність, готовий інтерфейс.	35%	6 місяців

Отже, в результаті досить глубокого аналізу ринкового та конкурентного середовища, факторів загроз та можливостей, сильних та слабких сторін продукту можемо зробити висновок, що на ринку на момент дослідження сприятливі умови для впровадження товару і, що даний товар відповідає вимогам користувачів. 4.3 Розроблення ринкової стратегії проекту.

Розроблення ринкової стратегії поперше за все передбачає визначення стратегії охоплення ринку. Для цих цілей ми охарактеризуємо цільові групи потенційних споживачів (таблиця 4.15).

Таблиця 4.15 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовністьспоживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті
1	Необхідність аналізу коментарів власного продукту	Висока	30%	Висока
2	Необхідність аналізу коментарів власного контенту	Висока	20%	Низька
3	Необхідність аналізу коментарів контенту чи продукту користувачам	Низька готовність	5%	Низька
Які цільові групи обрано:1,2				

У якості стратегією охоплення ринку було обрано масовий маркетинг - компанія концентрує свою увагу на всіх сегментах споживачів (таблиці 4.16, 4.17, 4.18).

Таблиця 4.16 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку *
	1 та 2	Стратегія масового маркетингу	Висока універсальність, висока якість, ціна.	Масовий маркетинг

Таблиця 4.17 – Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
+	+	-	Стратегія виклику лідера

Таблиця 4.18 – Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)

Продовження таблиці 4.18

Легкість у використанн, швідкисть роботи та точність	Покращення точності та позиціювання себе як продукт з новими технологіями	Точність аналізу, простота.	Іміджеві Доступність та простота Універсальність

4.5 Висновки

В цьому розділі було в цілому реализовано перший етап реалізації стартап проекту, а саме, виконано маркетинговий аналіз стартап проекту. Опираючись на нього можна сказати що є певна можливість ринкового успіху проекту, адже він не має конкурентів як таких, також варто сказати що рентабельність проекту є досить високою. З огляду на потенційну групу клієнтів, а саме, авторів контенту, товарів та послуг є великі перспективи успіху цього продукту та випуск його на декілька платформа у вигляді додатку, наприклад на телефон.

Тож, робота може бути прийнята як стартап що спрямований на незайняті ринки з високою імовірністю на успіх. Про це говорить відсутність конкурентів та величезна і досить активна потенційна аудиторія. Враховуючи вище наведені аргументи, вважаю необхідним продовжувати досліджувати та розвивати наведені у третьому розділі нейронні мережі, а також планувати інтерфейс додатку.

ВИСНОВОК

У ході цієї роботи було проведено дослідження якості роботи та тренування нейронних мереж котрі можна використовувати у якості рішення для задачі розпізнавання емоційної тональності тексту у досить великих об’ємах. А також було порівняно результати котрі отримані від мереж.

Також було проведено огляд та розбір літератури про пов’язаної зі штучними нейронними мережами, а також детально було розглянуто складові нейронних мереж, методи навчання, тестування та оптимізації.

Тож було зконфігуровано CNN мережу, LSTM мережу та MNB, наступним кроком було проведене тестування на заготовленій та та проведено випробування на підготованій, валідованій та виправленій спеціальними методами вибірці. Було задіяно та розглянуто принцип роботи моделі Word2Vec, навчено її та як результат використано у вище наведених мережах.

Як результат LSTM показала кращий результат по точності та помилці при роботі у порівнянні з іншими рішеннями, наступним кроком було визначено напрямок її оптимізації.

ПЕРЕЛІК ПОСИЛАНЬ

1. A Survey of the Recent Architectures of Deep Convolutional Neural Networks URL :(https://arxiv.org/pdf/1901.06032.pdf) (дата звернення: 15.03.2021);
2. Штучні нейронні мережі :обчислення URL :(http://www.immsp.kiev.ua/postgraduate/Biblioteka_trudy/ShtuchnNejronMeregNester2004.pdf) (дата звернення: 20.04.2021);
3. A Comprehensive Guide to Convolutional Neural Networks URL :(https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53) (дата звернення: 10.04.2021);
4. Zhang Y., Wallace B. A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification URL (https://arxiv.org/pdf/1510.03820.pdf) (дата звернення: 10.04.2021);
5. Шитиков В. К., Мастицкий С. Э. (2017) Классификация, регрессия, алгоритмы Data Mining с использованием R URL (https://ranalytics.github.io/data-mining/072-NBC.html) (дата звернення: 20.04.2021);
6. Горбачевская Е.Н. Классификация нейронных сетей. URL: https://cyberleninka.ru/article/n/klassifikatsiya-neyronnyh-setey/viewer) (дата звернення: 11.10.2021);
7. Булка Б.А. Особливості та імплікації застосування машинного навчання для створення музичного матеріалу. *Architecture and art*, 2020. URL: https://www.ukrlogos.in.ua/10.11232-2663-4139.10.04.html (дата звернення: 19.09.2021);

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

```
import pandas as pd
import numpy as np
n = ['id', 'date', 'name', 'text', 'typr', 'rep', 'rtw', 'faw', 'stcount', 'foll', 'frien',
'listcount']
data_positive = pd.read_csv('data/positive.csv', sep=';', error_bad_lines=False,
names=n, usecols=['text'])
data_negative = pd.read_csv('data/negative.csv', sep=';', error_bad_lines=False,
names=n, usecols=['text'])
sample_size = min(data_positive.shape[0], data_negative.shape[0])
raw_data = np.concatenate((data_positive['text'].values[:sample_size],
                        data_negative['text'].values[:sample_size]), axis=0)
labels = [1] * sample_size + [0] * sample_size
import re
def preprocess_text(text):
    text = text.lower().replace("ё", "е")
    text = re.sub('((www\.[^\s]+)|(https?://[^\s]+))', 'URL', text)
    text = re.sub('@[^\s]+', 'USER', text)
    text = re.sub('[^a-zA-Za-яА-Я1-9]+', ' ', text)
    text = re.sub(' +', ' ', text)
    return text.strip()
data = [preprocess_text(t) for t in raw_data]
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(data, labels, test_size=0.2,
random_state=2)
import sqlite3
# Відкриваємо бд
```

```
conn = sqlite3.connect('mysqlite3.db')
c = conn.cursor()
with open('data/tweets.txt', 'w', encoding='utf-8') as f:
    # Читаємо текст
    for row in c.execute('SELECT ttext FROM sentiment'):
        if row[0]:
            tweet = preprocess_text(row[0])
            # Пишемо текст у файл
            print(tweet, file=f)
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
SENTENCE_LENGTH = 26
NUM = 100000
def get_sequences(tokenizer, x):
    sequences = tokenizer.texts_to_sequences(x)
    return pad_sequences(sequences, maxlen=SENTENCE_LENGTH)
tokenizer = Tokenizer(num_words=NUM)
tokenizer.fit_on_texts(x_train)
x_train_seq = get_sequences(tokenizer, x_train)
x_test_seq = get_sequences(tokenizer, x_test)
import logging
import multiprocessing
import gensim
from gensim.models import Word2Vec
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s',
level=logging.INFO)
# Считуємо нерозмічені дані
data = gensim.models.word2vec.LineSentence('data/tweets.txt')
# Починаємо навчання Word2Vec
```

```
model = Word2Vec(data, size=200, window=5, min_count=3,
workers=multiprocessing.cpu_count())
model.save("models/w2v/model.w2v")
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
# Обмежуємо кількість слів у речені
SENTENCE_LENGTH = 26
NUM = 100000
def get_sequences(tokenizer, x):
    sequences = tokenizer.texts_to_sequences(x)
    return pad_sequences(sequences, maxlen=SENTENCE_LENGTH)
tokenizer = Tokenizer(num_words=NUM)
tokenizer.fit_on_texts(x_train)
x_train_seq = get_sequences(tokenizer, x_train)
x_test_seq = get_sequences(tokenizer, x_test)
from gensim.models import Word2Vec
# Завантажуємо модель
w2v_model = Word2Vec.load('models/w2v/model.w2v')
DIM = w2v_model.vector_size
embedding_matrix = np.zeros((NUM, DIM))
# Додаємо NUM слів
for word, i in tokenizer.word_index.items():
    if i >= NUM:
        break
    if word in w2v_model.wv.vocab.keys():
        embedding_matrix[i] = w2v_model.wv[word]
from keras.layers import Input
from keras.layers.embeddings import Embedding
tweet_input = Input(shape=(SENTENCE_LENGTH,), dtype='int32')
```

```
tweet_encoder = Embedding(NUM, DIM,
input_length=SENTENCE_LENGTH,
                    weights=[embedding_matrix], trainable=False)(tweet_input)
from keras import backend as K
def precision(y_true, y_pred):
    """Precision metric.
    Only computes a batch-wise average of precision.
    Computes the precision, a metric for multi-label classification of
    how many selected items are relevant.
    """
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision
def recall(y_true, y_pred):
   """Recall metric.
    Only computes a batch-wise average of recall.
    Computes the recall, a metric for multi-label classification of
    how many relevant items are selected.
    """
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall



def f1(y_true, y_pred):
    def recall(y_true, y_pred):
        """Recall metric.
        Only computes a batch-wise average of recall.
```

```
        Computes the recall, a metric for multi-label classification of
        how many relevant items are selected.
        """
        true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
        possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
        recall = true_positives / (possible_positives + K.epsilon())
        return recall
    def precision(y_true, y_pred):
        """Precision metric.
        Only computes a batch-wise average of precision.
        Computes the precision, a metric for multi-label classification of
        how many selected items are relevant.
        """
        true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
        predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
        precision = true_positives / (predicted_positives + K.epsilon())
        return precision
    precision = precision(y_true, y_pred)
    recall = recall(y_true, y_pred)
    return 2 * ((precision * recall) / (precision + recall + K.epsilon()))
from keras import optimizers
from keras.layers import Dense, concatenate, Activation, Dropout
from keras.models import Model
from keras.layers.convolutional import Conv1D
from keras.layers.pooling import GlobalMaxPooling1D
branches = []
# Добавляем dropout-регуляризацию
x = Dropout(0.2)(tweet_encoder)
for size, filters_count in [(2, 10), (3, 10), (4, 10), (5, 10)]:
    for i in range(filters_count):
```

```
    # Додаємо згортку
    branch = Conv1D(filters=1, kernel_size=size, padding='valid',
activation='relu')(x)
    # Додаємо субдекскритизацію
    branch = GlobalMaxPooling1D()(branch)
    branches.append(branch)
# Поєднуємо ознаки
x = concatenate(branches, axis=1)
# Додаємо дропаут
x = Dropout(0.2)(x)
x = Dense(30, activation='relu')(x)
x = Dense(1)(x)
output = Activation('sigmoid')(x)
model = Model(inputs=[tweet_input], outputs=[output])
model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=[precision, recall, f1])
model.summary()
from keras.utils.vis_utils import plot_model
plot_model(model, to_file='model.png')
from keras.callbacks import ModelCheckpoint
checkpoint = ModelCheckpoint("models/cnn/cnn-frozen-embeddings-
{epoch:02d}-{val_f1:.2f}.hdf5",
                        monitor='val_f1', save_best_only=True, mode='max',
period=1)
history = model.fit(x_train_seq, y_train, batch_size=32, epochs=10,
validation_split=0.25, callbacks = [checkpoint])
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('ggplot')
def plot_metrix(ax, x1, x2, title):
```

```
    ax.plot(range(1, len(x1) + 1), x1, label='train')
    ax.plot(range(1, len(x2) + 1), x2, label='val')
    ax.set_ylabel(title)
    ax.set_xlabel('Epoch')
    ax.legend()
    ax.margins(0)
def plot_history(history):
    fig, axes = plt.subplots(ncols=2, nrows=2, figsize=(16, 9))
    ax1, ax2, ax3, ax4 = axes.ravel()
    plot_metrix(ax1, history.history['precision'], history.history['val_precision'],
'Precision')
    plot_metrix(ax2, history.history['recall'], history.history['val_recall'],
'Recall')
    plot_metrix(ax3, history.history['f1'], history.history['val_f1'], "$F_1$")
    plot_metrix(ax4, history.history['loss'], history.history['val_loss'], 'Loss')
    plt.show()
plot_history(history)
model.load_weights('models/cnn/cnn-frozen-embeddings-09-0.76.hdf5')
from sklearn.metrics import classification_report
predicted = np.round(model.predict(x_test_seq))
print(classification_report(y_test, predicted, digits=5))
from keras import optimizers
model.layers[1].trainable = True
adam = optimizers.Adam(lr=0.0001)
model.compile(loss='binary_crossentropy', optimizer=adam,
metrics=[precision, recall, f1])
model.summary()
checkpoint = ModelCheckpoint("models/cnn/cnn-trainable-{epoch:02d}-
{val_f1:.2f}.hdf5",monitor='val_f1', save_best_only=True, mode='max',
period=1)
```

```
    history_trainable = model.fit(x_train_seq, y_train, batch_size=32, epochs=5,
    validation_split=0.25, callbacks = [checkpoint])
model.load_weights('models/cnn/cnn-trainable-05-0.77.hdf5')
    predicted = np.round(model.predict(x_test_seq))
    print(classification_report(y_test, predicted, digits=5))
    print(predicted)
    print(x_test_seq)
    plot_history(history_trainable)
    n = ['id', 'date','name','text','typr','rep','rtw','faw','stcount','foll','frien','listcount']
    data_positive = pd.read_csv('data/positive.csv', sep=';',error_bad_lines=False,
    names=n, usecols=['text'])
    data_negative = pd.read_csv('data/negative.csv', sep=';',error_bad_lines=False,
    names=n, usecols=['text'])
    sample_size = min(data_positive.shape[0], data_negative.shape[0])
    raw_data = np.concatenate((data_positive['text'].values[:sample_size],
                        data_negative['text'].values[:sample_size]), axis=0)
    labels = [1]*sample_size + [0]*sample_size
    import re

    def preprocess_text(text):
        text = re.sub('((www\.[^\s]+)|(https?://[^\s]+))','URL', text)
        text = re.sub('@[^\s]+','USER', text)
        text = text.lower().replace("ё", "е")
        text = re.sub('[^a-zA-Za-яA-Я1-9]+', ' ', text)
        text = re.sub(' +',' ', text)
        return text.strip()

    data = [preprocess_text(t) for t in raw_data]
    from sklearn.pipeline import Pipeline
```

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.model_selection import train_test_split, GridSearchCV

text_clf = Pipeline([('vect', CountVectorizer()),
                     ('tfidf', TfidfTransformer()),
                     ('clf', MultinomialNB())])

tuned_parameters = {
    'vect__ngram_range': [(1, 1), (1, 2), (2, 2)],
    'tfidf__use_idf': (True, False),
    'tfidf__norm': ('l1', 'l2'),
    'clf__alpha': [1, 1e-1, 1e-2]
}
x_train, x_test, y_train, y_test = train_test_split(data, labels, test_size=0.33,
random_state=42)
from sklearn.metrics import classification_report

score = 'f1_macro'
print("# Tuning hyper-parameters for %s" % score)
print()
np.errstate(divide='ignore')
clf = GridSearchCV(text_clf, tuned_parameters, cv=10, scoring=score)
clf.fit(x_train, y_train)

print("Best parameters set found on development set:")
print()
print(clf.best_params_)
print()
```

```
print("Grid scores on development set:")
print()
for mean, std, params in zip(clf.cv_results_['mean_test_score'],
                             clf.cv_results_['std_test_score'],
                             clf.cv_results_['params']):
    print("%0.3f (+/-%0.03f) for %r" % (mean, std * 2, params))
print()

print("Detailed classification report:")
print()
print("The model is trained on the full development set.")
print("The scores are computed on the full evaluation set.")
print()
print(classification_report(y_test, clf.predict(x_test), digits=4))
print()
```