Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

факультет прикладних інформаційних технологій та електроінженерії
(повна назва факультету)
кафедра радіотехнічних систем
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Метод розпізнавання нечітких символів з використанням нейронної мережі

Виконав(ла): студент(ка) 6 курсу, групи РРм-61
спеціальності 172 Телекомунікації та радіотехніка
(шифр і назва спеціальності)

	(підпис)	Щіпський А.В (прізвище та ініціали)
Керівник	(підпис)	Дунець В.Л. (прізвище та ініціали)
Нормоконтроль	(підпис)	Хвостівська Л.В. (прізвище та ініціали)
Завідувач кафедри	(підпис)	Дунець В.Л. (прізвище та ініціали)
Рецензент	(підпис)	Хвостівський М.О. (прізвище та ініціали)

Тернопіль
2021

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет *прикладних інформаційних технологій та електроінженерії*
(повна назва факультету)

Кафедра *радіотехнічних систем*
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ *Дунець В.Л.*
(підпис) (прізвище та ініціали)
« » 2021 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *магістр*
(назва освітнього ступеня)

за спеціальністю *172 Телекомунікації та радіотехніка*
(шифр і назва спеціальності)

студенту *Щіпському Анатолію Володимировичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Метод розпізнавання нечітких символів з використанням нейронної мережі*

Керівник роботи *Дунець Василь Любомирович, к.т.н.*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*30*» *листопада* 20*21* року № *4/7-1019*

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи *Об'єкт дослідження: процес розпізнавання нечітких символів для оптичного зчитування зображень. Предмет дослідження: методи розпізнавання мовних сигналів з використанням нейронної мережі.*

4. Зміст роботи (перелік питань, які потрібно розробити)

1. Аналітична частина
2. Основна частина
3. Науково-дослідна частина
4. Охорона праці та безпека в надзвичайних ситуаціях

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці та	*Клепчик В.М., стар. викл. каф. ОХ*		
безпека в надзвичайних			
ситуаціях			

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	*Отримання завдання*	30.10.2021	*виконано*
2	*Аналіз завдання*	02.11.2021	*виконано*
3	*Виконання розділу 1*	09.11.2021	*виконано*
4	*Виконання розділу 2*	15.11.2021	*виконано*
5	*Виконання розділу 3*	25.11.2021	*виконано*
6	*Виконання розділу 4*	28.11.2021	*виконано*
7	*Оформлення пояснювальної записки*	30.11.2021	*виконано*
8	*Оформлення графічного та презентаційного матеріалу*	02.12.2021	*виконано*
9	*Перевірка роботи на антиплагіат*	18.12.2021	*виконано*
10	*Попередній захист*	18.12.2021	*виконано*
11	*Захист*	21.12.2021	*виконано*

Студент _____ *Щіпський А.В.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Дунець В.Л.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Метод розпізнавання нечітких символів з використанням нейронної мережі» // Кваліфікаційна робота // Щіпський Анатолій Володимирович // Тернопільський національний технічний університет імені Івана Пулюя, факультет прикладних інформаційних технологій та електроінженерії, група РРм-61 // Тернопіль, 2021 // с. – 73, рис. – 37, табл. – 1, додат. – 2, бібліогр. – 51.

Ключові слова: РОЗПІЗНАВАННЯ ТЕКСТУ, ЛОКАЛІЗАЦІЯ ТЕКСТУ, НЕЙРОННІ МЕРЕЖІ.

В кваліфікаційній роботі запропоновано підхід до розпізнавання локалізованого тексту, що базується на поєднанні рекурентних, згорткових нейронних мереж (CRNN) та алгоритму CTC-loss. Ця архітектура нейронних мереж реалізована за допомогою мови програмування Python. Проведено експеримент на двох наборах даних, за результатами якого були побудовані графіки зміни протягом навчання функції втрат, відстані Левенштейна та точності розпізнавання на тренувальному та двох тестових наборах даних.

ANNOTATION

Theme of qualification work: "Method of fuzzy character recognition using a neural network" // Qualification work // Shchipsky Anatoly Vladimirovich // Ternopil National Technical University named after Ivan Pulyuy, Faculty of Applied Information Technologies and Electrical Engineering, group PPm-61 // Ternopil, 2021 // with. - 73, fig. - 37, table. - 1, appendix. - 1, bibliogr. - 51.

Keywords: TEXT RECOGNITION, TEXT LOCALIZATION, NEURAL NETWORKS.

The qualification paper proposes an approach to localized text recognition based on a combination of recurrent, convolutional neural networks (CRNN) and the CTC-loss algorithm. This neural network architecture is implemented using the Python programming language. An experiment was performed on two data sets, based on the results of which graphs of changes during training of the loss function, Levenstein distance and accuracy of recognition on the training and two test data sets were constructed.

ЗМІСТ

ВСТУП

Актуальність роботи. Складні графічні сцени – це зображення, що характеризуються відсутністю чітких критеріїв відхилення фону від тексту, неоднорідністю фону, великою ймовірністю різноманітних спотворень та зашумлення [1]. Такий текст може бути змінної якості, різного шрифту, нахилу, форми, товщини та текстури. Прикладами таких сцен можуть бути фотографії вуличних вивісок, кадри з фільмів із субтитрами, дані в системах навігації, тощо.

Системи розпізнавання тексту (т.зв. OCR-системи), що існують на даний момент, орієнтовані на паперові документи, в яких текст розташовується на однорідному контрастному тлі і легко піддається сегментації та бінаризації. Розпізнавання тексту на складних графічних сценах, на відміну розпізнавання тексту на паперових документах, ускладнюється неможливістю явної локалізації (detection) тексту. Система немає апріорної інформації у тому, де як на зображенні розташований текст, і локалізація тексту – це окремий етап роботи таких систем. Також дане завдання актуальна тим, що на даних зображеннях часто зустрічаються специфічні слова (назви компаній, марки автомобілів, жаргони в тексті реклами і т.д.). З цієї причини ускладнюється використання методів із заздалегідь заданим словником, що містить всі можливі варіації слів, що розпізнаються.

На сьогодні вже існують системи локалізації тексту, що використовуються у розпізнаванні номерних знаків автомобілів та зазначень дати та часу на фотографіях. Вирішення таких завдань полегшується тим, що текст, що розпізнається, має фіксовану довжину і алфавіт (набір можливих символів), а також знаходиться на контрастному фоні і може бути опрацьований алгоритмами сегментації. У роботі вирішується задача розпізнавання вже локалізованого тексту.

Метою роботи розроблення метод розпізнавання нечітких символів з використанням нейронної мережі.

Для досягнення поставленої мети нам потрібно вирішити такі завдання:

1. Провести огляд та аналіз існуючих на даний момент методів локалізації та розпізнавання тексту на зображеннях складних графічних зображень;

2. Обґрунтувати підхід із застосуванням CRNN архітектури нейронних мереж до розпізнавання локалізованого на зображенні тексту;

3. Проведення експеримент з реалізацією CRNN архітектури нейронних мереж та порівняння результатів на двох наборах даних з іншими підходами;

Об’єкт дослідження: процес розпізнавання нечітких символів для оптичного зчитування зображень.

Предмет дослідження: методи розпізнавання мовних сигналів з використанням нейронної мережі.

Методи дослідження: рекурентні нейронні мережі, згорткові нейронні мережі та алгоритм CTC-loss.

Наукова новизна отриманих результатів. Обґрунтування застосування нейронної мережі для розпізнавання нечітких символів, що підвищує точність розпізнавання нечітких символів

Публікації.

Викладені в роботі результати доповідалися та обговорюватися на 9-ій Науково-технічній конференції «Інформаційні моделі, системи та технології» (м. Тернопіль, 8-9 грудня 2021 р.).

РОЗДІЛ 1
АНАЛІТИЧНА ЧАСТИНА

1.1. Огляд методів розпізнавання нечітких символів на графічних стендах.

Оптичне розпізнавання символів (OCR) – це електронне зчитуванняя зображень друкованого, машинописного, рукописного тексту, в текстові дані. Ця технологія дає змогу редагувати текст, виконує машинний переклад, семантичну обробку тексту, а також заощаджувати пам'ять при зберіганні документів.

Завдання розпізнавання тексту не обмежується обробкою паперових документів, актуальне також завдання розпізнавання тексту на складних графічних сценах, зображених на фотографіях і кадрах відеозйомки.

Основна відмінність – це розташування тексту на зображенні. OCR-системи мають апріорне знання розташування тексту в документі. У тексті виділяються окремі рядки, слова та символи, які після попередньої обробки подаються на вхід алгоритму класифікації.

Прикладами складних графічних сцен можуть бути фотографії вуличних вивісок, кадри з фільмів із субтитрами, дані у системах навігації роботів тощо. Вирішення завдання розпізнавання подібного тексту дозволить використовувати такі системи на мобільних платформах, відео- та фотокамерах, у пошукових системах та багато іншого.

Локалізувати текст на зображенні зі сценою набагато складніше. Крім самого тексту на зображенні можуть бути інші об'єкти – дерева, автомобілі, люди тощо, і завдання локалізації тексту – відділення тексту на зображенні з інших об'єктів.

Розпізнавання тексту на графічній сцені поділяється на три етапи:

- локалізація тексту на зображенні;
- розпізнавання тексту на локалізованому зображенні;

- Перевірка знайденого слова за словником.

Рис. 1.1. Приклади зображень складних графічних сцен

Рис. 1.2. Приклад некоректної роботи алгоритму бінаризації

1.2. Локалізація тексту

Проведемо огляд методів, запропонованих у науково-технічній літературі на вирішення завдання локалізації тексту на зображеннях складних графічних сцен.

Робота Y. Kunishige та співавторів [2] спирається на ідею використання т.зв. "контексту оточення" (environmental context). Основний принцип полягає у використанні інформації про те, що оточує область – «кандидата». Пропонується аналізувати той фон, на якому знаходиться регіон зображення,

що, можливо, є текстовим. Ідея базується на емпіричному припущенні, що можливість наявності тексту, наприклад, на трав'яному покриві або на небі - низька.

Другий матеріал того ж автора [3], а також стаття авторського колективу Lao, Du та ін [4] присвячені розробці ідеї так званих feature-точок.

У роботі [5] описується ідея автоматичної генерації ознак, що використовуються для розпізнавання. Основна ідея у тому, що ознаки для алгоритму класифікації можна складати вручну, а створювати їх з допомогою машинного навчання. Як база для навчання пропонується використовувати так звані патчі, що мають розмір 8x8 пікселів. Подальша обробка зображення зводиться до обчислення навчених ознак на областях зображення, що цікавлять.

Основна ідея ряду робіт (наприклад, [6]) полягає в тому, що букви та слова на зображенні, як правило, мають постійну товщину штриха. Тому виявлення таких об'єктів, на думку авторів, перспективно використовувати алгоритм SWT. Ширина штриха в букві повинна бути приблизно постійна, так само як і букви в одному слові повинні мати схожу ширину. Ознака тим паче корисний, оскільки використовувати його можна як однієї з ознак при класифікації регіонів, і як ознаки при «збиранні» регіонів у слова.

Кордони символів у межах описаного підходу можуть обчислюватися, наприклад, Canny Egde Detector. Потім у напрямку градієнта знаходиться парна гранична точка. Якщо градієнт у цій точці приблизно колінеарен градієнту в першій точці, то всі точки між ними заповнюються значенням знайденої ширини.

Потім, при другому проході знайденим на першому проході відрізках обчислене значення ширини обмежується медіанним значенням вздовж відрізка. При цьому слід пам'ятати, що запропонований метод вимагатиме певних додаткових обчислювальних витрат для боротьби з помилками на кутах та деякими іншими специфічними для нього ефектами.

Рис. 1.3. Візуалізація результатів методу локалізації тексту, запропонованого у статті [5]

У роботі [4] до аналогічного завдання запропоновано дещо інший підхід. Там також вирішується завдання пошуку точкового тексту, але для виявлення точок, що становлять літери, застосовується досить добре відомий алгоритм FAST. Потім проводиться евристична фільтрація хибних кандидатів, об'єднання точок у букви, букв - у слова, після чого застосовується класифікатор SVM (Support Vector Machine) для детектування текстових областей.

Автори роботи [7] пропонують використовувати згорткові нейронні мережі для вирішення цього завдання. Під час навчання на вхід такої мережі подається зображення всієї графічної сцени, а як очікуваний вихід використовується локалізований текст цього ж зображення.

1.3. Розпізнавання тексту

Розпізнавання тексту на складній графічній сцені є процес декодування отриманого на етапі локалізації зображення в послідовність символів.

Такі методи умовно поділяються на три категорії [8]:

- методи на основі символів (character-based) [9, 10];
- методи на основі слів (word-based) [11];
- методи на основі послідовностей (sequence-based) [12].

Методи на основі символів (character-based) спочатку, як правило, локалізують окремі символи, після чого класифікують їх та поєднують у слова.

Рис. 1.4. Приклад некоректної роботи алгоритму локалізації символів

У роботі [13] пропонується наступний підхід до розпізнавання послідовності цифр: глибокої згортковою мережею з вхідного зображення X розміром 128 x 128 x 3 витягується вектор ознак $H \in R4096$, які далі передаються 6-ти незалежним (без weight sharing) для обчислення розподілу ймовірностей $P(S1|H)$,…, $P(S5|H)$ кожного символу в послідовності та ймовірності $P(L|H)$ для визначення довжини послідовності. Приклади зображень з номерними знаками будинків, що розпізнаються даною системою, наведено на рисунку 5, візуалізація обчислювального графа даної мережі - на рисунку 7.

Рис. 1.5. Приклади зображень з номерами будинків, що розпізнаються даною системою

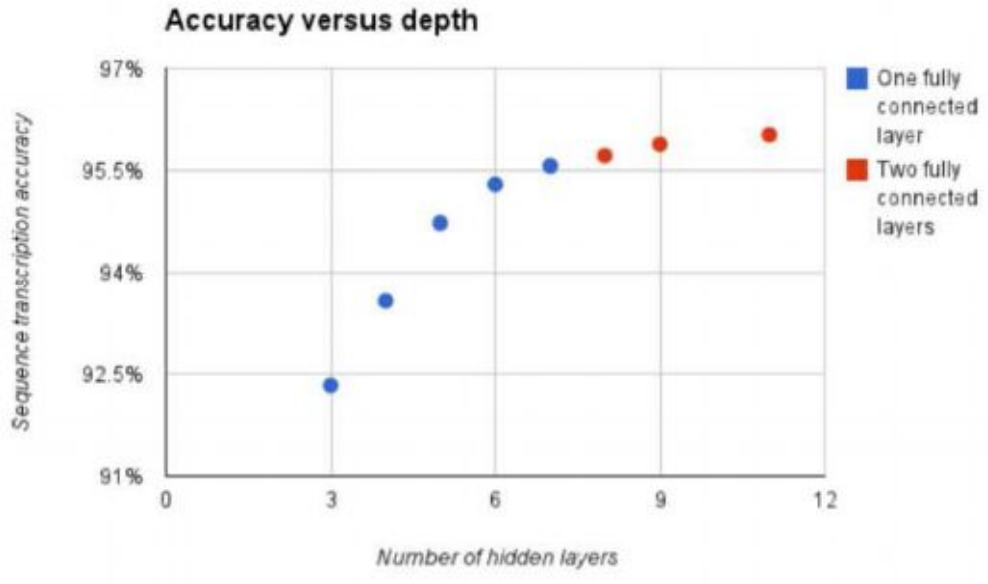

Рис. 1.6. Графік залежності точності розпізнавання від глибини мережі

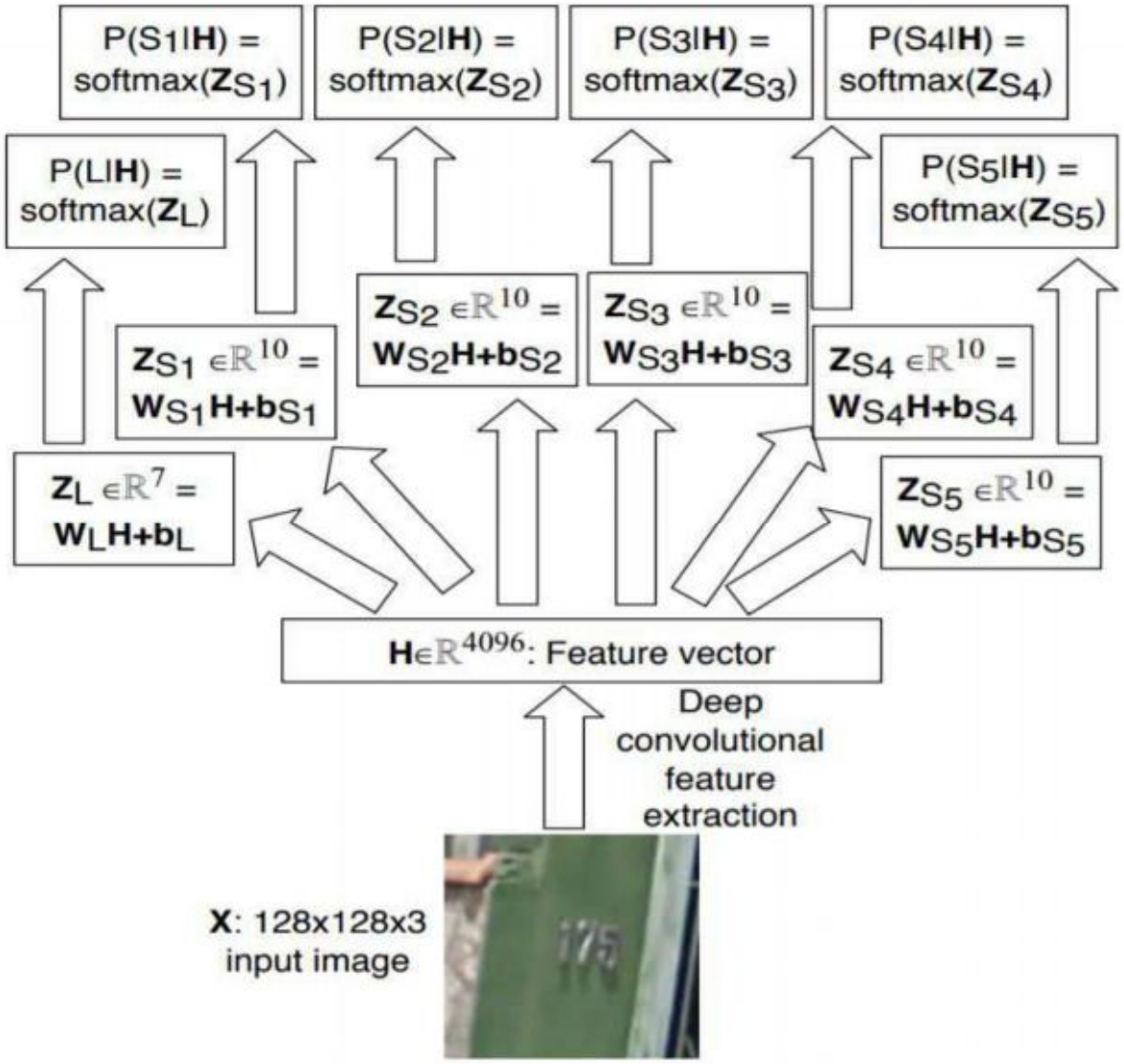

Рис. 1.7. Візуалізація обчислювального графа нейронної мережі із статті [13]

Методи з урахуванням слів(word-based) не локалізують окремі символи, а розпізнають слова цілком [11]. Max Jaderberg, у спільній роботі з іншими авторами [12], пропонує використовувати згорткові нейронні мережі із заздалегідь визначеним словником слів. Вихідний шар цієї мережі є вектором розподілу ймовірностей розміром 1x1xN, де N – число слів у словнику. Цей підхід простий у реалізації, але недостатньо гнучкий практично при розпізнаванні слів, які у словнику.

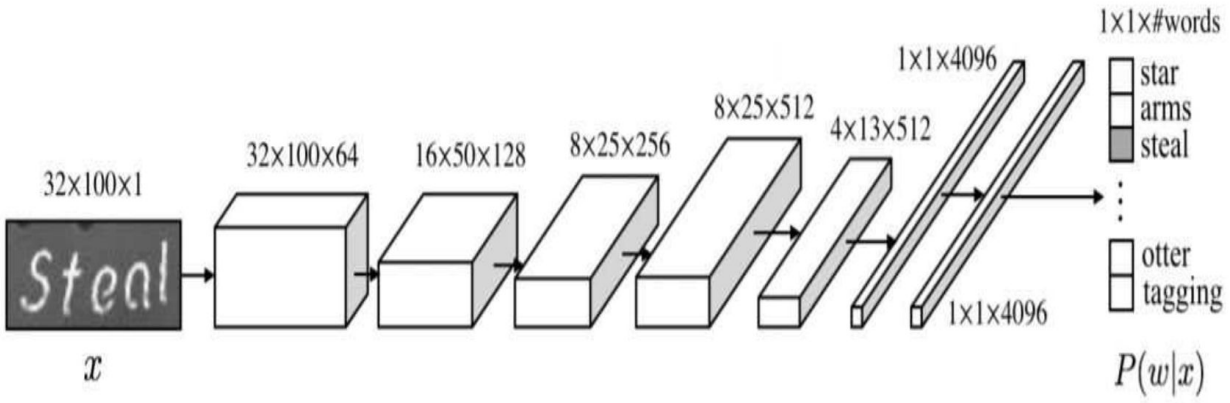

Рис.1.8. Архітектура нейронної мережі у вигляді згортки, запропонованої в статті [12]

Методи на основі послідовностей (sequence-based) базуються на поданні вхідного зображення у вигляді заданої послідовності та її декодування [14]. До цих методів відноситься і використання архітектури нейронних мереж, що описується в цій роботі (CRNN + CTC-loss) [15].

У випадках, коли застосування описаних вище методів не забезпечує необхідну точність розпізнавання, застосовується перевірка розпізнаного слова за словником з метою підвищення точності класифікації. При такому підході проводиться посимвольна перевірка кожного розпізнаного слова та зіставлення його зі словами у наперед складеному словнику.

У статті про CRNN[15] авторами пропонується поєднувати згорткові та рекурентні мережі, а як функцію втрат використовувати CTC loss. Даний підхід на момент публікації показував state-of-the-art результати у вирішенні даного завдання, як і його подальші модифікації. На рисунку 9 наведено візуалізацію CRNN.

У роботі [16] пропонується використовувати нейронні мережі з увагою (attention) перед рекурентними шарами. Архітектура такої мережі показана рисунку 10.

Авторами статті [17] пропонується спочатку за допомогою ковзного вікна виділяти на вихідному зображенні n ділянок з подальшим виділенням ознак згортковими шарами, класифікацією та передачею алгоритму CTC-loss. Архітектура нейронної мережі представлена рисунку 11.

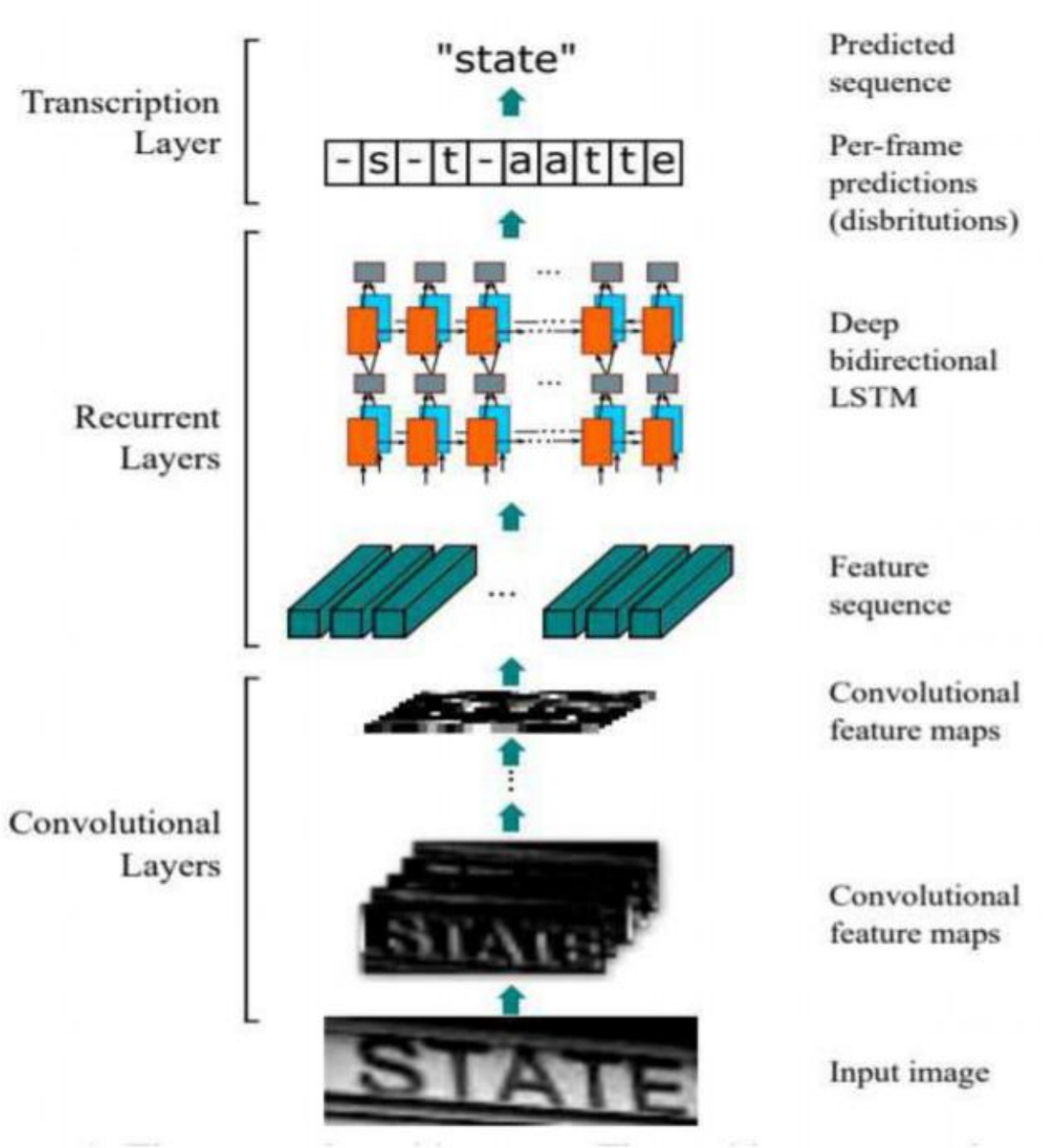

Рис. 1.9. Візуалізація архітектури CRNN

У статті [18] автори пропонують використовувати нейронну мережу без рекурентних шарів для покращення паралелізації обчислень. Вхідне зображення обробляється ковзним вікном, далі згортковими шарами витягуються ознаки і передаються спеціальному згортковому шару з увагою (attention) для кодування і декодування послідовностей. Ця архітектура представлена рисунку 12.

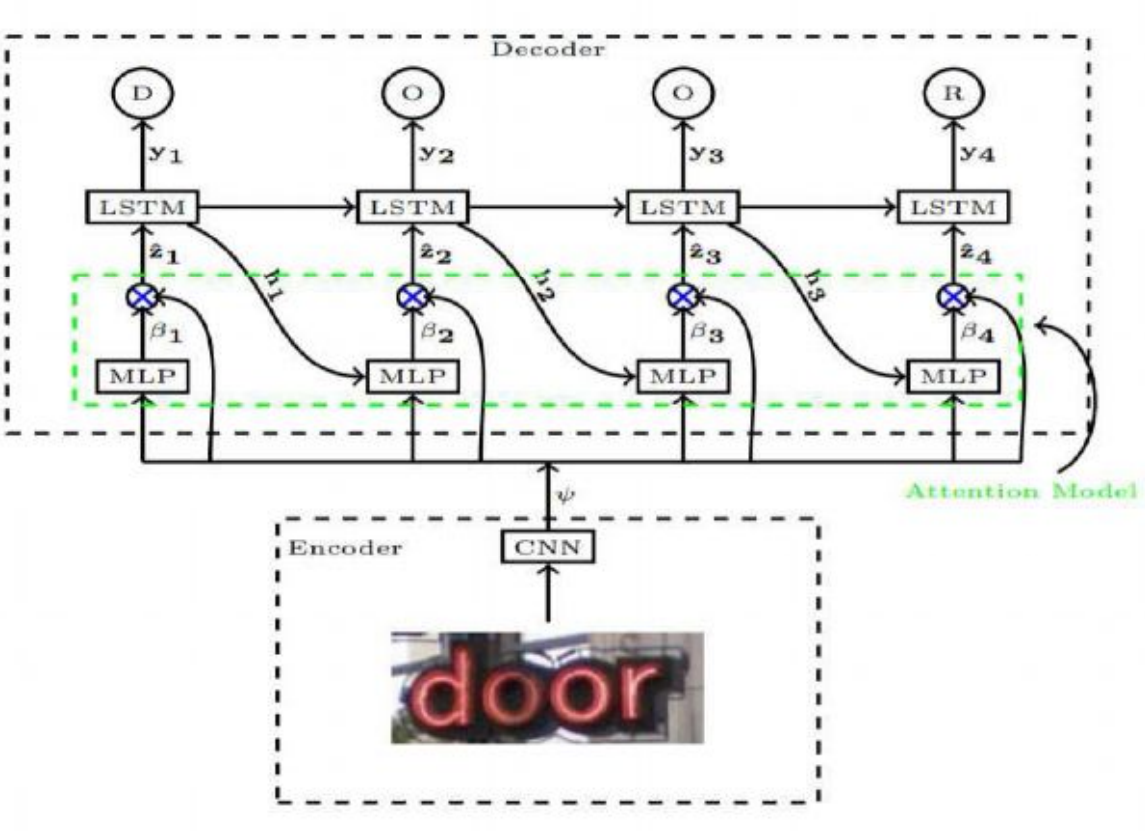

Рис. 1.10. Архітектура нейронної мережі з увагою (attention)

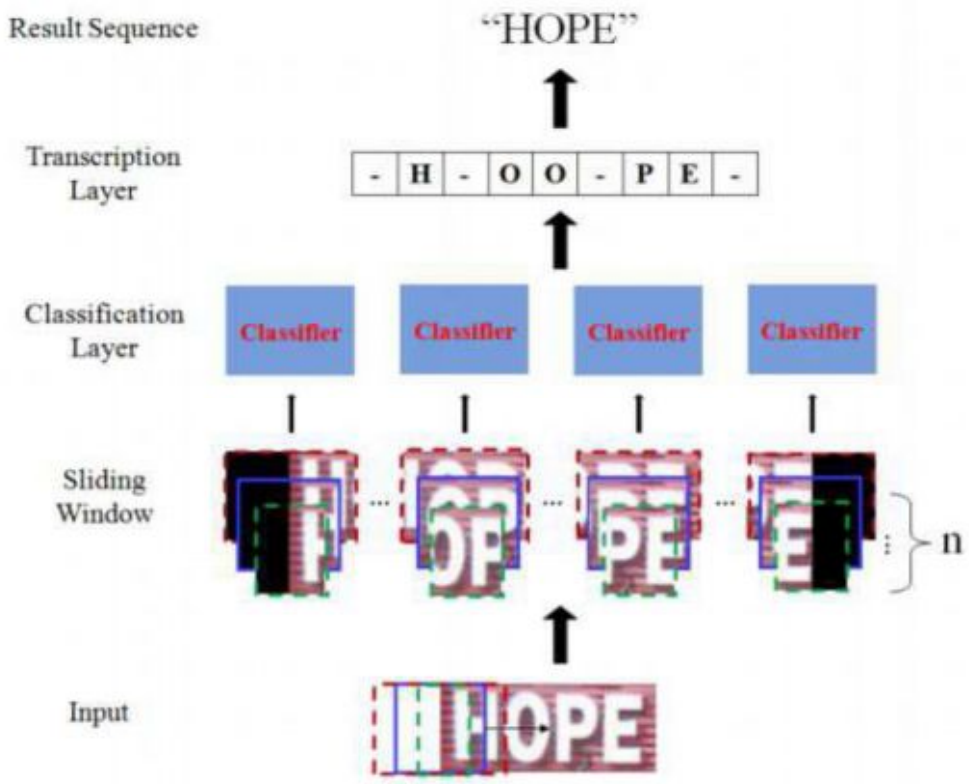

Рис. 1.11. Архітектура нейронної мережі, запропонованої авторами [17]

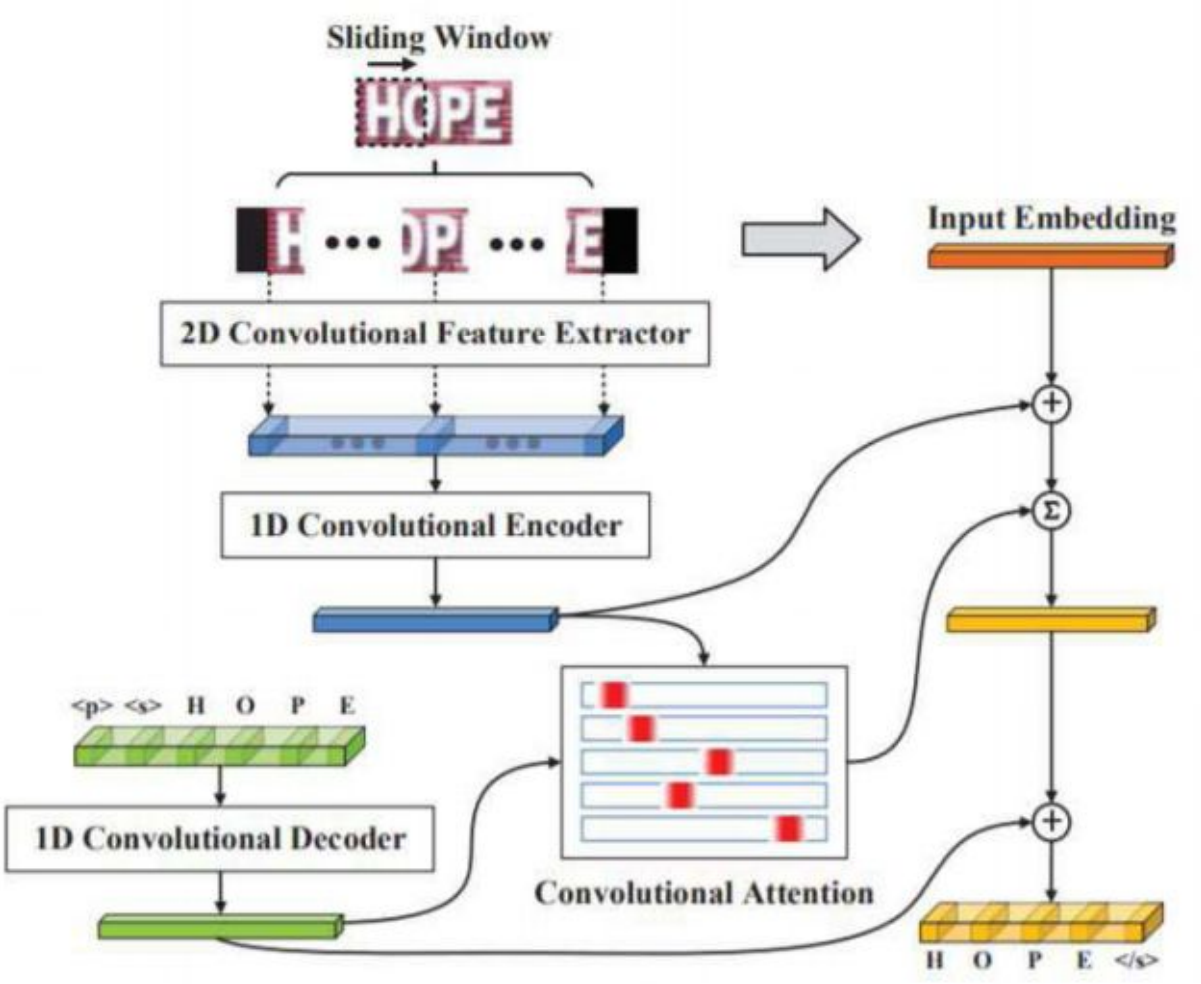

Рис. 1.12. Архітектура нейронної мережі, запропонованої у статті [18]

У роботі [19] пропонується використовувати згорткові нейронні мережі без рекурентних шарів з 2D-Attention. Візуалізація обчислювального графа цієї мережі представлена рисунку 13.

Автори роботи [20] пропонують масштабувати вхідне зображення по ширині n разів, і подавати всі n зображень на вхід згорткової мережі із загальними вагами для всіх зображень (weight sharing). Далі витягнуті згортковою мережею ознаки обробляються механізмом уваги (attention). Архітектура мережі представлена рисунку 14.

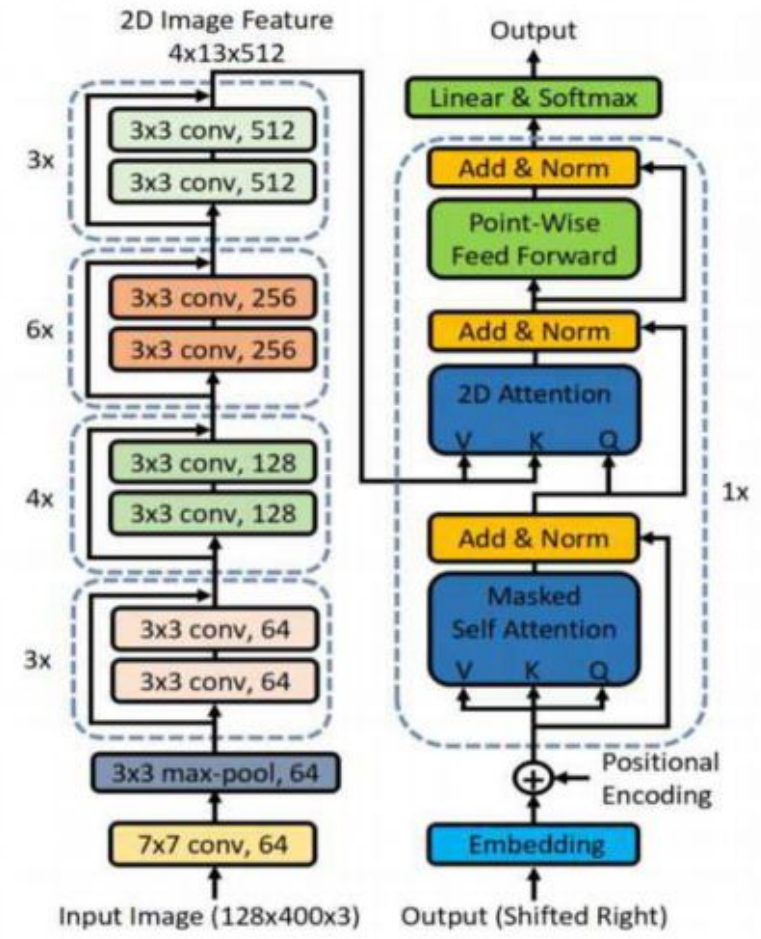

Рис. 1.13. Візуалізація обчислювального графа нейронної мережі запропонованої в [19]

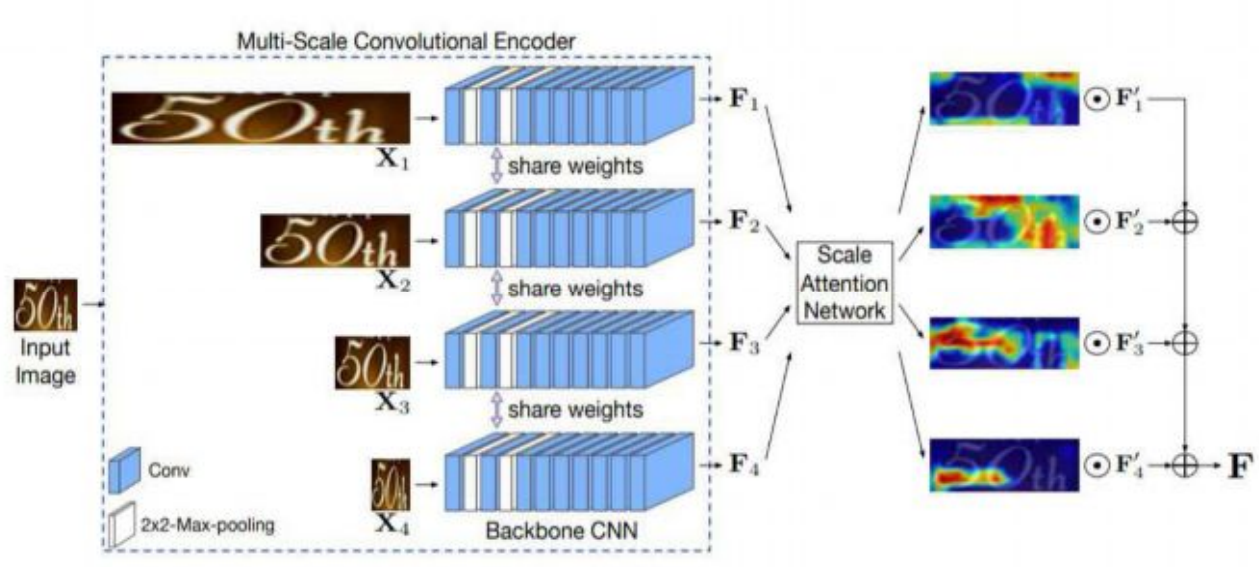

Рис. 1.14. Архітектура мережі з різним масштабуванням вхідного зображення по ширині

У багатьох із оглядових підходів після навчання нейронної мережі декодоване слово перевіряється за заздалегідь заданим обмеженим словником. Це часто дозволяє досягти більшої точності розпізнавання, але робить систему нездатною до розпізнавання слів, які відсутні у словнику.

1.4. Висновки до розділу 1

У розділі проведено огляд літературних джерел та аналіз існуючих на даний момент методів локалізації та розпізнавання тексту. Такі методи поділяються на три категорії: методи на основі символів, методи на основі слів, методи на основі послідовностей. Проте такі методи не дають змоги розпізнавати нечіткі символи для оптичного зчитування зображень, тому постає задача розроблення нового методу що підвищить точність розпізнавання нечітких символів.

РОЗДІЛ 2
ОСНОВНА ЧАСТИНА

2.1. Підхід із застосуванням CRNN-архітектури нейронних мереж

Для вирішення задач класифікації локалізованого тексту на складній графічній сцені пропонується використовувати поєднання згорткової нейронної мережі (CNN), рекурентної нейронної мережі (RNN) та спеціалізованої Connectionist Temporal Classification функції помилки (CTC-loss) [14, 15].

Нижче наведено принцип роботи всіх шарів, що використовуються в даній архітектурі мережі.

2.2. Повно зв'язний шар

Обчислення прямого поширення повнозв'язкового шару нейронної мережі – це лінійне перетворення вектора вхідних даних x=(x1, x2, …, xm) з матрицею ваг W розмірності n x (m+1), де кожен елемент перетвореного вектора згодом передається нелінійній функції активації f:

$$f(Wx) = \begin{pmatrix} f(w_1^T x) \\ \vdots \\ f(w_n^T x) \end{pmatrix}, \qquad (2.1)$$

де x - Вектор вхідних даних;

b - Вектор зсуву шару;

n – кількість нейронів у шарі;

f – функція активації шару;

W – матриця ваг шару.

Насправді часто замість додавання результату лінійного перетворення окремого вектора зміщення нейронів (bias) розмірності n матриця ваг W

конкатенується зі стовпцем з одиниць, що математично рівнозначно.

$$W = \begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} w_{11} & \dots & w_{1m} & 1 \\ \vdots & \ddots & \vdots & \vdots \\ w_{n1} & \dots & w_{nm} & 1 \end{pmatrix}, \tag{2.2}$$

де n – кількість нейронів у шарі;

m – розмірність вхідних даних.

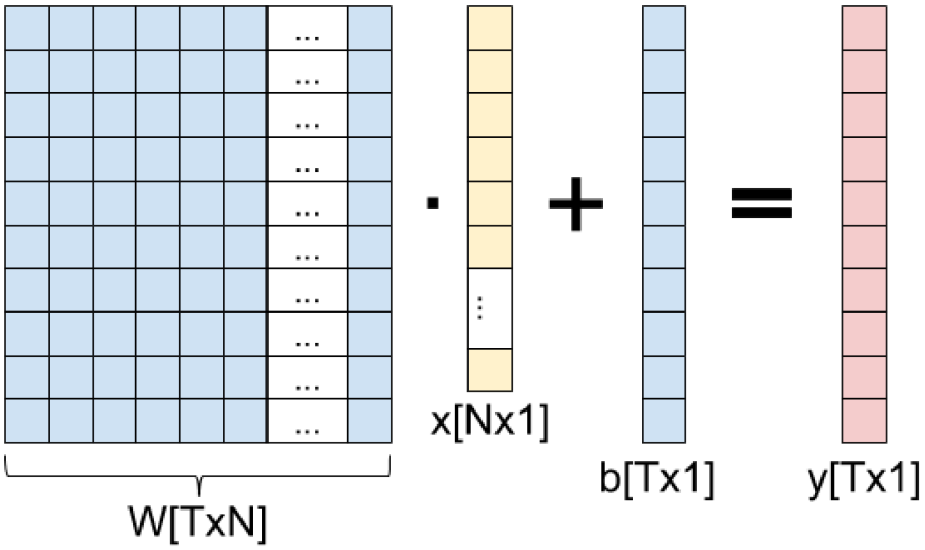

Рис. 2.1. Візуалізація обчислення повнозв'язного шару15

Зворотне поширення помилки по повно зв'язного шару є градієнтом композиції функцій:

$$\frac{\partial t}{\partial x} = \frac{\partial f}{\partial x} * \frac{\partial t}{\partial f}; \tag{2.3}$$

$$\frac{\partial t}{\partial w} = \frac{\partial f}{\partial w} * \frac{\partial t}{\partial f}, \tag{2.4}$$

де t – значення нейронів наступного шару;

f – функція активації даного шару;

x – значення нейронів даного шару;

W – матриця ваг цього шару.

Градієнт по шару x - це помилка, яка згодом передається на попередній шар мережі, а градієнт по матриці ваг W використовується для зміни значень ваг за допомогою оптимізаційного алгоритму.

2.3. Згортковий шар

Нейронна мережа у вигляді загорткового шару є зваженою за допомогою матриці ваг W суму значень попереднього шару. Математично пряме поширення згорткового шару з матрицею ваг W розмірності $(2d+1)$ x $(2d+1)$ для елемента і-го рядка та j-го стовпця вхідної матриці x описується так:

$$y_{i,j} = \sum_{-d \le a,b \le d} W_{a,b} x_{i+a,j+b}, \quad (2.5)$$

де $y_{i,j}$ – результат згортки для елемента з координатами (i, j);

$x_{i+a,\,j+b}$ – елемент вхідної матриці з координатами $(i+a, j+b)$;

$W_{a,b}$ – елемент матриці ваг із координатами (a, b).

Принцип роботи згорткового шару представлений на рисунку 16. Розмір матриці наступного шару розраховується за такою формулою:

$$(w, h) = (mW - kW + 1,\ mH - kH + 1), \quad (2.6)$$

де w - ширина матриці наступного шару;

h - висота матриці наступного шару;

mW - ширина матриці попереднього шару;

mH - висота матриці попереднього шару;

kW - ширина ядра згортки;

kH - висота ядра згортки.

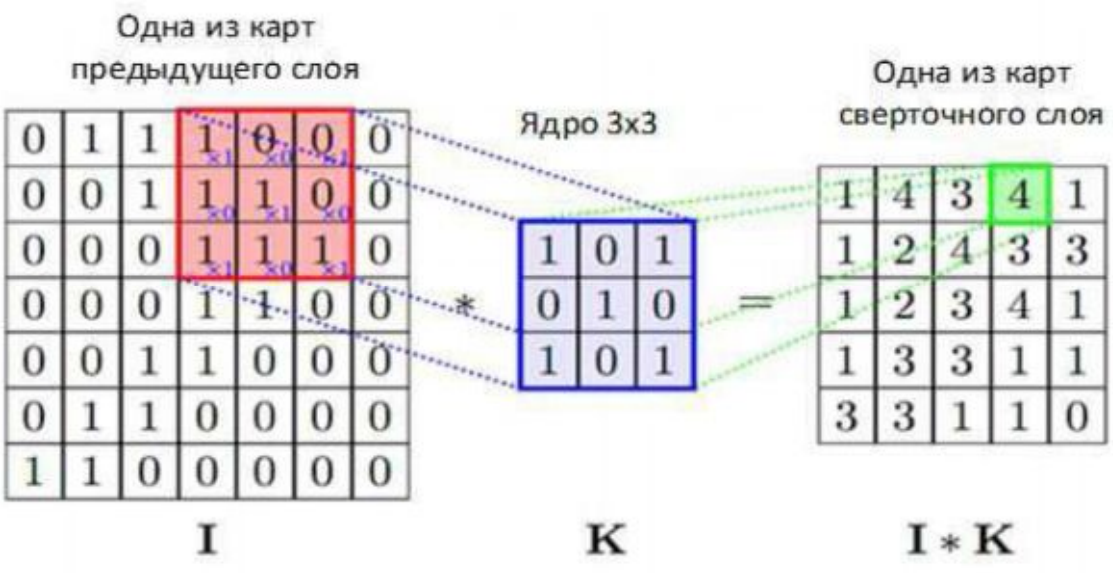

Рис. 2.2. Принцип роботи згорткового шару16

Часто для досягнення тієї ж розмірності, що і на вході згорткового шару використовується т.зв. padding – заповнення країв вихідної матриці нулями до необхідної розмірності. Як правило, на одному згортковому шарі обчислюється відразу кілька карт ознак, кожна з власними вагами щодо кожної вхідної карти ознак.

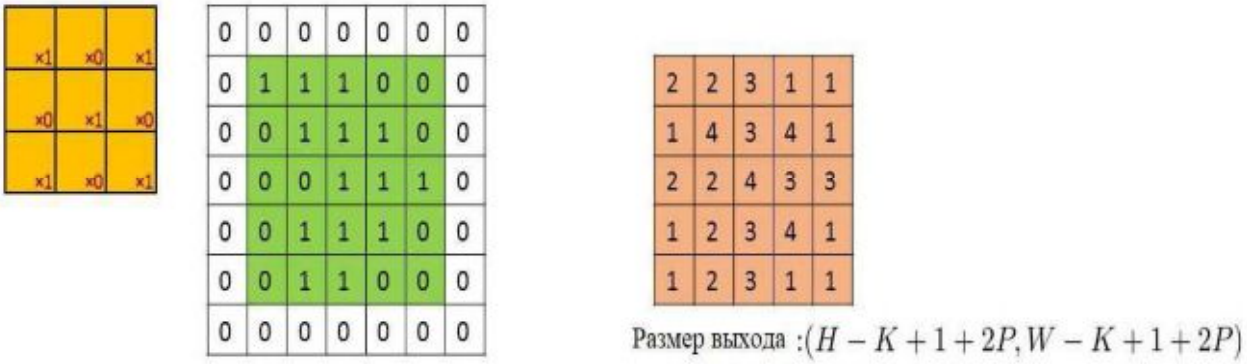

Рис. 2.17. Візуалізація принципу роботи операції згортки з padding

2.4. Шар субдискретизації.

Шар субдискретизації (підвибірковий шар, pooling-шар), як правило, розташовується після згорткового шару, переданого нелінійної функції активації. Призначення цього шару - це зниження розмірності карт ознак з

метою економії обчислювальних потужностей під час навчання мережі та регуляризації. Найбільш поширений варіант субдискретизації - це max-pooling, застосування операції взяття максимуму для плаваючого вікна із заздалегідь заданим кроком та розміром.

Принцип роботи шару субдискретизації з вікном розміру 2x2 і кроком 2 представлений на рисунку 18:

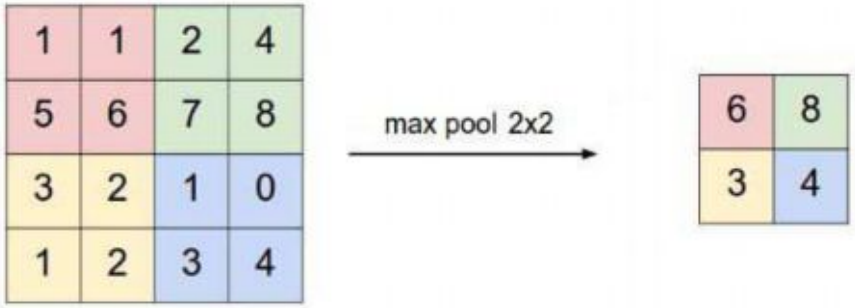

Рис. 2.3. Візуалізація принципу роботи шару субдискретизації 18

Операція max-pooling' а описується такою формулою:

$$y_{i,j} = max_{-d \le a \le d, -d \le b \le d}(x_{i+a,j+b}), \quad (2.7)$$

де x - вихід попереднього шару;

d - розмір вікна субдискретизації;

$y_{i,j}$ - результат застосування max-pooling'а для елемента з координатами (i,j).

2.5. Шар нормалізації з міні-батчів.

Запропонований у статті [21] метод нормалізації з міні-батч використовується зараз у більшості архітектур нейронних мереж. Мережі з використанням нормалізації по міні-батчам значно стійкіші до ініціалізації ваг і менш схильні до перенавчання. Для кожного з нейронів обчислюється математичне очікування μx та стандартне відхилення σx по міні-батчу $x=\{x_1, x_2, \ldots, x_m\}$.

$$\mu_x = \frac{1}{m}\sum_{i=1}^{m} x_i ; \quad (2.8)$$

$$\sigma_x^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_x)^2, \quad (2.9)$$

де m – розмір міні-батчу;

x – безліч значень нейрона по всьому міні-батчу;

Потім дані нормалізуються за допомогою віднімання математичного очікування та поділу на стандартне відхилення.

$$x_i' = \frac{x_i - \mu_x}{\sqrt{\sigma_x^2 + \epsilon}}; \quad (2.10)$$

де ε – додаткова константа, що вводиться для запобігання поділу на нуль.

Обчислюється кінцевий результат з урахуванням додаткового масштабування та зсуву нормалізації.

$$y_i = \gamma x_i + \beta, \quad (2.11)$$

де γ – додатковий оптимізований параметр для масштабування нормалізації кожного елемента;

β – додатковий оптимізований параметр для зсуву нормалізації по кожному елементу.

Також нейронні мережі, в яких використовується нормалізація з міні-батч сходяться на більшості завдань швидше, ніж мережі без неї. Автори оригінальної статті з CRNN [15] використовують 2 шари батч-нормалізації у своїй архітектурі.

2.6. Рекурентний шар.

В архітектурі CRNN ознаки, що витягуються згортковими шарами, є послідовностями, оскільки в тексті, що розпізнається, важливий порядок

слідування символів один за одним. Їх послідовний характер повинен враховуватися під час навчання мережі, і найпоширеніший спосіб враховувати його – використовувати рекурентні шари.

За характером послідовностей завдання машинного навчання умовно поділяються на 5 типів:

- один одного (one to one) – один вхід моделі відповідає одному виходу (наприклад, завдання класифікації образів);

- один до багатьох (one to many) – один вхід моделі відповідає послідовності виходів (наприклад, завдання анотування зображень);

- багато хто до одного (many to many) – послідовність входів відповідає одному виходу (наприклад, завдання аналізу тональності тексту);

- багато хто до багатьох (many to many) - послідовність входів відповідає послідовності виходів (наприклад, машинний переклад);

- багато хто з синхронізацією (many to many) – синхронізована послідовність входів і виходів (наприклад, класифікація відео з маркуванням кожного кадру).

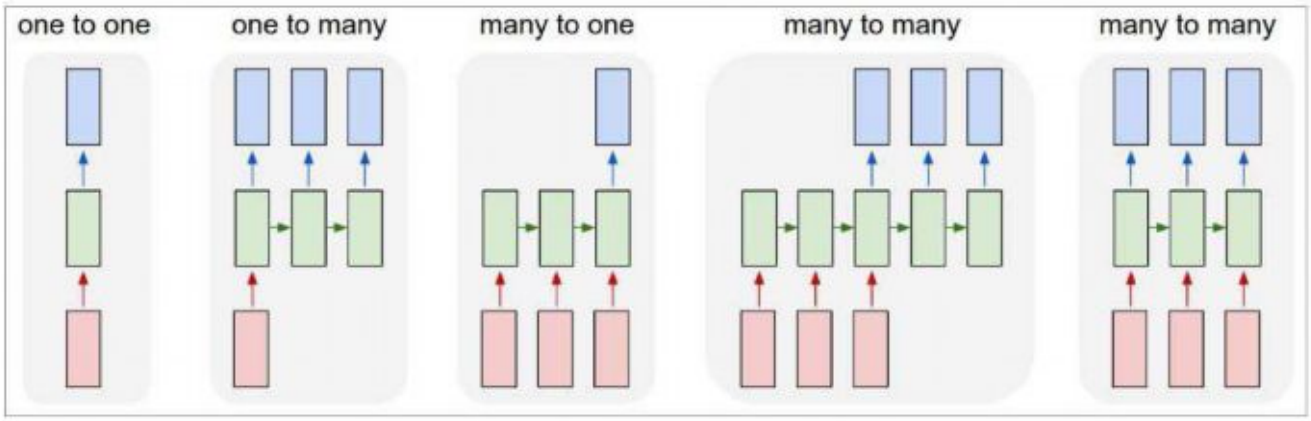

Рис. 2.4. Умовне розподіл завдань машинного навчання за характером послідовностей19

Розв'язана задача відноситься до 4-го типу, в якому послідовність входів відповідає послідовності виходів без взаємної синхронізації. Послідовність ознак, що витягуються згортковими шарами, відповідає змінної довжини послідовності вихідних символів у слові, що розпізнається.

Основний принцип роботи рекурентних шарів – використання значень нейронів, отриманих на попередніх ітераціях навчання мережі. У шарі визначається деякий прихований стан s, який зберігається і передається від попереднього до наступного елементу послідовності.

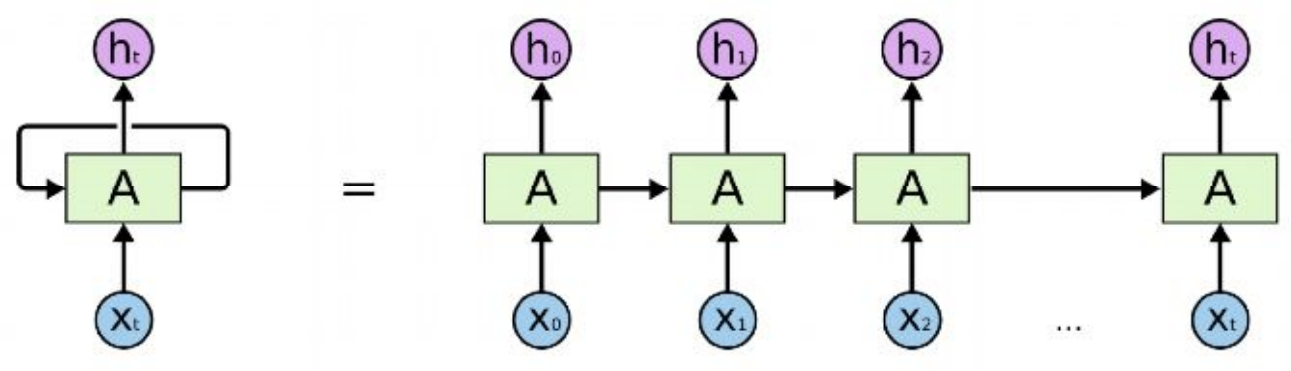

Рис. 2.5. Принцип роботи рекурентного шару та його розгорнуте у часі уявлення 20

Математично класичний рекурентний шар (Vanilla RNN) нейронної мережі описується так

$$s_t = f(Ws_{t-1} + Ux_t + b); \tag{2.12}$$

$$h_t = g(Vs_t + c), \tag{2.13}$$

де x_t – вхідні дані на поточному годинниковому кроці;

s_{t-1} – значення прихованого стану на попередньому тимчасовому кроці;

W – матриця ваг для прихованого стану;

U – матриця ваг для вхідних даних;

b – усунення даних перед нелінійністю;

f – нелінійна функція активації рекурентного шару (зазвичай використовується гіперболічний тангенс, tanh);

s_t – значення прихованого стану на поточному часовому кроці;

V – матриця ваг на виході шару;

c – зсув даних на виході шару;

g – функція не лінійності на виході (наприклад, softmax);

h_t – вихідні значення шару.

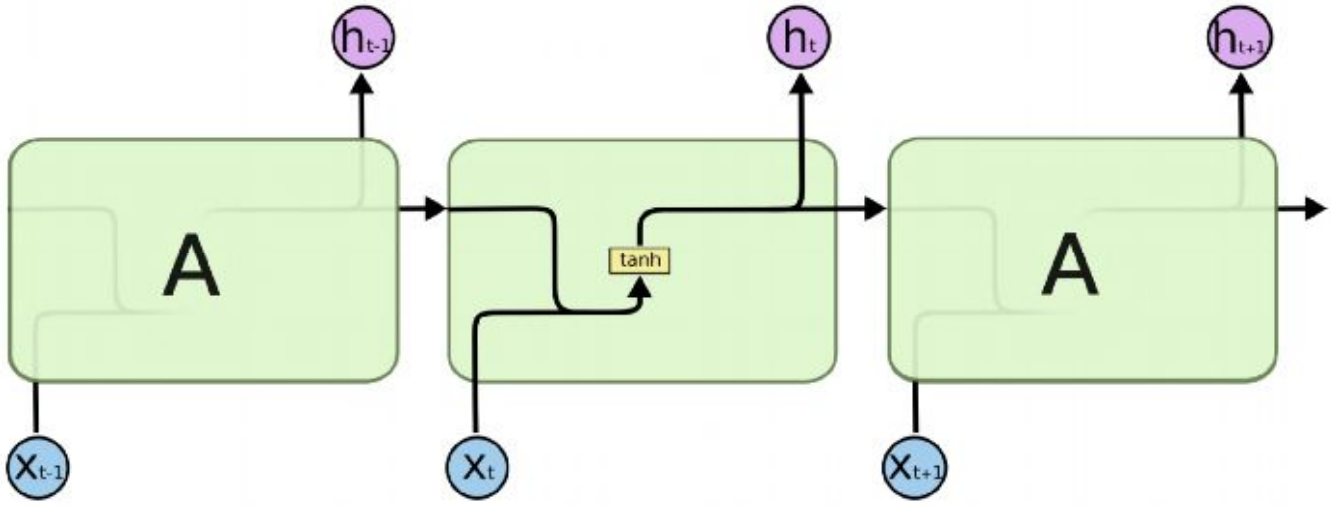

Рис. 2.6. Структура класичного рекурентного шару21

Але класична архітектура рекурентних мереж має кілька проблем. Одна з них – проблема довгих залежностей. При матричному перемноженні інформація про довгострокові залежності на більш ранніх тимчасових кроках постійно перезаписуватиметься, і мережа таким чином матиме більш “коротку” пам'ять і нездатна відтворювати довгострокові залежності. Ця проблема частково вирішується застосуванням складніших архітектур, одна з них – LSTM (Long Short-Term Memory). У мережі LSTM є 3 т.зв. вузла або гейта (gate): забуваючий гейт (forget gate), вхідний гейт (input gate), вихідний гейт (output gate) і сам рекурентний осередок із прихованим станом (cell state).

Забуваючий гейт f_t необхідний контролю даних, які з попередніх часових кроків. Він описується такою формулою:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{2.14}$$

де h_{t-1} – прихований стан на попередньому тимчасовому кроці;

x_t – вхідні дані на поточному часовому кроці;

W_f – матриця ваг забуває;

b_f – зміщення забуваючий гейт;

σ – нелінійна функція активації (сигмоїда).

Вхідний гейт it обробляє вхідні дані для поточного тимчасового кроку та описується формулою:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{2.15}$$

де W_i – матриця ваг вхідного гейту;

b_i – зсув вхідного гейту;

Після обчислення значень забувального та вхідного гейта обчислюється значення-кандидат на зміну стану комірки (candidate cell gate) та стан комірки Ct (cell state) LSTM:

$$C'_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C); \tag{2.16}$$

$$C_t = \mathrm{f_t} * \mathrm{C_{t-1}} + \mathrm{i_t} * \mathrm{C'_t}, \tag{2.17}$$

де W_C – матриця ваг стану осередку;

b_C – усунення стану осередку;

tanh – нелінійна функція активації (гіперболічний тангенс).

У вихідному гейте обчислюється прихований стан в момент часу h_t і значення вихідного гейта o_t

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o); \tag{2.18}$$

$$h_t = \mathrm{o_t} * \tanh(\mathrm{C_t}), \tag{2.19}$$

де W_o – матриця ваг вихідного гейту;

b_o – зсув вихідного гейту.

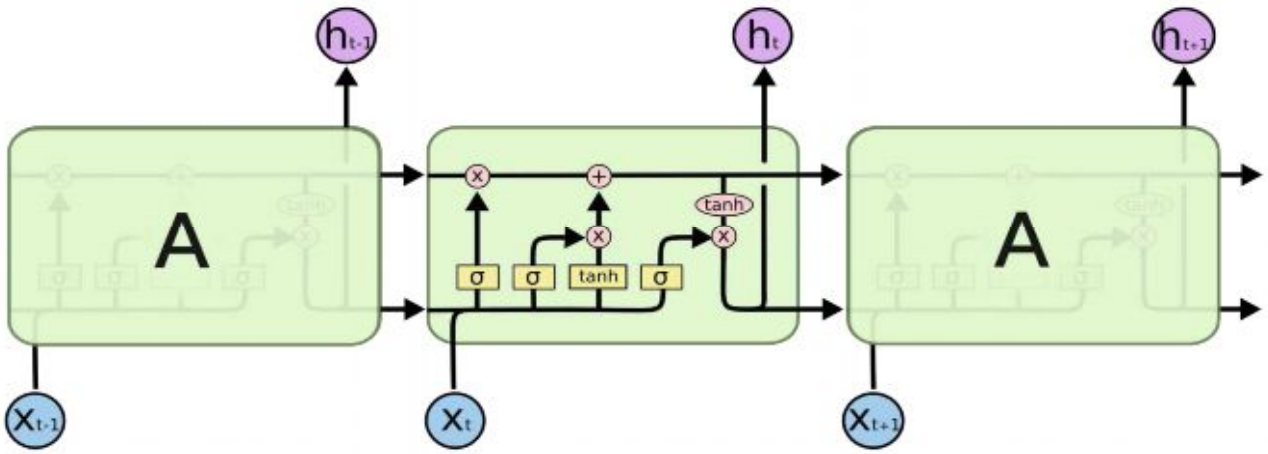

Рис. 2.7. Структура рекурентного LSTM (long short-term memory) шару22

Оскільки при подачі зображення на вхід мережі відоме як початок послідовності, так і її кінець – використовуються двонаправлені (bidirectional) LSTM шари. Принцип роботи bidirectional LSTM такий – задаються 2 незалежних LSTM шару, у одному з яких прихований стан передається від початку послідовності до її кінця, а іншому навпаки – від кінця послідовності до її початку, після чого ці 2 шару конкатенуються друг з одним. Це дозволяє мережі в процесі навчання враховувати для кожного символу як інформацію про попередні символи, а й інформацію про наступні.

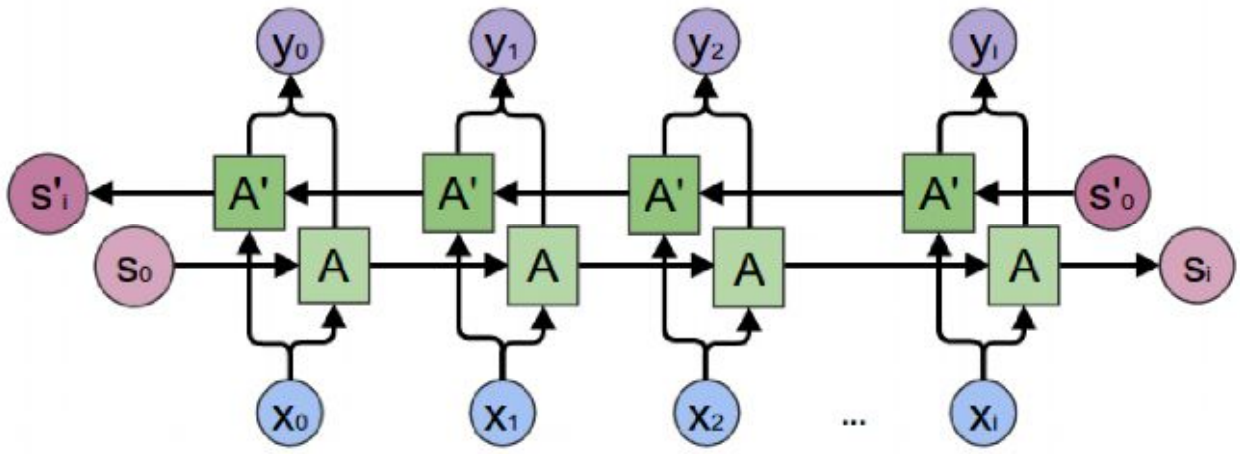

Рис. 2.8. Принцип роботи двонаправленого (bidirectional) LSTM шару23

2.7. CTC LOSS

Перш ніж описати безпосередньо CTC (Connectionist temporal classification) loss, розглянемо повністю архітектуру CRNN.

Вхідне зображення подається на вхід шарам згорткової нейронної мережі [22, 34] для подальшого отримання ознак.

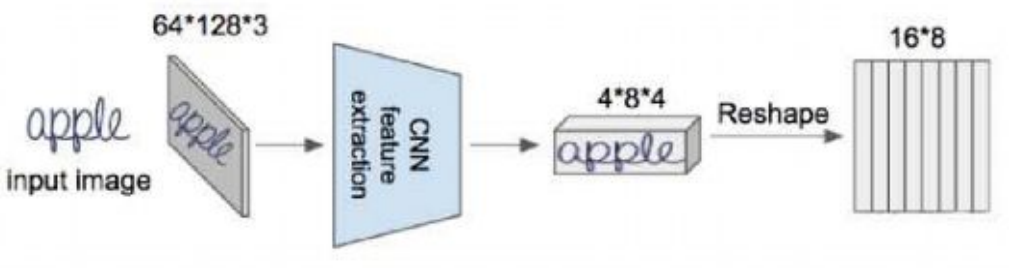

Рис. 2.9. Приклад отримання ознак за допомогою згорткової нейронної мережі24

Вихід згорткової мережі є тривимірним масивом, який необхідно перетворити до двомірного, друга розмірність якого повинна дорівнювати максимальній кількості символів у вхідному зображенні.

Після перетворення кожен стовпець матриці подається на вхід відповідної LSTM-клітини [23]. Обробка одержаних ознак рекурентним шаром візуалізовано рисунку 25.

Рекурентний шар необхідний представлення вхідних ознак як послідовності і розпізнавання мережею символів у тому порядку, де вони перебувають у шуканому тексті.

Після обробки рекурентним шаром кожен вектор передається повнозв'язного шару та активаційної функції softmax [22]. Розмірність повнозв'язного шару дорівнює довжині масиву всіх можливих символів, що розпізнаються, з додаванням blank-маркера.

Цей маркер алгоритм CTC-помилки використовує позначення відсутності символу у тексті, т.к. Більшість всіх поданих на вхід послідовностей знаків буде по довжині менше максимально можливої. Softmax-функція представляє

вхідний вектор у вигляді вектора розподілу ймовірностей для всіх можливих символів, включаючи і маркер відсутності символу (blank) [15]. Приклад роботи декодуючого алгоритму наведено рисунку 26.

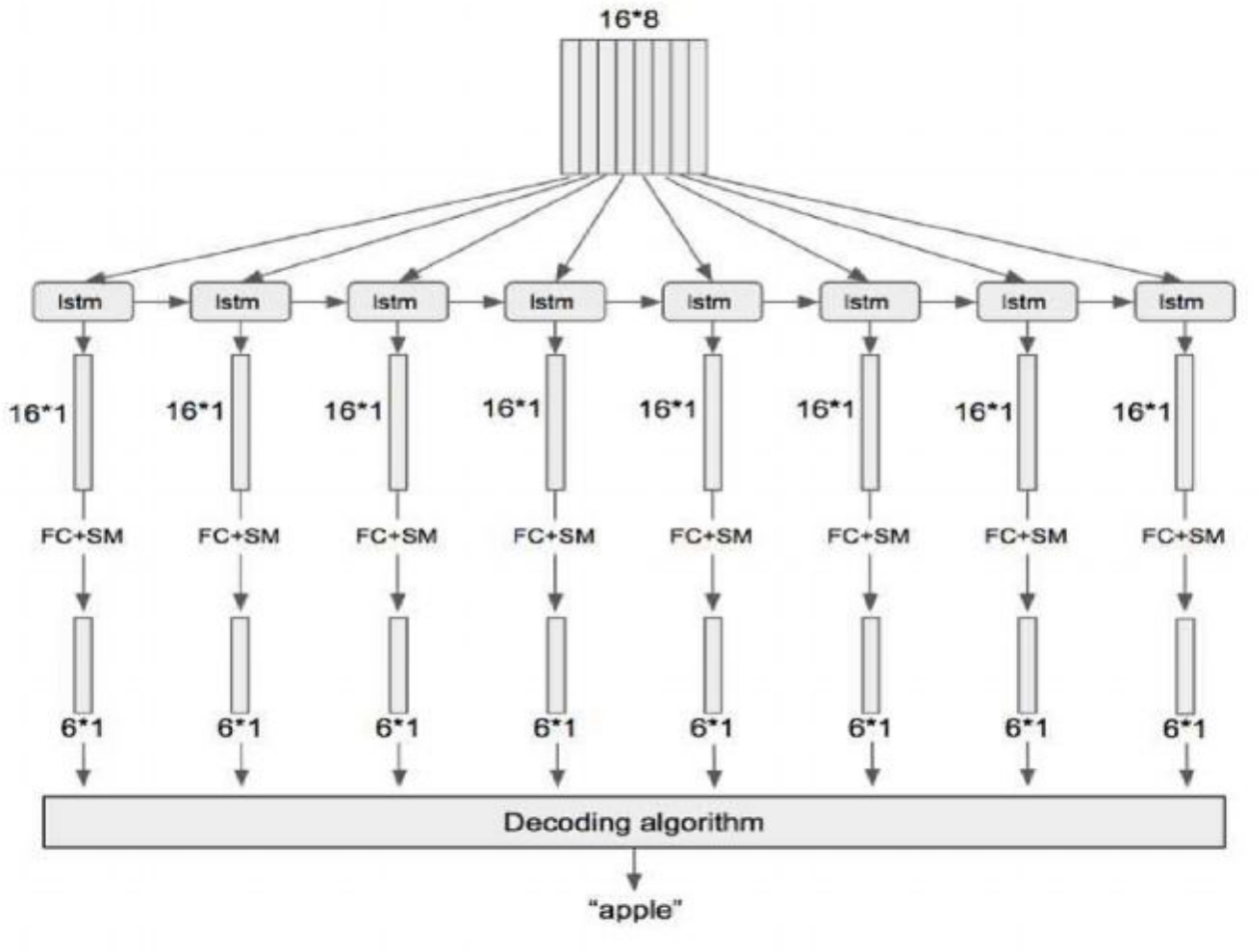

Рис. 2.10. Обробка одержаних ознак рекурентним шаром25

Вихід є матрицею з розподіленими функцією softmax по стовпцях ймовірностями, де кількість стовпців – максимально можлива довжина слова, а кількість рядків – довжина "алфавіту" L + 1(blank-маркер). Дана матриця подається на вхід алгоритму, що декодує, який спочатку видаляє всі повторювані символи, а потім blank-маркери. На рисунку нижче наведено приклад декодування вхідного шляху.

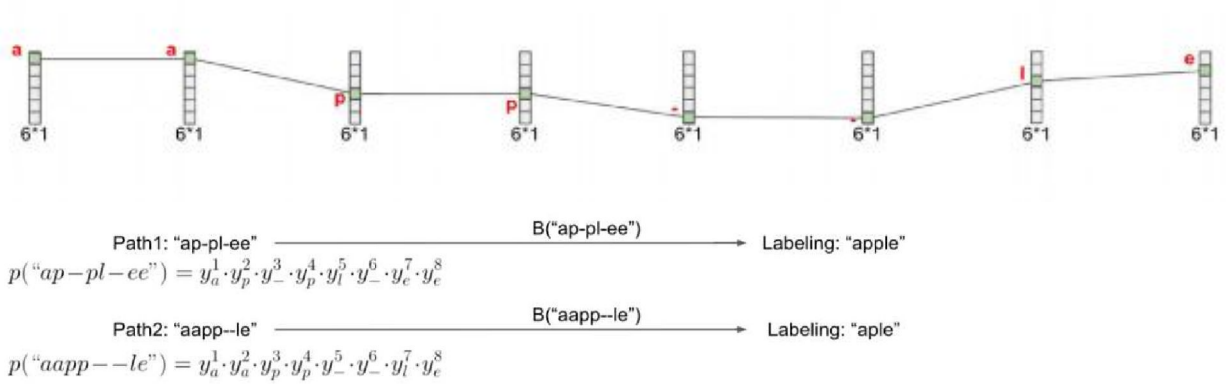

Рис. 2.11. Візуалізація роботи алгоритму декодування27

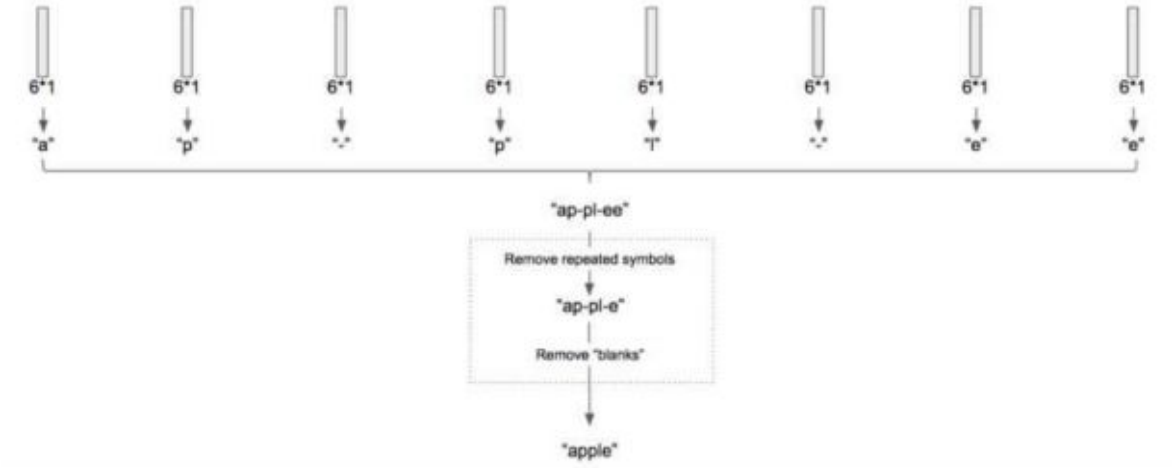

Рис. 2.12. Приклад роботи алгоритму декодування26

Підсумковий алфавіт - це безліч усіх можливих символів у слові та blank-маркер:

$$L' = L \cup \{\text{blank}\}; \tag{2.20}$$

Імовірність шляху Y дорівнює добутку ймовірностей всіх активованих осередків кожного стовпця матриці:

$$p(Y) = \prod_{t=1}^{T} y_{Y_t}^t, \quad \forall Y \in L'; \tag{2.21}$$

Приклад розрахунку ймовірності колії “ap-pl-ee”:

$$p(\text{"ap-pl-ee"}) = y_a^1 * y_p^2 * y_-^3 * y_p^4 * y_l^5 * y_-^6 * y_e^7 * y_e^8; \quad (2.21)$$

Одне слово завжди відповідає безліч шляхів, й у розрахунку ймовірності слова необхідно розрахувати суму ймовірностей з усіх можливих шляхах [14]. Сама функція помилки нагадує бінарну крос ентропію, тільки як ймовірність класу тут використовується ймовірність слова.

$$\text{CTC loss} = -\ln(p(\text{"apple"})); \quad (2.23)$$

Для прикладу (в алфавіті 6 символів, максимальна кількість символів - 8) це 6^8=1679616 можливих шляхів. На практиці ж довжина алфавіту та максимальна кількість символів часом у рази більша, тому для знаходження ймовірності слова набагато ефективніше використовувати динамічне програмування.

Створюється нова матриця, подібна до отриманої на виході softmax-шару.

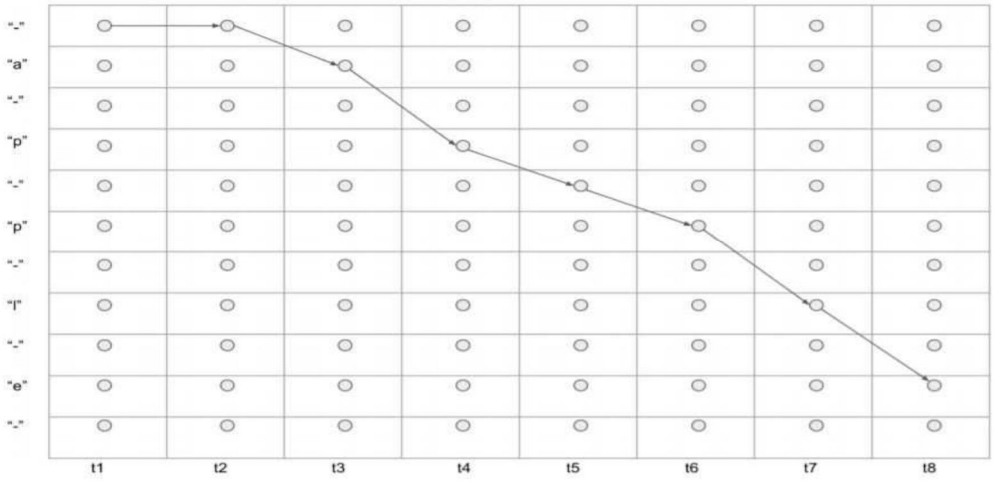

Рис. 2.13. Нова матриця для побудови безлічі всіх можливих шляхів28

З її допомогою можна побудувати безліч всіх можливих шляхів для потрібного слова, а також розрахувати коефіцієнт α. Розрахунок коефіцієнта має вигляд:

$$\alpha_t(s) = \left(\alpha_{t-1}(s) + \alpha_{t-1}(s-1)\right) * y^t_{seq(s)}; \quad (2.24)$$

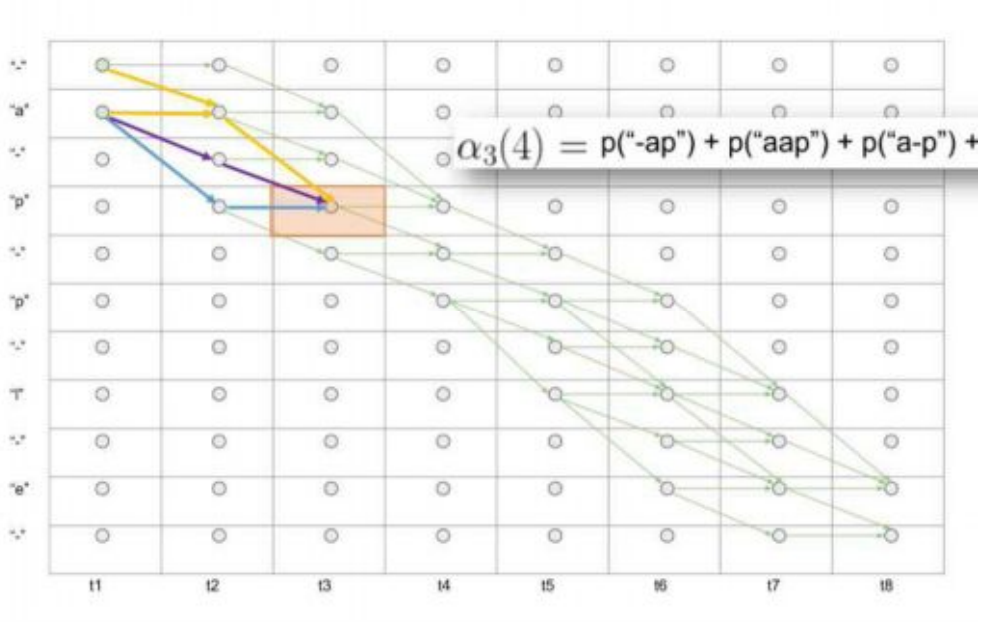

Рис. 2.14. Візуалізація всіх можливих шляхів для слова “apple”29

Для кожного елемента t-го стовпця та s-го рядка матриці розраховується сума всіх шляхів, що ведуть до цього елемента, після чого множиться на ймовірність самого елемента. Ця операція повторюється до останнього стовпця, значення якого підсумовуються та передаються функції помилки.

Даний підхід дозволяє ефективно обчислити суму всіх можливих шляхів до необхідного слова та отримати його ймовірність, що необхідно для навчання нейронної мережі.

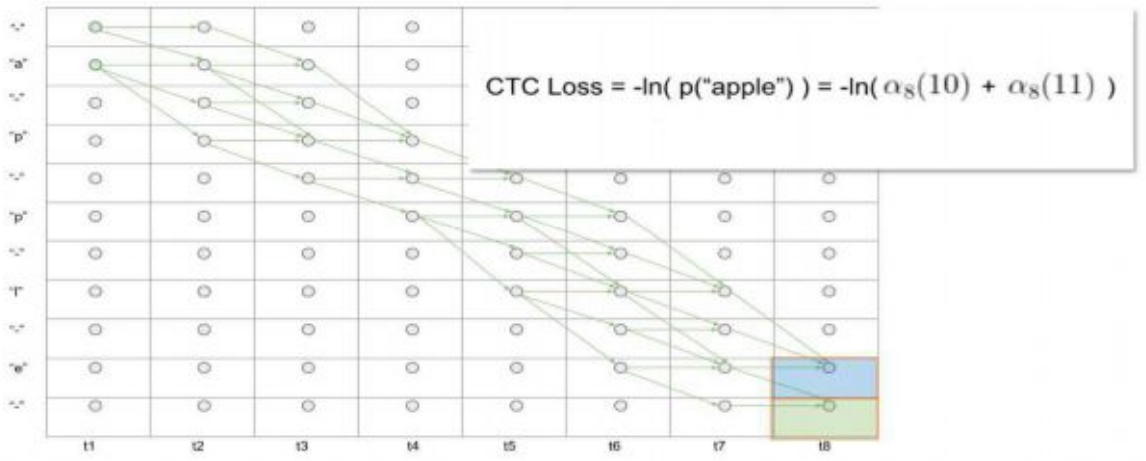

Рис. 2.15. Розрахунок суми всіх шляхів із застосуванням динамічного програмування30

Коефіцієнт вводиться для прямого поширення, для зворотного використовується інший коефіцієнт – β.

$$\beta_t(s) = \left(\beta_{t+1}(s) + \beta_{t+1}(s+1)\right) * y^t_{seq(s)}; \quad (2.25)$$

Коефіцієнт β для елемента матриці розраховується для всіх шляхів, що виходять з даного елемента. Так, поділивши твір α_s^t і β_s^t на можливість елемента y_s^t і просумувавши вздовж *t*-го стовпця матриці, отримаємо суму всіх можливих шляхів:

$$p(\text{"apple"}) = \sum_{s=1}^{seq} \frac{\alpha_s^t * \beta_s^t}{y_s^t}; \quad (2.26)$$

Для навчання мережі градієнтним спуском необхідно продиференціювати функцію помилки з усіх виходів мережі y_k^t:

$$\frac{\partial(-\ln(p(\text{"apple"}))}{\partial y_k^t} = -\frac{1}{p(\text{"apple"})} * \frac{\partial p(\text{"apple"})}{\partial y_k^t}; \quad (2.26)$$

Розглядаються лише шляхи, що йдуть через символ *k* у момент *t*:

$$\frac{\partial p(\text{"apple"})}{\partial y_k^t} = -\frac{1}{{y_k^t}^2} * \sum_{seq(s)=k} \alpha_t(s) * \beta_t(s); \quad (2.26)$$

Таким чином, архітектура CRNN є повністю диференційованою системою (end-to-end) і є одним обчислювальним графом.

Для поліпшення точності розпізнавання, декодоване слово часто перевіряють за заданим словником. У цій роботі для цього між декодованим словом і кожним словом зі словника обчислюється відстань Левенштейна, після чого вибирається кілька слів з найменшою відстанню.

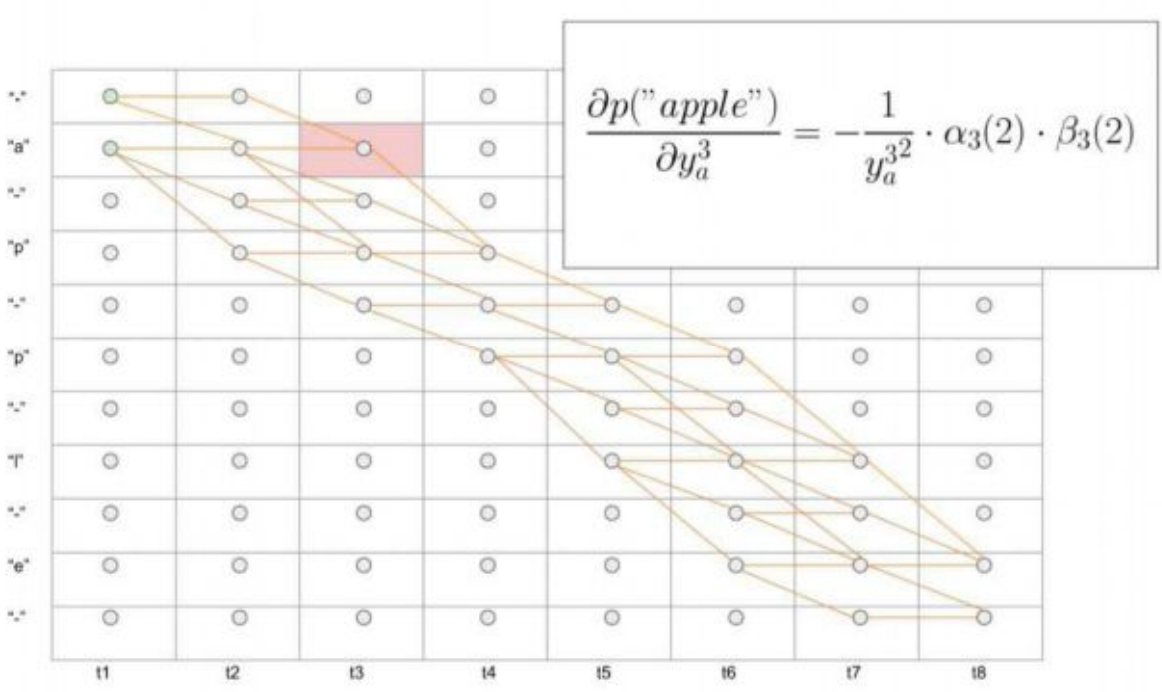

Рис. 2.16. Приклад диференціювання за символом "a" в момент "З"31

2.5. Висновки до розділу 2

У розділі розглянуто підходи до застосуванням архітектури нейронних мереж для вирішення задач класифікації локалізованого тексту на складній графічній сцені. Зокрема, розглянуто згорткову нейронну мережу (CNN), рекурентну нейронну мережу (RNN) та спеціалізовану функцію помилки (CTC-loss).

РОЗДІ З
НАУКОВО-ДОСЛІДНА ЧАСТИНА

3.1. Метод розпізнавання нечітких символів

Оскільки з існуючих на даний момент наборів даних для цього завдання немає наборів з кількістю зображень, достатньою для якісного навчання мережі, було прийнято рішення використовувати набір Synth 90k, запропонований та розроблений у статті [2]. Набір даних є програмно згенерованими зображеннями в градаціях сірого змінної ширини і висотою в 31 піксель. У тренувальній частині набору міститься понад 7 мільйонів зображень із 90000 унікальними англійськими словами. Приклади зображень представлені рисунку 3.1.

Рис. 3.1. Приклади зображень із набору даних Synth 90k

Для оцінки якості натренованої мережі використовувалися два набори даних. Перший набір – IIIT 5K. Даний набір є локалізованим текстом на фотографіях магазинних вітрин, рекламних щитів, оголошень, вуличних знаків і т.п. У наборі міститься 2000 тренувальних та 3000 тестових зображень. Приклади зображень представлені рисунку 3.2.

Рис. 3.2. Приклади зображень із набору даних IIIT5K

Другий набір даних – SVT (Street View Text). У наборі даних за аналогією з IIIT 5K-Word переважають зображення з текстом на магазинних вітринах, рекламних щитах, вивісках різних закладів. Набір даних містить 249 зображень графічних сцен та 647 зображень локалізованого тексту. Під тренувальну вибірку виділено 80% зображень, під тестову – 20%. Приклади зображень представлені рисунку 3.3.

Рис. 3.3. Приклади зображень із набору даних SVT (Street View Text)

3.2. Експериментальні дослідження

Було розроблено програму на ЕОМ з реалізацією CRNN+CTC-loss архітектури мовою програмування Python з використанням фреймворку Tensorflow. Вхідне зображення наводилося до розміру 31×100 пікселів у градаціях сірого, після чого подавалося на вхід нейронної мережі разом із відповідним словом. Розроблена нейронна мережа має таку архітектуру:

- 1-й згортковий шар, ReLU, ядро 3x3 (31, 100, 64);
- 1-й max-pooling шар, ядро 2x2, крок 2x2, (16, 50, 64);
- 2-й згортковий шар, ReLU, ядро 3x3 (16, 50, 128);
- 1-й шар батч-нормалізації;
- 2-й max-pooling шар, ядро 2x2, крок 2x2, (8, 25, 128);
- 3-й згортковий шар, ReLU, ядро 3x3 (8, 25, 256);
- 2-й шар батч-нормалізації;
- 4-й згортковий шар, ReLU, ядро 3x3 (8, 25, 256);
- 3-й max-pooling шар, ядро 2x2, крок 1x2 (8, 13, 256);
- 5-й згортковий шар, ReLU, ядро 3x3 (8, 13, 512);
- 3-й шар батч-нормалізації;
- 6-й згортковий шар, ReLU, ядро 3x3 (8, 13, 512);

- 4-й max-pooling шар, ядро 2x2, крок 1x2, (8, 7, 512);

- 7-й згортковий шар, ReLU, ядро 3x3 (8, 7, 512);

- 1-й bidirectional-LSTM шар (256, 256);

- 2-й bidirectional-LSTM шар, (256, 256);

- 1 повний зв'язний шар, функція активації Softmax;

– CTC-loss.

Як оптимізатор обраний Adam з кроком навчання 0.001, розмір вхідного пакета (batch), для якого відбувається одноразове коригування ваги - 64. Довжина алфавіту - 43 символи, максимально можлива довжина слова -16 символів. Вага мережі до навчання ініціалізується нормально розподіленими значеннями з математичним очікуванням, рівним 0 і стандартним відхиленням, рівним 0.1. Для крайніх пікселів у згорткових шарах використовується автоматичне заповнення значень нулями для отримання на виході шару тієї ж розмірності, що і на вході. Нейронна мережа навчалася 1.5 епохи (7224612 зображень із Synth 90k) на графічному процесорі NVIDIA GeForce GTX 650, процес навчання зайняв 170 годин (7 днів).

Графік зміни помилки на тренувальних даних (Synth 90k), SVT та IIIT5K наведено на рисунку 3.4.

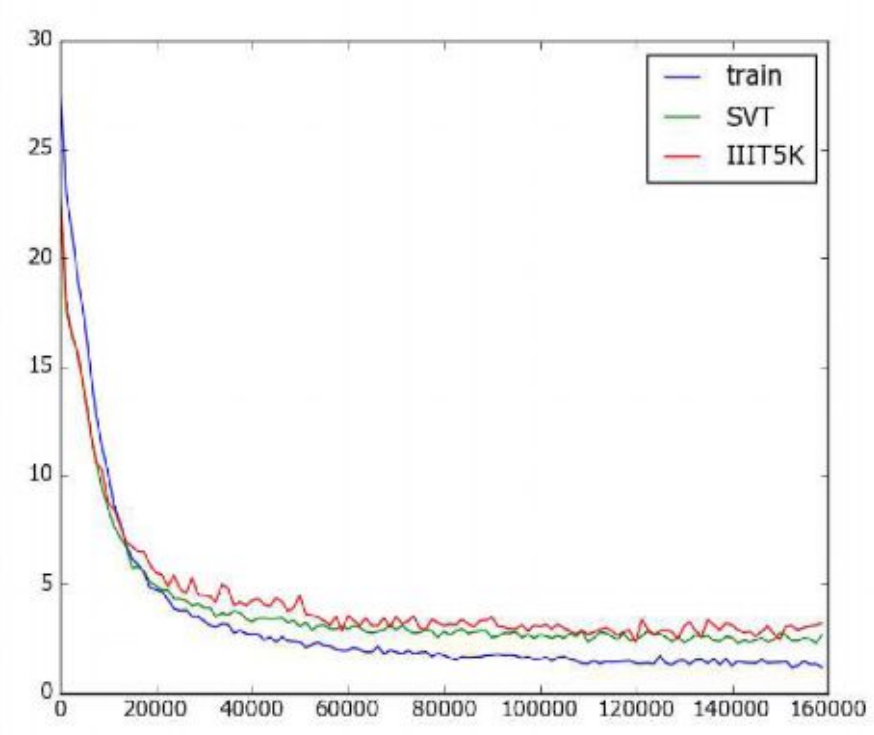

Рис. 3.4. Графік зміни помилки нейронної мережі

Графік зміни відстані Левенштейна між передбаченими та еталонними словами наведено на рисунку 3.5.

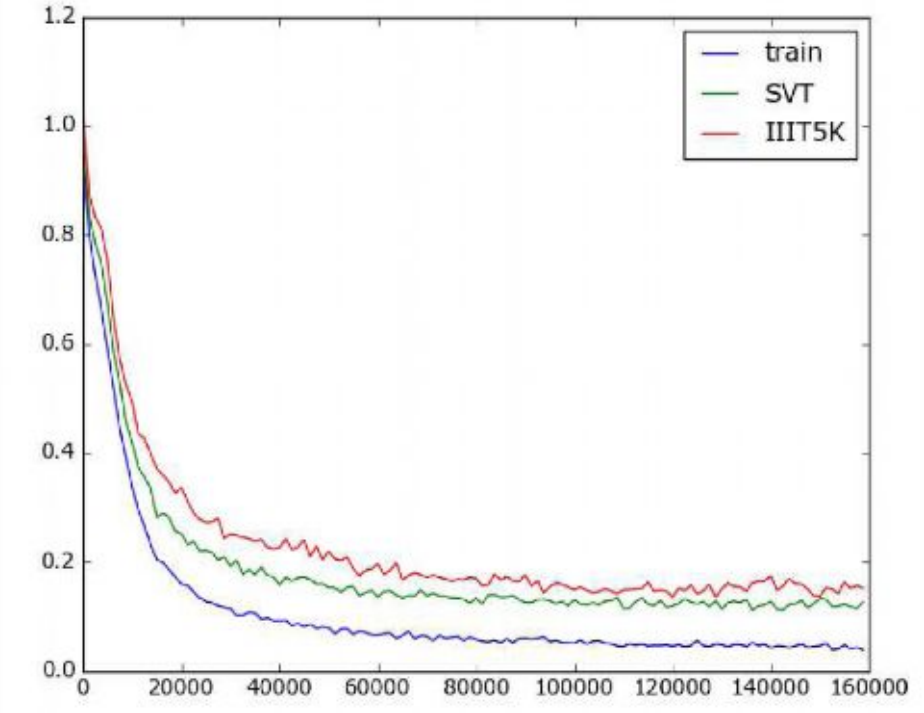

Рис. 3.5. Графік зміни відстані Левенштейна

Графік зміни точності (стосунки правильно передбачених слів до загального числа) наведено на рисунку 3.6.

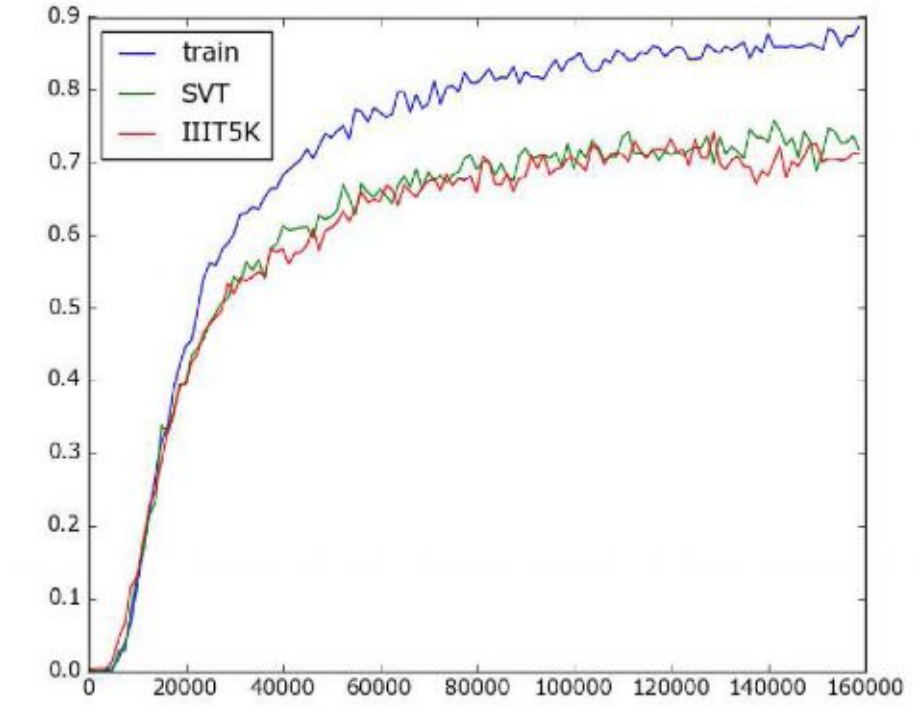

Рис. 3.6. Графік зміни точності

У таблиці 1 наведено результати експерименту та порівняння з результатами інших підходів. Система показує результати, близькі до state-of-the-art рішень, при цьому навчаючись швидше за CRNN з [4] завдяки додатковим шарам батч-нормалізації.

Таблиця 3.1

Результати експерименту та порівняння підсумкової точності з іншими підходами на двох наборах даних

Метод	IIIT5K			SVT	
	50 слів	1000 слів	без словника	50 слів	без словника
ABBYY	24.3	-	-	35.0	-
Wang и др.	-	-	-	57.0	-
Mishra и др.	64.1	57.5	-	73.2	-
Wang и др.	-	-	-	70.0	-
Rodrguez-Serrano и др.	76.1	57.4	-	70.0	-
Bissacco и др. [5]	-	-	-	-	78.0
Jaderberg и др.	-	-	-	86.1	-
Jaderberg и др. [2]	95.5	89.6	-	93.2	71.7
Jaderberg и др.	97.1	92.7	-	95.4	80.7
Baoguang Shi и др. (CRNN) [4]	97.6	94.4	78.2	96.4	80.8
R^2AM	96.8	94.4	78.4	-	80.7
CA-FCN	98.9	99.8	92.0	-	82.1
FACLSTM	99.5	98.6	90.5	-	82.2
Запропонований підхід	96.7	93.2	74.0	95.3	76.2

3.3. Висновки до розділу 3

В розділі описано програмне середовище, яке розроблено на мові програмування Python та проведено експериментальні дослідження з розпізнавання нечітких символів. Складено таблицю значень точності розпізнавання тексту. За підсумками експерименту на наборі даних SVT без використання словника було досягнуто точності 76.2%, з використанням словника розміром 50 слів – 95.3%.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Охорона праці

Пожежна сигналізація і зв'язок. Засоби гасіння пожеж. Протипожежне водопостачання. Первинні засоби пожежогасіння. Автоматичні засоби пожежогасіння на об'єктах галузі.

Швидке виявлення та сигналізація про виникнення пожежі, своєчасний виклик пожежних підрозділів та оповіщення про пожежу людей, що перебувають у зоні можливої небезпеки, дозволяє швидко локалізувати осередки пожежі, здійснити евакуацію та вжити необхідних заходів щодо гасіння пожежі. Тому підприємства повинні бути забезпечені засобами зв'язку та системами пожежної сигналізації та оповіщення.

Для передачі повідомлення про пожежу в будь-який час доби можна використовувати телефони спеціального та загального призначення, радіозв'язок, централізовані установки пожежної сигналізації. Системи оповіщення про пожежу повинні забезпечувати у відповідності з розробленими планами евакуації передачу сигналів оповіщення одночасно по всьому будинку (споруді) а при необхідності - послідовно або вибірково в окремі його частини (поверхи секції тощо). Кількість оповіщувачів (динаміків), їх розміщення та потужність повинні забезпечити необхідну чутність у всіх місцях перебування людей Для передачі текстів оповіщення та керування евакуацією допускається використовувати внутрішні радіотрансляційні мережі. Приміщення, з якого здійснюється керування системою пожежного оповіщення, належить розміщувати на нижніх поверхах будівель, біля входу на сходові клітки, у місцях з цілодобовим перебуванням чергового персоналу.

Найбільш швидким та надійним засобом виявлення ознак займання та сигналізації про пожежу вважається автоматична установка пожежної сигналізації (АУПС), яка повинна працювати цілодобово. Залежно від схеми

з'єднання розрізняють променеві (радіальні) та кільцеві АУПС (рис. 4.1). Принцип роботи АУПС полягає в наступному: при спрацюванні хоча б одного зі сповіщувачів на приймально-контрольний прилад надходить сигнал "Пожежа".

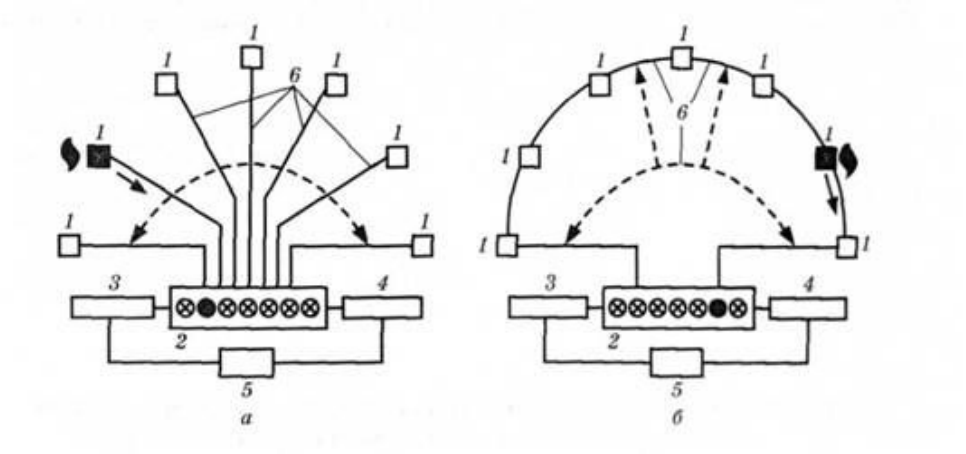

Рис. 4.1. Схеми променевого (а) та кільцевого (б) з'єднання в АУПС**:**
1 - сповіщувачі; 2 - приймально-контрольний прилад; 3 - блок живлення від електромережі; 4 - блок аварійного живлення; 5 - система перемикання живлення; 6 - з'єднувальні проводи

Засоби гасіння пожежі поділяють на первинні, автоматичні і спеціальні.

До *первинних засобів гасіння пожежі* належать пожежні відра і діжки з водою, ломовий інструмент (ніж, сокира, гак, лом, багор - розміщуються на пожежному щиті), ящики і відро з піском, совки і лопати, протипожежна тканина, ручні насоси, пожежні крани внутрішнього водопроводу з рукавами і стволами, ручні вогнегасники усіх типів.

Розміщують їх на спеціальних щитах. Щити встановлюють з таким розрахунком, щоб до найдальшої будівлі було не більше 100м, а від сховищ із вогненебезпечними матеріалами — не більше 50 м, або з розрахунку — один щит на 5000 м^2 Фарбують їх у сигнальний червоний колір, а написи на них та на щитах роблять контрастним білим кольором.

Необхідну кількість первинних засобів пожежогасіння визначають окремо для кожного поверху та приміщення. Вибір типу і визначення потрібної кількості вогнегасників здійснюється залежно від їх вогнегасної спроможності.

Заклади охорони здоров’я повинні мати на кожному поверсі не менше двох переносних вогнегасників. Переносні вогнегасники необхідно розміщувати шляхом: навішування на вертикальні конструкції на висоті не більше 1,5 м від рівня підлоги до нижнього торця вогнегасника і на відстані від дверей, достатній для її повного відчинення; встановлювання в пожежні шафи поруч із пожежними кранами, у спеціальні тумби або на пожежні щити (стенди).

Автоматичні засоби гасіння пожежі бувають різні: повітряно-пінні, газові, порошкові, водяні. Автоматичні установки при виникненні пожежі приводяться в дію відповідним давачем (сповіщувачем) або спонукальним пристроєм, а напівавтоматичні та ручні - людиною. Зараз найширше застосовуються автоматичні установки пожежогасіння, призначені для виявлення осередку пожежі, забезпечення подавання і випускання вогнегасної речовини в захищуване приміщення та оповіщення про пожежу.

Спеціальні засоби гасіння пожежі включають обладнання пожежних частин: пожежні машини і насоси, гідранти, піногенератори різних типів та установки гасіння пожежі.

4.2. Безпека в надзвичайних ситуаціях

Медичний захист населення у надзвичайних ситуаціях.

Серед способів захисту населення у надзвичайних ситуаціях особливе місце займає медичний захист. Виходячи з досвіду, надзвичайні ситуації, як правило, призводять до масової загибелі людей та їх ураження. Для зменшення ступеня ураження необхідно приймати невідкладні заходи щодо надання медичної допомоги потерпілим, відповідно до вимог статті 36 Кодексу цивільного захисту України, Закону України «Про екстрену медичну допомогу».

Наданням цієї допомоги займається медична служба цивільного захисту (ЦЗ), яка організовується на базі Міністерства охорони здоров’я України та його структурних підрозділів і призначена для медичного забезпечення

населення, що постраждало внаслідок стихійного лиха, аварій та катастроф. Вона виконує такі основні завдання:

— своєчасне надання потерпілому населенню усіх видів медичної допомоги та лікування потерпілих з метою їх повного одужання;

— попередження виникнення і розповсюдження серед населення масових інфекційних захворювань;

— забезпечення санітарного благополуччя населення та виключення несприятливих санітарних наслідків виробничих аварій та стихійних лих.

Ці завдання вирішуються шляхом проведення комплексу організаційних, лікувально-профілактичних, лікувально-евакуаційних, санітарно-гігієнічних і протиепідемічних заходів. Основними з них є:

— підготовка органів та установ охорони здоров'я до роботи в умовах великих виробничих аварій та стихійних лих;

— організація і підготовка пересувних медичних формувань для проведення рятувальних робіт, а також підготовка медичних установ до лікування потерпілих і хворих;

— організація і проведення лікарняно-евакуаційних, санітарно-гігієнічних та протиепідемічних заходів в осередках ураження і на етапах медичної евакуації;

— навчання медичного персоналу з медичних питань;

— розроблення планів підготовки органів і об'єктів охорони здоров'я до виконання заходів ЦЗ;

— організація забезпечення формувань та установ медичної служби медичним, господарським і спеціальним майном;

— навчання населення вмінню надавати само- та взаємодопомогу при отриманні різних травм.

При виникненні надзвичайних ситуацій, незалежно від їх масштабу, як правило, першими для надання медичної допомоги до осередку катастрофи прибувають і починають працювати, згідно із своїми функціональними обов'язками, бригади швидкої медичної допомоги (ШМД), які обслуговують населення даної території. Вони надають першу лікарську (фельдшерські

бригади – долікарську) допомогу постраждалим і евакуюють їх у стаціонарні медичні заклади.

Якщо ліквідувати медико-санітарні наслідки надзвичайної ситуації силами штатних бригад ШМД неможливо, то розгортаються сили першого етапу медичної евакуації. У район надзвичайної ситуації додатково до діючих штатних бригад ШМД направляються медичні бригади постійної готовності першої черги, територіального рівня (а якщо вимагають обставини, то і державного рівня).

При неможливості забезпечення кваліфікованої та спеціалізованої допомоги всім постраждалим діючими лікарняними установами розгортаються лікувальні заклади другого етапу медичної евакуації. Ліжкофонд, що розгортається додатково, забезпечується запасами м'якого інвентарю, медикаментів, продуктів харчування, предметів догляду за хворими, медобладнання тощо. Спеціалізовані бригади постійної готовності другої черги, що прибувають на посилення медичного персоналу забезпечені майном за рахунок закладів до яких вони належать.

Координацію роботи служби на державному рівні здійснює Центральна координаційна комісія МОЗ України, а на територіальному рівні — територіальні координаційні комісії. Вони є постійно діючими дорадчими позаштатними органами, які створюються з метою погодження дій медичних сил різних відомств в умовах надзвичайних ситуацій. До складу комісій входять представники усіх міністерств і відомств, відповідних рівнів Служби.

Головою центральної координаційної комісії є Міністр охорони здоров'я України, а територіальних координаційних комісій — начальники управлінь (відділів)охорони здоров'я відповідних адміністративних територій.

Особливе місце в роботі медичної служби займає захист населення від інфекційних захворювань.

Для запобігання розповсюдження інфекційних захворювань в осередку інфекційної хвороби встановлюється режим карантину або обсервації.

Карантин — адміністративні та медико-санітарні заходи, що застосовуються для запобігання поширенню особливо небезпечних

інфекційних захворювань;

Рішення про встановлення карантину, а також про його відміну негайно доводиться до відома населення відповідної території через засоби масової інформації.

У рішенні про встановлення карантину зазначаються:

— обставини, що призвели до цього;

— визначаються межі території карантину;

— затверджуються необхідні профілактичні, протиепідемічні та інші заходи, їх виконавці та терміни проведення;

— встановлюються тимчасові обмеження прав фізичних і юридичних осіб та додаткові обов'язки, що покладаються на них.

Встановлення карантину передбачає:

— повну ізоляцію осередку інфекційної хвороби;

— встановлення охорони на зовнішніх кордонах;

— заборону виходу людей, тварин та вивезення майна;

— дозвіл в'їзду лише спеціальним формуванням призначеним для проведення профілактичних та протиепідемічних заходів;

— заборону транзитного проїзду;

— розподіл населення на дрібні групи і доставку продуктів харчування, води в окремі квартири та будинки;

— припинення роботи всіх підприємств та установ, крім тих, які мають значення для життєзабезпечення населення;

— проведення профілактичних заходів серед населення та лікування хворих;

— проведення санітарної обробки населення, дезінфекції, дезінсекції, дератизації;

— використання засобів індивідуального захисту.

Об'єкти які продовжують роботу в зонах карантину переходять на особливий режим праці:

— робітники та службовці переводяться на казармене положення з виконанням протиепідемічних заходів;

— зміни розподіляються на окремі групи (меншої чисельності), контактування між ними та вихід з приміщень забороняється;

— харчування та відпочинок організовується групами у спеціально відведених приміщеннях.

Карантин встановлюється на період, необхідний для ліквідації епідемії чи спалаху особливо небезпечної інфекційної хвороби. Коли інфекційна хвороба не відноситься до групи особливо небезпечних вводяться обмежувальні протиепідемічні заходи – режим обсервації.

Обмежувальні протиепідемічні заходи встановлюються місцевими органами виконавчої влади та органами місцевого самоврядування за поданням відповідного головного державного санітарного лікаря у разі, коли в окремому населеному пункті, у дитячому виховному, навчальному чи оздоровчому закладі виник спалах інфекційної хвороби або склалася неблагополучна епідемічна ситуація, що загрожує поширенням інфекційних захворювань. Обмеженням підлягають ті види господарської та іншої діяльності, що можуть сприяти поширенню інфекційних захворювань. Види і тривалість обмежувальних протиепідемічних заходів встановлюються залежно від особливостей перебігу інфекційної хвороби, стану епідемічної ситуації та обставин, що на неї впливають.

4.3.Висновок до розділу 4

В цьому розділі розглянуто такі питання: "Пожежна сигналізація і зв'язок. Засоби гасіння пожеж. Протипожежне водопостачання. Первинні засоби пожежогасіння. Автоматичні засоби пожежогасіння на об'єктах галузі". У підрозділі проаналізовано медичний захист населення у надзвичайних ситуаціях.

ВИСНОВКИ

У роботі проведено огляд та аналіз існуючих методів локалізації та розпізнавання тексту на зображеннях складних графічних сцен. Запропоновано підхід до розпізнавання локалізованого тексту, що базується на поєднанні загорткових, рекурентних нейронних мереж та алгоритму CTC-loss. Ця архітектура нейронних мереж реалізована за допомогою мови програмування Python із застосуванням бібліотеки Tensorflow. Проведено експеримент на двох наборах даних, за результатами якого були побудовані графіки зміни протягом навчання функції втрат, відстані Левенштейна та точності розпізнавання на тренувальному та двох тестових наборах даних. Складено таблицю значень точності розпізнавання тексту. Розпізнавання проводилося двома способами: з використанням додаткового словника фіксованого розміру та без його використання. Для набору даних IIIT5K використовувалися 2 словники – розмірами 50 та 1000 слів, для набору даних SVT використовувався словник розмірів 50 слів.

За підсумками експерименту на наборі даних SVT без використання словника було досягнуто точності 76.2%, з використанням словника розміром 50 слів – 95.3%. На наборі даних IIIT5K без словника було досягнуто точність 74.0%, зі словником розміром 50 слів – 96.7%, зі словником розміром 1000 слів – 93.2. Проведено порівняння з результатами інших робіт, розроблена система показує результати, близькі до state-of-the-art рішень з меншими обчислювальними витратами завдяки використанню додаткових шарів нормалізації міні-батчів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Адрианов А.И. Локализация текста на изображениях сложных графических сцен // Современные проблемы науки и образования. – № 3. – 2013.

2 Y. Kunishige, F. Yaokai, S. Uchida. Scenery Character Detection with Environmental Context // The 11th International Conference on Document Analysis and Recognition (ICDAR), 2011. pp. 1049 – 1053.

3 S. Uchida, Y. Shigeyoshi, Y. Kunishige, F. Yaokai. A Keypoint-Based Approach Toward Scenery Character Detection // The 11th International Conference on Document Analysis and Recognition (ICDAR), 2011. pp. 819 – 823.

4 Y. Du, H. Ai, S. Lao. Dot Text Detection Based on FAST Points // The 11th International Conference on Document Analysis and Recognition (ICDAR), 2011. pp. 435 – 439.

5 A. Coates, B. Carpenter, C. Case, S. Satheesh, B. Suresh, T. Wang, D. Wu, A. Ng. Text Detection and Character Recognition in Scene Images with Unsupervised Feature Learning // The 11th International Conference on Document Analysis and Recognition (ICDAR), 2011. pp. 440 – 445.

6 B. Epshtein, E. Ofek, Y. Wexler, Detecting Text in Natural Scenes with Stroke Width Transform // 23rd IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol.V. San Francisco, 2010.

7 Tong He, Weilin Huang, Yu Qiao, Jian Yao. Text-attentional convolutional neural network for scene text detection // IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

8 Pengyuan Lyu, Minghui Liao, Cong Yao, Wenhao Wu, Xiang Bai. Mask TextSpotter: An End-to-End Trainable Neural Network for Spotting Text with Arbitrary Shapes. Huazhong University of Science and Technology, 2018.

9 Bissacco, A., Cummins, M., Netzer, Y., Neven, H.: Photoocr: Reading text in uncontrolled conditions // Proc. ICCV, 2013. pp. 785–792.

10 Jaderberg, M., Vedaldi, A., Zisserman, A. Deep features for text spotting //

Proc. ECCV, 2014. pp. 512–528.

11 Jaderberg, M., Simonyan, K., Vedaldi, A., Zisserman, A.: Synthetic data and artificial neural networks for natural scene text recognition. CoRR, 2014.

12 Max Jaderberg, Karen Simonyan, Andrea Vedaldi, Andrew Zisserman. Reading Text in the Wild with Convolutional Neural Networks. // International Journal of Computer Vision. – Hingham, MA, USA, 2016. – pp. 1-20.

13 I. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, V.Shet. Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks // arXiv:1312.6082, 2014.

14 Shi, B., Bai, X., Yao, C. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition // IEEE Trans. Pattern Anal. Mach. Intell. 39(11), 2017. pp. 2298–2304.

15 Alex Graves, Santiago Fernandez, Faustino Gomez, Jurgen Schmidhuber. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. // ICML '06 Proceedings of the 23rd international conference on Machine learning. – Pittsburgh, Pennsylvania, 2006. – pp. 369-376.

16 S. Ghosh, E. Valveny, A. Bagdanov. Visual attention models for scene text recognition // 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), 2017.

17 F. Yin, Y. Wu, X. Zhang, C. Liu. Scene Text Recognition with Sliding Convolutional Character Models // arXiv:1709.01727, 2017.

18 Y. Wu, F. Yin, X. Zhang, L. Liu, C. Liu. SCAN: Sliding Convolutional Attention Network for Scene Text Recognition // ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), 2019.

19 P. Wang, L. Yang, H. Li, Y. Deng, C. Shen, Y. Zhang. A Simple and Robust Convolutional-Attention Network for Irregular Text Recognition // arXiv:1904.01375, 2019.

20 W. Liu, C. Chen, K. Wong. SAFE: Scale Aware Feature Encoder for Scene Text Recognition // Asian Conference on Computer Vision (ACCV), 2018.

21 S. Ioffe, C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift // ICML'15 Proceedings of the 32nd

International Conference on International Conference on Machine Learning, 2015.

22 Sheng Qian, Hua Liu, Cheng Liu, Si Wu, Hau San Wong. Adaptive activation functions in convolutional neural networks // Neurocomputing, Volume 272, 2018. – pp. 204-212. 16 Kohonen, T. (1988), Learning Vector Quantization, Neural Networks, 1 (suppl 1), 303.

23 S. Hochreiter, Jurgen Schmidhuber. Long short-term memory // Neural Computation, 1997. – pp. 1735–1780.

24 Baoguang Shi, Xiang Bai, Cong Yao. An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition // IEEE Transactions on Pattern Analysis and Machine Intelligence. – University of Toronto, Canada, 2015. – pp. 99.

25 Профессиональный информационно-аналитический ресурс, посвященный машинному обучению, распознаванию образов и интеллектуальному анализу данных. [Электронный ресурс]: режим доступа - http://www.machinelearning.ru.

26 Многослойные нейронные сети [Электронный ресурс] : Материал из Википедии — свободной энциклопедии : Версия 89620593, сохранённая в 18:53 UTC 12 декабря 2017 / Авторы Википедии // Википедия, свободная энциклопедия. — Электрон. дан. — Сан-Франциско: Фонд Викимедиа, 2017. — Режим доступа: http://ru.wikipedia.org/?oldid=89620593 wikipedia.org

27 Bottou, Leon. (2011). From Machine Learning to Machine Reasoning. Computing Research Repository - CORR. 94. . 10.1007/s10994-013-5335-x.

28 Уоссермен Ф. Нейрокомпьютерная техника: Теория и практика = Neural Computing. Theory and Practice. — М.: Мир, 1992. — 240 с. — ISBN 5-03-002115-9.

29 Медведев В.С. Нейронные сети / В.С. Медведев, В.Г. Потемкин – М.: Диалог МИФИ, 2002.

30 Carlos Affonso, Andre Luis Debiasso Deep Learning for biological image classification – Expert Systems with Applications, Volume 85, 2017, pp. 114-122.

31 Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions”. arXiv:1409.4842

2014

32 C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, “Rethinking the Inception Architecture for Computer Vision”. arXiv:1512.00567 2015

33 Статистическая обработка данных [Электронный ресурс] – Режим доступа: http://www.intuit.ru/studies/courses/3632/874/info.

34 Классические методы статистики:метод главных компонент[Электронный ресурс] – Режим доступа: http://r-analytics.blogspot.ru/2012/08/blog-post.html.

35 Профессиональный информационно-аналитический ресурс, посвященный машинному обучению, распознаванию образов и интеллектуальному анализу данных. [Электронный ресурс]– Режим доступа: http://www.machinelearning.ru/

36 Статистический анализ медицинских данных. Применение пакета прикладных программ STATISTICA / Реброва О.Ю. – Москва: Медиа Сфера, 2002. – 312 с.

37 Статистические методы построения эмпирических формул. / Львовский Е.Н. – Москва: Высшая школа, 19. 239 с.

38 Нейронная сеть Кохонена [Электронный ресурс] : Материал из Википедии — свободной энциклопедии : Версия 89620593, сохранённая в 18:53 UTC 12 декабря 2017 / Авторы Википедии // Википедия, свободная энциклопедия. — Электрон. дан. — Сан-Франциско: Фонд Викимедиа, 2017. — Режим доступа: http://ru.wikipedia.org/?oldid=89620593 wikipedia.org.

39 Kohonen, T. (1988), Learning Vector Quantization, Neural Networks, 1 (suppl 1), 303.

40 Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel: Backpropagation Applied to Handwritten Zip Code Recognition, Neural Computation, 1(4):541-551, Winter 1989.

41 Jain, V. and Seung, S. H. (2008). Natural image denoising with convolutional networks. In NIPS’2008.

42 Graham, Benjamin (2014-12-18), "Fractional Max-Pooling", arXiv:1412.6071 [cs.CV].

43 Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. (2009a). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In ICML’2009.

44 Zeiler, M., Krishnan, D., Taylor, G., and Fergus, R. (2010). Deconvolutional networks. In CVPR’2010.

45 Галушкин А. И. Синтез многослойных систем распознавания образов. — М.: Энергия, 1974.

46 Martinetz T. M., Berkovich S. G., Schulten K. J. Neural-gas network for vector quantization and its application to time-series prediction // IEEE Trans. on Neural Networks, 1993, No. 4. — P. 558—569.

47 Linda G. Shapiro and George C. Stockman (2001): «Computer Vision», pp 279—325, New Jersey, Prentice-Hall, ISBN 0-13-030796-3.

48 Ивченко Г. И., Медведев Ю. И. Введение в математическую статистику. — М. : Издательство ЛКИ, 2010. — §2.2. Выборочные моменты: точная и асимптотическая теория. — ISBN 978-5-382-01013-7.

49 Гилл Ф., Мюррей У., Райт М. Практическая оптимизация = Practical Optimization. — М.: Мир, 1985.

50 Горбань А. Н. Обучение нейронных сетей. — М.: СП ПараГраф, 1990.

51 Masakazu Iwamura. Advances of Scene Text Datasets // Department of Computer Science and Intelligent Systems Graduate School of Engineering, Osaka Prefecture University, 2018.

ДОДАТКИ

ДОДАТОК А

Вихідний код розробленої програми на ЕОМ

config.py

```
char_vector = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ-'.!?,\""
num_classes = len(char_vector) + 1
batch_size = 64
max_word_len = 16
img_w = 100
img_h = 31
```

utils.py

```
import numpy as np import config
def label_to_array(label):
return [config.char_vector.index(x) for x in label]
def ground_truth_to_word(ground_truth):
try:
ground_truth_int = ground_truth.astype(np.int16) word =
".join([config.char_vector[int(i)] for i in ground_truth_int if i in
np.arange(len(config.char_vector))])
except TypeError:
print(ground_truth) print(ground_truth_int)
return word
def batch_ground_truth_to_word(batch_ground_truth):
batch_words = np.empty(config.batch_size, dtype=np.object)
try:
for j in np.arange(config.batch_size):
batch_words[j] = ".join([config.char_vector[int(i)] for i in batch_ground_truth[j] if i
in np.arange(len(config.char_vector))])
except TypeError:
print(batch_ground_truth) print(batch_ground_truth[0])
```

```
return batch_words
def to_max_len(word):
while len(word) < config.max_word_len:
    word += '-' return word
def sparse_tuple_from(sequences, dtype=np.int32):
indices = [] values = []
for n, seq in enumerate(sequences):
indices.extend(zip([n]*len(seq), [i for i in range(len(seq))])) values.extend(seq)
indices = np.asarray(indices, dtype=np.int64) values = np.asarray(values,
dtype=dtype)
    shape = np.asarray([len(sequences), np.asarray(indices).max(0)[1]+1],
dtype=np.int64)
return indices, values, shape
data.py
import numpy as np
import pandas as pd
from imageio import imread
from scipy.misc import imresize
from utils import label_to_array, to_max_len, sparse_tuple_from
from tensorflow.keras.utils import to_categorical
from mat4py import loadmat
import xmltodict
import random
import config
import tensorflow as tf
def synth_word_generator(path='90kDICT32px/', mode='train'):
if mode == 'train':
    file_path = 'annotation_train.txt' if mode == 'val':
    file_path = 'annotation_val.txt' if mode == 'test':
file_path = 'annotation_test.txt'
with open(path + file_path, 'r') as file:
```

```
    file_raws = file.readlines() dataset_len = len(file_raws) j = 0
while True:
j += 1
# Random files for this batch
batch_raws = [random.choice(file_raws) for _ in np.arange(config.batch_size)]
x_batch = np.empty([config.batch_size, config.img_h, config.img_w, 1])
y_batch = np.empty(config.batch_size, dtype=np.object)
dt_batch = np.empty(config.batch_size, dtype=np.object)
for i, raw in enumerate(batch_raws):
try:
img_path = raw.split(' ')[0]
label = img_path.split('_')[1].upper()
target = label_to_array(label)
img = imread(path + img_path)[:, :, 0]
except (IndexError, SyntaxError, ValueError, OSError):
raw = random.choice(file_raws)
img_path = raw.split(' ')[0]
label = img_path.split('_')[1].upper()
target = label_to_array(label)
img = imread(path + img_path)[:, :, 0]
img = imresize(img, (config.img_h, config.img_w))[:, :, np.newaxis]
x_batch[i, :, :, :] = img
y_batch[i] = label dt_batch[i] = target
x_batch /= 255
dt_batch = sparse_tuple_from(np.reshape(np.array(dt_batch), (-1)))
yield x_batch, y_batch, dt_batch
def svt_word_generator(path='SVT/'):
xml_path = 'test.xml'
with open(path + xml_path, 'rb') as file:
xmlDict = xmltodict.parse(file)
df = pd.DataFrame.from_dict(xmlDict)
```

```
x_dataset = [] y_dataset = [] tg_dataset = [] lex_dataset = []
for image in df['tagset']['image']:
img_path = image['imageName']
lexicon = image['lex']
full_img = imread(path + img_path)
if isinstance(image['taggedRectangles']['taggedRectangle'], list):
for word in image['taggedRectangles']['taggedRectangle']:
label = word['tag'] target = label_to_array(label) height = int(word['@height']) width
= int(word['@width'])
x = int(word['@x']) y = int(word['@y']) word = full_img[y:y + height, x:x + width, :]
if (word.shape[0] == 0) or (word.shape[1] == 0): continue
word = imresize(word, (config.img_h, config.img_w))
x_dataset.append(word)
y_dataset.append(label)
tg_dataset.append(target)
lex_dataset.append(lexicon)
else:
word = image['taggedRectangles']['taggedRectangle']
label = word['tag']
target = label_to_array(label)
height = int(word['@height'])
width = int(word['@width'])
x = int(word['@x'])
y = int(word['@y'])
word = full_img[y:y + height, x:x + width, :]
word = imresize(word, (config.img_h, config.img_w))
x_dataset.append(word) y_dataset.append(label) tg_dataset.append(target)
lex_dataset.append(lexicon)
   x_dataset = np.mean(np.array(x_dataset, dtype=np.float32), axis=-1)[:, :, :,
np.newaxis]
x_dataset = np.reshape(x_dataset[:-6], [10, 64, 31, 100, 1]) x_dataset /= 255
```

```
y_dataset = np.array(y_dataset, dtype=np.object) y_dataset = np.reshape(y_dataset[:-
6], [10, 64]) tg_dataset = np.array(tg_dataset, dtype=np.object) tg_dataset =
np.reshape(tg_dataset[:-6], [10, 64]) lex_dataset = np.array(lex_dataset,
dtype=np.object) lex_dataset = np.reshape(lex_dataset[:-6], [10, 64])
for i in np.arange(10):
x_batch = x_dataset[i]
y_batch = y_dataset[i]
tg_batch = tg_dataset[i]
lex_batch = lex_dataset[i]
tg_batch = sparse_tuple_from(np.reshape(np.array(tg_batch), (-1)))
yield x_batch, y_batch, tg_batch, lex_batch
def IIIT5K_word_generator(path=TIIT5K/'):
file_path = 'testdata.mat' key = 'testdata'
x_dataset = [] y_dataset = [] tg_dataset = [] lex50_dataset = [] lexlOOO_dataset = []
file = loadmat(path + file_path) dataset_len = len(file[key]['ImgName'])
for i in np.arange(dataset_len):
img_path = file[key]['ImgName'][i] label = file[key]['GroundTrath'][i] lexicon_50 =
file[key]['smallLexi'][i] lexicon_1000 = file[key]['mediumLexi'][i] target =
label_to_array(label)
img = imread(path + img_path) if img.ndim == 2:
   img = np.stack([img, img, img], axis=2) img = imresize(img, (config.img_h,
config.img_w))
x_dataset.append(img) y_dataset.append(label) tg_dataset.append(target)
lex50_dataset.append(lexicon_50) lex 1000_dataset. append(lexicon_ 1000)
   x_dataset = np.mean(np.array(x_dataset, dtype=np.float32), axis=-l)[:, :, :,
np.newaxis]
x_dataset = np.reshape(x_dataset[:-56], [46, 64, 31, 100, 1]) x_dataset /= 255
y_dataset = np.array(y_dataset, dtype=np.object) y_dataset = np.reshape(y_dataset[:-
56], [46, 64]) tg_dataset = np.array(tg_dataset, dtype=np.object) tg_dataset =
np.reshape(tg_dataset[:-56], [46, 64]) lex50_dataset = np.array(lex50_dataset,
dtype=np.object) lex50_dataset = np.reshape(lex50_dataset[:-56], [46, 64])
```

```
lexl000_dataset = np.array (lex 1000_dataset, dtype=np.object)
lex1000_dataset = np.reshape(lex1000_dataset[:-56], [46, 64, 1000])
for i in np.arange(46):
x_batch = x_dataset[i]
y_batch = y_dataset[i]
tg_batch = tg_dataset[i]
tg_batch = sparse_tuple_from(np.reshape(np.array(tg_batch), -1))
lex50_batch = lex50_dataset[i]
lex1000_batch = lex1000_dataset[i]
yield x_batch, y_batch, tg_batch, lex50_batch, lex1000_batch
train_model.py
import numpy as np
import tensorflow as tf
from tensorflow.contrib import rnn
from utils import ground_truth_to_word, batch_ground_truth_to_word
from data import synth_word_generator, svt_word_generator,
IIIT5K_word_generator
import config
save_path = '.'
train_len = 7224612
train_epoch_len = train_len // config.batch_size
iteration_count = train_epoch_len * 5
inputs = tf.placeholder(tf.float32, [config.batch_size, config.img_h, config.img_w,
1],
name='inputs')
targets = tf.sparse_placeholder(tf.int32, name='targets')
seq_len = tf.placeholder(tf.int32, [None], name='seq_len')
# Feature extraction
conv1 = tf.layers.conv2d(inputs=inputs, filters = 64, kernel_size = (3, 3), padding =
"same", activation=tf.nn.relu, name='conv_1')
pool1 = tf.layers.max_pooling2d(inputs=conv1, pool_size=[2, 2], strides=2,
```

```
name='pool_1')
conv2 = tf.layers.conv2d(inputs=pool1, filters = 128, kernel_size = (3, 3), padding =
"same", activation=tf.nn.relu, name='conv_2')
pool2 = tf.layers.max_pooling2d(inputs=conv2, pool_size=[2, 2], strides=2,
name='pool_2')
conv3 = tf.layers.conv2d(inputs=pool2, filters = 256, kernel_size = (3, 3), padding =
"same", activation=tf.nn.relu, name='conv_3')
bnorm1 = tf.layers.batch_normalization(conv3, name='bn_1')
conv4 = tf.layers.conv2d(inputs=bnorm1, filters = 256, kernel_size = (3, 3), padding
=
"same", activation=tf.nn.relu, name='conv_4')
pool3 = tf.layers.max_pooling2d(inputs=conv4, pool_size=[2, 2], strides=[1, 2],
padding="same", name='pool_3')
conv5 = tf.layers.conv2d(inputs=pool3, filters = 512, kernel_size = (3, 3), padding =
"same", activation=tf.nn.relu, name='conv_5')
bnorm2 = tf.layers.batch_normalization(conv5, name='bn_2')
conv6 = tf.layers.conv2d(inputs=bnorm2, filters = 512, kernel_size = (3, 3), padding
=
"same", activation=tf.nn.relu, name='conv_6')
pool4 = tf.layers.max_pooling2d(inputs=conv6, pool_size=[2, 2], strides=[1, 2],
padding="same", name='pool_4')
conv7 = tf.layers.conv2d(inputs=pool4, filters = 512, kernel_size = (2, 2), padding =
"valid", activation=tf.nn.relu, name='conv_7')
# Reshape for LSTM and max char count
reshaped_cnn_output = tf.reshape(conv7, [config.batch_size, -1, 512],
name='reshaped_cnn')
max_char_count = reshaped_cnn_output.get_shape().as_list()[1]
max_char_count_name = tf.constant(max_char_count, name='max_char_count')
print('='*20, max_char_count)
with tf.variable_scope(None, default_name="bidirectional-rnn-1"):
#     Forward
```

```
lstm_fw_cell_1 = rnn.BasicLSTMCell(256)
#     Backward
lstm_bw_cell_1 = rnn.BasicLSTMCell(256)
#     First bidirectional LSTM
inter_output, _ = tf.nn.bidirectional_dynamic_rnn(lstm_fw_cell_1, lstm_bw_cell_1,
reshaped_cnn_output, seq_len, dtype=tf.float32) inter_output =
tf.concat(inter_output, 2)
with tf.variable_scope(None, default_name="bidirectional-rnn-2"):
#     Forward
lstm_fw_cell_2 = rnn.BasicLSTMCell(256)
#     Backward
lstm_bw_cell_2 = rnn.BasicLSTMCell(256)
#     Second bidirectional LSTM
lstm_outputs, _ = tf.nn.bidirectional_dynamic_rnn(lstm_fw_cell_2, lstm_bw_cell_2,
inter_output, seq_len, dtype=tf.float32) lstm_outputs = tf.concat(lstm_outputs, 2)
logits = tf.reshape(lstm_outputs, [-1, 512], name='reshaped_lstm')
W = tf.Variable(tf.truncated_normal([512, config.num_classes], stddev=0.1),
name="W")
b = tf.Variable(tf.constant(0., shape=[config.num_classes]), name="b")
logits = tf.matmul(logits, W) + b
logits = tf.reshape(logits, [config.batch_size, -1, config.num_classes], name='logits')
# Final layer, the output of the BLSTM
logits = tf.transpose(logits, (1, 0, 2), name='transpose_logits')
# Loss and cost calculation
loss = tf.nn.ctc_loss(targets, logits, seq_len) cost = tf.reduce_mean(loss, name='cost')
optimizer = tf.train.AdamOptimizer(learning_rate=1e-4).minimize(cost)
decoded, log_prob = tf.nn.ctc_beam_search_decoder(logits, seq_len,
merge_repeated=False)
dense_decoded = tf.sparse_tensor_to_dense(decoded[0], default_value=-1,
name='dense_decoded')
acc = tf.reduce_mean(tf.edit_distance(tf.cast(decoded[0], tf.int32), targets),
```

```
name='accuracy')
with tf.Session() as sess:
init = tf.global_variables_initializer() sess.run(init)
synth_train_gen = synth_word_generator('90kDICT32px/', mode='train') svt_gen =
svt_word_generator(path='SVT/') IIIT5K_gen =
IIIT5K_word_generator(path='IIIT5K/')
train_loss_hist = [] val_loss_hist = [] train_acc_hist = [] val_acc_hist = []
svt_loss_hist = [] svt_acc_hist = [] IIIT5K_loss_hist = [] IIIT5K_acc_hist = []
train_wer_hist = [] val_wer_hist = [] svt_wer_hist = [] IIIT5K_wer_hist = []
# Train
for i in np.arange(iteration_count):
x_batch, y_batch, dt_batch = synth_train_gen.__next__()
      op, decoded_value, loss_value, acc_value = sess.run([optimizer, dense_decoded,
cost, acc], feed_dict={inputs: x_batch, targets: dt_batch, seq_len: [max_char_count]
* config.batch_size})
if i % 50 == 0:
print('True synth train:\t{}'.format(y_batch[0]))
print('Pred synth train:\t{}'.format(ground_truth_to_word(decoded_value[0])))
         print('[{}] Synth 90k iteration. Train loss: {}\tTrain accuracy: {}'.format(i,
loss_value, acc_value))
         train_wer = np.sum(np.equal(y_batch,
batch_ground_truth_to_word(decoded_value)))/config.batch_size
print('Train WER:\t{}'.format(train_wer))
train_wer_hist.append(train_wer)
train_loss_hist.append(loss_value)
train_acc_hist.append(acc_value)
x_batch_val, y_batch_val, dt_batch_val = synth_val_gen.__next__()
val_decoded_value, val_loss_value, val_acc_value = sess.run([dense_decoded,
cost, acc], feed_dict={inputs: x_batch_val, targets: dt_batch_val, seq_len:
[max_char_count] * config.batch_size})
print('True synth val:\t{}'.format(y_batch_val[0]))
```

```
        print('Pred synth
val:\t{}'.format(ground_truth_to_word(val_decoded_value[0])))
        print('[{}] Synth 90k iteration. Val loss: {}\tVal accuracy: {}'.format(i,
val_loss_value, val_acc_value))
        val_wer = np.sum(np.equal(y_batch_val,
batch_ground_truth_to_word(val_decoded_value)))/config.batch_size
print('Val WER:\t{}'.format(val_wer))
val_wer_hist.append(val_wer)
val_loss_hist.append(val_loss_value)
val_acc_hist.append(val_acc_value)
x_batch_svt, y_batch_svt, dt_batch_svt = svt_gen.__next__() svt_decoded_value,
svt_loss_value, svt_acc_value, _, _ =
sess.run([dense_decoded, cost, acc], feed_dict={inputs: x_batch_svt, targets:
dt_batch_svt, seq_len: [max_char_count] * config.batch_size})
print('True svt:\t{}'.format(y_batch_svt[0]))
print('Pred svt:\t{}'.format(ground_truth_to_word(svt_decoded_value[0])))
        print('[{}] SVT iteration. Test loss: {}\tTest accuracy: {}'.format(i,
svt_loss_value, svt_acc_value))
        svt_wer = np.sum(np.equal(y_batch_svt,
batch_ground_truth_to_word(svt_decoded_value)))/config.batch_size
print('svt WER:\t{}'.format(svt_wer))
svt_wer_hist.append(svt_wer)
svt_loss_hist.append(svt_loss_value)
svt_acc_hist.append(svt_acc_value)
x_batch_IIIT5K, y_batch_IIIT5K, dt_batch_IIIT5K = IIIT5K_gen.__next__()
IIIT5K_decoded_value, IIIT5K_loss_value, IIIT5K_acc_value, _, _ =
sess.run([dense_decoded, cost, acc], feed_dict={inputs: x_batch_IIIT5K, targets:
dt_batch_IIIT5K, seq_len: [max_char_count] * config.batch_size})
print('True IIIT5K:\t{}'.format(y_batch_IIIT5K[0]))
        print('Pred
IIIT5K:\t{}'.format(ground_truth_to_word(IIIT5K_decoded_value[0])))
```

```
        print('[{}] IIIT5K iteration. Test loss: {}\tTest accuracy: {}'.format(i,
IIIT5K_loss_value, IIIT5K_acc_value))
        IIIT5K_wer = np.sum(np.equal(y_batch_IIIT5K,
batch_ground_truth_to_word(IIIT5K_decoded_value)))/config.batch_size
print('IIIT5K WER:\t{}'.format(IIIT5K_wer))
IIIT5K_wer_hist.append(IIIT5K_wer)
IIIT5K_loss_hist.append(IIIT5K_loss_value)
IIIT5K_acc_hist.append(IIIT5K_acc_value)
if i % 5000 == 0:
saver = tf.train.Saver(tf.global_variables(), max_to_keep=10) saver.save(sess,
save_path)
evaluate.py
import numpy as np
import tensorflow as tf
from utils import ground_truth_to_word, batch_ground_truth_to_word
from test_data import svt_word_generator, IIIT5K_word_generator
from editdistance import distance
import config
def get_predict_gen(dataset='SVT'):
with tf.Session() as sess:
#       Load model
saver = tf.train.import_meta_graph('..meta') saver.restore(sess,
tf.train.latest_checkpoint('./')) graph = tf.get_default_graph()
#       Get tensors
inputs = graph.get_tensor_by_name("inputs:0") targets_shape =
graph.get_tensor_by_name("targets/shape:0") targets_values =
graph.get_tensor_by_name("targets/values:0") targets_indices =
graph.get_tensor_by_name("targets/indices:0") targets =
tf.SparseTensor(targets_indices, targets_values, targets_shape) seq_len =
graph.get_tensor_by_name("seq_len:0") max_char_count =
graph.get_tensor_by_name("reshaped_cnn:0") .get_shape().as_list()[1]
```

```
acc = graph.get_tensor_by_name("accuracy:0")
dense_decoded = graph.get_tensor_by_name("dense_decoded:0")
if dataset == 'SVT':
dataset_gen = svt_word_generator(path='SVT/')
for i in np.arange(10):
        x_batch_svt, y_batch_svt, dt_batch_svt, lex_batch_svt =
dataset_gen.__next__()
        acc_value, svt_decoded_value = sess.run([acc, dense_decoded],
feed_dict={inputs: x_batch_svt, targets: dt_batch_svt, seq_len: [max_char_count] *
config.batch_size})
        yield svt_acc_value, batch_ground_truth_to_word(svt_decoded_value),
y_batch_svt, lex_batch_svt
if dataset == 'IIIT5K':
dataset_gen = IIIT5K_word_generator(path='IIIT5K/') for i in np.arange(46):
        x_batch_IIIT5K, y_batch_IIIT5K, dt_batch_IIIT5K, lex50_batch_IIIT5K,
lex1000_batch_IIIT5K = dataset_gen.__next__()
        IIIT5K_acc_value, IIIT5K_decoded_value = sess.run([acc,
dense_decoded], feed_dict={inputs: x_batch_IIIT5K, targets: dt_batch_IIIT5K,
seq_len: [max_char_count] * config.batch_size})
        yield IIIT5K_acc_value,
batch_ground_truth_to_word(IIIT5K_decoded_value), y_batch_IIIT5K,
lex50_batch_IIIT5K, lex1000_batch_IIIT5K
if __name__ == '__main__':
svt_gen = get_predict_gen(dataset='SVT') IIIT5K_gen =
get_predict_gen(dataset='IIIT5K') total_svt_acc = [] total_svt_acc_d50 = []
total_IIIT5K_acc_d50 = [] total_IIIT5K_acc_d1000 = [] total_IIIT5K_acc = []
for i in np.arange(10):
svt_acc, svt_pred, svt_true, lexicon = next(svt_gen) svt_found_words = []
for j in np.arange(config.batch_size):
current_lex = lexicon[j].split(',') pred_word = str(svt_pred[j])
distances = [distance(pred_word, lex_word) for lex_word in current_lex] found_word
```

```
= current_lex[np.argmin(distances)] svt_found_words.append(found_word)
total_svt_acc.append(np.sum(np.equal(svt_true, svt_pred))/config.batch_size)
total_svt_acc_d50.append(np.sum(np.equal(svt_true,
svt_found_words))/config.batch_size)
print('Total SVT accuracy\n', np.mean(total_svt_acc))
print('Total SVT accuracy with 50 words dict', np.mean(total_svt_acc_d50))
for i in np.arange(46):
IIIT5K_acc, IIIT5K_pred, IIIT5K_trae, IIIT5K_lex50, IIIT5K_lexlOOO =
next(IIIT5K_gen)
IIIT5K_found_words50 = [] IIIT5K_found_wordslOOO = []
for j in np.arange(config.batch_size):
current_lex50 = IIIT5K_lex50[j] current_lexlOOO = IIIT5K_lexl000[j] pred_word =
str(IIIT5K_pred[j])
distances50 = [distance(pred_word, lex_word) for lex_word in current_lex50]
distances 1000 = [distance(pred_word, lex_word) for lex_word in current_lex 1000]
found_word50 = current_lex50[np.argmin(distances50)] found_wordl000 =
current_lexl000[np.argmin(distancesl000)]
IIIT5K_found_words50.append(found_word50) IIIT5 K_found_words
1000.append(found_word 1000)
        total_IIIT5K_acc.append(np. sum(np.equal(IIIT5 K_trae,
IIIT5K_pred))/config.batch_size)
        total_IIIT5K_acc_d50.append(np.sum(np.equal(IIIT5K_trae,
IIIT5K_found_words50))/config.batch_size)
        total_IIIT5K_acc_d 1000.append(np. sum(np.equal(IIIT5 K_true,
IIIT5K_found_words 1000))/config.batch_size)
print('Total IIIT5K accuracy\n', np.mean(total_IIIT5K_acc))
    print('Total IIIT5K accuracy with 50 words dictW,
np.mean(total_IIIT5K_acc_d50))
    print('Total IIIT5K accuracy with 1000 words dictW,
np.mean(total_IIIT5K_acc_dl000))
```

ДОДАТОК А

Копія тези конференції

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

IX НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

8–9 грудня 2021 року

ТЕРНОПІЛЬ
2021

ЗМІСТ

УДК 621.376

А. Марценюк, Б. Андрейчук, А. Щіпський
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

МОДЕЛЮВАННЯ РАДІОСИГНАЛУ ІЗ АМПЛІТУДНОЮ МОДУЛЯЦІЄЮ ДЛЯ ОЦІНЮВАННЯ ЗАВАДОСТІЙКОСТІ ЗВ'ЯЗКУ

UDC 621.376

A. Martsenyuk, B. Andriychuk, A. Shchipsky

MODELING OF RADIO SIGNAL WITH AMPLITUDE MODULATION FOR ASSESSMENT OF COMMUNICATION DIFFICULTY

При створенні систем передачі даних в більшості випадків виявляється, що спектр початкового передаючого сигналу зосереджений зовсім не на тих частотах, які ефективно пропускає наявний канал зв'язку. Вирішення вказаної проблеми досягається шляхом використання модуляції. В процесі передачі даних по каналам із певним типом модуляції відбувається вплив на нього різного роду завад, що негативно впливає на його завадозахищеність і відповідно спотворення передаючих даних. Тому визначення завадостійкості каналу зв'язку як показника якості функціонування за призначенням без погіршення робочих характеристик під впливом різного роду завад є актуальною задачею.

В роботі представлено результати оцінювання каналу зв'язку із амплітудною модуляцією за блок-схемою дослідження, яку зображена на рисунку.

Застосовуючи програмне середовище Matlab здійснено процес амплітудної модуляції над синусоїдальним сигналом із різними значеннями несучої частоти. Для організації процесу оцінювання завадостійкості каналу зв'язку з амплітудною модуляцією при різних несучих частотах здійснено процес впливу завади типу білого шуму на амплітудно-модульовані реалізації шляхом її адитивного додавання.

Основним параметром завадостійкості каналу зв'язку є відношення сигнал-шум (SNR), яке обчислюється виразом:

$q = 10\log\left(\frac{E_{\text{ВИХІДНОГО}}}{A_{\text{ВИХ. ШУМУ}}}\right)$, де $E_{сигналу}$ – енергія сигналу, $E_{вих.шуму}$ – енергія вихідного шуму.

Із отриманої залежності (відношення сигнал-шум (SNR)) встановлено, що при збільшенні середньоквадратичного відхилення білого шуму відношення сигнал/шум як показник завадостійкості каналу зменшується із різними значенням частот несучого коливання.

Література.

1. Дунець В. Л. Метод оптимального виявлення сигналів в каналах зв'язку / В.Л. Дунець, Т.І. Цимбала, Р.В. Ракуш // Збірник тез доповідей V Міжнародної науково-технічної конференції молодих учених та студентів „Актуальні задачі сучасних технологій", 17–18 листопада 2016 року. – Т. : ТНТУ, 2016. – Том II. – С. 37–38.