КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра обчислювальної математики

Кваліфікаційна робота
на здобуття ступеня бакалавра

за спеціальністю 113 Прикладна математика

на тему:

РОЗПІЗНАВАННЯ МАТЕМАТИЧНИХ ВИРАЗІВ ЗА ДОПОМОГОЮ ПІДХОДІВ МАШИННОГО НАВЧАННЯ

Студента 4-го курсу
кафедри обчислювальної математики
Завади Сергія Олександровича

_____

Науковий керівник:
асистент
Денисов Сергій Вікторович

_____

«____» _____ 2021 р.

Роботу розглянуто й допущено до захисту на засіданні кафедри обчислювальної математики «____» _____ 2021 р., протокол № ___

Завідувач кафедри Ляшко С. І.

_____

Київ – 2021

ЗМІСТ

ВСТУП

Математичні формули часто зустрічаються в багатьох галузях науки, техніки, техніки та математики. Можливість витягувати математичні формули з цифрових документів та перекладати їх на мови розмітки дуже корисна для широкого кола завдань пошуку інформації. PDF - це фактично стандартний формат для різних типів публікації, що робить розповсюдження документів дуже простим та надійним. Хоча людина порівняно легко може розпізнати математичні формули, комп'ютерне розпізнавання математичних формул у документах PDF залишається головною проблемою. Це головним чином тому, що формат PDF не містить інформації про його математичний вміст. Розпізнавати математичні формули з документів PDF складно через наявність незвичних математичних символів та складних структур макета. Крім того, математичні формули в документах PDF можуть бути частково представлені блоками графічних зображень, безпосередньо відтвореними з гліфів PDF, що зберігає правильні форми, але втрачає текстове представлення токенів. Ці проблеми можна було б легко вирішити, якщо б були доступні джерела розмітки документів PDF. Хорошим прикладом є сховища препринтів arXiv.org, які надають читачам доступ до файлів LaTeX разом із файлами PDF, але вони містять лише незначну частину існуючих цифрових публікацій. Для переважної більшості цифрових документів необхідні передові методи перекладу математичного вмісту PDF у джерела розмітки. Будучи структурованою математичною мовою опису, LaTeX може використовуватися для отримання математичних формул і може бути легко перетворений в інші формати, такі як MathML, для підтримки додатків високого рівня. Найдавнішими зусиллями, датованими 1967 р, були розроблені різні підходи для відновлення математичного змісту з різним рівнем успіху. Нещодавні досягнення в технологіях оптичного розпізнавання символів (OCR) дозволили розпізнавати текст у цифрових документах з високою точністю. Однак розпізнавати математичні формули важко,

оскільки, крім розпізнавання окремих математичних символів, необхідно також визначати структурні взаємозв'язки між символами, такими як верхні/нижні індекси, вкладені дроби, матриці тощо. Дослідники розробили правила на основі методів структурного аналізу та синтаксичних аналізаторів для перетворення математичних формул у мови розмітки. Одним з успішних прикладів є система INFTY, яка була розроблена для перетворення документів у структуровані формати, такі як LaTeX, а згодом перетворена на комерційне програмне забезпечення під назвою InftyReader для цифрової обробки документів. Переклад зображень математичних формул у послідовності LaTeX - це перетин галузей обробки зображень та обробки тексту, яке нещодавно набуло підвищеного інтересу у спільноті глибокого навчання. Модель послідовності до послідовності (seq2seq), яку також називають архітектурою кодера-декодера, успішно застосована на перетині цих галузей. Кодер для таких додатків, як правило, є згортковою нейронною мережею (CNN), яка кодує вхідні зображення у вигляді абстрактних характеристик, а декодер, як правило, є рекурентною нейронною мережею (RNN), яка представляє мовну модель для переведення вихідних даних кодера в послідовність лексем. Ця архітектура робить розмір вхідних зображень та вихідних послідовностей гнучкими. Модель Seq2seq була успішно використана в задачах створення субтитрів до зображень. Нещодавно автори [2] успішно застосували модель seq2seq на основі уваги для перекладу зображень на LaTeX, що продемонструвало здатність моделі обробляти структурний вміст. Використовуючи успіх у моделі [1], у цій роботі розглянуто модель seq2seq під назвою MI2LS[22] (Math Image to LaTeX Sequence), яка зосереджена на вирішенні трьох ключових проблем, які не досліджувались у попередніх роботах. По-перше, щоб допомогти моделі краще диференціювати двовимірну просторову взаємозв'язок математичних символів, автори запропонували розширити карти характеристик зображення, додавши синусоїдальне позиційне кодування для більш широкого представлення просторової інформації про місцевість. По-друге,

запропоновано цільову функцію на рівні послідовності, засновану на оцінці BLEU [10], яка може краще охопити взаємозв'язок між різними маркерами в послідовності LaTeX, ніж перехресної ентропії на рівні токена. Знаючи, що оцінка на рівні послідовності є дискретною і недиференційованою, автори у розглянутій роботі пропонують вирішити проблему оптимізації на основі алгоритму Gradient Policy [11] при навчанні підкріплення для навчання моделі.

РОЗДІЛ 1

1.1 Структура та призначення нейронних мереж згортки

Згорткові нейронні мережі застосовуються досить широко і в різних областях. Першим і, по суті, найбільш тривіальним завданням, яке навчилися вирішувати за допомогою згорткових нейронних мереж, стала класифікація зображень. Але, по суті, можна класифікувати будь-які сигнали.

CNN складається з шарів входу та виходу, а також із декількох прихованих шарів. Приховані шари CNN складаються зі шарів згортки, агрегації, та повноз'єднаних шарів та шарів нормалізації.

Шари згортки застосовують до входу операцію згортки, передаючи результат до наступного шару. Згортка імітує реакцію окремого нейрону на зоровий стимул. Кожен згортковий нейрон обробляє дані лише для свого рецептивного поля. Хоч повноз'єднані нейронні мережі прямого поширення й можливо застосовувати як для навчання ознак, так і для класифікування даних, застосування цієї архітектури до зображень є непрактичним. Було би необхідним дуже велике число нейронів, через дуже великі розміри входу, пов'язані з зображеннями, де кожен піксель є відповідною змінною. Наприклад, повноз'єднаний шар для зображення розміром 100 × 100 має 10 000 ваг. Операція згортки дає змогу розв'язати цю проблему, оскільки вона зменшує кількість вільних параметрів, дозволяючи мережі бути глибшою за меншої кількості параметрів. Наприклад, незалежно від розміру зображення, області замощування розміру 5 × 5, кожна з одними й тими ж спільними вагами, вимагають лише 25 вільних параметрів. Таким чином, це розв'язує проблему «затухання» або «вибуху» градієнтів у тренуванні традиційних багатошарових нейронних мереж з багатьма шарами за допомогою зворотного поширення.

Згорткові мережі можуть включати шари агрегування, які об'єднують виходи кластерів нейронів одного шару до одного нейрону наступного шару. Наприклад, агрегування принципом максимізації (max pooling) використовує

максимальне значення з кожного з кластерів нейронів попереднього шару. Іншим прикладом є агрегування принципом усередненя (англ. average pooling), що використовує усереднене значення з кожного з кластерів нейронів попереднього шару.

Повноз'єднані шари (Fully connected) з'єднують кожен нейрон шару з кожним нейроном наступного шару. Це, в принципі, є тим же, що й традиційна нейронна мережа багатошарового перцептрону (БШП).

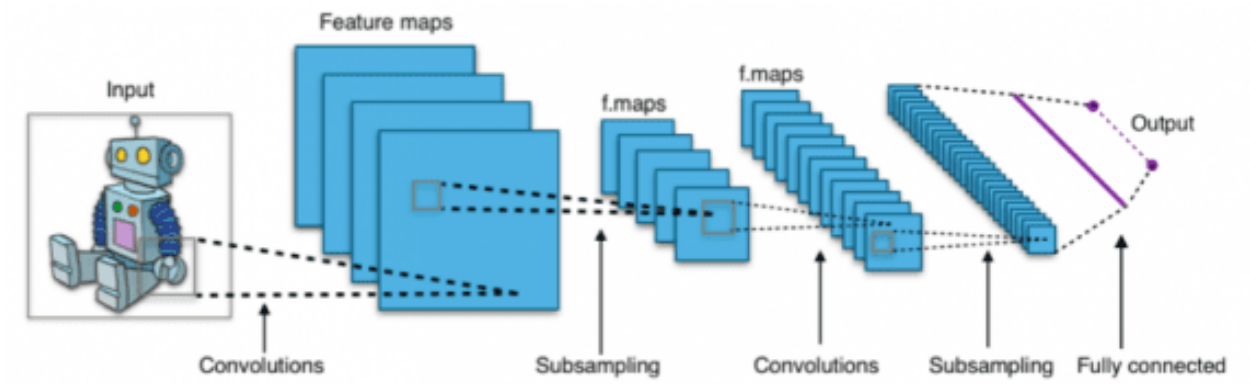

Рисунок 1.1.1 – Схема CNN

1.2 Структура та призначення LSTM

Довга короткочасна пам’ять (LSTM) , рисунок 1.1 - це архітектура штучної рекурентної нейронної мережі (RNN), що використовується в галузі глибокого навчання. На відміну від стандартних нейронних мереж прямого зв'язку, LSTM як різновид нейронної мережі RNN має зворотній зв'язок, за допомогую якого реалізується механізм пам’яті. Він може обробляти не тільки окремі точки даних (наприклад, зображення), але й цілі послідовності даних (наприклад, мовлення або відео). LSTM може бути застосованний до таких завдань, як розпізнавання рукописного вводу, розпізнавання мови та виявлення аномалій у мережевому трафіку або IDS (системи виявлення вторгнень). Мережі LSTM добре підходять для класифікації, обробки та прогнозування на основі даних часових рядів, оскільки між важливими подіями в часових рядах можуть бути затримки невідомої тривалості. LSTM були розроблені для вирішення проблеми зникаючого градієнта, з якою можна зіткнутися при навчанні традиційних RNN. Відносна нечутливість до

довжини послідовності є перевагою LSTM перед RNN, прихованими марковськими моделями та іншими методами навчання послідовності.

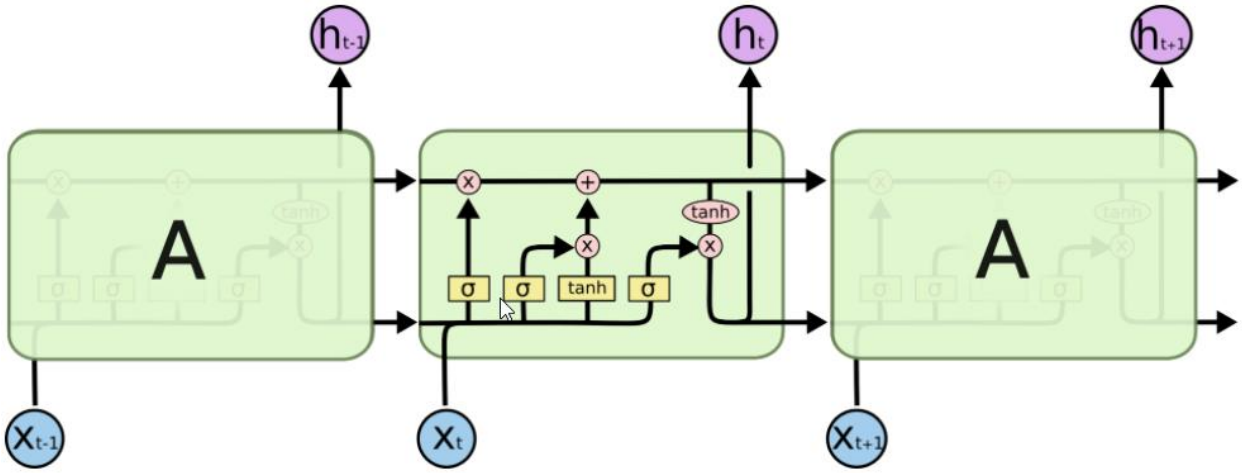

Рисунок 1.2.1

Основна ідея LSTM – це стан комірки (cell state) (рис. 1.2). Саме завдяки цього стану LSTM запам’ятовую довгострокові залежності.

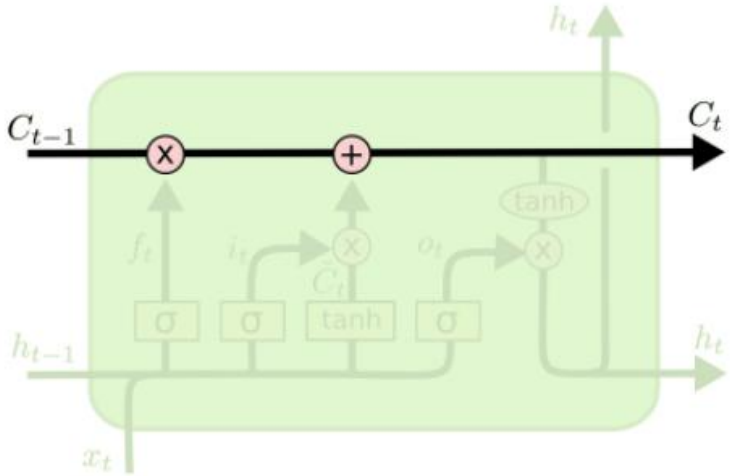

Рисунок 1. 2.2

Далі розглянемо принцип роботи LSTM в середині комірки. Перший крок в LSTM полягає у визначенні яку інформацію можна викинути зі стану комірки C_{t-1}, це рішення приймає сигмоїдний шар, який прийнято називати « шар фільтра забування» рис 1.3.

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

f_t приймає значення від 0 до 1, де 1 – «повністю запам’ятати», 0 – «повністю забути».

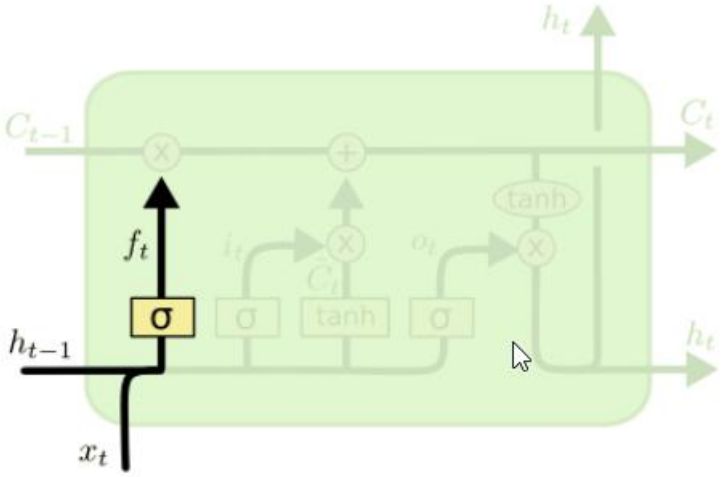

Рисунок 1. 2.3

Наступним крок (рис 1.4) полягає у вирішенні питання, яка інформація буде збережена у комірки стану. Цей крок складається з двох частин. Спочатку сигмоїдний шар під назвою «шар вхідного фільтра» визначає які значення треба оновити i_t. Потім *tanh*-шар , будує вектор $\widetilde{C_t}$, який може бути доданий до стану комірки.

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

$$\widetilde{C_t} = \tanh(W_C * [h_{t-1}, x_t] + b_C)$$

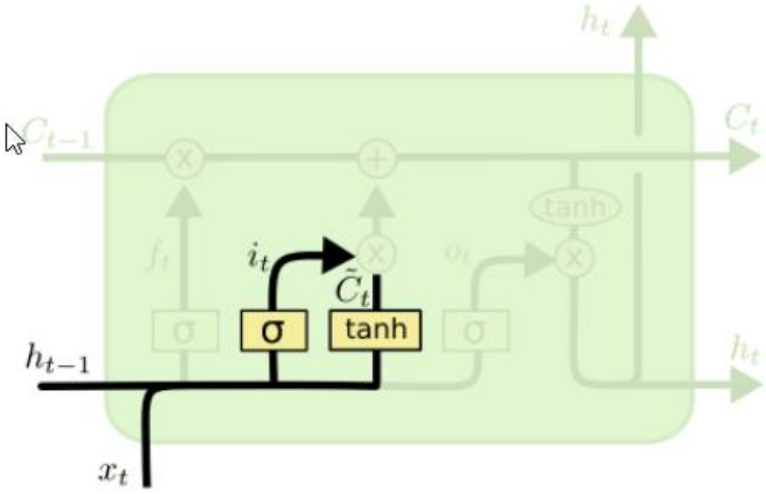

Рисунок 1. 2.4

Наступним кроком генеруємо стан комірки C_t рис 1.5.

$$C_t = C_{t-1} * f_t + \tilde{C}_t * i_t$$

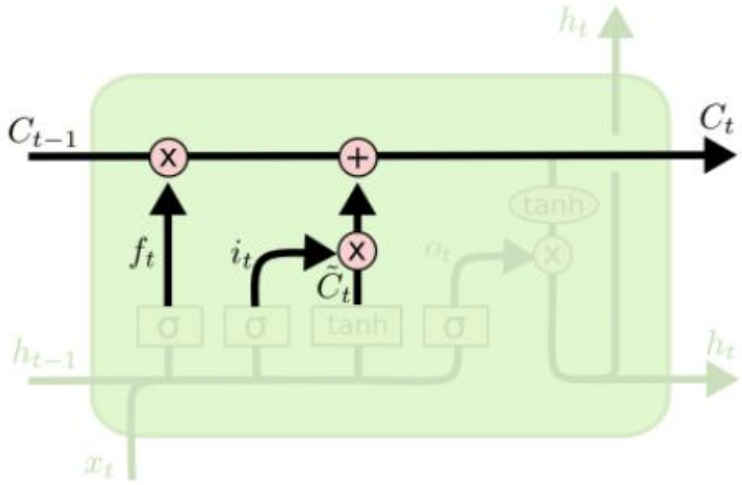

Рисунок 1. 2.5

На останньому кроці генеруємо вектор, який буде результатом роботи комірка. Вихідні данні генеруються з вектора стану комірки до якого застосовані деякі фільтри. Спочатку застосовуємо сигмоїдальний шар, котрий вирішує, яку інформацію зі стану комірки потрібно вивести, потім значення стану комірки проходить через tanh-фільтр, щоб отримати на виході значення від -1 до 1 та множиться з вихідними значеннями сигмоїдного шару (рис. 1.6).

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

$$h_t = o_t * \tanh C_t$$

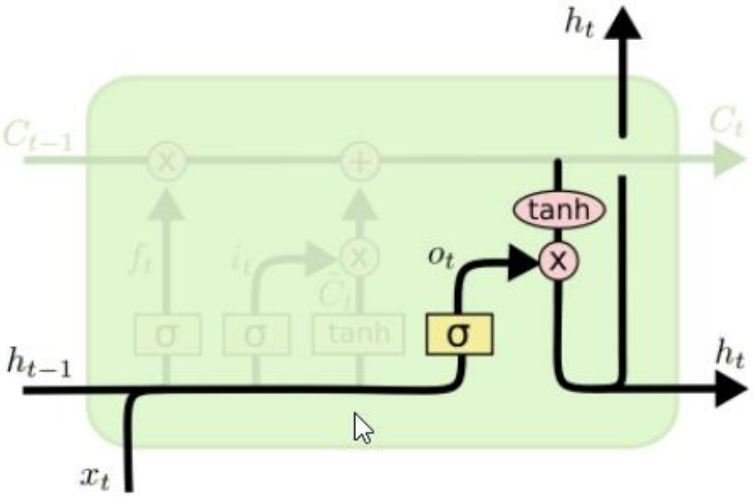

Рисунок 1. 2.6

РОЗДІЛ 2

2.1 Архітектура нейронної мережі WYGIWYS

Модель WYGIWYS для завдання розпізнання математичних виразів складається з кількох стандартних нейронних компонентів машинного зору та обробки природної мови. Спочатку мережа витягує просторові характеристики зображення за допомогою згорткової нейронної мережі (CNN), параметри моделі такі як і у кодера представленого у пункті 2.2. , і впорядковує елементи в сітці. Потім кожен рядок кодується за допомогою рекурентної нейронної мережі (RNN). Потім закодовані характеристики використовуються декодером RNN із механізмом зорової уваги (детальніше у пункті 2.2). Дешифратор реалізує мовну модель над словником V, пошук цільової функції на рівні токена базується на методі максимальної правдоподібності (детальніше у пункті 2.2). Повна структура проілюстрована на рисунок 2.1.1.

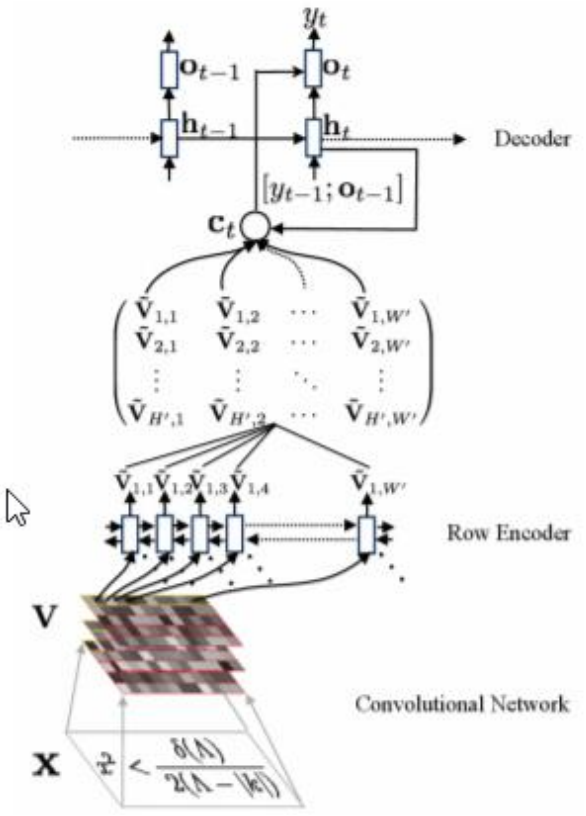

Рисунок 2.1.1 – Архітектура мережі WYGIWYS

2.2 Архітектура нейронної мережі MI2LS

У цьому розділі представлено формулювання постановки задачі. Далі представлено запропоновану архітектуру seq2seq моделі [22] як показано на рисунку 2.2.1, і пояснення структури енкодера, який є згортковою нейронною мережею, доповненою позиційним кодуванням, і декодер, який є стековою двоспрямованною довгою короткочасною пам'яттю (LSTM). Врешті-решт, пояснений м’який механізм уваги.

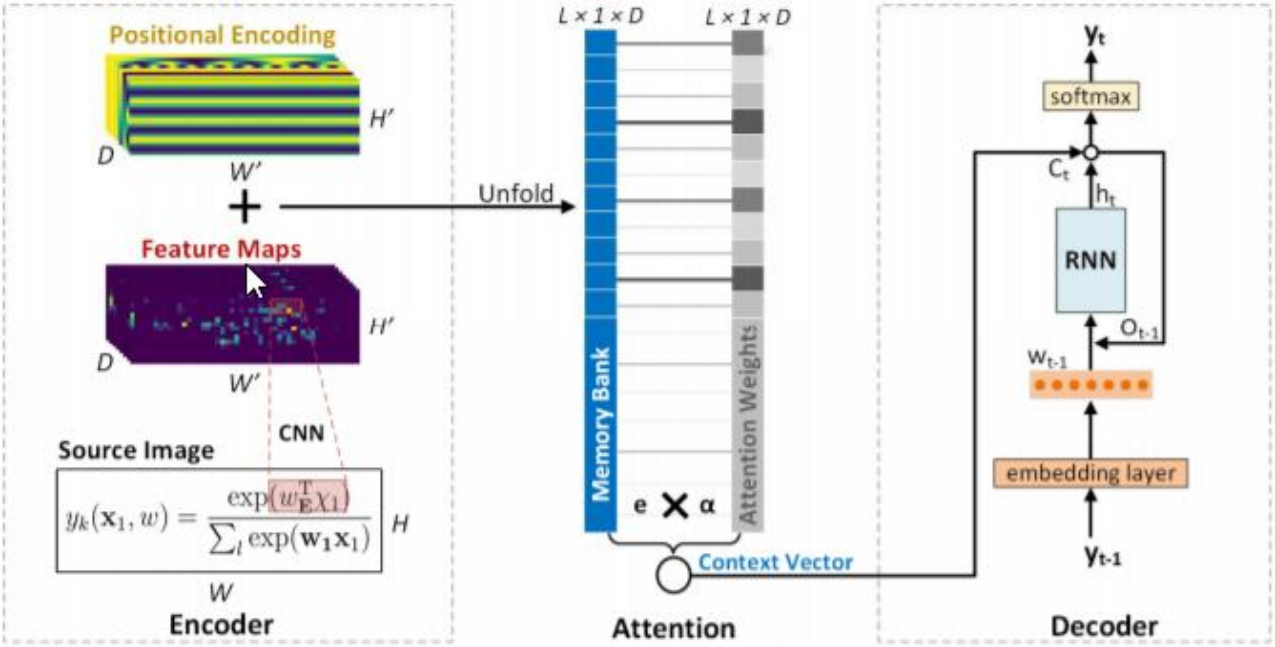

Рисунок 2.2.1 - Архітектура нейронної мережі MI2LS

2.2.1 Постановка задачі

Проблема розпізнавання математичної формули сформульована як задача передбачення послідовності. Нехай (x, y) - пара послідовностей зображення-LaTeX. $x \in R^{H \times W}$- зображення у відтінках сірого з висотою H та шириною W. $y = [y_1, y_2, \dots, y_T]$ – послідовність LaTeX, що складається з лексем, що позначає математичну формулу на зображенні. x може бути відтворений з y за допомогою стандартного компілятора TeX. Основа мета задачі полягає у відтворенні y з відомого зображення x, тобто у знаходженні такої функції $f(x)$, що $f(x) \rightarrow y$. Враховуючи набір з N пар істинних образів

зображень-LaTeX $G = \{x^i, y^i\}_{i=1}^N$, використовуємо контрольований тренінг для побудови послідовності функцій $\hat{f}$, що апроксимують f. Під час тестування використовуємо $\hat{f}(x) \rightarrow \hat{y}$ для прогнозування послідовності LaTeX y, яка реконструює вхідне зображення x. Оцінка виконується шляхом вимірювання подібності між $\hat{y}$ та істинною послідовністю y, а також між відтвореним зображенням $\hat{x}$ та істинним зображенням x.

2.2.2 Енкодер

Енкодер використовується для кодування вхідних зображень в абстрактні подання об'єктів. Він складається із згорткової нейронної мережі (CNN) та позиційного кодування.

2.2.2.1 Згорткова нейронна мережа

Ми використовуємо CNN для вилучення образів із вхідних зображень. CNN складається з шарів згортки, об'єднання та активації. На кожному шарі згортки вхідне зображення згортається з набором ядер, які діють як фільтри зображень. Значення ядра піддаються навчанню, що робить фільтри для пошуку образів залежними від даних, тому їх не треба генерувати вручну. Шар об'єднання зазвичай складається з функції максимального об'єднання або середньої функції об'єднання, що зменшує розмір зображення та збільшує розмір сприйнятливого поля, що дає змогу шукати більш абстрактні образи. Активаційний рівень додає нелінійність мережі. Зазвичай це ReLU, яка замінює негативні входи на 0 і зберігає позитивні входи без змін. Використаємо архітектуру CNN, засновану на VGG-VeryDeep, яка була спеціально адаптована для програм OCR. Детальніше про конфігурацію CNN можна знайти в ТАБЛИЦІ 1. Карти об’єктів перетворені на 2D-матрицю замість сплощеного вектора об’єктів, щоб зберегти інформацію про просторову місцевість, як показано на рисунку 1. Ця практика також дозволяє моделі приймати на вхід зображеня довільного розміру. При даній конфігурації CNN, де ширина W’, і висота H’ розмір вихідних карт образів у

8 разів менше, ніж у вхідного зображення, і кожна комірка є вектором розмірності D (D = 512 у нашій конфігурації).

Таблиця 1. Структура CNN для енкодера, maps – кількість фільтрів для шару згортки, k – розмір комірки ядра, p – розмір заповнення після згортки, s – крок, BN – порційна нормалізація. Розмір у дужках – (висота, ширина)

Type	**maps**	**k**	**p**	**s**
BN	-			
Convolution	512	(3,3)	(1,1)	(1,1)
MaxPooling		(2,1)	(0,0)	(2,1)
BN	-			
Convolution	512	(3,3)	(1,1)	(1,1)
MaxPooling		(1,2)	(0,0)	(1,2)
Convolution	256	(3,3)	(1,1)	(1,1)
BN	-			
Convolution	256	(3,3)	(1,1)	(1,1)
MaxPooling		(2,2)	(0,0)	(2,2)
Convolution	128	(3,3)	(1,1)	(1,1)
MaxPooling		(2,2)	(0,0)	(2,2)
Convolution	64	(3,3)	(1,1)	(1,1)
Input	Gray-scale image			

2.2.2.2 Позиційне кодування

Для розпізнавання тексту можна просто розгорнути карти образів (feature maps) з енкодера в масив і подати його в декодер RNN, не враховуючи явно просторову локалізацію, оскільки RNN здатний фіксувати порядок розташування зліва направо. Однак у математичних формулах просторовий зв’язок між символами охоплює різні напрямки: ліворуч-праворуч, зверху-вниз, підпис і напис, вкладення тощо. Позиційні відносини між математичними символами несуть критичну математичну семантику. Таким чином, необхідні особливі зусилля для збереження просторової

компоненти. У моделі застосовується 1-D позиційна техніка кодування, запропонована в моделі Transformer [21], до 2-D наступним чином:

$$PE(x, y, 2i) = \sin\left(x\ /10000^{4i/D}\right)$$

$$PE(x, y, 2i + 1) = \cos\left(x\ /10000^{4i/D}\right)$$

$$PE(x, y, 2j + D/2) = \sin\left(y\ /10000^{4j/D}\right)$$

$$PE(x, y, 2j + 1 + D/2) = \cos\left(y\ /10000^{4j/D}\right),$$

Де x та y визначають горизонтальне та вертикальне положення, а $i, j \in [0, \frac{D}{4})$ визначають розмірність. Ці характеристики додаються до карт образів (feature maps).

Позиційне кодування має такий самий розмір і розмірність, як і карти образів. Кожен вимір позиційного кодування складається з синусоїдального сигналу певної частоти та фази, що представляє або горизонтальний, або вертикальний напрямки. Використаємо шкалу часу від 1 до 10000. Кількість різних часових шкал дорівнює D / 4, що відповідає різним частотам. Для кожної частоти генеруємо синусоїдальний / косинусний сигнал у горизонтальному / вертикальному напрямку. Перша половина розмірів кодує горизонтальні положення, а друга половина - вертикальні положення.

Позиційне кодування та карти образів додаються разом, а потім розгортаються в одновимірний масив $\vec{E} \in R^{L\times 1\times D}$, де $L = H' \times W'$ - довжина масиву. Кожен вектор $e_i \in \vec{E}$ має розмірність D. Кожен такий вектор відповідає певній частині вхідного зображення. Зазначимо, що ця техніка кодування позиції має ту перевагу, що не додає нових тренованих параметрів до нейронної мережі. Крім того, у порівнянні з тренувальним позиційним представленням, синусоїдальне кодування може бути масштабовано до довжини, що перевищує ту, що є в навчальних даних.

2.2.3 Декодер

RNN добре підходить для завдань передбачення послідовності, оскільки вона зберігає історію попередніх передбачень і здатна переходити

від початку до кінця послідовності довільної довжини. Нехай $\vec{V}$- це словник, що містить усі дозволені токени LaTeX. Використаємо RNN для апроксимації мовної моделі $p(y_t|y_1,\dots,y_{t-1},\vec{E})$, яка робить прогноз щодо розподілу ймовірності токена $y_t \in \vec{V}$ в момент t на основі історії прогнозування $y_{i<t}$ та виводу енкодера $\vec{E}$. Далі введемо представлення лексем і структуру RNN.

2.2.3.1 Кодування LaTeX токенів

Припустимо, що джерело LaTeX вже розділено на маркери $y_1, y_2, \dots, y_T$. Токен може подаватися в RNN у різних поданнях. Найпростішим варіантом є подання кожного токена як унітарний код (one-hot vector), що означає, що токени ортогональні один одному, і, отже, при такому кодуванні токенів можемо пропустити важливу мовну семантику. Подібно до слів природної мови, багато токенів LaTeX взаємопов'язані. Наприклад, '{' та '}' можуть мати вищу кореляцію, оскільки їх потрібно використовувати в парі на основі граматики LaTeX. В результаті у роботі [22] запропоновано додати шар кодування слів, який зазвичай використовується в NLP, де токен y_t проектується у вектор великої розмірності w_t:

$$w_t = embedding(y_t)$$

Це кодування піддається навчанню і здатне охопити взаємозв'язок між різними лексемами.

2.2.3.2 Двоспрямована LSTM

У мережі використовується модель декодера, засновану на двох шарах двонаправлених комірок довгострокової пам'яті (LSTM). Стекова двонаправлена LSTM збільшує глибину RNN і, таким чином, допомагає захопити більш складну семантику мови. Використання двонаправлених комірок у кожному шарі допомагає охопити контексти як у прямому, так і в зворотному напрямку між маркерами. Рисунок 2.2.3.2.1(a) показує структуру стекової двоспрямованної LSTM, яка використовувалась. Для зручності просто будемо називати цю мережу RNN.

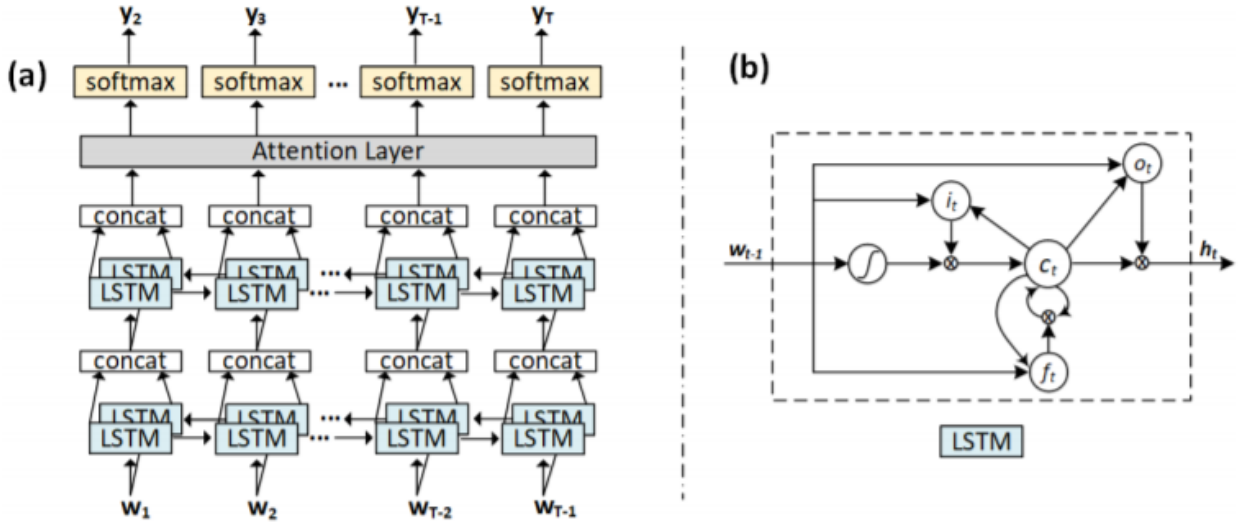

Рисунок 2.2.3.2.1 – LSTM шари (a), LSTM комірки (b)

LSTM краще обробляє довгі послідовності, ніж стандартна RNN, як має проблему зникаючого градієнта зі збільшенням довжини послідовності. На Рисунок 2.2.3.2.1(b) показано структуру комірки LSTM. Ядром LSTM є стан комірки c_t, який реєструє інформацію, яка спостерігалася під час t. LSTM здатний додавати або видаляти інформацію із стану комірки через три типи шлюзів: вхідний шлюз i_t, шлюз забування f_t і вихідний шлюз o_t. Як випливає з їх назв, ці вектора контролюють зчитування поточного входу, забуття поточного значення стану комірки або вихід поточного значення комірки. Кожен вектор є результатом роботи шарів нейронної мережі с сигмоїдною функцією активації і поточкового множення, вираженого, як показано нижче:

$$i_t = \sigma(W_{ix}w_{t-1} + W_{ih}h_{t-1})$$
$$f_t = \sigma(W_{fx}w_{t-1} + W_{fh}h_{t-1})$$
$$o_t = \sigma(W_{ox}w_{t-1} + W_{oh}h_{t-1})$$
$$c_t = f_t * c_{t-1} + i_t * \tanh(W_{cx}w_{t-1} + W_{ch}h_{t-1})$$
$$h_t = o_t * c_t$$

де h_t являє собою прихований стан RNN в момент часу t, σ - сигмоїдна функція, а W - матриця вагів

У додатках NLP початковий прихований стан і стан комірки декодера зазвичай є вихідними даними кодера RNN. Однак у нашій моделі кодером є

CNN, який не дає такого виходу. Для того, щоб отримати інформативні початкові стани для декодера RNN, у моделі додаються додаткові шари для підготовки початкових станів на основі вихідних даних кодера, як показано нижче:

$$h_0 = \tanh(W_h\left(\frac{1}{L}\sum_{i=1}^{L}\overrightarrow{e_i}\right) + b_h)$$

$$c_0 = \tanh(W_c\left(\frac{1}{L}\sum_{i=1}^{L}\overrightarrow{e_i}\right) + b_c)$$

2.2.4. Увага

Теоретично LSTM можна масштабувати для захоплення довготривалої пам'яті за необхідності. Однак нерідкі випадки, коли розмітка складної математичної формули охоплює понад сотню токенів LaTeX. У таких випадках початкового вектора прихованого стану в RNN буде недостатньо для стиснення всієї інформації з енкодера. Ця проблема є ще більш помітною у цій моделі, оскільки кодер CNN не має пам'яті. Для вирішення цієї проблеми був запроваджений механізм уваги, який став широко прийнятим підходом для підвищення продуктивності довгих послідовностей. По суті, він отримує повний вихід енкодера, а саме банк пам'яті $\vec{E}$, на основі якого обчислюється вектор контексту C_t для декодера на кожному кроці t часу. У роботі використаний механізм м'якої уваги, що означає, що контекстний вектор C_t обчислюється як лінійна комбінація векторів $\vec{e} \in \vec{E}$ в банку пам'яті:

$$C_t = \sum_{i=1}^{L}\alpha_{it}e_i,$$

Де $\alpha_{it} -$ це i-та вага в момент часу t.

Ваги уваги обчислюються за допомогою додаткового шару прямої передачі шляхом подачі в попередній прихований стан LSTM h_{t-1} та банк пам'яті $\vec{E}$, а потім передають його через шар softmax для нормалізації:

$$a_{it} = \beta^T \tanh(W_1 h_{t-1} + W_2 e_i)$$

$$\alpha_{it} = \text{softmax}(a_{it})$$

,де функція softmax використовується для генерування розподілу ймовірностей, визначеного як

$$\text{softmax}(a_{it}) = \frac{e^{a_{it}}}{\sum_{k=1}^{L} e^{a_{kt}}}$$

Вагові коефіцієнти уваги вказують, на які частини банку пам'яті слід зосередитись на поточному етапі часу, що допомагає моделі краще захопити виділені частини вхідного зображення.

Щоб включити інформацію контекстного вектора C_t в RNN, у моделі обчислюється інший прихований вектор стану O_t на основі контексту вектора C_t і поточного прихованого стану h_t. O_t називається вектором прихованого стану уваги, який подається назад на наступний крок часу RNN. Він також використовується для обчислення розподілу ймовірностей наступного токена:

$$O_t = \tanh(W_3[h_t, C_t])$$

У моделі також застосовується підхід подачі вхідних даних, в якому вхідний вектор кодування поєднується з вектором уваги з попереднього кроку часу як вхід для RNN. Таким чином, рішення приймаються з урахуванням минулої інформації про вирівнювання.

$$h_t = RNN(h_{t-1}, [w_{t-1}, O_{t-1}])$$

Ймовірність прогнозування стає:

$$p(y_t) = \text{softmax}(W_4 O_t)$$

що представляє розподіл ймовірності наступного маркера над словником $\vec{V}$.

2.2.5. Начання нейронної мережі

Ідеальна цільова функція повинна бути побудована на рівні послідовності через обмеження граматики LaTeX. Зауважимо, що алгоритм зворотного розповсюдження помилки (backpropagation) повинен слідувати напрямку градієнта цільової функції для оновлення параметрів моделі. Таким чином, окрім точного вимірювання якості прогнозування, дуже бажано, щоб

цільова функція була диференційованою. У цьому пункті описано запропонований у моделі дизайн цільової функції на рівні послідовності та методи обчислення її похідної на основі алгоритму Policy Gradient. Також зазначимо, що неможливо навчити нейронну мережу з цільовою функцією на рівні послідовності з випадкового початку, оскільки нейронна мережа може не збігатися в умовах поганого випадкового прогнозування. Щоб подолати ці проблеми, почнемо з навчання нейронної мережі з цільової функції на рівні токена, поки вона не сходиться. Це формує початковий стан для навчання на рівні послідовності, оскільки така модель може зосередитись на значно меншому просторі пошуку.

2.2.5.1 Цільова функція на рівні токена

Пошук цільової функції на рівні токена базується на методі максимальної правдоподібності (maximum likelihood estimation). Даний навчальний набір є послідовністю пар $\{x^i, y^i\}_{i=1}^N$ довжиною N, де x^i, y^i є i-тим вхідним зображенням та послідовністю LaTeX токенів відповідно, цілю навчання є пошук набору параметрів θ, таких що максимізують логарифм правдоподібності на навчальному наборі:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\{L_{MLE}(\theta)\},$$

Де

$$L_{MLE}(\theta) = \sum_{i=1}^{N} p(x^i, y^i) = \sum_{i=1}^{N}\sum_{t=1}^{T} p(y_t^i | y_1^i, y_2^i, \dots, y_{t-1}^i, x^i)$$

Це еквівалентно мінімізації перехресної ентропії (XENT):

$$L_{XENT}(\theta) = -\frac{1}{N}\left(\sum_{i=1}^{N} y^i * \log(\hat{y}^i)\right)$$

Де $\hat{y}$ – передбачення. Похідна перехресної ентропії може бути безпосередньо використана як градієнт.

Цільова функція токена має два обмеженнями. По-перше, вона максимізує ймовірність наступного правильного токена, не враховуючи

контексти на послідовності. По-друге, щоб уникнути проблеми зі зміщенням токена, токен с датасету потрібно подавати в RNN на кожному кроці під час навчання, замість того, щоб використовувати попередній прогноз RNN. Однак під час прогнозування попереднє передбачення від RNN використовується як наступний вхід, оскільки дані про істинність токенів вже недоступні. Як результат, розподіл ймовірностей, під час тренування, становить $p(y_t|y_{i<t}, \vec{E})$, але розподіл ймовірностей під час тестування має вигляд $p(y_t|\hat{y}_{i<t}, \vec{E})$. Тобто не має даних для того щоб передати в наступний шар, тому необхідно використовувати прогноз попереднього шару, проблема стає більш суттєвою, якщо попередній шар повертає поганий прогноз, що безпосередньо впливає на всю послідовність. Ця розбіжність відома як проблема «exposure bias». Тренування цільової функції на рівні послідовності спрямоване на подолання цих проблем.

2.2.5.1 Цільова функція на рівні послідовності

Формулювання цілі навчання на рівні послідовності починається з її метрики продуктивності на рівні послідовності. Нехай (x^i, y^i) і-та пара для тренування, а $\hat{y}^i$ - прогноз. Нехай $R(x^i, y^i) \rightarrow [0,1]$ функція, яка зіставляє прогнозовану послідовність з скалярною нагородою, де більші значення вказують на кращу продуктивність. $R(x^i, y^i)$ може бути оцінкою BLEU або іншою метрикою. Ціль оптимізації - максимізувати очікувану нагороду на наборі даних:

$$L_R(\theta) = \sum_{i=1}^{N} E_{p_\theta(\hat{y}^i|x^i)}\left[R(y^i, \hat{y}^i)\right] = \sum_{i=1}^{N} \sum_{\hat{y}^i \epsilon \Upsilon(x^i)} p_\theta(\hat{y}^i|x^i) R(y^i, \hat{y}^i),$$

Де $E(\cdot)$ позначає очікування, а $\Upsilon(x^i)$ множина можливих прогнозованих послідовностей для вхідного зображення x^i. Тоді цілю навчання буде:

$$\hat{\theta}_R = \underset{\theta}{\operatorname{argmax}}\{L_R(\theta)\}$$

Ця цільова функція на рівні послідовності спрямована на оптимізацію передбачення окремих лексем у контексті всієї послідовності. Це також

дозволяє усунути проблему упередженості експозиції, оскільки оптимізація базується вже не на кожному окремому маркері, а на всій послідовності, отже, більше не потрібно подавати токен з датасета на кожному кроці, щоб гарантувати вирівнювання токена. Можемо просто закрити цикл зворотного зв'язку, подаючи передбачуваний токен замість токена з датасета на наступний крок часу під час навчання.

Зауважемо, що оптимізувати $L_R(\theta)$ на основі вичерпного пошуку внаслідок експоненціального зростання простору пошуку $\Upsilon(x^i)$ обчислювально неможливо. Тим часом градієнтний спуск тут безпосередньо не застосовується, оскільки функція винагороди $R(y^i, \hat{y}^i)$ є дискретною функцією прогнозу, отже, не диференціюється. Для вирішення цієї проблеми у спільноті NLP були запропоновані рішення, які пропонують сформулювати цю проблему оптимізації як навчальну проблему з підкріпленням. У цьому налаштуванні модель прогнозування трактується як «agent». Прогноз на наступний токен - це «action». Після завершення передбачення передбачена послідовність порівнюється із послідовністю з датасету, щоб отримати оцінку рівня послідовності, яка є винагородою («reward»). Параметри нейронної мережі визначають поведінку. Незважаючи на те, що $L_R(\theta)$ не можна диференціювати, «Policy Gradient» алгоритм може бути використаний для перетворення градієнта очікувань як очікування градієнта, щоб уникнути використання похідної над функцією винагороди:

$$\nabla_\theta L_R(\theta) = \sum_{i=1}^{N} E_{p_\theta(\hat{y}^i|x^i)}\left[R(y^i, \hat{y}^i)\nabla_\theta \log p_\theta(\hat{y}^i|x^i)\right]$$

В принципі, можна використати алгоритм REINFORCE для оцінки вищезазначених сподівань на основі методів вибірки. Зокрема, очікуване значення можна наблизити, взявши одну вибірку з розподілу $\tilde{y} \sim p_\theta(y^i|x^i)$. На жаль, нейронній мережі важко сходитися таким чином через велику дисперсію в оцінці градієнта. Одним із методів зменшення дисперсії є

віднімання середньої винагороди $\bar{r}$ із передбачуваної винагороди. Таким чином, передбачувана похідна стає:

$$\widetilde{\nabla}_\theta L_R(\theta) = \sum_{i=1}^{N} [R(y^i, \hat{y}^i) - \bar{r}] \cdot \nabla_\theta \log p_\theta(\tilde{y}|x^i)$$

Середня винагорода $\bar{r}$ була оцінена шляхом навчання окремого рівня нейронної мережі. У роботі, у якій описана модель, автори просто використовували метод Монте-Карло, щоб оцінити $\bar{r}$, тобто взявши k вибірок з поліноміального розподілу та обчисливши середнє значення. Тепер, коли похідну можна отримати, алгоритм зворотного розповсюдження можна використовувати для навчання на рівні послідовності.

РОЗДІЛ 3

3.1 Попередня обробка навчальної вибірки

Я використовував загальнодоступний набір даних IM2LATEX-100K [2], який побудований на основі джерел публікацій LaTeX, відсканованих з теми фізики високих енергій в розділі теорії arXiv.org. Набір даних містить загалом 103 556 послідовностей LaTeX, що представляють різні математичні формули. Довжина символів кожної послідовності коливається від 38 до 997, середнє значення 118 і медіана 98. Кожна математична формула компілюється у формат PDF за допомогою інструмента pdfLaTeX, а потім перетворюється на зображення у градаціях сірого у форматі PNG із роздільною здатністю 1654 × 2339. Набір даних розділений на навчальний набір із 83 883 формул, набір перевірок з 9 319 формул та набір тестів із 10 354 формул.

Навчання моделі починається з побудови словника $\vec{V}$. Простий підхід до токенізації джерел LaTeX полягає в тому, щоб розглядати кожен окремий символ як токен. Більш складний підхід полягає в аналізі джерел LaTeX на найкоротші зарезервовані слова LaTeX. Наприклад, '\ psi' означає "ψ" у LaTeX, що трактується як один токен, а не як чотири окремі "\", "p", "s", "i". Другий підхід має очевидні переваги зменшення довжини послідовності та уникнення непотрібних помилок прогнозування та обчислень. Однак цей підхід не є тривіальним, оскільки він повинен мати повний перелік зарезервованих слів LaTeX та ефективний алгоритм розбору для сегментації джерел LaTeX. Тут я використовував парсер LaTeX, розроблений у [2]. Цей синтаксичний аналізатор спочатку перетворює джерело LaTeX в абстрактне дерево синтаксису за допомогою KaTex [20], а потім генерує маркери шляхом обходу дерева синтаксису. Також під час цього процесу можна застосувати нормалізацію послідовностей LaTeX. Цей крок нормалізації може зменшити поліморфну неоднозначність LaTeX, оскільки одне і те ж зображення математичної формули може бути створене з різних

послідовностей джерела LaTeX. Також до словника додаються дві лексеми службових програм <START> та <END>, щоб маркувати початок послідовності та кінець послідовності відповідно. Декодер ініціалізується маркером <START> і продовжує робити прогнози, поки не зустріне маркер <END>. У нас виходить словниковий запас розміром 496. Зображення попередньо обробляються шляхом обрізання до області формули, а потім зменшуються до половини їх початкових розмірів для більш ефективного використання пам'яті.

3.2 Критерії оцінки

Для вимірювання точності прогнозування використовуються два типи показників ефективності. Перший - це оцінка BLEU між передбачуваною та істинною послідовністю. BLEU широко використовується для вимірювання якості машинного перекладу природних мов, оцінка BLEU вимірює перекриття N-грамм (n послідовних лексем). Я використав 4-грамовий показник BLEU, який зазвичай використовується в літературі. Друга текстова метрика це відстань Левенштейна для істиної та передбачуваної послідовності. Через неоднозначність граматики LaTeX, x_i^j можна виразити або як x_i^j, або як x^j_i, тому потрібні додаткові метрики для зображень, так як текстові метрику не покривають деякі випадки, наприклад наведений вище. Тому я вимірював подібність між істинним та відтвореним зображенням із передбачуваної послідовності LaTeX на основі двох різних метрик: точна відповідність і точна відповідність після видалення пробілів. На основі цих показників ефективності метод порівнюється з комерційним програмним забезпеченням InftyReader [1] та трьох робіт, заснованих на глибокому навчанні: WYGIWYS [2], Double Attention [3].

3.3 Результати

Нейронна мережа навчалась на GPU Nvidia GeForce GTX 1650 super за допомогою програмного пакета Torch приблизно 36 годин с показником навчання «learning rate» рівним 0,1. Розмір пакету дорівнював 2 (таке

маленьке значення викликано обмеженням розміру доступної оперативної пам’яті відеокарти). Процес навчання був зупинений, так як похибка моделі, а саме похибка перехресної ентропії, припинила зменшуватися. Напевно це сталося із-за того, що при навчанні нейронної мережі методом градієнтного спуску метод знайшов локальний екстремум. Зауважимо, що похибка не може бути рівною 1, це викликано неоднозначністю граматики LaTeX (див. приклад про LaTex представлення x_i^j наведений вище).

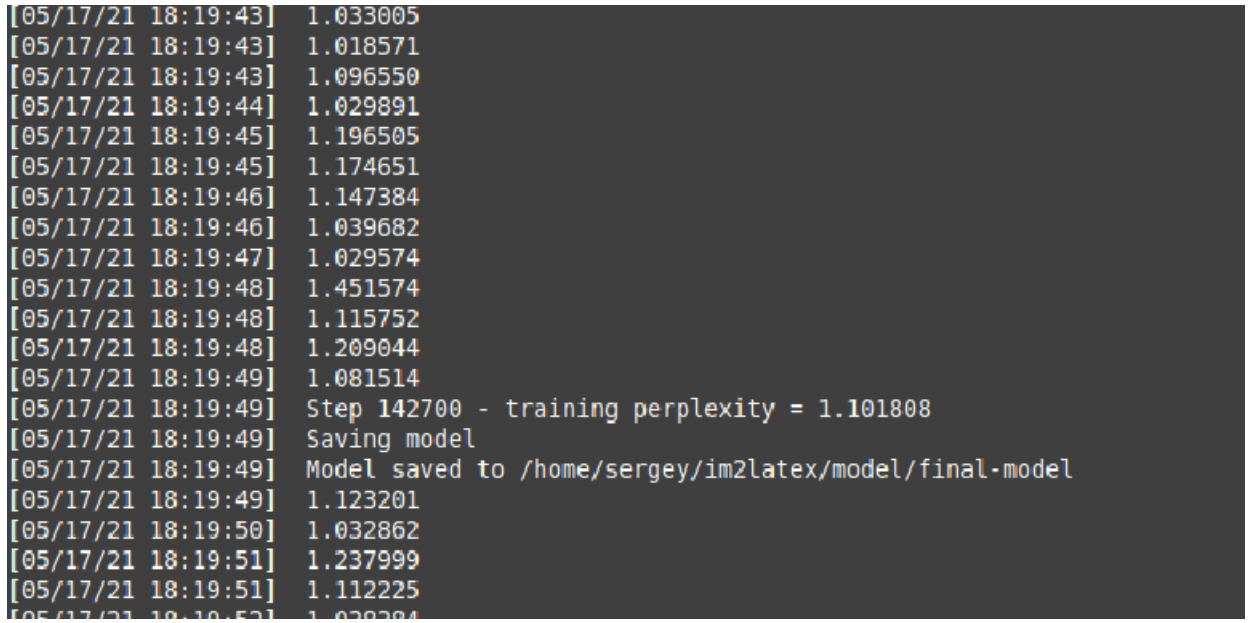

Рисунок 3.3.1 – Навчання нейронної мережі. Момент зупинки

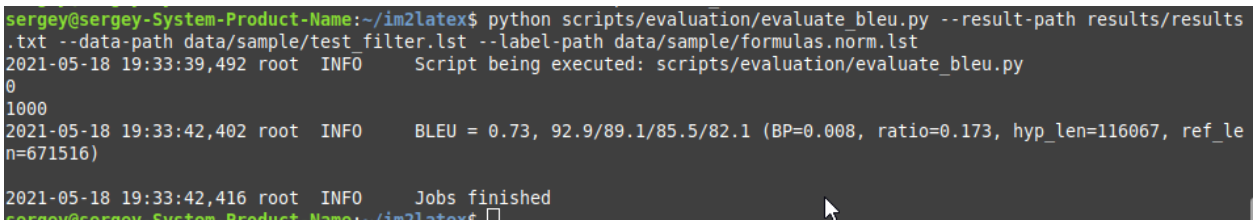

Рисунок 3.3.2 – Оцінка BLEU

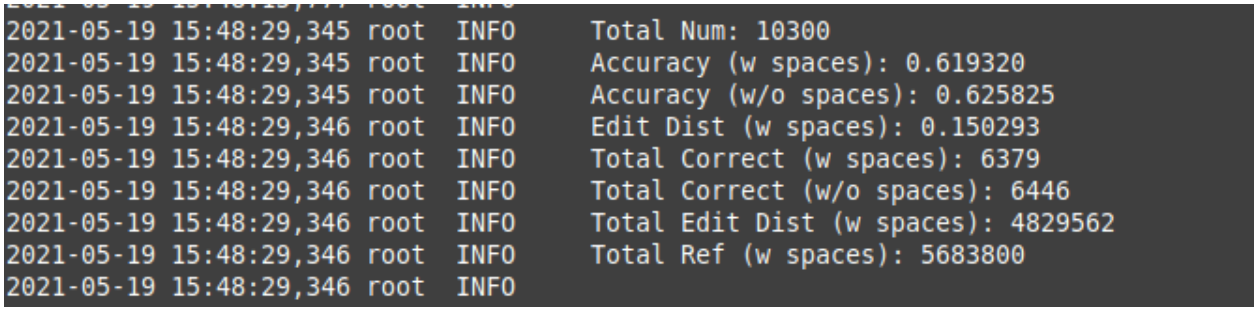

Рисунок 3.3.3 – Оцінка точної відповідності до та після видалення пробілів для оригінальних і відтворених з прогнозованої послідовності зображень

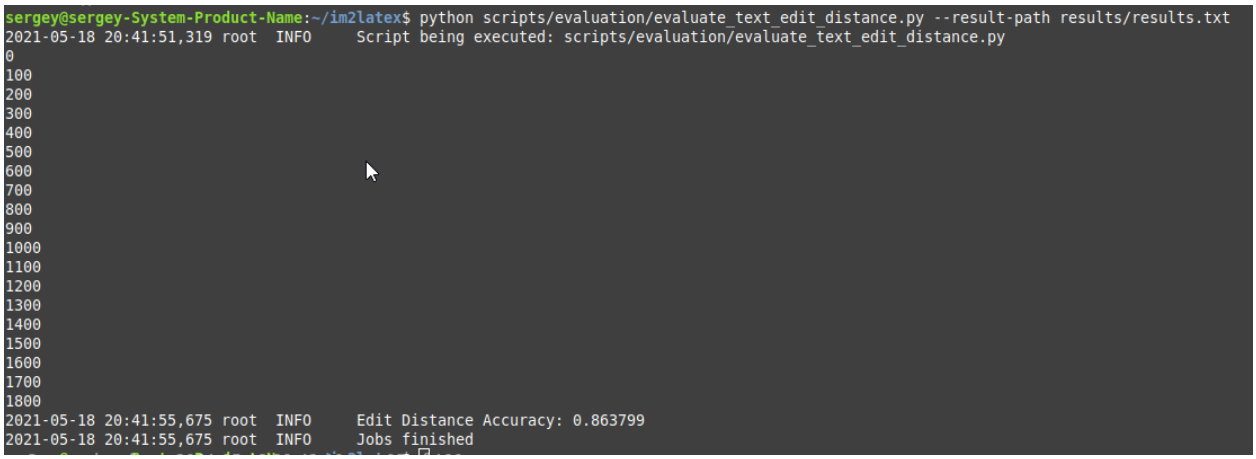

Рисунок 3.3.4 – Оцінка відстані Левенштейна для LaTeX послідовностей

Таблиця 1 – Оцінка ефективності різних моделей навчених на наборі даних IM2LATEX-100K.

Модель	BLEU	Відстань Левенштейна	Точна відповідності	Точна відповідності (-бп)
INFTY [1]	66.65	53.82	15.60	26.66
WYGIWYS [2]	87.73	87.60	77.46	79.88
Double Attention [3]	88.42	88.57	79.81	-
WYGIWYS(моя реалізація)	73	86.37	61.93	62.58
MI2LS with Reinforce	90.28	92.28	82.33	84.79

Основні експериментальні результати з прогнозування математичних виразів наведені в таблиці 1. Ці результати порівнюють кілька різних програмних продуктів декомпіляції зображень у LaTeX послідовності. Класична система INFTY має достатньо високу точності текстових оцінок, але низьке значення метрик зображення. Цільова модель має значно кращі показники оцінки зображень порівняно з INFTY, та прийнятні відносно інших розглянутих моделей.

\mathcal	⇒	-	\, (	⇒	\left(
\right)	⇒	)	\left[	⇒	[
\left(	⇒	(	\left(	⇒	\, (
)	⇒	\right)	\big	⇒	-
\mathbf	⇒	-	\right)	⇒	) \,
(	⇒	\left(	\prime	⇒	-
\mathrm	⇒	-	_ {	⇒	-
\right]	⇒	]	\widetilde	⇒	-
\mathrm	⇒	\operatorname	\bot	⇒	\perp

Рисунок 3.3.5 – Найпоширеніші помилки, що впливають на візуалізацію набору даних LaTeX.

Основні помилки моделі, що не мають інтервалів, наведені на рисунку 3.3.5. Ці приклади показують найпоширеніші лексеми, на яких модель часто помиляється. Більшість помилок виникають із-за проблем із шрифтами, наприклад, використання малих дужок замість великих, або рідких специфічних символів тощо. На рисунках 3.3.6 - 3.3.8 показано кілька прикладів типових помилок. Зазвичай більша частина структури виразу зберігається, але є з одна або дві помилками розпізнавання символів.

gold

$$e^{i\mathbf{k}\cdot\mathbf{r}} = e^{ikr\cos(\theta-\Theta)} = \sum_{l=-\infty}^{\infty} i^l J_l(kr)\, e^{il(\theta-\Theta)},$$

pred

$$e^{i\mathbf{k}\cdot\mathbf{r}} = e^{ikr\cos(\theta-\mathbf{g})} = \sum_{l=-\infty}^{\infty} i^l J_l(kr)\, e^{il(\theta-\mathbf{g})},$$

Рисунок 3.3.6 – Перший приклад помилкового прогнозування

gold

$$\phi'(x) = \langle \phi' | \phi_x \rangle_U = \int_{U_J} \phi'(y)\overline{\phi}_x(y)|\psi_0|^2(y)\exp\big(-\Phi(y)\big)\frac{\omega^n}{n!}(y),$$

pred

$$\phi'(x) = \langle \phi' | \phi_x \rangle_b = \int_{U_J} \phi'(y)\overline{\phi}_x(y)|\psi_0|^2(y)\exp\big(-\Phi(y)\big)\frac{\omega^n}{n!}(y),$$

Рисунок 3.3.7 – Другий приклад помилкового прогнозування

gold

$$K' = \sqrt{c-2f}\ , \quad K'' = -\frac{1}{\sqrt{c-2f}}\ ,$$

pred

$$K' = \sqrt{c-2f}\ , \quad K'' = -\frac{1}{\sqrt{c-2/}}\ ,$$

Рисунок 3.3.8 – Третій приклад помилкового прогнозування

Також були позиційні помилки, незважаючи на те що в при навчанні використовувалось позиційне кодування, це каже про не ідеальність алгоритму позиційного кодування використаного в програмі, і є приводом для подальших досліджень. На рисунку 3.3.9 наведено приклад типової помилки позиційного кодування.

gold

$$\varepsilon^{(1)}(x) = \frac{g_\varepsilon \lambda^2}{4\pi c \tau_S} \frac{\cos(\sqrt{\omega_S^2/c^2 + 1/\lambda^2}\,|\vec{x} - \vec{x}_S| - \omega_S t)}{|\vec{x} - \vec{x}_S|}.$$

pred

$$\varepsilon^{(1)}(x) = \frac{g_\varepsilon \lambda^2}{4\pi c \tau_S} \frac{\cos(\sqrt{\omega_S^2/c^2 + 1/\lambda^2}\,\vec{x} - \vec{x}_S| - \omega_S t)}{|\vec{x} - \vec{x}_S|}.$$

Рисунок 3.3.9 – Помилка позиційного кодування

ВИСНОВКИ

У даній роботі я розглянув два передових методи розпізнання математичних виразів, що базуються на методах машинного навчання: WYGIWYS (2015) і MI2LS(2019). Обидва ці методи є моделями seq2seq архітектури і складається с енкодера (у моделі WYGIWYS – це звичайна CNN мережа, у моделі MI2LS – CNN с позиційним кодуванням) та декодера (у обох випадках це різновид RNN, а саме LSTM с шаром уваги). Реалізував програмний додаток моделі WYGIWYS, провів тестування, та оцінку стандартних метрик для прогнозованої послідовностей, а саме текстову метрику BLEU, та метрики для зображень Extract Match, Extract Match without whitespaces. Порівняв значення метрик с реалізацією авторів моделей WYGIWYS, MI2LS, та іншими моделями, що не були описані у цій роботі, такими як INFTY [1], та Double Attention [3]. У результаті отримав, що модель WYGIWYS значно уступає моделі MI2LS, незалежно від реалізації. Насамперед це пов’язано з тим, що модель MI2LS реалізує техніку позиційного кодування, яка дозволяє враховувати вкладеність та інші просторові характеристики математичних виразів. Відсутність позиційного кодування у моделі WYGIWYS, породжує окремий клас помилкових прогнозувань, які мають спільні візуальні характеристики. Наприклад Рисунок 3.3.9, на якому видно, що символи та їх послідовність нейрона мережа розпізнала вірно, але є проблеми с вертикальним положенням токенів у формулі. Також гарний результат моделі MI2LS можна пояснити особливою технікою навчання, де цільова функція оптимізується не на знаходження наступного токена, а на прогнозування всієї послідовності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. M. Suzuki, F. Tamari, R. Fukuda, S. Uchida, and T. Kanahori, "INFTY: an integrated OCR system for mathematical documents," in Proceedings of the 2003 ACM symposium on Document engineering, 2003, pp. 95-104:ACM.

2. Y. Deng, A. Kanervisto, and A. M. Rush, "What you get is what you see: A visual markup decompiler," arXiv preprint arXiv:1609.04938, vol. 10, pp. 32-37, 2016.

3. W. Zhang, Z. Bai, and Y. Zhu, "An Improved Approach Based on CNN-RNNs for Mathematical Expression Recognition," in Proceedings of the 2019 4th International Conference on Multimedia Systems and Signal Processing, 2019, pp. 57-61: ACM.

4. J. Wang, Y. Sun, and S. Wang, "Image To Latex with DenseNet Encoder and Joint Attention," Procedia computer science, vol. 147, pp. 374-380, 2019.

5. P. Ion, R. Miner, S. Buswell, and A. Devitt, Mathematical Markup Language (MathML) 1.0 Specification. World Wide Web Consortium (W3C), 1998.

6. R. H. Anderson, "Syntax-directed recognition of handprinted two-dimensional mathematics," in Symposium on Interactive Systems for Experimental Applied Mathematics: Proceedings of the Association for Computing Machinery Inc. Symposium, 1967, pp. 436-459: ACM.

7. M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep structured output learning for unconstrained text recognition," arXiv preprint arXiv:1412.5903, 2014.

8. O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 3156-3164.

9. K. Xu et al., "Show, attend and tell: Neural image caption generation with visual attention," in International conference on machine learning, 2015, pp. 2048-2057.

10. K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: a method for automatic evaluation of machine translation," in Proceedings of the 40th annual meeting on association for computational linguistics, 2002, pp. 311-318: Association for Computational Linguistics.

11. R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in Advances in neural information processing systems, 2000, pp. 1057-1063.

12. M. A. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," arXiv preprint arXiv:1511.06732, 2015.

13. K.-F. Chan and D.-Y. Yeung, "Mathematical expression recognition: a survey," International Journal on Document Analysis and Recognition, vol. 3, no. 1, pp. 3-15, 2000.

14. U. Garain, B. Chaudhuri, and A. R. Chaudhuri, "Identification of embedded mathematical expressions in scanned documents," in Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004., 2004, vol. 1, pp. 384-387: IEEE.

15. Z. Wang, D. Beyette, J. Lin, and J.-C. Liu, "Extraction of Math Expressions from PDF Documents based on Unsupervised Modeling of Fonts," in IAPR International Conference on Document Analysis and Recognition (ICDAR), Sydney, Australia, 2019: IEEE.

16. X. Wang, Z. Wang, and J.-C. Liu, "Bigram Label Regularization to Reduce Over-Segmentation on Inline Math Expression Detection," in IAPR International Conference on Document Analysis and Recognition (ICDAR), Sydney, Australia, 2019: IEEE.

17. L. Gao, X. Yi, Y. Liao, Z. Jiang, Z. Yan, and Z. Tang, "A Deep Learning-Based Formula Detection Method for PDF Documents," in 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), 2017, pp. 553-558: IEEE.

18. H. M. Twaakyondo and M. Okamoto, "Structure analysis and recognition of mathematical expressions," in Proceedings of 3rd International Conference on Document Analysis and Recognition, 1995, vol. 1, pp. 430-437: IEEE.

19. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in neural information processing systems, 2012, pp. 1097-1105.

20. (2019, Aug 25). KaTex. Available: https://katex.org/

21. A. Vaswani et al., "Attention is all you need," in Advances in neural information processing systems, 2017, pp. 5998- 6008.

22. Zelun Wang, Jyh-Charn Liu "Translating Math Formula Images to LaTeX Sequences Using Deep Neural Networks with Sequence-level Training", arXiv preprint arXiv: 1908.11415, pp. 3-6, 2019.