МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики
(повна назва факультету)
Кафедра дискретного аналізу та інтелектуальних систем
(повна назва кафедри)

ДИПЛОМНА РОБОТА

Розпізнавання тексту з картинки за допомогою нейронних мереж

Виконав: студент групи ПМі-45

Спеціальності 122 - Комп'ютерні науки
(шифр і назва спеціальності)

Віпшовський Ю.А.
(підпис) (прізвище та ініціали)

Керівник: Притула М.М.
(підпис) (прізвище та ініціали)

Рецензент:
(підпис) (прізвище та ініціали)

2023

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

1. CPU - Central Processing Unit - Центральний процесор
2. GPU - Graphics Processing Unit - Графічний процесор
3. TPU - Tensor Processing Unit - Тензорний процесор
4. НМ - нейронна мережа
5. CNN – Convolution Neural Network – згорткова нейронна мережа
6. RNN – Recurrent Neural Network – рекурентна нейронна мережа
7. LSTM – Long Short-Term Memory – довга короткочасна пам’ять
8. CRNN - Convolutional recurrent neural network – Згорткова рекурентна нейронна мережа
9. Дата сет – набір тренувальних даних, для навчання нейронної мережі

ВСТУП

За останнє десятиліття людство дуже сильно розвинуло технології штучного інтелекту. Зараз штучний інтелект все більше і більше залучається у різні аспекти нашого життя. Його використовують для того, щоб досягнення найрізноманітніших цілей: аналіз даних, прогнозування курсів, розпізнавання образів та багато іншого.

Великі компанії уже давно інтегрують у різні свої сервіси технології штучного інтелекту, які допомагають користувачам у роботі з програмами, або значно спрощують різноманітну рутинну роботу. Також різні фірми використовують технології штучного інтелекту для збільшення ефективності роботи своїх робочих. Наприклад у нас є робітник, який перевіряє якість певного товару, нехай це будуть якісь плати, на які припаюють різноманітні чіпи та інші електронні компоненти. Раніше для нашого робітника потрібна була лупа і багато часу, щоб акуратно перевірити сотні плат на день. Коли фірма додала йому на допомогу систему на базі штучного інтелекту. Ця система аналізує фотографію плати, і якщо припій для системи виглядає якось не стандартно, вона передає плату на аналіз робітнику і він уже вирішує що з платою робити. Тепер робітник повинен перевірити у день значну меншу кількість плат, у результаті його продуктивність значно збільшилась, і його завантаженість зменшується.

Аналогічну систему можна створити для розпізнавання, наприклад номерних знаків авто, для автоматичної фіксації правопорушень. Така система облегшить роботу поліцейських при оформленні різноманітних протоколів.

У цій роботі описана мережа для розв’язання задачі розпізнавання коротких текстів із картинок.

1 ПОСТАНОВКА ЗАДАЧІ

У моїй роботі розроблена система для розв'язання задачі розпізнавання англійського тексту із фотографій.

Задача розпізнавання тексту полягає у тому щоб маючи картинку із текстом, виділити і розпізнати букви цього тексту, та отримати на виході у текстовому вигляді слово що було на зображенні. При розв’язанні цієї задачі система виділяє характерні ознаки букв на зображенні слова й на основі цих ознак виділяє окремі букви слова, а потім об’єднує їх у слово.

При розпізнаванні тексту, система розпізнає окремі букви цього тексту і повертає їх у порядку розпізнання. У результаті користувач отримує слово у текстовому вигляді.

2 ТЕОРЕТИЧНА ЧАСТИНА

2.1 Система штучного інтелекту

З розвитком електронно обчислювальних машин учені почали ставити собі питання чи можливо наділити ці машини людським інтелектом? У першу чергу це було необхідно для розв’язання різних класів інтелектуальних задач:

- розпізнавання образів
- логічне мислення;
- аналіз ситуації
- розуміння нової інформації
- навчання і самонавчання
- планування цілеспрямованих дій

Людина у процесі еволюції розвинула власний інтелект і розв’язування задач такого типу для нас не створює значних труднощів. Розпізнавати образи можуть малі діти, а надпотужний комп’ютер, що розв’язує величезні системи рівнянь за долі секунди – ні.

Саме тому учені ще у минулому столітті докладали значних зусиль, щоб розв'язати цю проблему. Прогрес у цьому ділі стався з появою математичної моделі штучного нейрона запропонованою Ворреном Маккалохом та Волтером Піттсом. Які показали що їх модель здатна виконувати логічні та числові операції. Розвинувши цю модель і побудувавши із неї різноманітні нейронні мережі, учені змогли навчити комп’ютер розв’язувати інтелектуальні задачі.

2.2 Математичний нейрон

Основою будь-яких НМ є нейрон – як одиниця обробки нейронної інформації. Математичний нейрон – це математична модель яка описує біологічний нейрон, функція такого нейрона повернути на вихід певний сигнал у залежності від вхідних сигналів.[1]

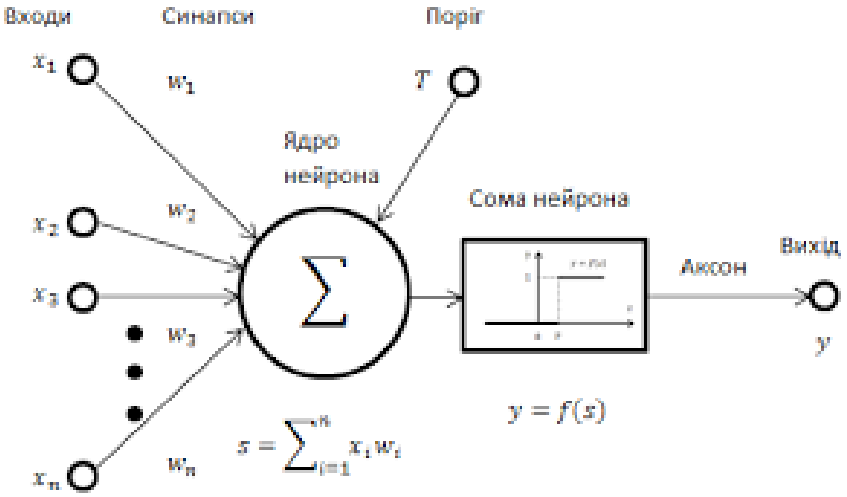


Рисунок 2.2 -1 Математична модель нейрона

На рисунку 2.2-1 показала модель такого нейрона. Він складається з 6 частин:

- Входи – набір вхідних сигналів до нейрона - x_i
- Синапси – вага або сила даного вхідного сигналу. У математичній моделі це коефіцієнт та який множиться на вхідний сигнал - w_i
- Ядро нейрона – суматор, обчислює суму з усіх входів після дії на них синоптичних ваг синапсів -

$$s = \sum_{i=1}^{n} x_i * w_i$$

- Поріг – це величина яка характеризує збільшення, або зменшення рівня сигналу що подається з суматора до функції активації. - T
- Сома нейрона, або функція активації – функція яка обчислює сигнал який подається на вихід нейрона – y = f(s+T)
- Вихід – вихідний сигнал із нейрона, результат його роботи.

2.2.1 Функція активації

Вибір функції активації для нейрона – це найголовніше у його проєктуванні. Головна задача будь-якої функції активації – обмежити мінімальне і максимальне значення на виході нейрона (зазвичай використовується проміжок [0,1], інколи використовуються [-1,1]), також від функції залежить чи поширить сигнал нейрон, та якою буде сила цього сигналу.

Розглянемо основні функції активації для нейронів:

1. Порогова функція

   Найпростіша функція активації. При отриманні на вхід достатнього рівня сигналу повертає 1, якщо рівень сигналу замалий нейрон не активується і функція повертає на вихід 0.

$$\varphi(s) = \begin{cases} 1, & s \geq 0 \\ 0, & s \leq 0 \end{cases}$$

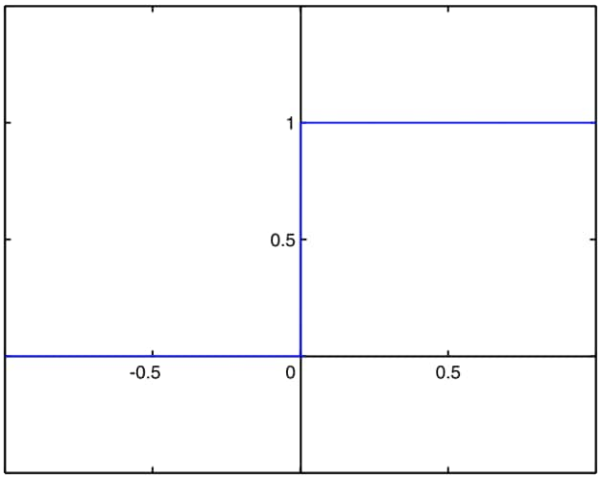


Рисунок 2.2.1 -1 Графік «Порогової функції»

2. Кусково-лінійна функція

   Функція активації, яка занулює усі від'ємні значення менші за -0.5 та обмежує вихідні значення більші за 0.5 одиницею. Сигнал на виході нейрона лінійно пов'язаний із ваговою сумою сигналів на його вході.

$$\varphi(s) = \begin{cases} 1, & s \geq 0.5 \\ |s|, & -0.5 < s < 0.5 \\ 0, & s \leq -0.5 \end{cases}$$

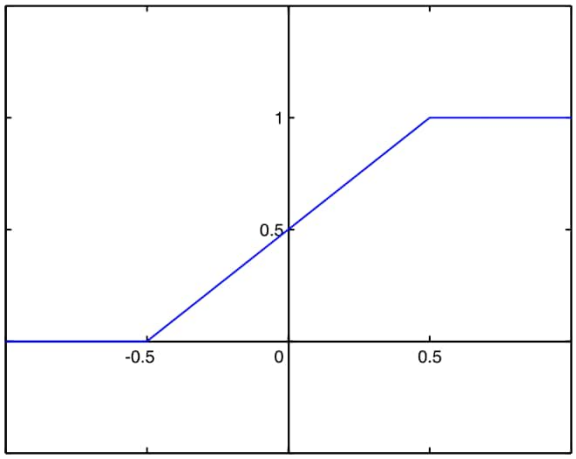


Рисунок 2.2.1-2 Графік «Кусково-лінійної функції»

3. Сигмоїдальна функція

Недоліками крокової та Кусково-лінійної активаційних функцій можна назвати те, що вони не є диференційованими на всій числової осі, а отже не можуть бути використані при навчанні за деякими алгоритмами. Тому найчастіше використовується сигмоїдальна функція у нейронних мережах, особливо у мережах для задач класифікації.

$$\varphi(s) = \frac{1}{1 + e^{-s}}$$

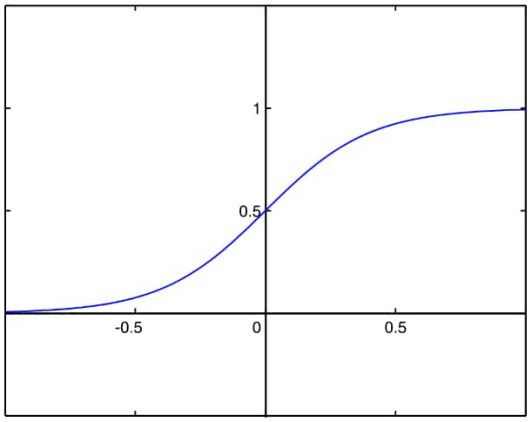


Рисунок 2.2.1-3 Графік «Сигмоїдальної функції»

4. ReLU (Випрямлена лінійна функція)

Основними перевагами даної функції є :

- Швидкість тренування

- Краще градієнтне поширення : рідше виникає проблема зникання градієнта[1] у порівнянні з сигмоїдальною функцією

$$\varphi(s) = \begin{cases} 0, s < 0 \\ s, s \geq 0 \end{cases}$$

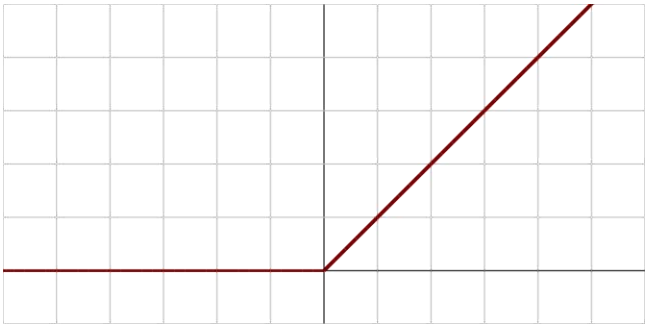


Рисунок 2.2.1-4 Графік функції ReLU

5. Softmax

Функція що найчастіше використовується в останньому шарі класифікаторів на основі нейронних мереж. Вона стискає вхідний вектор зі значеннями що повернув останній прихований шар до вектора зі значеннями в проміжку [0,1] що у сумі дають 1. Ці значення вектора дають нам розподіл імовірностей для кожного з класів класифікації.

$$f_i(\vec{s}) = \frac{e^{s_i}}{\sum_{i=1}^{J} e^{s_i}}$$

2.3 Штучні нейронні мережі

Штучні нейронні мережі, як і їхній біологічний аналог, складаються з з'єднаних між собою нейронів, які у тій чи іншій мірою мають елементарні функції біологічних нейронів.

Це наділяє ШНМ такими властивостями[1]:

[1] Проблема зникання градієнта – може виникати при навчанні мережі методом зворотного поширення помилки, коли градієнт не доходить до перших шарів мережі, тим самим сповільнюючи або взагалі зупиняючи навчання мережі

1. Адаптивне навчання - здатність нейронної мережі змінювати свої характеристики, закладені їй у програмі.
2. Узагальнення – навчена мережа є нечутливою до несильно змінених вхідних даних, що дозволяє використовувати її у задачах класифікації, та відновлення інформації.
3. Стійкість до перебоїв – НМ може продовжувати виконувати завдання при руйнуванні декількох вузлів.

Можливість НМ узагальнювати різні вхідні сигнали, дає нам можливість розв’язувати ряд задач описаних вище, що непосильні для комп’ютера. Але для розв'язання цих задач мережу спочатку навчити, підібрати правильні ваги, так щоб нас задовольнила точність її роботи, що довгий час було дуже складно для багатошарових НМ.

2.4 Архітектура нейронних мереж

2.4.1 Одношарова нейронна мережа

Найпростіший варіант НМ повинен складатись з одного шару нейронів, у якому буде всього один нейрон. Вона може виконувати різні обчислення з вхідними сигналами та видавати один вихідних сигнал. Найчастіше від такої системи очікують відповідь так чи ні.

У трохи складнішому варіанті у шарі буде декілька нейронів, а отже і вихідних сигналів буде декілька. Це дозволяє використовувати їх для класифікації вхідних сигналів на більш ніж два класи.

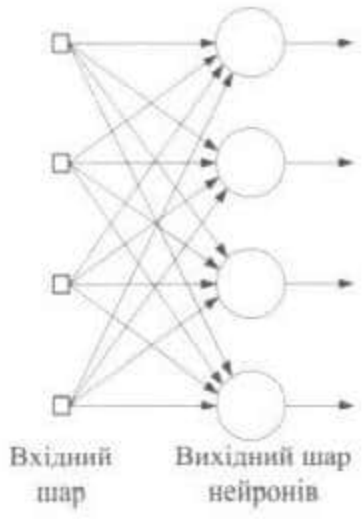


Рисунок 2.4.1 Одношарова нейронна мережа

Хоч мережа і називається одношаровою, але формально вона містить два шари – вхідний та вихідних. Це два базові шари, які містяться у кожні нейронній мережі. Вхідний шар використовується для того, щоб передати нейронній мережі вхідний вектор. Вихідний шар відповідає за отримання результатів роботи нейронної мережі. Нейрони

одношарової нейронної мережі знаходяться у вихідному шарі, вони виконують усі обчислення і повертають на вихід їх результати.

Очевидно, що розв'язувати складні задачі на нейронних мережах з такою архітектурою неможливо, адже проаналізувавши просто кожен біт певного вектора без аналізу їхніх взаємозв’язків - щось сказати про сам вектор складно.

2.4.2 Багатошарова нейронна мережа

Розвитком одношарової нейронної мережі є мережа з додатковими прихованими шарами – багатошарова нейронна мережа. Така мережа уже здатна виділяти певні зв’язки між елементами вхідного вектора. Це дає значний вплив на результат, якщо кількість вхідних даних велика.

Вхідний вектор, є інформацією для першого прихованого шару нейронів. Результат обчислень цього шару є входом для наступного прихованого шару нейронів (якщо такий є) і так до вихідного шару, результати якого вже передаються назовні.

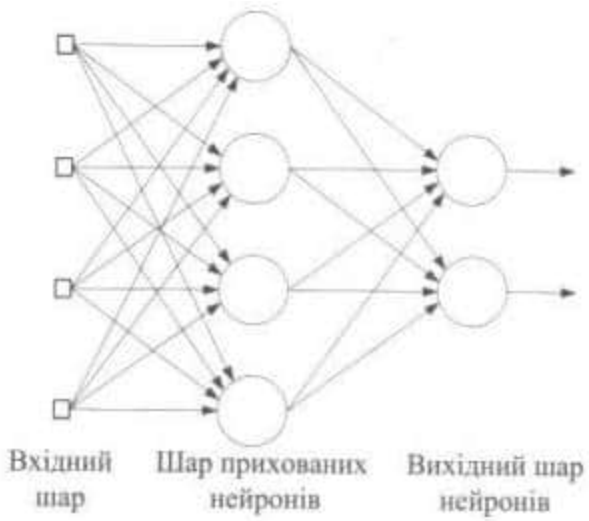


Рисунок 2.4.2 Багатошарова нейронна мережа

Довгий час реалізувати таку архітектуру було неможливо, через неможливість навчання такої мережі відомими на цей час методами. Але з проявою метода зворотного поширення похибки, нейронні мережі даного типу почали використовувати. А прискорення обчислень на GPU, дозволило проводити навчання таких мереж достатньо швидко та ускладнювати їх архітектуру для досягнення кращих результатів.

2.5 Згорткова нейронна мережа(CNN)

Згорткові нейронні мережі є дуже ефективними нейронними мережами для аналізу зображень. Це нейронна мережа прямого поширення[2], яка у своїй роботі виділяє із зображення характерні ознаки з допомогою згорткових шарів, та зменшує розмір матриці цих ознак з допомогою агрегувальних шарів. У результаті її роботи із зображення виділяються ознаки, які мережа потім класифікує повно зв'язним шаром.

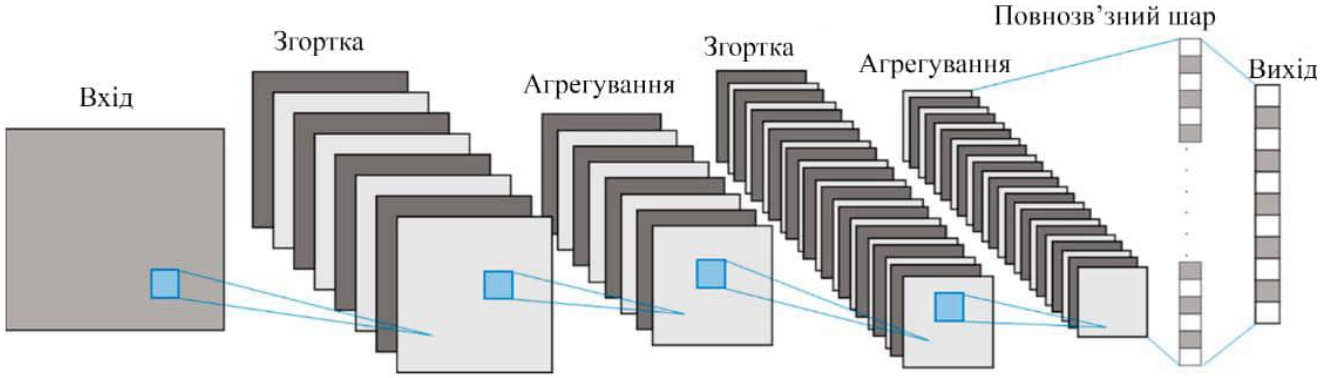


Рисунок 2.5 Архітектура згорткової нейронної мережі

Розглянемо роботу такої мережі на кожному шарі.

2.5.1 Вхідний шар

Даний шар є у кожній нейронній мережі, його завдання отримати інформацію про вхідні дані, що надаються мережі, та передати їх наступному шару на обробку. Також він виконує первинну обробку цих даних.

У мережах, що працюють із картинками цей шар містить стільки нейронів, скільки є пікселів у зображені – для зображень у градаціях сірого. У випадку кольорового зображення нейронів стане у 3 рази більше, адже потрібно ще отримати інформацію про кожен з трьох каналів – червоний, синій, зелений.

При класифікації тексту нас не цікавить його колір, тому будь-яку кольорову фотографію доцільно перетворювати у градації сірого, що дозволяє зменшити

[2] Нейронна мережа прямого поширення – мережа у якій дані поширюються від вхідного шару до вихідного, в одному напрямку.

кількість нейронів не тільки у вхідному шарі, а й у решті шарів. Таке перетворення позитивно впливає і на точність нейронної мережі, і на час її навчання.

2.5.2 Шар згортки

Даний шар використовує ядро згортки для отримання інформації про різні елементи зображення. На вхід такого шару поступає матриця зображення (або уже обробленого попередніми шарами зображення), і елемент за елементом обробляється ядром.[4]

Ядро згортки – це матриця, основна функція якої, виділити необхідні для подальшого аналізу характерні риси зображення. На початкових шарах це можуть бути характеристики ліній – пряма, чи опукла. А на глибших шарах, коли в декількох елементах матриці зберігається уже значна інформація про початкове зображення уже можна виділяти характерні ознаки того чи іншого предмета, наприклад колеса машини, чи пальці руки.

Найпростішим прикладом матриці згортки, є матриця що виділяє границі об’єкта.

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Рисунок 2.5.2-1 Матриця для виділення об’єкта на картинці

Під час обробки ядром відбувається множення під матриці виділеної з вхідної матриці на матрицю ядра, для кожного елемента вхідної матриці. У результаті ми отримаємо матрицю на виході оброблену ядром, у якій будуть виділені потрібні нам характеристики зображення.

Перевагою таких шарів у нейронних мережах є їхня можливість до навчання (вони самі створюють ядра, для потрібних їм ознак), та незалежність від положення об’єкта на зображенні (така мережа знайде потрібні ознаки об’єкта незалежно від їх положень на картинці).

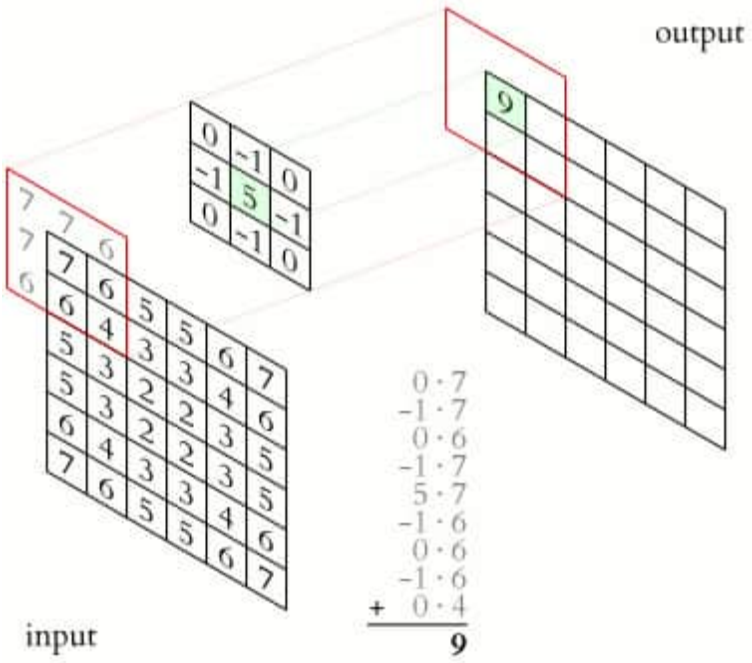


Рисунок 2.5.2-2 Приклад роботи шару згортки

2.5.3 Агрегувальний шар

Після шару згортки повинен іти агрегувальний шар, який зменшить розмірність матриці ознак.[4] Це необхідно, щоб з уже виділених ознак зображення вибрати найбільш важливі, зменшити кількість нейронів, та обчислень на наступних шарах, щоб вони виділяли більш глобальні ознаки з зображення.

Найчастіше використовуються шари, що вибирають максимальне або середнє значення з сітки. Я використовував шар MaxPooling – який вибирає максимальне значення із сітки. Принцип його дії можна побачити на рисунку 2.5.3. Шар поділяє вхідну матрицю на комірки вказаного розміру(на рисунку ці комірки розміру 2x2) та із кожної комірки вибирає найбільше значення.

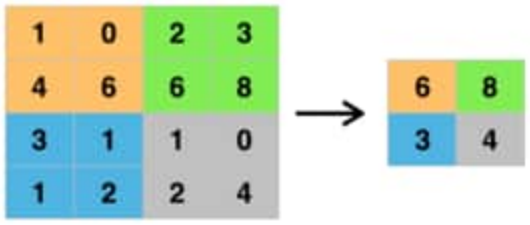


Рисунок 2.5.3 Робота агрегувального шару, що вибирає максимуми, розміром 2x2 на матриці 8x8

2.5.4 Повно зв’язний шар

Кінцевий шар нейронної мережі, що виконує завдання обчислення імовірностей, до якого класу відноситься вхідне зображення.

У цьому шарі на вхід кожного нейрона поступає сигнал від кожного нейрона з попереднього шару. Кожен нейрон відповідає за певний клас, і під час навчання у цього нейрона підбираються такі ваги синапсів, щоб він давав найбільшу імовірність для свого класу зображень.

У результаті роботи цього шару над вхідним вектором з особливостями зображення, нейронна мережа видає вектор з розподілом імовірностей для кожного з заданих класів. Клас з найбільшою імовірністю є класом до якого віднесла наша нейронна мережа зображення.

2.6 Рекурентна нейронна мережа(RNN)

Нейронні мережі, які у своїй архітектурі мають зворотні зв’язки називаються - рекурентними нейронними мережами. Такий підхід дозволяє рекурентним нейронним мережам отримати пам’ять на відміну від мереж прямого поширення. Це дозволяє їм ефективніше обробляти входи різної довжини. При обробці картинок рекурентні нейронні мережі обробляють ділянки картинок і на них шукають знайомі їм образи(у нашому випадку букви або частини букв) і маючи інформацію про минулий результат обробки, стараються оприділити що на цьому елементі картинки зображено.

2.6.1 Довга короткочасна пам’ять(LSTM)

Система LSTM була запропонована Зеппом Хохрайтером та Юргеном Шмідхубером у 1997 році.[2] Ця система базується на знаннях про свої минулі результати, та з допомогою них та нових вхідних даних аналізує картину повністю. Така система дуже ефективна в аналізі потокової інформації, та при аналізі зображень, коли необхідно із великого зображення частинами виділити якусь інформацію, що підходить у нашому випадку із розпізнаванням слів.

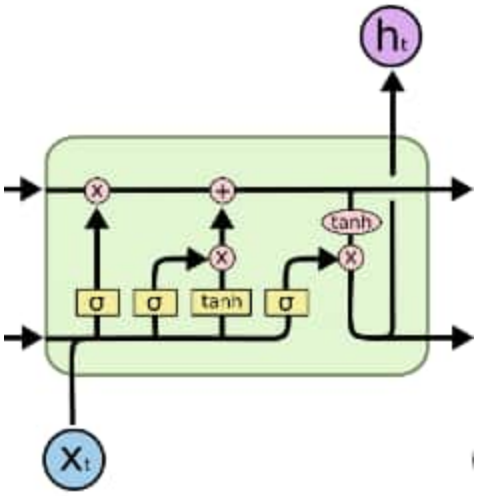


Рисунок 2.6.1-1 Схема комірки LSTM

LSTM у своїй роботі базується на таких компонентах: забувальний шар, вхідний шар, стану комірки, вихідний шар.

Перший крок у роботі комірки – вирішити яку інформацію необхідно забути у забувальному шарі. Для цього зазвичай використовується сигмоїдальна функція, яка оприділяє значенням від 0 до 1 для кожного стану комірки скільки інформації необхідно забути, де 0 – забути усю інформацію, 1 – запам’ятати усю інформацію. Це значення множиться на стан комірки й у результаті значення стану зменшується.[3] У моделі розпізнавання тексту із картинки цей шар відповідає за розпізнавання окремих букв. Коли в комірку приходить якась інформація, яка ніяк не зв’язана із попереднім результатом, це означає що буква змінилась, а отже попередній результат необхідно забути.

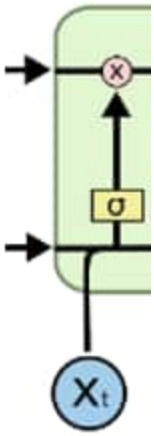


Рисунок 2.6.1-2 Забувальний шар LSTM

Другий крок у роботі комірки – вирішити яку нову інформацію нам необхідно зберегти у стані комірки. Для цього у нас є вхідний шар, який містить сигмоїдальну функцію, яка вирішує, які значення ми будем оновлювати, а шар tanh вибирає значення, які варто додати до стану. Ці два шари об'єднуються й оновлюють стан комірки.[3] У результаті роботи двох попередніх шарів ми оновлюємо стан комірки. Перший шар вирішив на скільки необхідно забути старих дані комірки, а другий шар вирішив на скільки необхідно додати у стан комірки нових даних. Таким чином ми утримуємо у пам'яті інформацію про окремі букви вхідного слова.

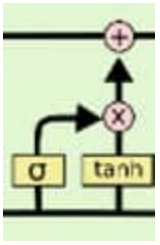


Рисунок 2.6.1-3 Вхідний шар LSTM

Третій крок у роботі комірки - вирішити яку інформацію показати на виході. Для цього є вихідний шар. Він бере дані з уже оновленого стану комірки, та з допомогою сигмоїдальної функції вирішує, як вхідні дані впливають на пам'ять комірки.[3] У результаті на виході ми отримуємо наш прогноз із даної ітерації. Таким чином ми на кожній ітерації отримуємо дані про букву й у результаті після проходженню по усьому зображенню ми можемо об'єднати отримані дані про букви та сказати яке слово у нас на фотографії.

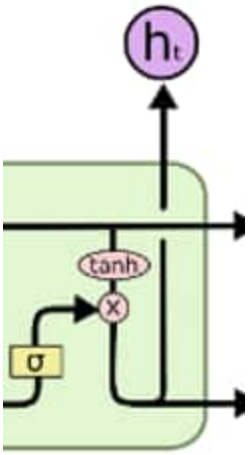


Рисунок 2.6.1-4 Вихідний шар LSTM

2.7 Згорткова рекурентна нейронна мережа(CRNN)

Як описано вище, згорткова нейронна мережа прекрасно підходить для виділення характерних ознак із зображення, а рекурентна мережа прекрасно підходить для того, щоб маючи якісь ознаки ділянок зображення, знаходити взаємозв'язки між цими ділянками та повертати у результаті текст зображений на картинці.

Поєднавши ці дві мережі ми отримаємо CRNN.

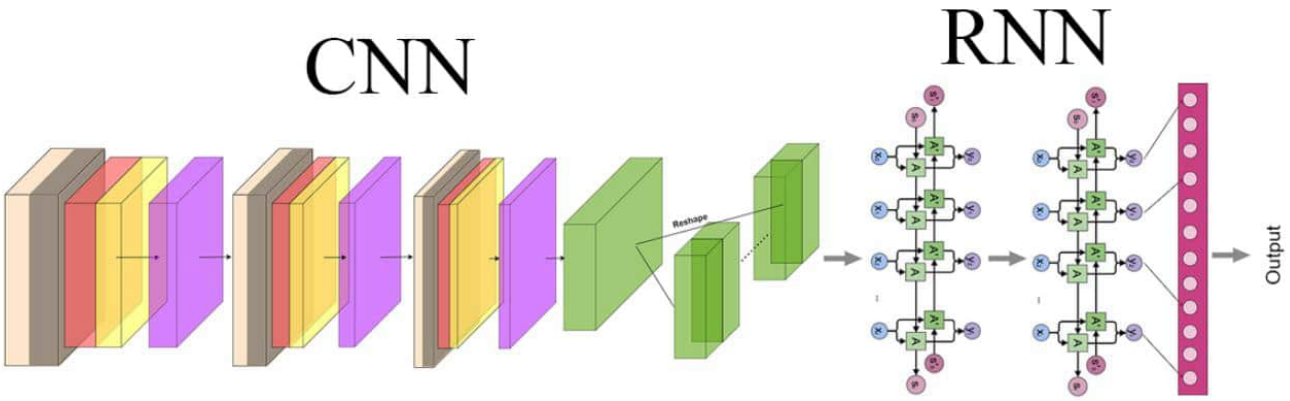


Рисунок 2.7 Архітектура згорткової рекурентної нейронної мережі

Дана мережа будується з декількох згорткових шарів, та зазвичай двох рекурентних шарів. Згорткові шари слугують для виділення характерних ознак зображення, у той час, як рекурентні шари уже із цих характерних ознак стараються виділити з усього зображення необхідну інформацію.

Перевагою даної системи є те що рекурентним шарам не потрібно обробляти складні взаємозв'язки між пікселями зображення. У результаті точність такої мережі суттєво покращується.

Отримуючи вхідне зображення дана мережа обробляє його згортковими шарами, виділяючи необхідні характерні ознаки, та зменшуючи розмірність вхідних даних. У результаті коли інформація доходить до рекурентних шарів вона є більш зрозумілою для них, та має меншу розмірність, що позитивно впливає на точність роботи системи. Перший рекурентний шар виділяє локальні залежності між ознаками. Другий шар уже працює із даними що повернув перший і визначає глобальніші залежності між елементами, у нашому випадку між буквами, або їх частинами.

2.8 Навчання нейронних мереж

Однією з основних переваг нейронних мереж є уміння навчатись. Головним алгоритмом навчання багатошарових нейронних мереж є – метод зворотного поширення помилки. Цей алгоритм поширює сигнал помилки від виходу до входу

нейронної мережі (тобто у зворотному напрямку) та мінімізує похибку змінюючи синоптичні ваги нейронів, на кожному кроці.

В основі цього методу лежить деякий алгоритм пошуку мінімуму. Найпростішим є метод градієнтного спуску – ітеративний метод пошуку мінімуму похибки у просторі ваг.

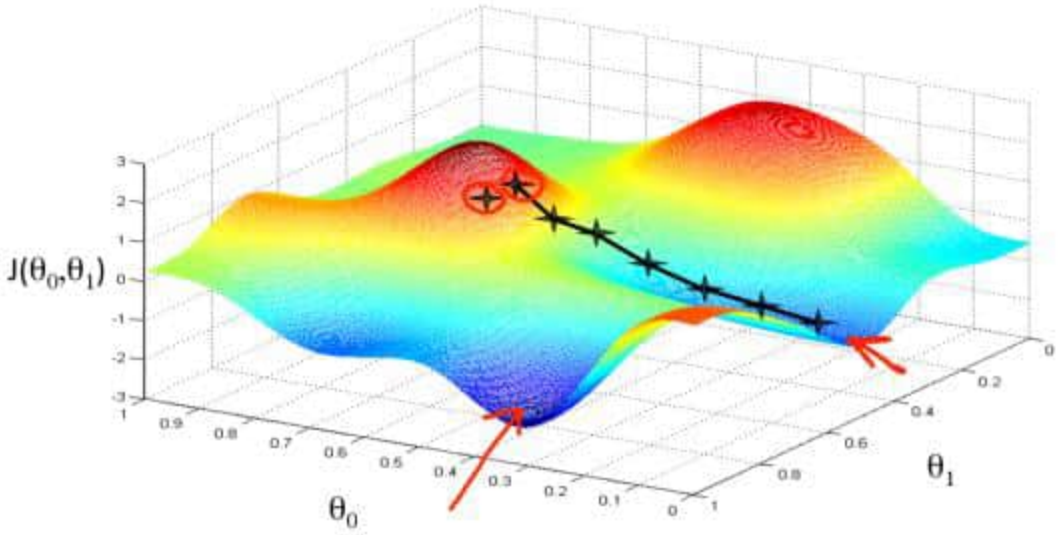


Рисунок 2.6 Метод градієнтного спуску, що попав у пастку локального мінімуму.

У цього метода є декілька недоліків:

1. Попадання у локальний мінімум – мережа попаде у незначний мінімум, коли поруч є значно глибший.
2. Проблема розміру кроку - якщо вибрати крок замалим, час навчання буде дуже великим. Якщо ж вибрати крок завеликим, то мережа буде скакати між різними частинами градієнта і мережа з кожною ітерацією не буде наближатись до бажаного результату.

   На перевагу цим недолікам виступає простота цього метода, а деякі підвиди тою чи іншою мірою виправляють ці недоліки.

2.8.1 Проблеми навчання нейронної мережі

При навчанні нейронної мережі головне не попасти в одну з двох основних пасток зв’язаних з навчальними даними:

1. Перенавчання
2. Недонавчання

Перша відбувається тоді коли нейронна мережа отримала або завелику кількість навчальної вибірки, або під час навчання вибірка була не рівномірною й один з класів зустрічався значно частіше за інші. В результаті мережа дуже сильно змістила ваги до навчальних даних. На рисунку 2.6.1 показано, як функція що представлення зеленою лінією огинає усі тренувальні дані максимально акуратно, тим самим створюючи для випадкових даних вищі похибки у порівнянні з функцією що представлена чорною лінією.

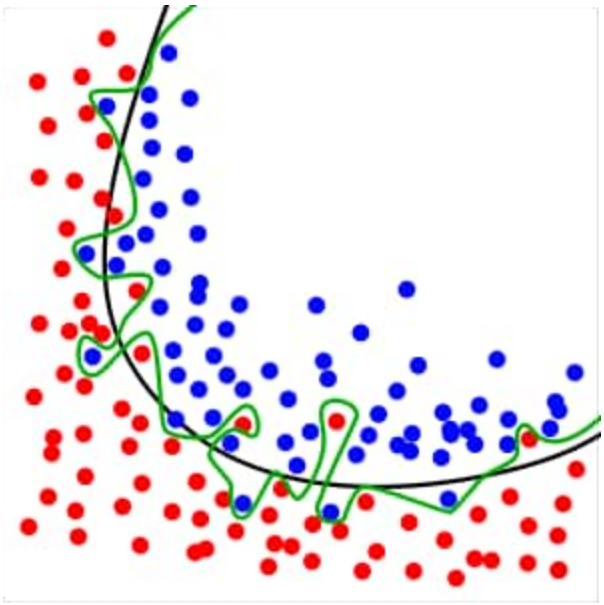


Рисунок 2.6.1 Результат перенавчання

Недонавчання трапляється тоді, коли навчальних даних було замало і модель не змогла сформувати правильні ваги для коректної роботи мережі з зовнішніми даними.

2.8.2 Проблема зникнення градієнта

Тренуючи нейронні мережі методом зворотного поширення помилки на основі градієнта існує проблема, що при великій кількості шарів мережі під час зворотного поширення помилки у них, похідні можуть бути менші за одиницю. У такому випадку при перемножуванні цих похідних градієнт стає надзвичайно малим, що перешкоджає вагам змінюватись, тим самим зупиняючи навчання нейронної мережі.

Для зменшення впливу цієї проблеми на навчання можна використовувати активаційну функцію ReLU. Вона не змінює позитивні значення, але занулює від’ємні значення, отже від’ємні значення не будуть зменшувати градієнт.

Також використовується метод нормалізації градієнта, для прикладу Batch Normalization. Ідея такого методу полягає у тому щоб за допомогою середнього значення, та значення стандартного відхилення невеликого набору вхідних прикладів при тренуванні нормалізувати градієнт, щоб не допустити його зникненню.

Архітектура рекурентної мережі LSTM також помагає у розв’язанні цієї проблеми. Завдяки використанню забувального та запам’ятовувального шарів під час навчання LSTM здатна контролювати градієнт і заважати його зникненню.

3 ОПИС ОСНОВНИХ ПРОГРАМНИХ ЗАСОБІВ

3.1 Python

Інтерпретована[3] об’єктно орієнтована мова програмування високого рівня із динамічною типізацією. Найпопулярніша мова програмування нині. Саме тому для неї існує велика кількість бібліотек, що значно спрощують написання програм.

Завдяки динамічній типізації та інтерпретованому методу виконання коду, Python є дуже зручною мовою програмування для написання коду із можливістю зразу бачити результат його роботи. Найпопулярнішим інтерактивним середовищем розробки є Jupyter Notebook. Це середовище є дуже зручним для розробки нейронних мереж, та інших проєктів штучного інтелекту, адже можна під час написання коду перевірити вхідні дані, чи значення якихось змінних просто вивівши їх на екран, що неможливо у компільованих мовах.

Завдяки наявності зручного інструментарію, простого синтаксису, та величезної кількості хороших сторонніх бібліотек, Python є популярною мовою для написання різноманітних аплікацій, включаючи проєкти із машинного навчання.

3.2 Numpy

Бібліотека з відкритим кодом для мови Python з простим та доступним синтаксисом. Створена для виконання швидких та зручних математичних обчислень з різними багатовимірними векторами. Написана мовою C, та добре оптимізована для швидких обчислень, завдяки використанню різноманітних технологій розподілених, паралельних, а також графічних обчислень.

[3] Інтерпольована мова програмування – мова програмування, в якій сирцевий код програми не перетворюється попередньо повністю у машинний код для виконання, як у компільованих мовах, а виконується рядок за рядком з допомогою спеціальної програми-інтерпретатора.

3.3 TenthorFlow

Платформа з відкритим кодом розроблена компанією Google для різноманітних задач машинного навчання. Ця платформа містить безліч інструментів, бібліотек та вбудованих ресурсів, що дозволяють створювати різноманітні аплікації з машинного навчання. TenthorFlow підтримує обчислення як на CPU та GPU, так і на TPU

TenthorFlow містить багато бібліотек різних рівнів для написання найрізноманітніших систем штучного інтелекту. Найпопулярніша серед початківців є бібліотека Keras - створена для того, щоб надати зручний та швидкий інструментарій для створення, та розширення нейронних мереж. З її допомогою програмісту не потрібно задумуватись про фундаментальні властивості штучних нейронів, їх зв'язки у мережі, а також про навчання цих мереж.

Завдяки своїй багатофункціональності, просторі у використанні, та наявності бібліотеки TenthorFlow для усіх популярних мов програмування, ця платформа отримала значну популярність серед різноманітних розробників систем штучного інтелекту. Дану платформу для реалізації різних задач штучного інтелекту використовує багато відомих компаній таких як Twitter, Spotify, Intel, ARM, та навіть Coca-Cola.

3.4 Django

Високорівневий вебфреймворк Python, з допомогою якого можна швидко та ефективно розробляти веб додатки різного рівня. Django використовує схему MVC[4], що дозволяє розбити програму на компоненти, спрощує розробку та підтримку коду.

[4] MVC – Model View Controller – патерн, що використовується при проєктуванні та розробці програмного забезпечення. Даний патерн передбачає поділ системи на три частини: модель даних(збереження та обробка вхідних даних та даних з бази даних), вигляд(інтерфейс взаємодії з користувачем), контролер(керує взаємодіями з моделлю та виглядом).

Django містить багато готових шаблонів, компонентів та бібліотек, що спрощують розробку веб додатків. Розробникам не потрібно переживати за безпеку, автентифікацію, авторизацію та інші аспекти веб розробки. Це значно спрощує розробку та надійність застосунку.

Django використовують багато різних відомих компаній, таких як: Instagram, Pinterest, Dropbox та навіть NASA.

4 РОЗРОБКА ДОДАТКА

Створення додатка можна розділити на дві частини:

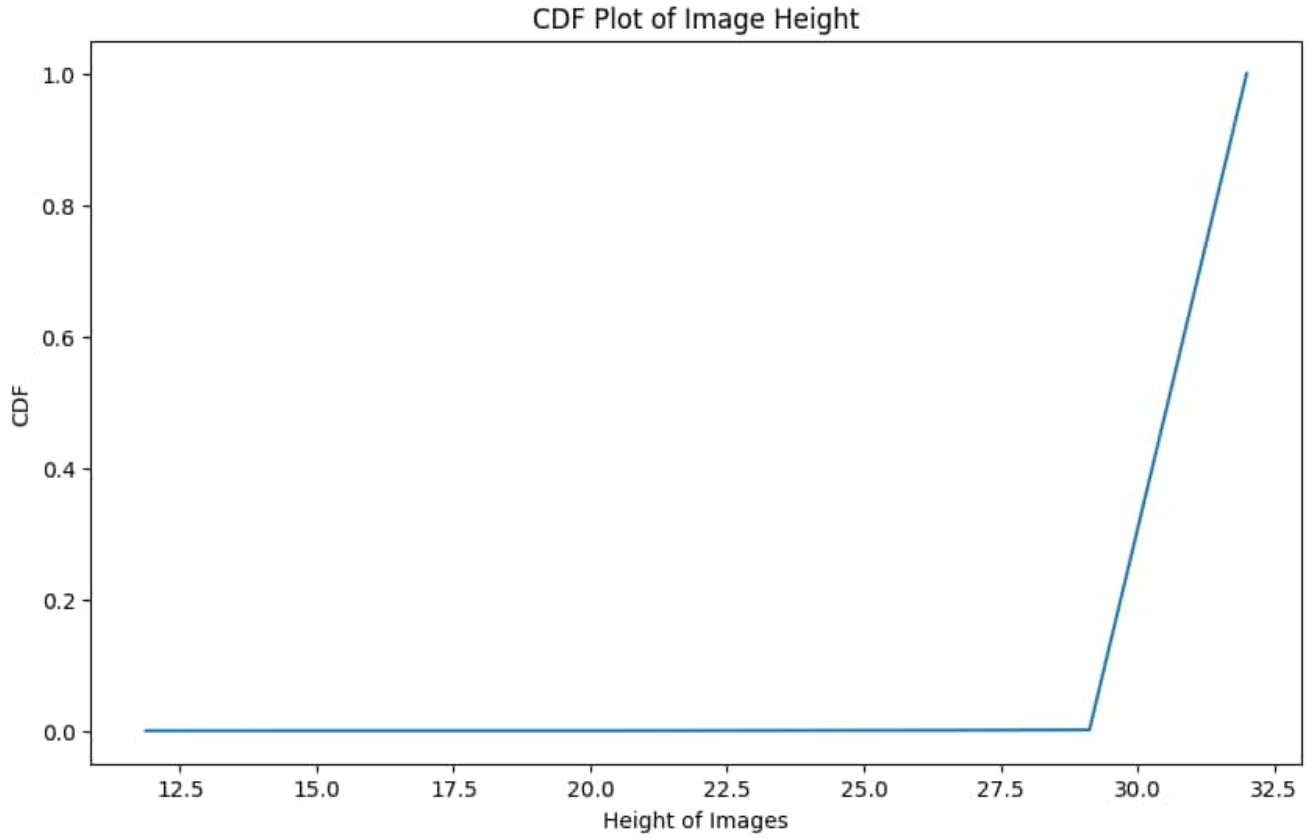
1. Розробка нейронної мережі для розпізнавання тексту із картинок.
2. Розробка веб-додатка, що використовуватиме цю нейронну мережу для класифікації зображень завантажених на сайт.

4.1 Розробка нейронної мережі

Для розробки нейронної мережі я використав бібліотеку TenthorFlow разом із Keras для мови програмування Python. Для написання мережі я використовував середовище доступне у Visual Studio Code що називається IPython Notebook(аналогічна система як Jupiter Notebook).

Спочатку я знайшов хороший дата сет з англомовними словами.[6] MJSynth Synthetic Word Dataset - цей дата сет має більше 4-ох мільйонів картинок англійських слів у відтінках сірого. Проаналізувавши картинки дата сету за допомогою бібліотеки numpy було зроблено висновок що більшість картинок мають висоту 31 піксель(на графіку 4.1-1 видно що висота картинок різко зростає

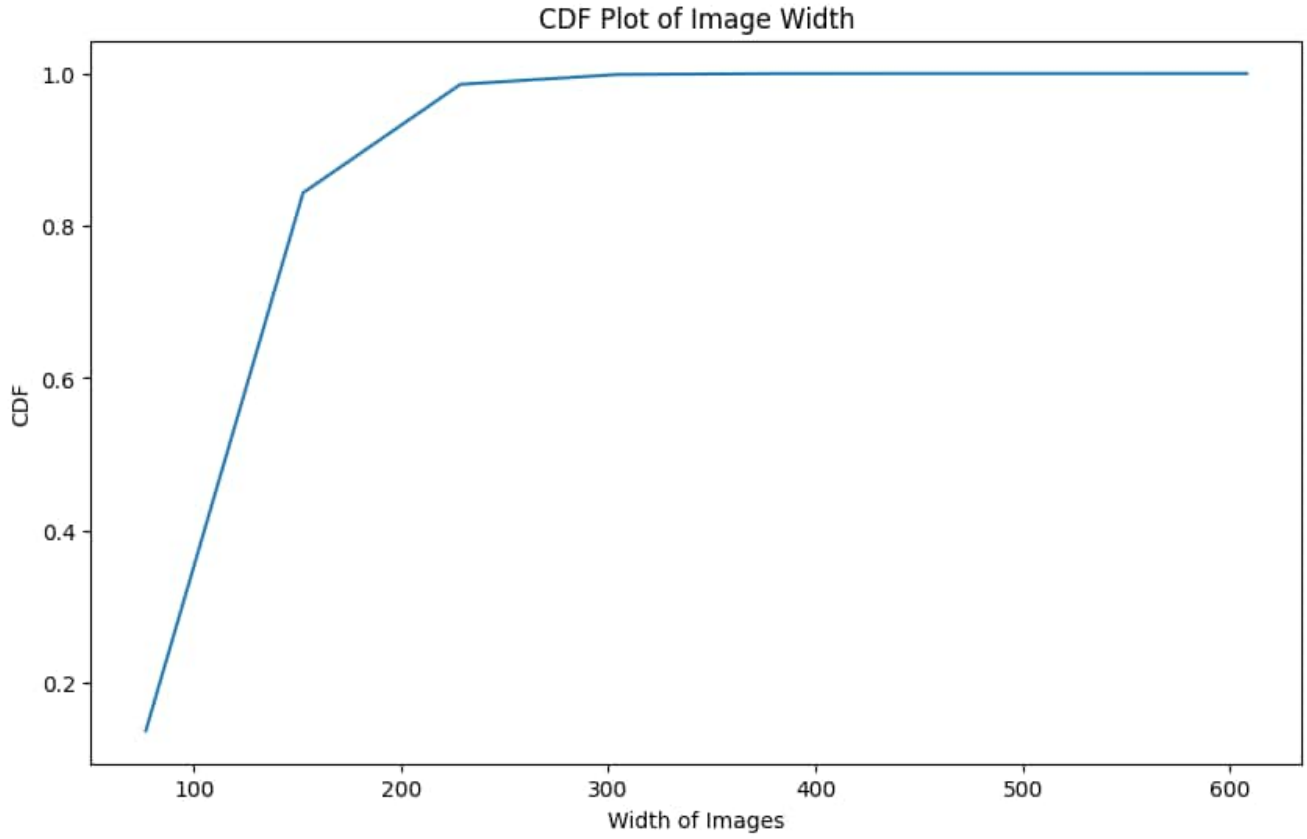
зі значення 29 пікселів, та рівномірно розподілена між 29 та 32 пікселями)



Графік 4.1-1 Акумулююча функція висоти картинок у пікселях

Аналогічне дослідження проведено для ширини картинок. З графіка 4.1-2 випливає, що 85% усіх картинок мають ширину до 160 пікселів, а решта картинок мають ширину від 160 до 240 пікселів. Тому було прийнято рішення, що ширина у 170

пікселів буде оптимальною.



Графік 4.1-2 Акумулююча функція ширини картинок у пікселях

У результаті кожну картинку дата сету перед навчанням та валідацією перетворено до розміру 170*31. Також прийнято рішення, що картинки із текстом який користувач схоче розпізнавати також необхідно перетворювати до такого розміру, та перетворити із кольорових до градацій сірого. Для цього використана Python бібліотека OpenCV.

Наступним кроком є створення моделі самої мережі. Я створив декілька CRNN моделей, що базується на різній кількості шарів згортки та двох, або одному рекурентному шару LSTM.

Найбільша та найефективніша модель містить 7 згорткових шарів та агрегувальних шарів відповідно. Згорткові шари мають багато фільтрів що дозволяє їм ефективно виділяти різноманітні особливості із зображення. У найбільшій мережі перший згортковий шар містить 64 фільтри із ядром 3*3, другий 128 фільтрів із ядром 3*3, третій та четвертий шари 256 фільтрів із ядром 3*3, п’ятий та шостий місять 512 фільтрів із ядром 3*3, а сьомий містить також 512

фільтрі, але ядро 2*2, два рекурентні шари LSTM, та Softmax на виході. Така велика модель має 7,350,373 параметрів із яких 7,345,893 необхідно навчити. Саме тому навчання відбувається дуже довго та необхідно мати великий дата сет.

Далі запустив мережу тренуватись на тренувальній вибірці. Для цього поділив вибірку на частини по 200000 картинок, та ці картинки поділив на корзинки по 64 елементи, у результаті одна епоха складалась із 3125 ітерацій по 64 картинки у кожній. Приблизний час навчання однієї епохи 3-4 години й ще приблизно годину на валідацію. Запустити навчання навіть на 10 епох за один раз було неможливо, тому було зроблено json файл із моделлю та після кожного епохи зберігались параметри моделі у фалі з розширенням .h5. У результаті комп'ютер на ніч навчався на дві епохи та зберігав їх у теці, а на наступний день завантаживши останню епоху, можна було продовжити навчання. Таким чином я натренував мережу на 10 епохах за 5 днів. Та натренував потім менші мережі для порівняння результатів.

4.2 Розробка веб-додатка для взаємодії із нейронною мережею

Для демонстрації можливостей нейронної мережі я написав веб-додаток з допомогою Django, який при запуску завантажує в оперативну пам'ять нейронну мережу, та дозволяє користувачеві завантажувати фотографії, які додаток обробляє та передає нейронній мережі для розпізнавання тексту.

Django додаток, що запускається на локальному сервері командою, після запуску зчитує json файл з архітектурою мережі, яка зберігається у теках веб-додатка, та з допомогою бібліотеки keras відновлює модель. Після цього до відновленої моделі завантажуються ваги натренованої перед тим моделі із файлу .h5.

Далі ми можемо перейти на сайт за посиланням http://127.0.0.1:8000/image_processing/process_image/ на якому маємо можливість

завантажити фотографію натиснувши на кнопку Вибрати файл, та завантажити із комп’ютера файл. (Рисунок 4.2-1 та Рисунок 4.2-2)

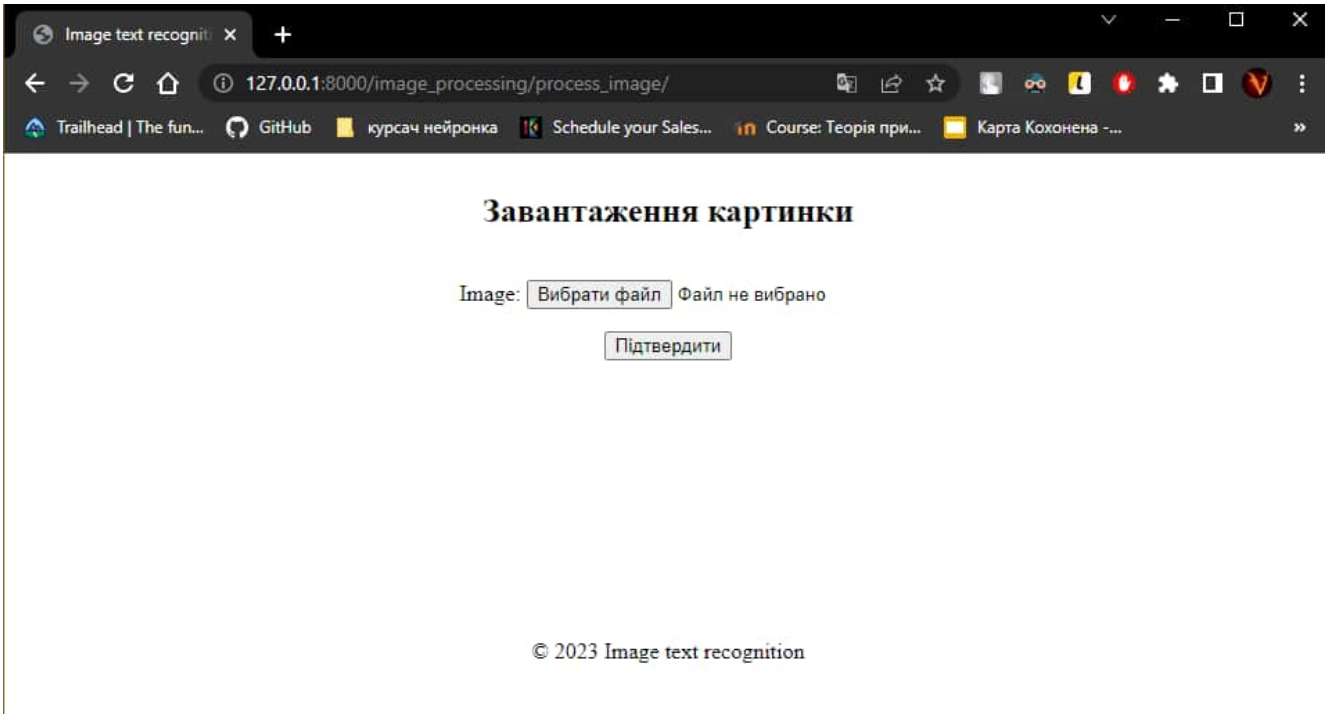


Рисунок 4.2-1 Скриншот головної сторінки сайту

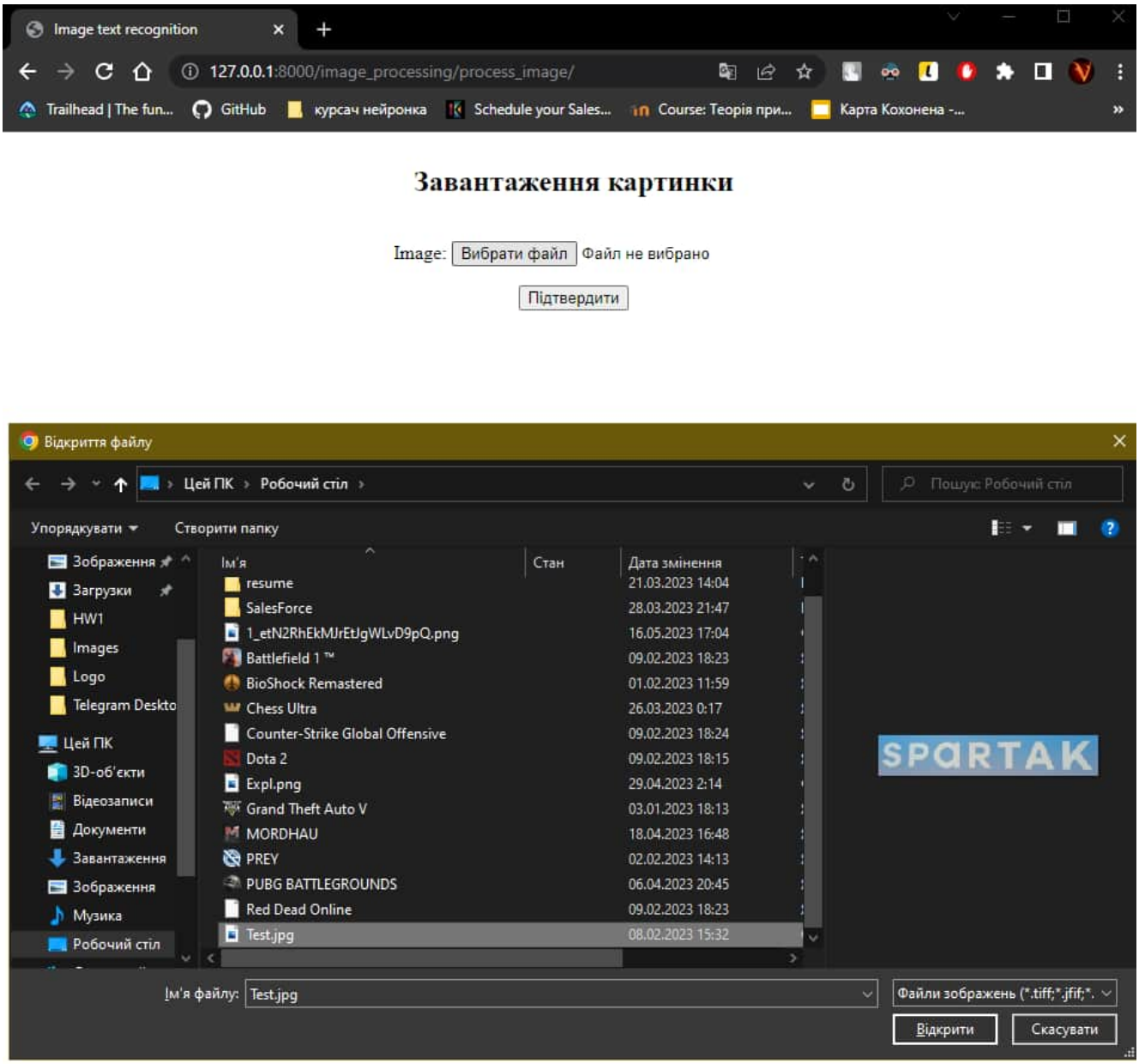


Рисунок 4.2-2 Скриншот вибору файлу

Після вибору файлу, біля кнопки «Вибрати файл» буде написана назва вибраного файлу. Далі необхідно натиснути кнопку «Підтвердити», тоді файл відправляється на сервер та проходить перед обробку(зменшення шумів, перетворення на градації сірого та зменшення розширення). Результат перед

обробки зображений на рисунку 4.2-3.

Рисунок 4.2-3 Зображення до перед обробки та після

Після цього файл відправляється на вхід нейронної мережі, яка витягує із нього текст, та повертає його на нову сторінку(Рисунок 4.2-4)

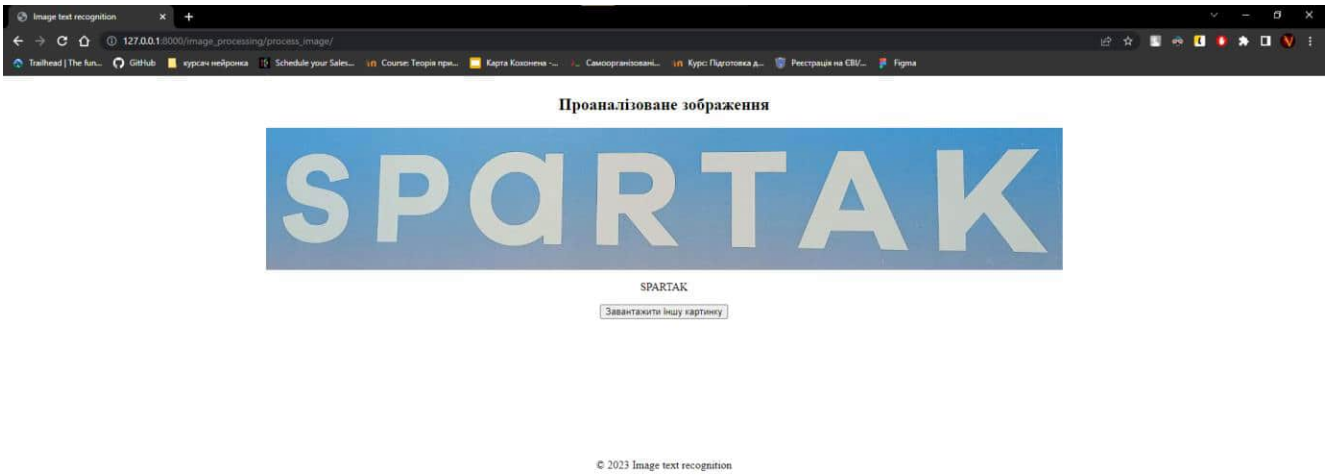


Рисунок 4.2-4 Скриншот з результатом аналізу картинки

Після цього можна натиснути кнопку «Завантажити іншу картинку» та потрапити назад на головну сторінку, щоб вибрати іншу картинку.

На рисунках нижче можна побачити інші приклади роботи додатка

Рисунок 4.2-5 Приклад розпізнавання тексту із логотипа Samsung

Приклад на рисунку 4.2-5 показує, що положення слова на картинці, та наявність незначних дефектів не впливає на його розпізнавання.

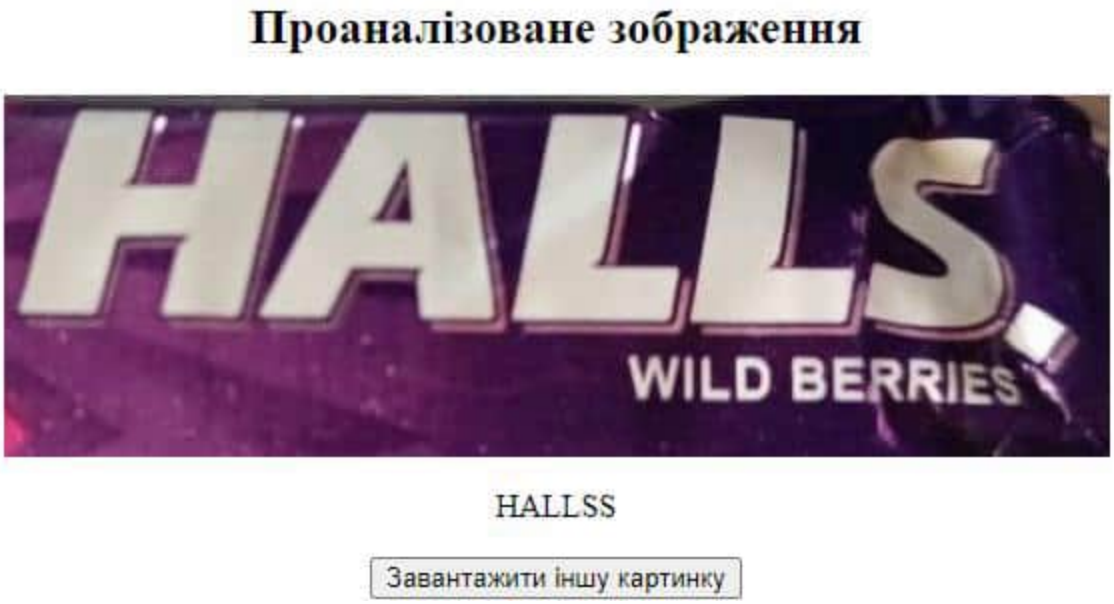


Рисунок 4.2-6 Не правильно розпізнане слово

Приклад на рисунку 4.2-6 показує, що мережа може не правильно розпізнати слово. Хоч усі букви розпізнані правильно, але імовірно через крапку та деформовану упаковку мережа ще дописала одну букву S у кінці.

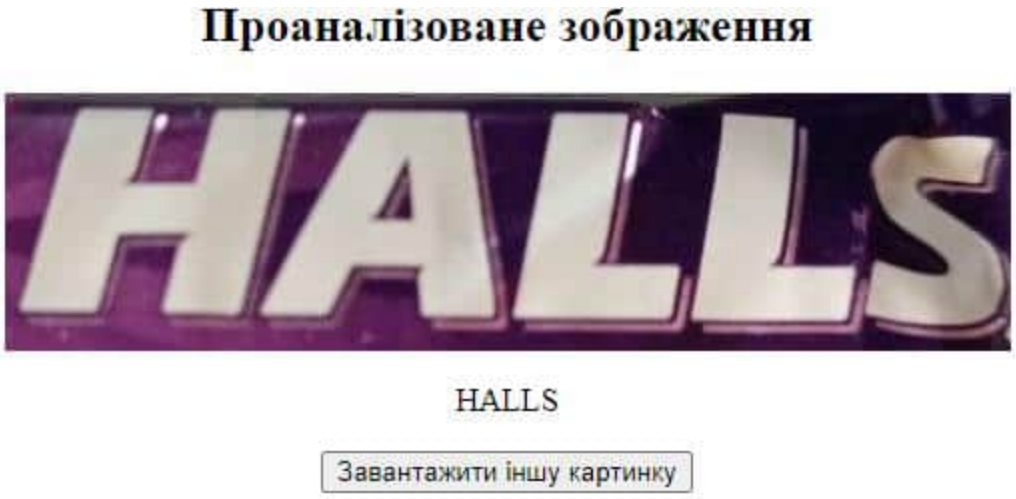


Рисунок 4.2-7 Правильно розпізнаний текст із цієї ж упаковки

На рисунку 4.2-7 показано, що уже обрізана картинка розпізнається правильно, не зважаючи на геометричні дефекти тексту.

Рисунок 4.2-8 Правильно розпізнаний текст малої роздільної здатності із різноманітними дефектами

На рисунку 4.2-8 показано, як текст із різними дефектами, а саме розмазаною буквою R, деформованими іншими буквами, та різними бліками на полотні правильно розпізнається.

На даних прикладах показано, що мережа добре справляється із задачею розпізнавання тексту.

5 АНАЛІЗ РЕЗУЛЬТАТІВ

Вартує проаналізувати не тільки точність розпізнавання, та швидкість розпізнавання, а і час навчання мережі, адже цей параметр також важливий, враховуючи те що навчаються ці мережі довго. Для цього було зроблено 4 мережі різної складності, а саме з різною кількістю згорткових шарів, та одна мережа із лише одним рекурентним шаром.

Найпростіша мережа: 3 згорткових шари - перший шар містить 64 фільтри та ядро 3*3, другий шар 128 фільтрів, ядро 3*3, третій шар містить 256 фільтрів та ядро 2*2. Та один рекурентний шар - 948,709 навчальних параметрів. Така ж модель, але з двома рекурентними шарами - 1,999,333 параметрів для навчання.

Середня мережа: 5 згорткових шари - перший шар містить 64 фільтри та ядро 3*3, другий шар 128 фільтрів, ядро 3*3, третій шар містить 256 фільтрів та ядро 3*3, четвертий шар 512 фільтрів та ядро 3*3, п’ятий шар містить 512 фільтрів та ядро 2*2. Кількість параметрів для навчання 4,394,469.

Найбільша мережа: 7 згорткових шарів - перший згортковий шар містить 64 фільтри із ядром 3*3, другий 128 фільтрів із ядром 3*3, третій та четвертий шари 256 фільтрів із ядром 3*3, п’ятий та шостий місять 512 фільтрів із ядром 3*3, а сьомий містить також 512 фільтрів та ядро 2*2. Кількість параметрів для навчання 7,345,893.

Порівняємо спочатку час навчання мереж:

Мережа з одним рекурентним шаром – приблизно 40 хвилин на епоху.

Найпростіша мережа – приблизно годину 20 хвилин на епоху

Середня мережа – приблизно годину 40 хвилин на епоху

Найбільша мережа – приблизно 2 години 20 хвилин на епоху.

Також не залежно від мережі на епоху навчання необхідно ще приблизно година додатково на валідацію. Отже, ефективніше навчати більші мережі, враховуючи факт із валідацією, та той факт що мережа у якої в 7 разів більше параметрів навчається всього в 4 рази довше.

Порівняємо точність після п’яти епох навчання кожної мережі. Для цього візьмем 150 випадкових картинок із дата сету, запустимо розпізнавання і зрівняємо результати із бажаними як для цілого слова, та і для кожної букви. У результаті отримаємо загальну точність мережі, та точність визначення окремих букв(слово вважається неправильно передбаченим, навіть якщо одна буква не правильна, у результаті мережа можна на слові із 10 букв неправильно визначити якусь одну, і уже слово не правильне, тому загальна точність буде завжди значно нижчою ніж точність визначення окремих букв).

Мережа з одним рекурентним шаром – загальна точність 68,7%, точність визначення окремих букв 84,7%, час визначення 150 картинок – 8,95 секунди, приблизно 27 мілісекунд на слово. Час навчання 366 хвилин.

Найпростіша мережа – загальна точність 74,0%, точність визначення окремих букв 87,2%, час визначення 150 картинок – 10,26 секунди, приблизно 33 мілісекунди на слово. Час навчання 528 хвилин.

Середня мережа - загальна точність 77,3%, точність визначення окремих букв 90,7%, час визначення 150 картинок – 11.69 секунди, приблизно 39 мілісекунд на слово. Час навчання 612 хвилин.

Найбільша мережа – загальна точність 80,0%, точність визначення окремих букв 91,3%, час визначення 150 картинок – 16.18 секунд, приблизно 50 мілісекунд на слово. Час навчання 845 хвилин.

Мережа	Загальна точність	Точність визначення окремих букв	Час розпізнавання слова	Час навчання 5 епох
3CNN+1RNN	68,7%	84,7%	27 мс	366 хв
3CNN+2RNN	74,0%	87,2%	33 мс	528 хв
5CNN+2RNN	77,3%	90,7%	39 мс	612 хв
7CNN+2RNN	80,0%	91,3%	50 мс	845 хв

Аналізуючи таблицю із результатами я дійшов висновку, що найбільший приріст по загальній точності від наявності другого рекурентного шару +5,3%, додавання згорткових шарів дає також приріст, але якщо шарів буде забагато, то з картинки можуть виділятись непотрібні особливості, та впливати на аналіз тексту. Також зі збільшенням шарів збільшується час розпізнавання тексту, та час навчання, а також зменшується ефективність навчання, адже більшій мережі необхідно більше даних, та існує проблема зникнення градієнта, а отже проблема у навчанні перших шарів мережі.

Збільшення точності визначення окремих букв при переході від архітектури 5CNN+2RNN до 7CNN+2RNN дуже не значна, що свідчить про те що додавати ще згорткові шари не є ефективно. Саме тому було прийнято рішення продовжити навчання цієї мережі. Після 10 епох навчання її точність досягла 87%, чого цілком достатньо на реальних прикладах показаних у розділі 4.2.

ВИСНОВОК

У моїй роботі розглянута згорткова рекурентна нейронна мережа для вирішення задачі розпізнавання англомовних слів із зображень. Описані особливості, та принцип роботи такої мережі.

У результаті з допомогою мови програмування Python та бібліотек Keras, і Tenthorflow розроблено мережу із згортковою рекурентною архітектурою, навчено її на великому наборі даних, протестовано на реальних зображеннях із текстом. Мережа показала хороші результати при обробці реальних прикладів.

Для зручності роботи із мережею було розроблено веб додаток з допомогою бібліотеки Django, який виконує перед обробку зображення, та ефективно й швидко передає його до мережі для отримання з нього тексту. Після обробки мережею зображення, сайт повертає користувачу текст із зображення.

Також був проведений аналіз ефективності роботи мережі. Було проаналізовано вплив від ускладнення архітектури на швидкість роботи мережі, точність визначення як окремих букв слова, так і слова у цілому, та вплив на швидкість навчання.

Нейронна мережа такого типу для розпізнавання тексту ефективно працює із розпізнаванням різноманітних елементів на зображеннях. Її можна навчити на різних дата сетах під різноманітні задачі (наприклад розпізнавати номерні знаки авто, чи певні ідентифікаційні коди) і мережа буде ефективно справлятись із заданою задачею. Розпізнавання тексту фіксованої довжини із картинок для такої мережі проста задача.

Така система спрощує перенесення даних із зображень у різні системи. Наприклад автоматичне паркування автомобілів, скануючи їхні номера, швидке

перенесення паспортних даних із фотографій паспортів, номерів телефонів та інших текстових даних із зображень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Трушевський В.М., Г.А. Шинкаренко, Н.М. Щербина Метод скінченних елементів і штучні нейронні мережі: теоретичні аспекти та застосування : ЛНУ імені Івана Франка 2014р. 198-208с.
2. Sepp Hochreiter, Jürgen Schmidhuber Long short-term memory. Neural Computation 9 1997р.
3. Understanding LSTM Networks 2015р. / Режим доступу: https://colah.github.io/posts/2015-08-Understanding-LSTMs/
4. Введение в свёрточные нейронные сети (Convolutional Neural Networks) [AndrewShmig] – 2019р. / Режим доступу: https://habr.com/ru/post/454986/
5. An Intuitive Explanation of Connectionist Temporal Classification Harald Scheidl 2018р. / Режим доступу: https://towardsdatascience.com/intuitively-understanding-connectionist-temporal-classification-3797e43a86c
6. Text Recognition Data / Режим доступу: https://www.robots.ox.ac.uk/~vgg/data/text/#sec-synth
7. Keras API reference / Режим доступу: https://keras.io/api/