МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

В.М. Шимкович

Дослідження і проектування інтелектуальних систем

Лабораторний практикум

Навчальний посібник

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня магістра
за освітньою програмою «Комп'ютерні системи та мережі»
спеціальності 123 «Комп'ютерна інженерія»

Електронне мережне навчальне видання

Київ
КПІ ім. Ігоря Сікорського
2022

Рецензент: *Кравець П.І.*, к.т.н., доцент, КПІ ім. Ігоря Сікорського, факультет інформатики та обчислювальної техніки, кафедра інформаційних систем та технологій

Відповідальний редактор: *Волокита А. М.*, к.т.н., доцент, КПІ ім. Ігоря Сікорського, факультет інформатики та обчислювальної техніки

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського

(протокол № від .)

за поданням Вченої ради факультету Інформатики та обчислювальної техніки (протокол № від .)

Навчальний посібник охоплює теоретичний матеріал та практичні завдання, які необхідні для виконання лабораторного практикуму з дисципліни «Дослідження і проектування інтелектуальних систем». Практикум виконується у комп'ютерному класі кафедри з використанням мови програмування python.
Робота може бути корисною студентам відповідних спеціальностей при вивченні дисциплін, пов'язаних із розробкою та використаннямм програмних засобів , а також при виконанні бакалаврських робіт, курсових проектів, магістерських робіт, в яких використовуються нейронні мережі. Останнє було враховано при оформленні роботи, яке виконано згідно вимог до конструкторської документації.

Реєстр. № НП 21/22-493. Обсяг 5,9 авт. арк.

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
проспект Перемоги, 37, м. Київ, 03056
https://kpi.ua
Свідоцтво про внесення до Державного реєстру видавців, виготовлювачів і розповсюджувачів видавничої продукції ДК № 5354 від 25.05.2017 р.

ЗМІСТ

ЛАБОРАТОРНА РОБОТА №1

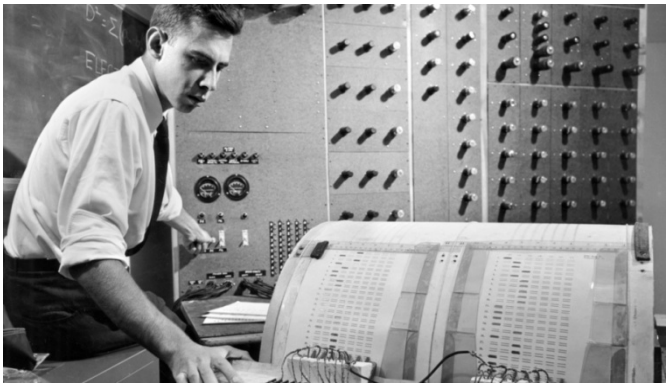
Парцептрон

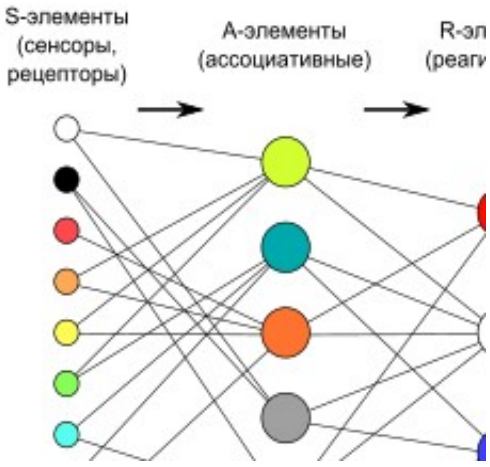
Завдання: Написати програму, що реалізує нейронну мережу Парцептрон та навчити її виконувати функцію XOR.

Перцептрон - математична або комп'ютерна модель сприйняття інформації мозком (кібернетична *модель мозку*), запропонована Френком Розенблатта в 1957 році і вперше реалізована у вигляді електронної машини «Марк-1» в 1960 році. Перцептрон став однією з перших моделей ШНМ, А «Марк-1» - першим у світі нейрокомп'ютером.

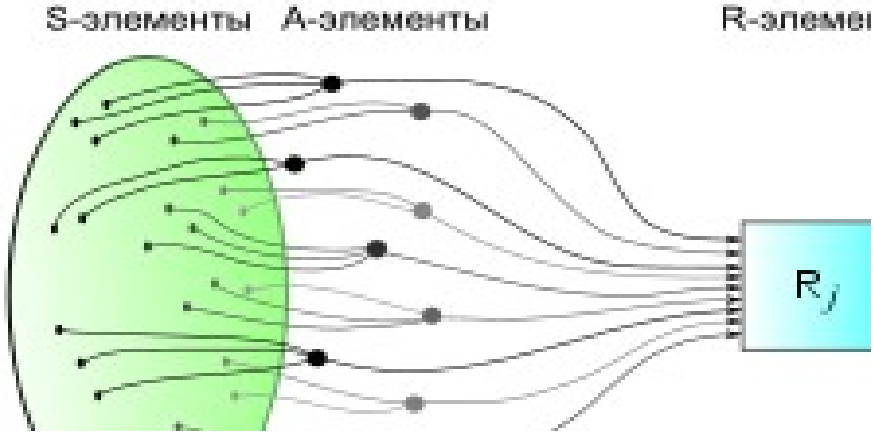
Перцептрон складається з трьох типів елементів, а саме: що надходять від датчиків сигнали передаються асоціативним елементам, а потім реагуючим елементам. Таким чином, перцептрони дозволяють створити набір «асоціацій» між вхідними стимулами і необхідної реакцією на виході. У біологічному плані це відповідає перетворенню, наприклад, зорової інформації в фізіологічний відповідь від рухових нейронів. Відповідно до сучасної термінології, перцептрони можуть бути класифіковані як штучні нейронні мережі:

- з одним прихованим шаром;
- з пороговою функцією активації;
- з прямим розповсюдженням сигналу.





Елементарний перцептрон складається з елементів трьох типів: S-елементів, A-елементів і *одного* R-елементу. S-елементи - це шар сенсорів або рецепторів. У фізичному втіленні вони відповідають, наприклад, світлочутливим клітинам сітківки ока або фоторезисторам матриці камери. Кожен рецептор може перебувати в одному з двох станів -*спокою* або *збудження*, і тільки в останньому випадку він передає одиничний сигнал в наступний шар, асоціативним елементам.



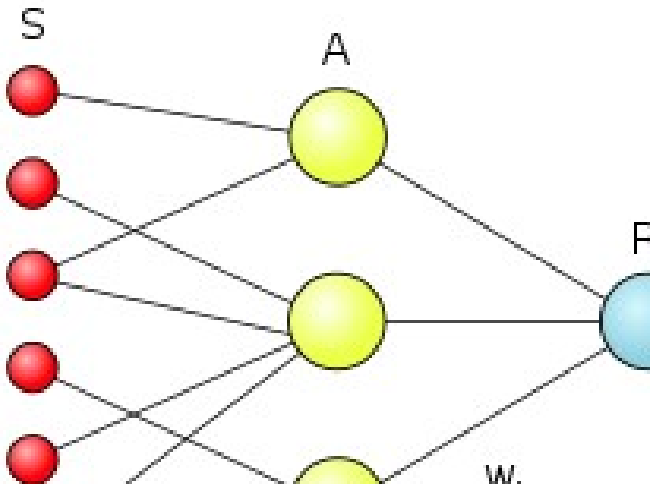
A-елементи називаються асоціативними, тому що кожному такому елементу, як правило, відповідає цілий набір (асоціація) S-елементів. A-елемент активізується, як тільки кількість сигналів від S-елементів на його вході перевищило деяку величину θ. Таким чином, якщо набір відповідних S-елементів розташовується на сенсорному полі в формі літери «Д», A-елемент активізується, якщо достатня кількість рецепторів повідомило про появу «білої плями світла» в їх околиці, тобто A-елемент буде як би асоційований з наявністю / відсутністю літери «Д» в деякій області.

Сигнали від збуджених A-елементів, в свою чергу, передаються в акумулятор R, причому сигнал від *i*-го асоціативного елемента передається з коефіцієнтом w_i. Цей коефіцієнт називається *вагою* A-R зв'язку.

Так само як і A-елементи, R-Елемент підраховує суму значень вхідних сигналів, помножених на ваги. R-елемент, а разом з ним і елементарний перцептрон, видає «1», якщо лінійна форма перевищує поріг θ, інакше на виході буде «-1». Математично, функцію, реалізовануиR-елементом, можна записати так:

навчання перцептрону полягає у зміні вагових коефіцієнтів w_i зв'язків A-R. Ваги зв'язків S-A (які можуть набувати значень {-1; 0; +1}) і значення порогів A-елементів вибираються випадковим чином на самому початку і потім не змінюються.

Після навчання перцептрон готовий працювати в режимі *розпізнавання* або *узагальнення*. У цьому режимі перцептроном пред'являються раніше невідомі йому об'єкти, і перцептрон повинен встановити, до якого класу вони належать. Робота перцептрону полягає в наступному: при пред'явленні об'єкта збуджені A-елементи передають сигнал R-елементом, який дорівнює сумі відповідних коефіцієнтів w_i. Якщо ця сума позитивна, то приймається рішення, що даний об'єкт належить до першого класу, а якщо вона негативна - то до другого.



Для початку визначимо складові елементи перцептрону, які є окремими випадками штучного нейрона з порогової функцією передачі.

- простим S-елементом(сенсором) є чутливий елемент, який від впливу будь-якого з видів енергії (наприклад, світла, звуку, тиску, тепла і т.д.) Виробляє сигнал. Якщо вхідний сигнал перевищує певний поріг θ, на виході елемента отримуємо +1, в іншому випадку - 0.

- простим А-елементом (асоціативним) називається логічний вирішальний елемент, який дає вихідний сигнал +1, коли алгебраїчна сума його вхідних сигналів перевищує деяку порогову величину θ (кажуть, що елемент *активний*), в іншому випадку вихід дорівнює нулю.
- простим R-елементом(реагуючим, тобто діючим) називається елемент, який видає сигнал +1, якщо сума його вхідних сигналів є строго позитивної, і сигнал -1, якщо сума його вхідних сигналів є строго негативною. Якщо сума вхідних сигналів дорівнює нулю, вихід вважається або рівним нулю, або невизначеним.

Якщо на виході будь-якого елементу ми отримуємо 1, то кажуть, що елемент активний або збуджений.

Всі розглянуті елементи називаються *простими*, так як вони реалізують *стрибкоподібні функції*. Розенблатт також стверджував, що для вирішення більш складних завдань можуть знадобитися інші види функцій, наприклад, лінійна.

ЛАБОРАТОРНА РОБОТА №2

Реалізація базових архітектур нейронних мереж

Мета роботи: Дослідити структуру та принцип роботи нейронної мережі. За допомогою нейронної мережі змоделювати функцію двох змінних.

Короткі теоретичні відомості

Штучні нейронні мережі (ШНМ) – математичні моделі, а також їхня програмна та апаратна реалізація, побудовані за принципом функціонування біологічних нейронних мереж – мереж нервових клітин живого організму. Системи, архітектура і принцип дії базується на аналогії з мозком живих істот. Ключовим елементом цих систем виступає штучний нейрон як імітаційна модель нервової клітини мозку – біологічного нейрона (рисунок 4.1). Даний термін виник при вивченні процесів, які відбуваються в мозку, та при спробі змоделювати ці процеси. Першою такою спробою були нейронні мережі Маккалока і Піттса. Після розробки алгоритмів навчання, отримані моделі стали використовуватися в практичних цілях: в задачах прогнозування, для розпізнавання образів, в задачах керування та інші.

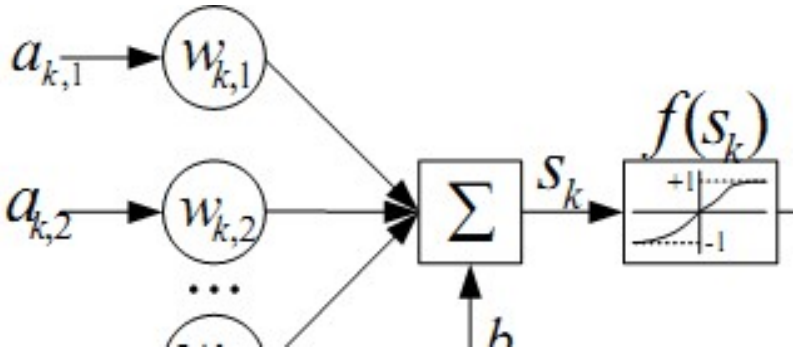


Рисунок 4.1 – Схема штучного нейрону

де q_k – вихідний сигнал k-го нейрона;

f – активаційна функція нейрона;

$a_{k,i}$ – вхідні сигнали k–го нейрона;

w_{ki} – синаптична вага k–го нейрона;

b_k – зміщення k–го нейрона.

ШНМ представляють собою систему з'єднаних і взаємодіючих між собою простих процесорів(штучних нейронів). Такі процесори зазвичай достатньо прості, особливо в порівнянні з процесорами, що використовуються в персональних комп'ютерах. Кожен процесор схожої мережі має справу тільки з сигналами, які він періодично отримує, і сигналами, які він періодично посилає іншим процесорам. І тим не менш, будучи з'єднаними в досить велику мережу з керованою взаємодією, такі локально прості процесори разом здатні виконувати достатньо складні завдання. З точки зору машинного навчання, нейронна мережа являє собою окремий випадок методів розпізнавання образів, дискримінантного аналізу, методів кластеризації тощо З математичної точки зору, навчання нейронних мереж – це багатопараметрична задача нелінійної оптимізації. З точки зору кібернетики, нейронна мережа використовується в задачах адаптивного управління і як алгоритми для робототехніки. З точки зору розвитку обчислювальної техніки та програмування, нейронна мережа – спосіб вирішення проблеми ефективного паралелізму . А з точки зору штучного інтелекту, ШНМ є основою філософської течії коннективізму і основним напрямком в структурному підході з вивчення можливості побудови (моделювання) природного інтелекту за допомогою комп'ютерних алгоритмів. Нейронні мережі не програмуються в звичайному розумінні цього слова, вони навчаються. Можливість навчання – одна з головних переваг нейронних мереж перед традиційними алгоритмами. Технічно навчання полягає в знаходженні коефіцієнтів зв'язків між нейронами. У процесі навчання нейронна мережа здатна виявляти складні залежності між вхідними даними і вихідними, а також виконувати узагальнення. Це означає, що у разі успішного навчання мережа зможе повернути вірний результат на підставі даних, які були відсутні в навчальній вибірці, а також неповних та / або «зашумленних», частково перекручених даних.

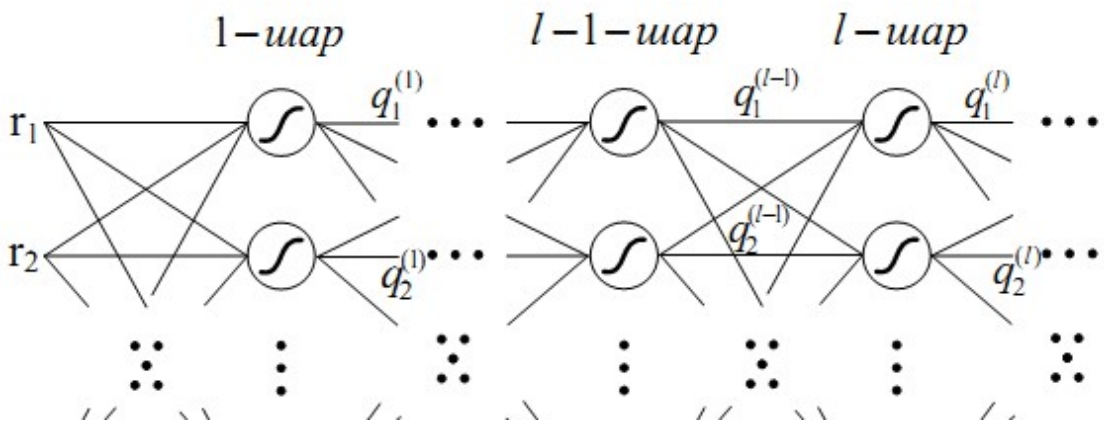


Рисунок 4.2 – Структурна схема багатошарової нейронної мережі прямого розповсюдження

Біологічна нейронна мережа складається з групи або декількох груп хімічно або функціонально пов'язаних нейронів, як показано на рисунку 4.2. Один нейрон може бути пов'язаний з багатьма іншими нейронами, а загальна кількість нейронів та зв'язків між ними може бути дуже великою. Зв'язки, які називаються синапсами, як правило формуються від аксонів до дендритів, хоча дендро-дендритичні мікросхем та інші зв'язки є можливими. Крім електричної передачі сигналів, також є інші форми передачі, які виникають з нейротрансмітерної(хімічний передавач імпульсів між нервовими клітинами) дифузії, і мають вплив на електричну передачу сигналів. Таким чином, нейронні мережі є надзвичайно складними.

Штучний інтелект і когнітивне моделювання намагаються імітувати деякі властивості біологічних нейронних мереж. Хоча аналогічні в своїх методах, перша має на меті вирішення конкретних завдань, а друга спрямована на створення математичних моделей біологічних нейронних систем.

У сфері штучного інтелекту, штучні нейронні мережі були успішно застосовані для розпізнавання мови, аналізу зображень та адаптивного управління, для того, щоб побудувати так званих програмних агентів (в комп'ютерних і відео ігор) або автономні роботи. На даний час, більшість розроблених штучних нейронних мереж для штучного інтелекту основі на

статистичних оцінках, класифікації оптимізації та теорії керування.

Сфера когнітивного моделювання включає в себе фізичне або математичне моделювання поведінки нейронних систем; від індивідуального нейроного рівня, через нейронний кластерний рівень до завершеного організму (наприклад, моделювання поведінки відповіді організму на подразники). Штучний інтелект, когнітивне моделювання і нейронні мережі є парадигмами обробки інформації натхненні системами біологічних нейроннів обробки інформації.

Порядок виконання роботи

Варіант завдання на дану лабораторну роботу вибираються з додатка Б за номером групи та номером студента в списку групи.

Для побудови та навчання нейронної мережі необхідно мати дані, які представляють собою набори значень виходів з відповідними їм значеннями входів. Тому шляхом моделювання функції двох змінних ці дані можуть бути зібрані з входу та виходу функції в масиви значень.

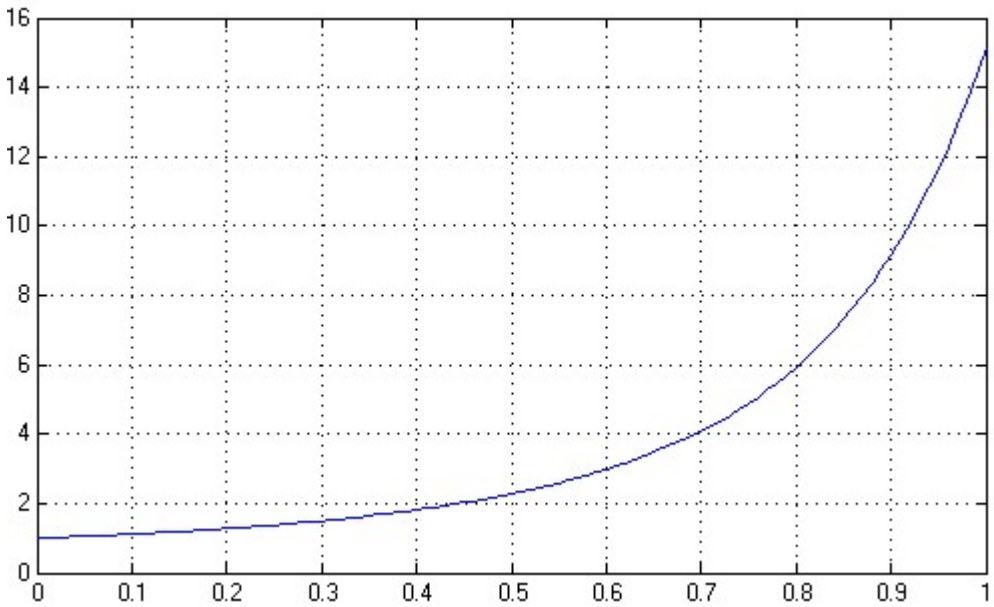


Рисунок 4.4 – Функція, що моделюється

Для нашої задачі (2 входи, 1 вихід) необхідно, щоб в першому шарі мережі було 2 нейрона і в останньому 1 (число нейронів в крайніх шарах

дорівнює кількості входів / виходів). Кількість середніх шарів і нейронів в них вибирається розробником мережі.

Створити та навчити нейронну мережу. Графік залежності помилки в мережі від епохи навчання, рисунок 4.5.

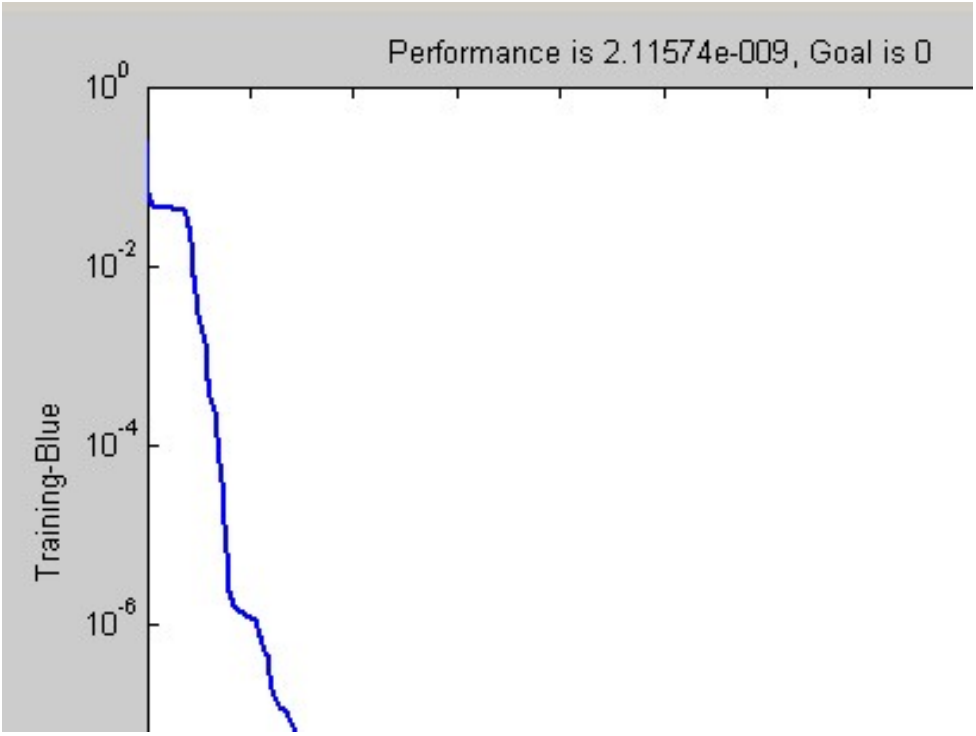


Рисунок 4.5 – Графік залежності помилки в мережі від епохи

Порівняти навчену нейромережеву модель з функцію двох змінних що моделюється.

Графіки, побудовані блоком Fcn і нейронною мережею представлені на рисунку 4.6 (похибка складає $\varepsilon = 0{,}0002219\%$):

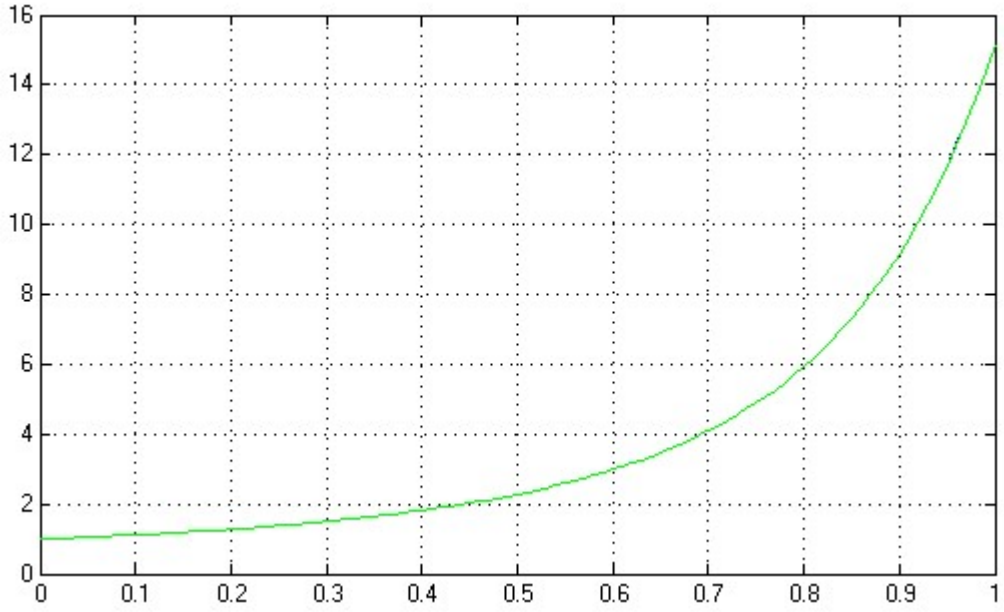


Рисунок 4.6 – Графіки функції двох змінних і нейронної мережі

Завдання: Написати програму, що реалізує нейронні мережі для моделювання функції двох змінних. Функцію двох змінних, типу $f(x+y) = x^2+y^2$, обрати самостійно. Промоделювати на невеликому відрізку, скажімо від 0 до 10.

Дослідити вплив кількості внутрішніх шарів та кількості нейронів на середню відносну помилку моделювання для різних типів мереж (feed forward backprop, cascade - forward backprop, elman backprop):

1. Тип мережі: feed forward backprop:
    a) 1 внутрішній шар з 10 нейронами;
    b) 1 внутрішній шар з 20 нейронами;
2. Тип мережі: cascade - forward backprop:
    a) 1 внутрішній шар з 20 нейронами;
    b) 2 внутрішніх шари по 10 нейронів у кожному;
3. Тип мережі: elman backprop:
    a) 1 внутрішній шар з 15 нейронами;
    b) 3 внутрішніх шари по 5 нейронів у кожному;
4. Зробити висновки на основі отриманих даних.

ЛАБОРАТОРНА РОБОТА №3

Нейронної мережі прямого розповсюдження для розпізнавання зображення

Завдання: Написати програму що реалізує нейронну мережу прямого розповсюдження для розпізнавання рукописних цифр.

У повнозв'язних гомогенних БНМПР виходи базових елементів кожного шару сполучені зі всіма входами всіх базових елементів наступного шару, а функція активації $f(s)$ приймається однаковою для всіх базових елементів нейронної мережі [42]. Структура багатошарових нейронних мереж прямого розповсюдження (БНМПР) зображена на Рис. 1.10.

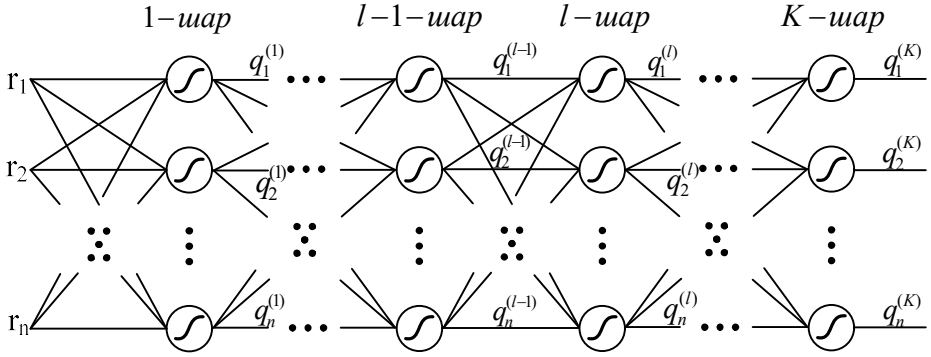


Рис. 1.10 – Структурна схема статичних БНМ

Проміжний l-й шар має n_i базових елементів. Зв'язки між базовими елементами в шарі відсутні. Виходи базових елементів l-го шару поступають на входи базових елементів тільки наступного (l+1)-го шару. Вихід i-го елементу в l-му шарі багатошарової нейромережі може бути визначений так само, як і для будь-якого базового елементу, у вигляді

$$q_i^{(l)} = f\left(\sum_{j=1}^{n_{l-1}} w_{i,j}^{(l)} q_1^{(l-1)} + w_{i,0}^{(l)} q_0^{(l-1)}\right) = f\left(s_i^{(l)}\right). \tag{1.5}$$

У векторній формі вихід l-го шару мережі дорівнює

$$\mathbf{q}_1^{(l)} = \mathbf{f}\left(\mathbf{W}_1^{(l)} \mathbf{q}_1^{(l-1)} + \mathbf{w}_0^{(l)} \mathbf{q}_0^{(l-1)}\right), \tag{1.6}$$

де $\mathbf{w}_0^{(l)} = col(w_{1,0}^{(l)},...,w_{n_l,0}^{(l)})$ – вектор вагових коефіцієнтів $\mathbf{q}_0^{(l-1)}$ у шарі l, який для всіх $l = \overline{1,K}$ приймається рівним +1; $\mathbf{q}_1^{(l)} = col(q_1^{(l)},...,q_{n_l}^{(l)})$, $\mathbf{q}_1^{(l-1)} = col(q_1^{(l-1)},...,q_{n_{l-1}}^{(l-1)})$ – вектори виходів базових процесорних елементів

шару l і виходів попереднього (l-1)-го шару, що поступають на входи базових елементів шару l. Для вихідного K-го шару маємо

$$\begin{aligned}\mathbf{q}_1^{(\mathrm{K})} &= \mathbf{W}_1^{(\mathrm{K})}\mathbf{q}_1^{(\mathrm{K}-1)} + \mathbf{w}_0^{(\mathrm{K})};\\ \mathbf{q}_1^{(\mathrm{K}-1)} &= \mathbf{f}\left(\mathbf{W}_1^{(\mathrm{K}-1)}\mathbf{q}_2^{(\mathrm{K}-1)} + \mathbf{w}_0^{(\mathrm{K}-1)}\right).\end{aligned} \tag{1.7}$$

Співвідношення (1.5) – (1.7) описують БНМПР з нелінійними функціями активації $f(s)$ пошарово. Функцію $s_i^{(l)}$ на i-х виходах суматорів базових елементів шару l називають дискримінантною. У загальному випадку – це відрізок багатовимірного ряду Тейлора, його найбільший ступінь визначає порядок базових елементів. Так, співвідношення (1.5) відповідає виходу елементу першого порядку, а вектор вагових коефіцієнтів відповідного ряду Тейлора $\mathbf{w}_0^{(l)} = col(w_{i,0}^{(l)},...,w_{i,n_{l-1}}^{(l)})$ є «пам'ять» базового процесорного елементу. Дискримінантна функція є зважена сума входів БНМПР з коефіцієнтами, рівними значенням «вагів» $w_{i,j}^{(l)}$. Дискримінантна функція розділяє вхідний вектор мінімум на два класи у разі лінійної мережі з функцією першого порядку.

БНМПР завдяки своїм властивостям можуть бути використані для побудови регулюючих та коректуючих пристроїв, еталонної, адаптивної, номінальної та інверсно-динамічної моделей об'єкта, на основі яких виконується спостереження та оцінка параметрів об'єкта, спостереження та оцінка величини діючих в системі збурень, пошук або обчислення оптимальної програми зміни керуючого впливу, ідентифікація об'єкта, прогнозування стану об'єкта, розпізнавання, класифікації, аналіз великої кількісті даних, що надходять з високою швидкістю з великої кількості пристроїв та датчиків. При виконанні цих функцій в сучасній техніці основними параметрами є швидкість обробки інформації, точність та кількість використаного ресурсу при апаратній реалізації. Тому підвищення ефективності роботи БНМПР при їх апаратній реалізації є актуальною задачею.

Корисне посилання: https://python-scripts.com/build-neural-network

ЛАБОРАТОРНА РОБОТА №4

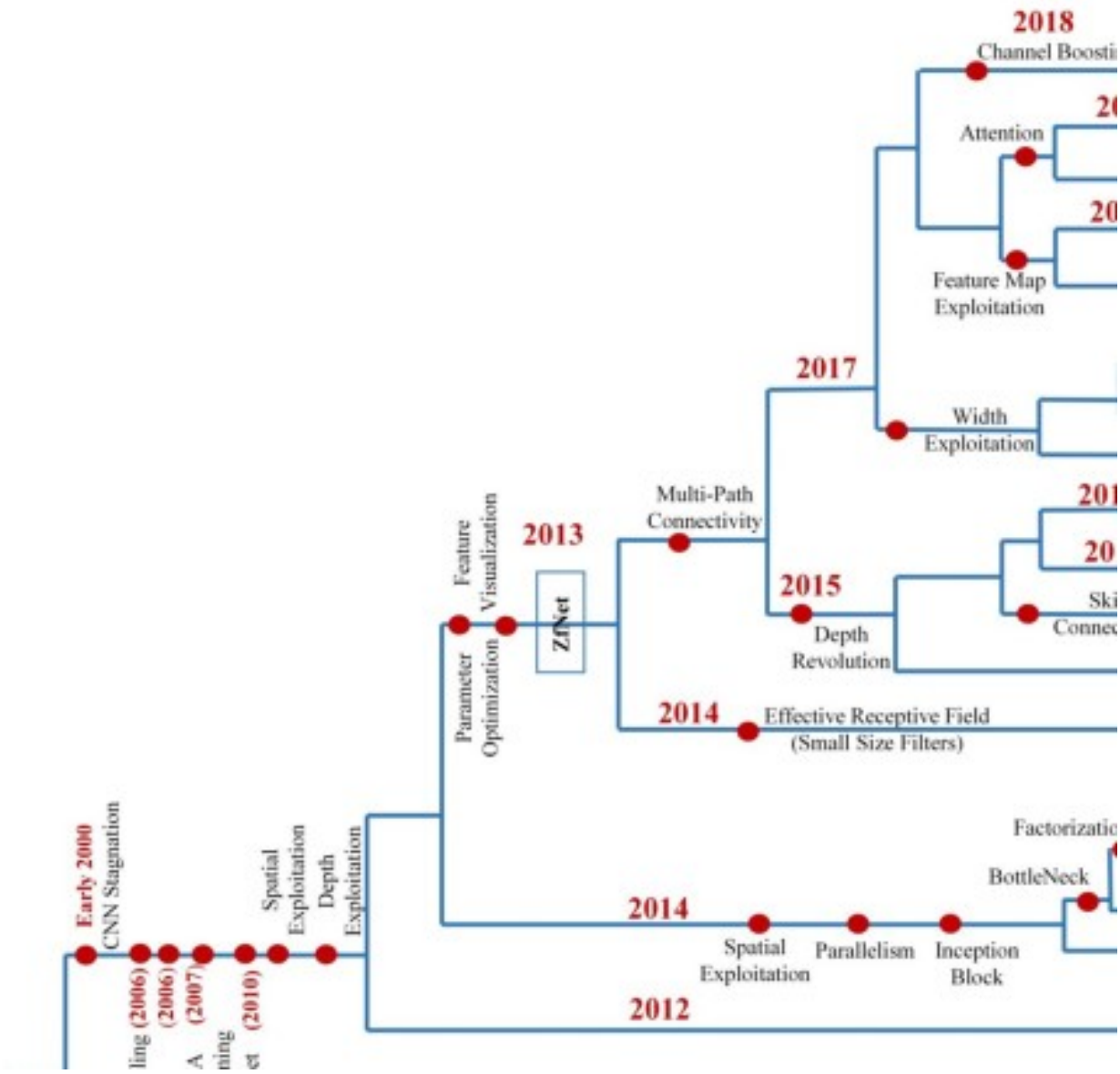
Згорткові нейронні мережі

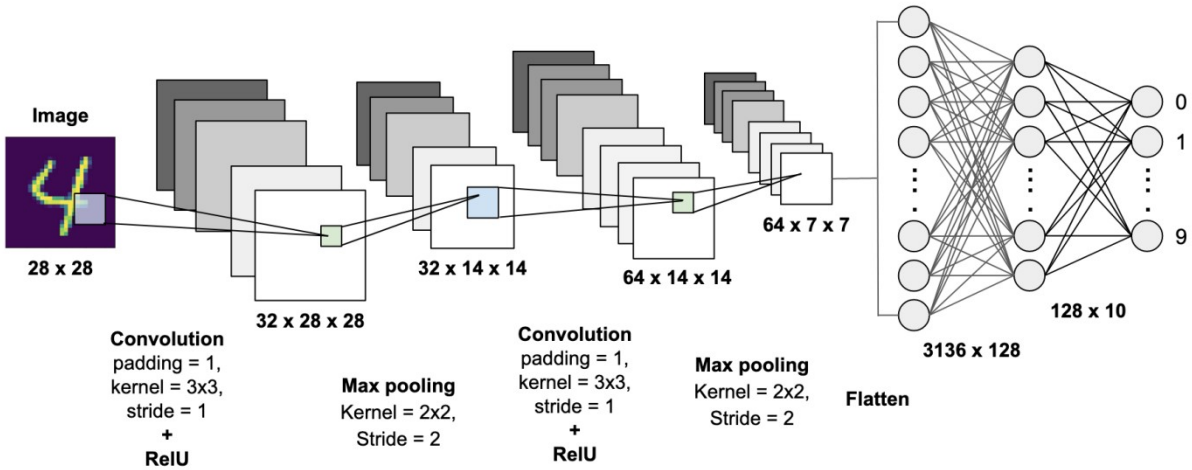
Завдання: Написати програму що реалізує згорткову нейронну мережу AlexNet для розпізнавання об’єктів з датасету ImageNet

Корисне посилання: https://towardsdatascience.com/implementing-alexnet-cnn-architecture-using-tensorflow-2-0-and-keras-2113e090ad98

Згорткова нейронна мережа (*CNN*) - спеціальна архітектура штучних нейронних мереж, запропонована Яном Лекуном в 1988 році і націлена на ефективне розпізнавання образів, входить до складу технологій глибокого навчання. Використовує деякі особливості зорової кори, в якій були відкриті так звані прості клітини, що реагують на прямі лінії під різними кутами, і складні клітини, реакція яких пов'язана з активацією певного набору простих клітин. Таким чином, ідея згорткових нейронних мереж полягає в чергуванні згорткових шарів і шарів підвибірки. Структура мережі - односпрямована (без зворотних зв'язків), принципово багатошарова. Для навчання використовуються стандартні методи, найчастіше метод зворотного поширення помилки. Функція активації нейронів (передавальна функція) - будь-яка, за вибором дослідника.

Назву архітектура мережі отримала через наявність операції згортки, Суть якої в тому, що кожен фрагмент зображення множиться на матрицю (ядро) згортки поелементно, а результат підсумовується і записується в аналогічну позицію вихідного зображення.





В згортковій нейронній мережі в *операції згортки* використовується лише обмежена матриця ваг невеликого розміру, яку «рухають» по всьому

оброблюваному шару (на самому початку - безпосередньо по вхідному зображенню), формуючи після кожного зсуву сигнал активації для нейрона наступного шару з аналогічною позицією. Тобто для різних нейронів вихідного шару використовуються одна і та ж матриця ваг, яку також називають *ядром згортки*. Її інтерпретують як графічне кодування якої-небудь ознаки, наприклад, наявність похилої лінії під певним кутом. Тоді наступний шар, що вийшов в результаті операції згортки такою матрицею ваг, показує наявність даної ознаки в оброблюваному шарі і її координати, формуючи так звану карту ознак (англ. *feature map*). Природно, в згортковій нейронній мережі набір ваг не один, а ціла гама, що кодує елементи зображення (наприклад лінії і дуги під різними кутами). При цьому такі ядра згортки не закладаються дослідником заздалегідь, а формуються самостійно шляхом навчання мережі класичним методом зворотного поширення помилки. Прохід кожним набором ваг формує свій власний примірник карти ознак, роблячи нейронну мережу багатоканальної (багато незалежних карт ознак на одному шарі). Також слід зазначити, що при переборі шару матрицею ваг її пересувають зазвичай не на повний крок (розмір цієї матриці), а на невелику відстань. Так, наприклад, при розмірності матриці ваг 5 × 5 її зрушують на один або два нейрона (пікселя) замість п'яти, щоб не «переступити» шуканий ознака.

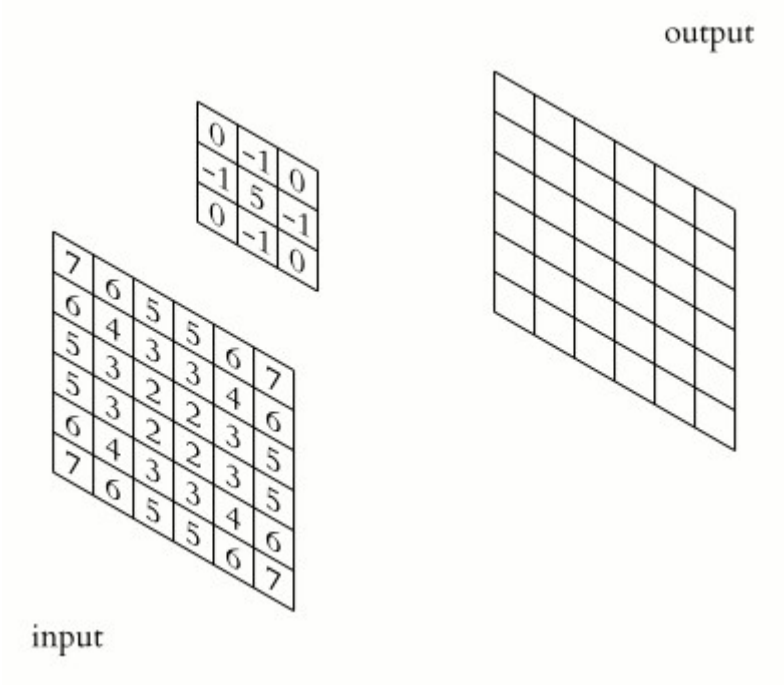
Операція підвибірки (англ. *pooling*), виконує зменшення розмірності сформованих карт ознак. У даній архітектурі мережі вважається, що інформація про факт наявності шуканої ознаки важливіше точного знання його координат, тому з кількох сусідніх нейронів карти ознак вибирається максимальний і приймається за один нейрон ущільненої карти ознак меншої розмірності. За рахунок цієї операції, крім прискорення подальших обчислень, мережа стає більш інваріантної до масштабу вхідного зображення.

Після початкового шару (вхідного зображення) сигнал проходить серію згорткових шарів, в яких чергується власне згортка і підвибірка. Чергування

шарів дозволяє складати «карти ознак» з карт ознак, на кожному наступному шарі карта зменшується в розмірі, але збільшується кількість каналів. На практиці це означає здатність розпізнавання складних ієрархій ознак. Зазвичай після проходження декількох шарів карта ознак вироджується в вектор або навіть скаляр, але таких карт ознак стають сотні. На виході згорткових шарів мережі встановлюють кілька шарів повнозвязної нейронної мережі, на вхід якого подаються кінцеві карти ознак.

Двовимірна згортка (2D convolution) - це досить проста операція: починаємо з ядра, що представляє із себе матрицю ваг. Ядро "ковзає" над двовимірним зображенням, поелементно виконуючи операцію множення з тією частиною вхідних даних, над якою воно зараз знаходиться, і потім підсумовує всі отримані значення в один вихідний піксель.

Ядро повторює цю процедуру з кожної локацією, над якою воно "ковзає", перетворюючи двовимірну матрицю в іншу все ще двовимірну матрицю ознак. Ознаки на виході є зваженими сумами (де ваги є значеннями самого ядра) ознак на вході, розташованих приблизно в тому ж місці, що і вихідний піксель на вхідному шарі.

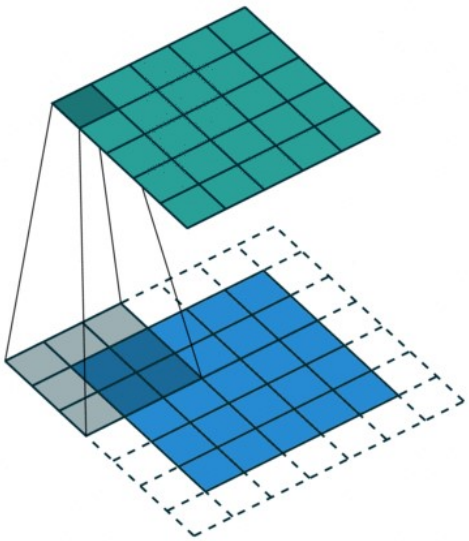


Незалежно від того, чи потрапляє вхідна ознака в "приблизно те ж місце", вона визначається в залежності від того, знаходиться вона в зоні ядра, що створює вихідні дані, чи ні. Це означає що розмір ядра згорткової нейронної мережі визначає кількість ознак, які будуть об'єднані для отримання нової ознаки на виході.

У прикладі, наведеному на рисунку попереднього слайду, ми маємо 6 * 6 = 36 ознак на вході і 3 * 3 = 9 ознак на виході. Для стандартного шару ми б мали вагову матрицю 36 * 9 = 324 параметрів, а кожна вихідна ознака була би зваженою сумою всіх ознак на вході. Згортка дозволяє зробити таку операцію з усього 9-ю параметрами, адже кожна ознака на виході виходить аналізом не кожної ознаки на вході, а тільки одного вхідного, що знаходиться в "приблизно тому ж місці".

Зверніть увагу на те, що в процесі ковзання краю по суті обрізаються, перетворюючи матрицю ознак розміром 5 * 5 в матрицю 3 * 3. Крайні пікселі ніколи не виявляються в центрі ядра, тому що тоді ядру немає над чим буде ковзати за краєм. Це зовсім не ідеальний варіант, так як ми хочемо, щоб розмір на виході дорівнював вхідному.

Padding додає до країв підроблені пікселі (зазвичай нульового значення, внаслідок цього до них застосовується термін "нульовий додаток" - "zero padding"). Таким чином, ядро при ковзанні дозволяє непідробним пикселям надаватися в своєму центрі, а потім поширюється на підроблені пікселі за межами краю, створюючи вихідну матрицю того ж розміру, що і вхідні.



Часто буває, що при роботі з згортковим шаром, потрібно отримати вихідні дані меншого розміру, ніж вхідні. Це зазвичай необхідно в згорткових нейронних мережах, де розмір просторових розмірів зменшується при збільшенні кількості каналів. Один із способів досягнення цього – використання шарів підвибірки. Наприклад, приймати середнє / максимальне значення кожної гілки розміром 2*2, щоб зменшити всі просторові розміри в два рази. Ще один спосіб домогтися цього – використовувати stride (Крок).

Ідея stride полягає в тому, щоб пропустити деякі області, над якими ковзає ядро. Крок 1 означає, що беруться прольоти через піксель, тобто за фактом кожен проліт є стандартною згорткою. Крок 2 означає, що прольоти відбуваються через кожні два пікселя, пропускаючи всі інші прольоти в процесі і зменшуючи їх кількість приблизно в 2 рази, крок 3 означає пропуск 3-х пікселів, скорочуючи кількість в 3 рази і т.д.

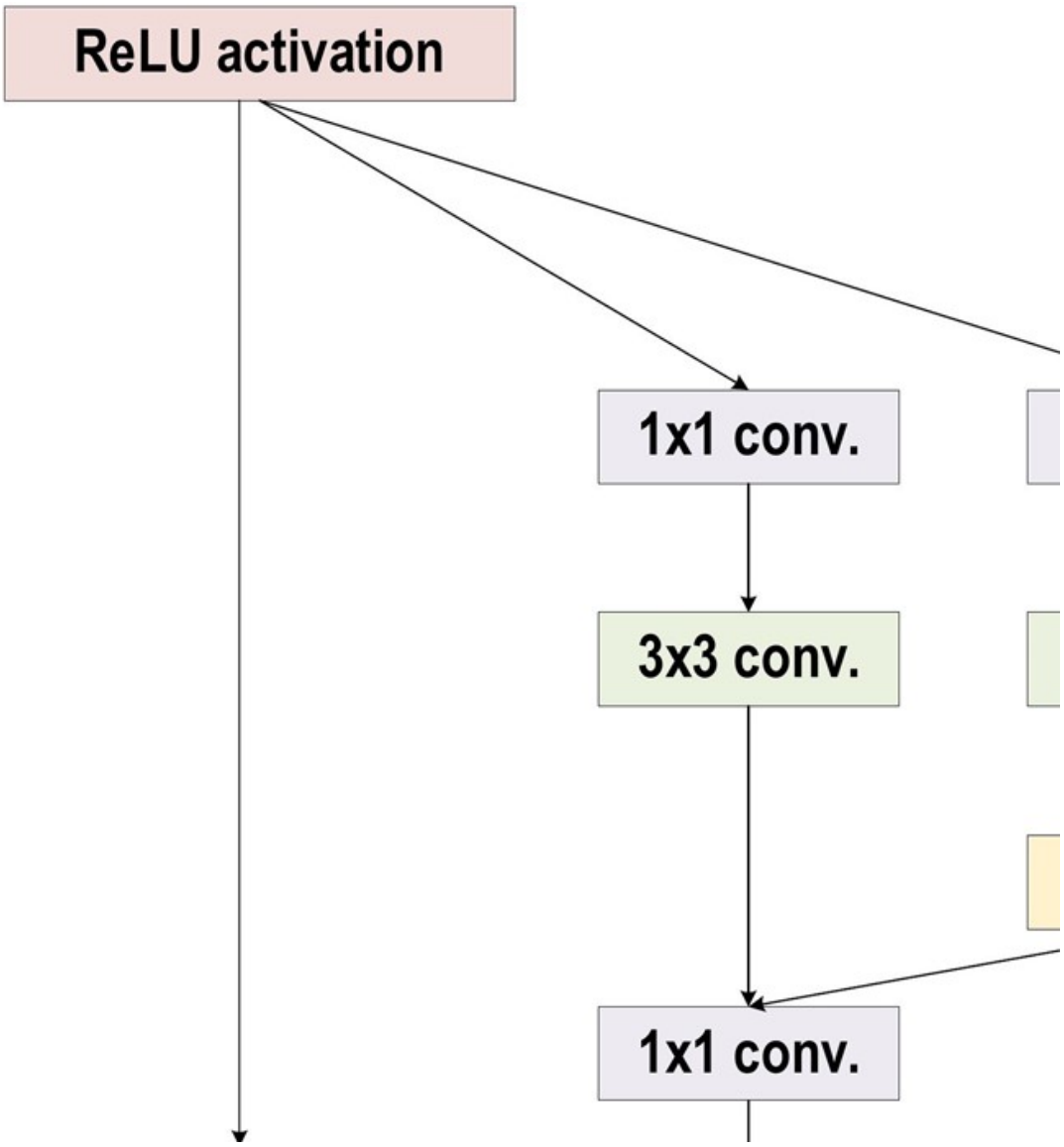
Більш сучасні мережі, такі як архітектури ResNet, повністю відмовляються від шарів підвибірки у внутрішніх шарах на користь згорток що чергуються, коли необхідно зменшити розмір на виході.

ЛАБОРАТОРНА РОБОТА №5

Згорткові нейронні мережі типу Inception

Завдання: Написати програму що реалізує згорткову нейронну мережу Inception V3 для розпізнавання об’єктів на зображеннях. Створити власний дата сет з папки на диску, навчити нейронну мережу на цьому датасеті розпізнавати породу Вашої улюбленої собаки чи кота. Навчену нейронну мережу зберегти на комп’ютер написати програму, що відкриває та аналізує зображення.

Szegedy et al. запропонував Inception-ResNet і Inception-V3/4 як оновлені типи Inception-V1/2. Концепція Inception-V3 полягала в тому, щоб мінімізувати обчислювальні витрати без впливу на більш глибоке узагальнення мережі. Так, Szegedy et al. використовували асиметричні фільтри малого розміру (1 × 5 і 1 × 7), а не фільтри великого розміру (7 × 7 і 5 × 5); більше того, вони використовували вузьке місце в 1 × 1 згортку до великого розміру фільтрів. Ці зміни роблять роботу традиційної згортки дуже схожою на міжканальну кореляцію. Раніше Lin et al. використовував потенціал фільтра 1×1 в архітектурі NIN. Згодом використав ту ж ідею розумним чином. Використовуючи згорткову операцію 1 × 1 в Inception-V3, вхідні дані відображаються в три або чотири ізольовані простори, які менші, ніж початкові вхідні простори. Далі всі ці кореляції відображаються в цих менших просторах через загальні згортки 5 × 5 або 3 × 3. На відміну від цього, в Inception-ResNet Szegedy об’єднує початковий блок і залишкову потужність навчання, замінюючи конкатенацію фільтра залишковим з’єднанням. Szegedy та ін. емпірично продемонстровано, що Inception-ResNet (Inception-4 із залишковими з’єднаннями) може досягти такої ж потужності узагальнення, як і Inception-V4 із збільшеною шириною та глибиною та без залишкових з’єднань. Таким чином, чітко показано, що використання залишкових з’єднань у навчанні значно прискорить навчання мережі Inception. На рисунку показана основна блок-схема початкової залишкової одиниці.

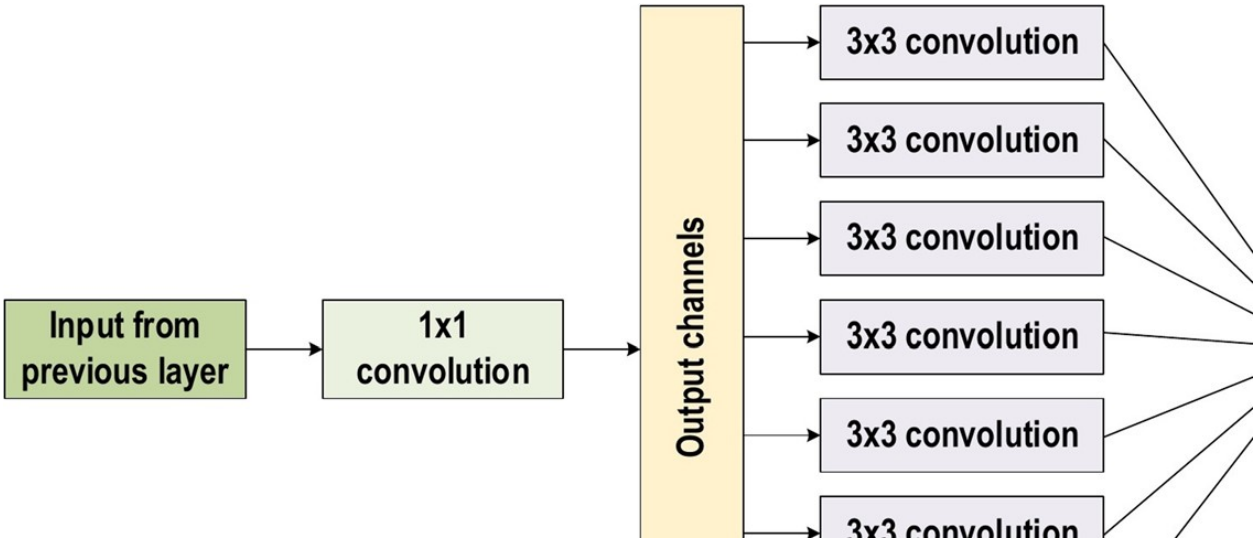

ReLU activation
1x1 conv.
3x3 conv.
1x1 conv.

ЛАБОРАТОРНА РОБОТА №6

Згорткові нейронні мережі типу Xception

Завдання: Написати програму що реалізує згорткову нейронну мережу Xception для розпізнавання об’єктів на відео. Створити власний дата сет з папки на диску, навчити нейронну мережу на цьому датасеті розпізнавати логотип вашого улюбленого бренду, скажімо Apple чи BMW. Навчену нейронну мережу зберегти на комп’ютер написати програму, що відкриває та аналізує відео, результат – час на якому з’являвся логотип.

Екстремальна початкова архітектура є основною характеристикою Xception. Основною ідеєю Xception є її глибинна роздільної згортки. Модель Xception скоригувала початковий блок, зробивши його ширшим і змінивши один вимір (3 × 3), а потім згортку 1 × 1, щоб зменшити складність обчислень. На рисунку показано архітектуру блоку Xception.



Мережа Xception стає додатково ефективною в обчислювальному відношенні завдяки використанню каналу розв’язки та просторової відповідності. Більше того, вона спочатку виконує відображення згорнутого виходу на короткий вимір вкладення, застосовуючи згортки 1 × 1. Потім вона виконує k просторових перетворень. Зауважте, що тут k представляє потужність, що визначає ширину, яка отримується за допомогою числа перетворень у Xception. Однак обчислення були спрощені в Xception завдяки чіткому згортанню кожного каналу навколо просторових осей. Ці осі згодом

використовуються як згортки 1 × 1 (точкова згортка) для виконання міжканальної відповідності. Згортка 1 × 1 використовується в Xception для упорядкування глибини каналу. Традиційна згортка операція в Xception використовує ряд сегментів перетворення, еквівалентних кількості каналів; Крім того, Inception використовує три сегменти трансформації, тоді як традиційна архітектура CNN використовує лише один сегмент трансформації. І навпаки, запропонований підхід трансформації Xception забезпечує додаткову ефективність навчання та кращу продуктивність, але не мінімізує кількість параметрів.

ЛАБОРАТОРНА РОБОТА №7

Рекурентні нейронні мережі LSTM

Завдання: Написати програму, що реалізує рекурентну нейронну мережу LSTM для розпізнавання емоційного забарвлення тексту, використати датасет Yelp Dataset

Корисне посилання:

Датасети: https://www.tensorflow.org/datasets/catalog/overview

https://keras.io/api/layers/recurrent_layers/

https://www.tensorflow.org/text/tutorials/text_classification_rnn

Довга короткочасна пам'ять (LSTM) змінила як машинне навчання, так і нейрокомп'ютери. Ця модель покращила розпізнавання мовлення Google, значно покращила машинний переклад у Google Translate та відповіді Alexa від Amazon. Цю нейронну систему також використовує Facebook, і станом на 2017 рік вона досягла понад 4 мільярдів перекладів на основі LSTM на день. Цікаво, що рекурентні нейронні мережі демонстрували досить дискретну продуктивність, поки не з'явились LSTM. Одна з причин успіху цієї рекурентної мережі полягає в її здатності впоратися з проблемою градієнта, що вибухає/зникає, що є складною проблемою, яку потрібно обійти під час навчання рекурентних або дуже глибоких нейронних мереж. У найблищих лекціях ми представляємо вичерпний огляд, який охоплює формулювання та навчання LSTM, застосування мереж LSTM, та код, що реалізує цю модель.

Рекурентні або дуже глибокі нейронні мережі важко тренувати, оскільки вони часто страждають від проблеми градієнта, що вибухає/зникає. Щоб подолати цей недолік під час вивчення довгострокових залежностей, була введена архітектура LSTM. Здатність до навчання LSTM вплинула на кілька областей як з практичної, так і теоретичної точки зору, так що вона стала найсучаснішою моделлю. Це призвело до того, що модель була використана Google для розпізнавання мовлення та покращення машинного перекладу в Google Translate. Amazon використовує цю модель для

покращення функціональних можливостей Alexa, а Facebook використовує її для понад 4 мільярдів перекладів на основі LSTM на день станом на 2017 рік.

Завдяки своїй високій застосовності та популярності ця нейронна архітектура також знайшла свій шлях у світ ігор. Наприклад, Google Deepmind створив AlphaStar, штучний інтелект, призначений для гри в Starcraft II. Протягом розробки AlphaStar вона почала освоювати гру, піднімаючись у світовому рейтингу, чого раніше не було. Дослідження в цій галузі, звичайно, не обмежуються Starcraft II, оскільки дослідницький інтерес охоплює весь ігровий жанр RTS через його складність. Щоб узагальнити тему навчання з підкріпленням в інших умовах, OpenAI вдалося створити руку робота під назвою Dactyl, яка сама навчилася маніпулювати об'єктами, як у людини.

Звичайно, нейронна архітектура не була б так добре прийнята на практиці без сильної теоретичної основи. Великий огляд кількох варіантів LSTM та їхніх характеристик щодо так званої ванільної моделі нещодавно провів Грефф. Vanilla LSTM інтерпретується як оригінальний блок LSTM з додаванням з'єднань вентилю забуття та комірки. Всього для експериментів було визначено вісім варіантів. Архітектура Vanilla LSTM добре справляється з низкою завдань, і жоден з восьми досліджених варіантів істотно не перевершує інші. Це виправдовує більшість застосувань, знайдених у літературі, для використання Vanilla LSTM .

Нещодавнє дослідження, надає огляд комірки LSTM, її функціональних можливостей та детальний огляд різних архітектур рекурентних мереж. Розрізняють нейронні мережі з домінуванням LSTM та інтегровані мережі LSTM, остання з яких додає інші компоненти, ніж LSTM, щоб скористатися перевагами його властивостей, отже, потенційно гібридизуючи нейронні мережі.

Тому в цій лекції ми представляємо вичерпний огляд моделі LSTM, який доповнює теоретичні висновки, що є в наданій статті. Наші три лекції

зосереджено на трьох основних напрямках, які переходять від теорії до практики.

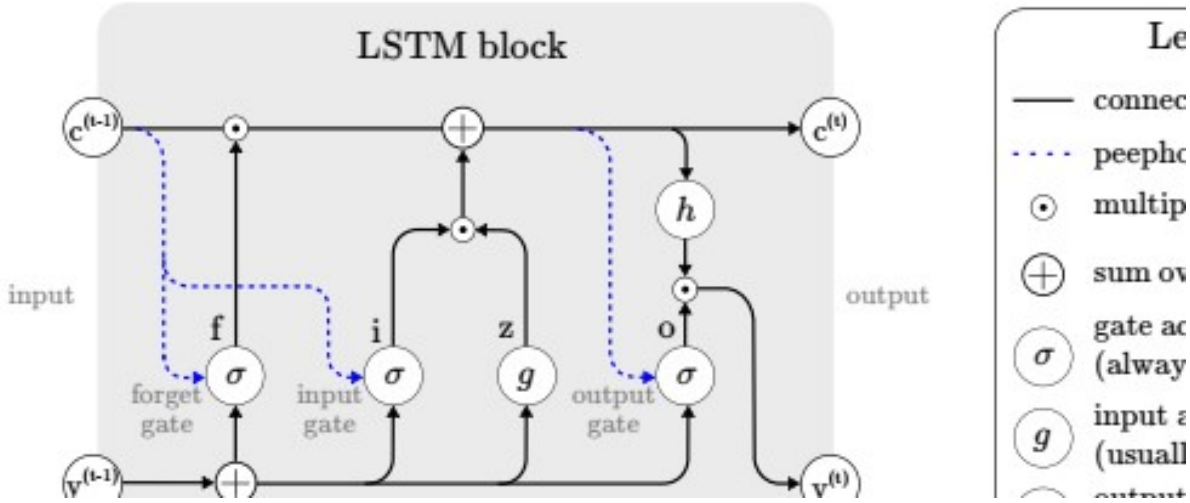
У першій лекції ми широко опишемо компоненти LSTM, як вони взаємодіють один з одним і як ми можемо оцінити параметри, які можна вивчати. Це корисно для студентів, які хочуть оволодіти моделлю з теоретичної точки зору, а не тільки бути досвідченими практиками. У другій лекції ми окреслимо цікаві додатки, які показують потенціал LSTM як беззаперечного сучасного методу в області глибокого навчання. Деякі цікаві домени додатків включають розпізнавання тексту, прогнозування часових рядів, обробку природної мови, комп'ютерний зір, а також субтитри для зображень і відео. В останній частині ми представляємо приклад коду в Tensorflow.

Рекурентні нейронні мережі LSTM мають велику популярність серед дослідників на даний час. Розробки по LSTM представлені на таких основних конференціях, як "*Конференція з нейронних систем обробки інформації"*, "*Конференція з комп'ютерного зору та розпізнавання образів*", "*Конференція AAAI зі штучного інтелекту*", "*Конференція з емпіричних методів у природній мові"* тощо. Грубий пошук, проведений у лютому 2020 року з використанням згаданих вище критеріїв пошуку, показав 11 931 документ, проіндексований Scopus.

Модель LSTM — це потужна рекурентна нейронна система, спеціально розроблена для подолання проблем з градієнтом, що вибухає/зникає, які зазвичай виникають при вивченні довготривалих залежностей, навіть якщо мінімальні затримки дуже тривалі. Загалом, цьому можна запобігти, використовуючи карусель з постійними помилками (CEC), яка підтримує сигнал помилки в кожній комірці. Насправді, такі комірки самі по собі є рекурентними мережами з цікавою архітектурою, оскільки CEC розширюється додатковими функціями, а саме вхідним і вихідним вентилем, утворюючи комірку пам'яті. Самоповторювані з'єднання вказують на зворотний зв'язок із запізненням на один часовий крок.

Базовий блок LSTM складається з комірки, вхідного вентиля, вихідного вентиля та вентиля забуття. Цей вентиль забуття спочатку не був частиною мережі LSTM, але був запропонований Герсом, щоб дозволити мережі скинути свій стан. Комірка запам'ятовує значення через довільні інтервали часу, а три вентилі регулюють потік інформації, пов'язаної з коміркою. Далі розглянемо версію рекурентої нейронної мережі LSTM, яка називається Vanilla, оскільки це найпопулярніша архітектура LSTM. Однак це не означає, що він також є кращим у кожній ситуації.

Архітектура LSTM складається з набору періодично підключених підмереж, відомих як блоки пам’яті. Ідея блоку пам’яті полягає в тому, щоб підтримувати свій стан з плином часу та регулювати потік інформації через нелінійні блоки стробування. На рис. показано архітектуру звичайного блоку LSTM, який включає вентилі, вхідний сигнал $\mathbf{x}^{(t)}$, вихідний $\mathbf{y}^{(t)}$, функції активації та з'єднання комірки. Вихід блоку періодично підключається до входу блоку та всіх вентилів.



Щоб прояснити, як працює модель LSTM, припустимо мережу, що складається з N блоків обробки та M входів. Нижче описано перехід вперед у цій повторюваній нейронній системі.

Вхідний блок. Цей крок присвячений оновленню вхідного компонента блоку, який поєднує поточний вхід $\boldsymbol{x}^{(t)}$ і вихід цього блоку LSTM $\mathbf{y}^{(t-1)}$ на останній ітерації. Це можна зробити, як показано нижче:

$$z^{(t)} = g(W_z x^{(t)} + R_z y^{(t-1)} + b_z)$$

де $\boldsymbol{W_z}$ і $\boldsymbol{R_z}$ — ваги, пов'язані з $\boldsymbol{x^{(t)}}$ і $\mathbf{y}^{(t-1)}$ відповідно, а $\boldsymbol{b_z}$ означає вектор ваги зміщення.

Вхідний вентиль. На цьому кроці ми оновлюємо вхідний вентиль, який поєднує поточний вхід $\boldsymbol{x^{(t)}}$, вихід цієї одиниці LSTM $\mathbf{y}^{(t-1)}$ і значення комірки $\boldsymbol{c^{(t-1)}}$ на останній ітерації. Наступне рівняння показує цю процедуру:

$$i^{(t)} = \sigma(W_i x^{(t)} + R_i y^{(t-1)} + p_i \odot c^{(t-1)} + b_i) \quad (2)$$

де коло з точкою в центрі позначає поточкове множення двох векторів, $\boldsymbol{W_i}$, $\boldsymbol{R_i}$ і $\boldsymbol{p_i}$ - це ваги, пов'язані з $\boldsymbol{x^{(t)}}$, $\mathbf{y}^{(t-1)}$ і $\boldsymbol{c^{(t-1)}}$, відповідно, тоді як $\boldsymbol{b_i}$ представляє вектор зміщення, пов'язаний з цим компонентом. На попередніх кроках рівень LSTM визначає, яка інформація повинна зберігатися в станах комірок мережі $\boldsymbol{c^{(t)}}$. Це включало вибір значень-кандидатів $\boldsymbol{z^{(t)}}$, які потенційно можуть бути додані до станів комірки, і значень активації $\boldsymbol{i^{(t)}}$ вхідних вентилів.

Вентиль забування. На цьому кроці блок LSTM визначає, яку інформацію слід видалити з попередніх станів комірки $\boldsymbol{c^{(t-1)}}$. Таким чином, значення активації $\boldsymbol{f^{(t)}}$ елементів забуття на етапі часу t обчислюються на основі поточного входу $\boldsymbol{x^{(t)}}$, виходів $\mathbf{y}^{(t-1)}$ і стану $\boldsymbol{c^{(t-1)}}$ осередків пам'яті. на попередньому кроці часу (t - 1), з'єднання вічка та умови зміщення $\boldsymbol{b_f}$ воріт забуття. Це можна зробити наступним чином:

$$f^{(t)} = \sigma(W_f x^{(t)} + R_f y^{(t-1)} + p_f \odot c^{(t-1)} + b_f) \quad (3)$$

де $\boldsymbol{W_f}$, $\boldsymbol{R_f}$ та $\boldsymbol{p_f}$ — ваги, пов'язані з $\boldsymbol{x^{(t)}}$, $\mathbf{y}^{(t-1)}$ і $\boldsymbol{c^{(t-1)}}$, відповідно, тоді як $\boldsymbol{b_f}$ означає вектор ваги зміщення.

Комірка. На цьому етапі обчислюється значення комірки, яке поєднує значення вхідного блоку $\boldsymbol{z^{(t)}}$, вхідного вентиля $\boldsymbol{i^{(t)}}$ і елемента забуття $\boldsymbol{f^{(t)}}$ з попереднім значенням комірки. Це можна зробити, як показано нижче:

$$c^{(t)} = z^{(t)} \odot i^{(t)} + c^{(t-1)} \odot f^{(t)}. \quad (4)$$

Вихідний вентель. На цьому етапі обчислюється вихідний вентиль, який поєднує поточний вхід $\boldsymbol{x}^{(t)}$, вихід цієї одиниці LSTM $\mathbf{y}^{(t-1)}$ і значення комірки c(t-1) на останній ітерації. Це можна зробити, як показано нижче:

$$o^{(t)} = \sigma(W_o x^{(t)} + R_o y^{(t-1)} + p_o \odot c^{(t)} + b_o) \quad (5)$$

де W_o, R_o і p_o – ваги, пов'язані з $\boldsymbol{x}^{(t)}$, $\mathbf{y}^{(t-1)}$ і $\boldsymbol{c}^{(t-1)}$, відповідно, тоді як b_o означає вектор ваги зміщення.

Вихідний блок. Нарешті, ми обчислюємо вихід блоку, який поєднує поточне значення комірки c(t) із поточним значенням вихідного отвору наступним чином:

$$y^{(t)} = g(c^{(t)}) \odot o^{(t)}.$$

На наведених вище кроках σ, g і h позначають поточкові нелінійні функції активації. Логістичний сигмоїд $\sigma(x) = 1/1+e^{1-x}$ використовується як функція активації вентиля, тоді як гіперболічний тангенс g(x) = h(x) = tanh(x) часто використовується як вхід і вихід блоку функція активації.

Незважаючи на те, що vanilla LSTM вже працює дуже добре, кілька робіт вивчають можливості покращення продуктивності. Наприклад, Су і Куо розробили модель Extended LSTM, яка ще більше підвищила точність прогнозів у кількох областях застосування за рахунок розширення пам'яті. Це показує, що теоретичні вдосконалення все ще можуть бути внесені до вже найсучаснішої архітектури виконання. У роботі Bayera вже тривали пошуки удосконалень моделі. Автори шукали архітектурну альтернативу LSTM, щоб оптимізувати можливості навчання послідовності. Їм вдалося розвинути структури осередків пам'яті, здатні вивчати контекстно-залежні формальні мови за допомогою градієнтного спуску, що багато в чому порівнянно з LSTM за продуктивністю. Автори в роботі побудували на основі повторюваних мереж хвилюючих нейронів, розробивши нейронні мережі довгострокової пам'яті (LSNN), включаючи адаптаційні нейрони. Під час

тестів, в яких розмір LSNN був порівнянний з розміром LSTM, було показано, що продуктивність дуже порівнянна з LSTM. Це ще одна ілюстрація того, наскільки точним є і залишається LSTM.

ЛАБОРАТОРНА РОБОТА №8

Нейронні мережі CNN-bi-LSTM для розпізнавання звуку

Завдання: Написати програму, що реалізує нейронну мережу типу CNN-bi-LSTM для розпізнавання мови в текст. Використати датасет librispeech: https://www.tensorflow.org/datasets/catalog/librispeech

Корисне посилання:

https://towardsdatascience.com/audio-deep-learning-made-simple-automatic-speech-recognition-asr-how-it-works-716cfce4c706

Коментар до лабораторних:

https://www.tensorflow.org/install

https://keras.io/getting_started/intro_to_keras_for_engineers/

Нейронну мережу реалізувати з нуля за допомогою бібліотек, брати готову не можна.

Вкрай рекомендується вчити на GPU. Якщо немає CUDA-сумісної відеокарти, використовуйте Google Colaboratory.