УДК 004.8
DOI https://doi.org/10.32782/IT/2024-3-10

Олег КОБИЛІН
кандидат технічних наук, доцент, завідувач кафедри інформатики, Харківський національний університет радіоелектроніки, пр. Науки, 14, м. Харків, Україна, 61166
***ORCID:** 0000-0003-0834-0475*

Ірина ВЕЧІРСЬКА
кандидат технічних наук, доцент кафедри інформатики, Харківський національний університет радіоелектроніки, пр. Науки, 14, м. Харків, Україна, 61166
***ORCID:** 0000-0001-7964-2361*

Олексій КРАВЧЕНКО
аспірант кафедри інформатики, Харківський національний університет радіоелектроніки, пр. Науки, 14, м. Харків, Україна, 61166
***ORCID:** 0009-0000-7999-1406*

Бібліографічний опис статті: Кобилін, О., Вечірська, І., Кравченко, О. (2024). Порівняння нейронних мереж типу RNN та LSTM. *Information Technology: Computer Science, Software Engineering and Cyber Security*, 3, 97–107, doi: https://doi.org/10.32782/IT/2024-3-10

ПОРІВНЯННЯ НЕЙРОННИХ МЕРЕЖ ТИПУ RNN ТА LSTM

Дослідження спрямоване на виявлення переваг і недоліків різних підходів до обробки послідовних даних, що є важливим аспектом у задачах обробки природної мови, таких як аналіз настроїв, машинний переклад та генерація тексту.

***Мета роботи.** Мета роботи полягає в дослідженні ефективності різних архітектур нейронних мереж для задачі класифікації настроїв, з акцентом на порівнянні моделей RNN та LSTM.*

***Методологія.** У роботі розглянуто теоретичні аспекти функціонування рекурентних нейронних мереж (RNN) та мереж довготривалої короткочасної пам'яті (LSTM), які є спеціалізованими варіантами RNN. Було проведено експериментальне порівняння чотирьох різних моделей нейронних мереж, що включають прості рекурентні мережі (RNN), мережі LSTM, а також згорткові нейронні мережі (CNN), які застосовувалися для задачі класифікації настроїв. Для експерименту було обрано набір даних imdb_reviews, що містить огляди фільмів, призначені для бінарної класифікації настроїв (позитивний або негативний відгук). Реалізація та навчання моделей було виконано за допомогою бібліотек TensorFlow та Keras, що забезпечують інструментарій для ефективного виконання машинного навчання. Процес навчання та тестування моделей відбувався із застосуванням стандартних підходів до попередньої обробки текстових даних, таких як токенізація та підготовка послідовностей.*

***Наукова новизна.** Показано, що основною перевагою LSTM є здатність вирішувати проблему довгострокових залежностей, що робить їх більш ефективними для задач, де важливо враховувати контекст на довгих послідовностях даних. Експериментально підтверджено, що час навчання рекурентних нейронних мереж суттєво більший порівняно з нерекурентними моделями, проте вони демонструють трохи кращу точність.*

***Висновки.** Результати дослідження свідчать про те, що використання LSTM мереж є більш ефективним підходом для вирішення складних задач, які потребують врахування контексту на рівні послідовностей, що перевищують за довжиною типові фрагменти тексту. LSTM переважають їх завдяки здатності зберігати довготривалі залежності, що особливо важливо в задачах, де необхідно враховувати взаємозв'язок між віддаленими елементами даних.*

***Ключові слова:** рекурентна нейронна мережа, LSTM, RNN, класифікація настроїв, довгострокові залежності*

Oleg KOBYLIN
Ph.D., Associate Professor, Head of the Department of Informatics, Kharkiv National University of Radio Electronics, 14, Nauky Ave., Kharkiv, Ukraine, 61166, oleg.kobylin@nure.ua
***ORCID:** 0000-0003-0834-0475*

Iryna VECHIRSKA
Ph.D., Associate Professor at the Department of Informatics, Kharkiv National University of Radio Electronics, 14, Nauky Ave., Kharkiv, Ukraine, 61166, iryna.vechirska@nure.ua
***ORCID:** 0000-0001-7964-2361*

Oleksii KRAVCHENKO
Postgraduate Student at the Department of Informatics, Kharkiv National University of Radio Electronics, 14, Nauky Ave., Kharkiv, Ukraine, 61166, oleksii.kravchenko@nure.ua
***ORCID:** 0009-0000-7999-1406*

To cite this article: Kobylin, O., Vechirska, I., Kravchenko, O. (2024). Porivniannia neironnykh merezh typu RNN ta LSTM [Comparison of RNN and LSTM neural networks]. *Information Technology: Computer Science, Software Engineering and Cyber Security*, 3, 97–107, doi: https://doi.org/10.32782/IT/2024-3-10

COMPARISON OF RNN AND LSTM NEURAL NETWORKS

The research aims to identify the advantages and disadvantages of different approaches to sequential data processing, which is an important aspect in natural language processing tasks such as sentiment analysis, machine translation, and text generation.

***The purpose of the work.** The purpose of the work is to investigate the effectiveness of different neural network architectures for the problem of sentiment classification, with an emphasis on comparing RNN and LSTM models.*

***Methodology.** The paper examines the theoretical aspects of the functioning of recurrent neural networks (RNN) and long-term short-term memory (LSTM) networks, which are specialized variants of RNN. An experimental comparison of four different neural network models, including simple recurrent networks (RNNs), LSTM networks, and convolutional neural networks (CNNs), applied to the sentiment classification task was conducted. For the experiment, the imdb_reviews dataset was chosen, which contains movie reviews intended for binary sentiment classification (positive or negative feedback). The implementation and training of the models was done using the TensorFlow and Keras libraries, which provide a toolkit for efficient machine learning. The process of training and testing the models took place using standard approaches to preprocessing textual data, such as tokenization and sequence preparation.*

***Scientific novelty.** It is shown that the main advantage of LSTM is the ability to solve the problem of long-term dependencies, which makes them more effective for tasks where it is important to take into account the context of long data sequences. It has been experimentally confirmed that the training time of recurrent neural networks is significantly longer compared to non-recurrent models, but they demonstrate slightly better accuracy.*

***Conclusions.** The results of the study indicate that the use of LSTM networks is a more effective approach for solving complex problems that require consideration of the context at the level of sequences exceeding in length typical fragments of text. LSTMs are superior to them due to the ability to preserve long-term dependencies, which is especially important in tasks where it is necessary to take into account the relationship between distant data elements.*

***Key words:** recurrent neural network, LSTM, RNN, sentiment classification, long-term dependencies.*

Актуальність проблеми. Рекурентні нейронні мережі (Recurrent Neural Network, RNN) – вид нейронних мереж, що використовуються в обробці природної мови (NLP) (Isakov, n.d.). Рекурентна нейромережа оцінює довільні пропозиції на основі того, як часто вони зустрічалися в текстах. З іншого боку, такі моделі генерують новий текст. Навчання моделі на поемах Шекспіра дозволить генерувати новий текст, схожий на Шекспіра.

Ідея RNN полягає у послідовному використанні інформації. У традиційних нейронних мережах мається на увазі, що це входи і виходи незалежні. Але для багатьох завдань це не підходить. Якщо ви хочете передбачити наступне слово у реченні, краще враховувати попередні слова. RNN називаються рекурентними, тому що вони виконують одну і ту ж задачу для кожного елемента послідовності, причому вихід залежить від попередніх обчислень.

Рекурентні нейронні мережі продемонстрували великий успіх у багатьох завданнях NLP. На цьому етапі слід згадати, що типом RNN, що найчастіше використовується, є LSTM, які набагато краще захоплюють (зберігають) довгострокові залежності, ніж RNN. LSTM – це, по суті, те саме, що й RNN, які ми розберемо в цьому досліджені, просто мають інший спосіб. обчислення прихованого стану.

Аналіз останніх досліджень і публікацій. В останні роки зростає інтерес до використання рекурентних нейронних мереж (RNN) у завданнях обробки природної мови (NLP), таких як аналіз настроїв, машинний переклад та генерація текстів. Значна частина досліджень зосереджена на покращенні здатності RNN працювати

з довгостроковими залежностями, що є критично важливим у таких завданнях. Однією з найважливіших подій стало введення мереж довготривалої короткочасної пам'яті (LSTM), які здатні вирішувати проблему зникнення градієнтів та забезпечувати ефективну обробку довгих послідовностей даних. Такі моделі були успішно застосовані в різних галузях, включаючи машинний переклад та розпізнавання мовлення. Дослідження, такі як роботи (Glek, n.d.; Hochreiter, 1991), заклали фундамент для подальшого розвитку цієї технології, а останні експерименти демонструють їхню високу ефективність у порівнянні з традиційними RNN.

Мета дослідження. Метою даного дослідження є порівняння ефективності різних архітектур рекурентних нейронних мереж для класифікації настроїв. Особлива увага приділяється порівнянню моделей RNN та LSTM на основі експериментальних результатів із використанням набору даних imdb_reviews. Важливим аспектом є вивчення здатності цих моделей обробляти довгострокові залежності, що впливають на точність класифікації настроїв у текстових послідовностях.

Виклад основного матеріалу дослідження. Люди не запускають розумовий процес із нуля у кожний момент часу. Читаючи статтю, ви розумієте значення кожного слова на основі значень попередніх слів. Думки мають властивість накопичуватися та впливати один на одного. Цей принцип використовується у мережах LSTM (Glek, n.d.).

Прості нейронні мережі не можуть цього зробити, і це серйозна вада. Уявіть, що ви хочете в реальному часі класифікувати події у фільмі. Незрозуміло, як звичайна нейронна мережа може використовувати знання про попередні події, щоб вивчити наступні.

Рекурентні нейронні мережі (РНМ) вирішують цю проблему. Через наявність циклів РНМ виглядають більш складними порівняно з простими нейронними мережами, але на справді між ними немає великої різниці. РНМ можна розглядати, як кілька копій однієї й тієї ж мережі, кожна із яких передає повідомлення наступнику.

В останні роки досягнуто успіху в застосуванні РНМ до широкого кола проблем: розпізнавання мовлення, лінгвістичне моделювання, переклад, опис зображень. На допомогу у вирішенні перерахованих задач прийшли LSTM (Bengio, Simard, & Frasconi, 1994). LSTM *(long short-term memory або довга короткострокова пам'ять) – тип рекурентної нейронної мережі, здатний навчатися довгостроковим залежностям. LSTM, які вперше було представлено в роботі (Hochreiter & Schmidhuber, 1997) та потім удосконалено та популяризовано іншими дослідниками, добро справляються з багатьма завданнями і досі широко застосовуються.* LSTM спеціально розроблено для усунення проблеми довгострокової залежності. Їхня спеціалізація – запам'ятовування інформації протягом тривалих періодів часу, тому їх практично не потрібно навчати (Hochreiter, 1991; Bengio, Simard, & Frasconi, 1994)!

Принцип роботи мережі LSTM. LSTM зменшує або збільшує кількість інформації про стан комірки, залежно від потреб. Для цього використовуються структури, що ретельно налаштовуються, які називаються гейтами.

Гейт – це «брама», яка пропускає або не пропускає інформацію. Гейти складаються з сигмовидного шару нейронної мережі та операції поточкового множення.

На виході сигмовидного шару видаються числа від нуля до одиниці, визначаючи скільки відсотків кожної одиниці інформації пропустити далі. Значення «0» означає «не пропустити нічого», значення «1» – «пропустити все».

Покрокова схема роботи мережі LSTM. LSTM має три такі гейти для контролю стану комірки.

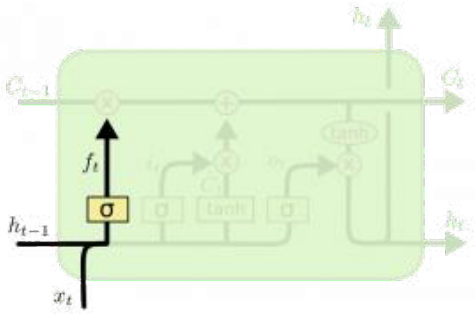
1. Шар втрати. На першому етапі LSTM потрібно вирішити, яку інформацію ми збираємося викинути зі стану комірки. Це рішення приймається сигмовидним шаром, званим шаром гейту втрати. Він отримує на вхід і видає число від 0 до 1 для кожного номера в стані комірки C. 1 означає «повністю зберегти», а 0 – «цілком видалити» (рис. 1).

2. Шар збереження. На наступному кроці потрібно вирішити, яку нову інформацію зберегти у стані комірки. Розіб'ємо процес на дві частини. Спочатку сигмоїдний шар, званий «шаром гейту входу», вирішує, які значення потрібно оновити. Потім шар tanh створює вектор нових значень-кандидатів C, які додаються до стан. На наступному етапі ми об'єднаємо ці два значення для оновлення стану (рис. 2).

3. Новий стан. Тепер оновимо попередній стан комірки для отримання нового стану **C**. Спосіб оновлення обрано, тепер реалізуємо саме оновлення.

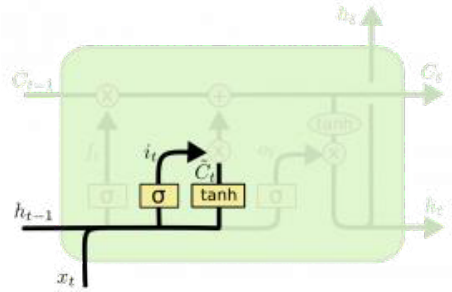
Помножимо старий стан на f**,** втрачаючи інформацію, яку вирішили забути. Потім додаємо i*C. Це нові значення кандидатів, які масштабуються залежно від того, як ми вирішили оновити кожне значення стану (рис. 3).

Нарешті, потрібно вирішити, що хочемо отримати на виході. Результат буде відфільтрованим



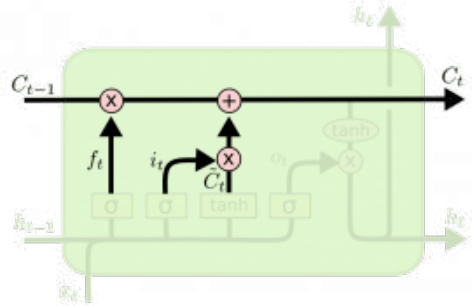
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Рис. 1. Шар втрати



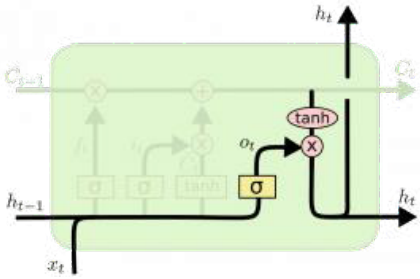
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Рис. 2. Шар збереження



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

Рис. 3. Шар нового стану



$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Рис. 4. Отримання результату на виході з комірки

станом комірки. Спочатку запускаємо сигмоїдний шар, який вирішує, які частини стану комірки виводити. Потім пропускаємо стан комірки через tanh (щоб розмістити всі значення в інтервалі [-1, 1]) і множимо його на вихідний сигнал сигмовидного гейту (рис. 4).

Описана вище схема традиційна для LSTM. Але не всі LSTM ідентичні. Насправді майже в кожній статті використовуються версії, що відрізняються. Відмінності незначні, але варто згадати деякі з них.

У популярному варіанті LSTM, представленому в (Gers & Schmidhuber, 2000), ми

дозволяємо шарам гейтів переглядати стан комірки (рис. 5).

На діаграмі вгорі «око» є у всіх гейтів, але в багатьох статтях він є лише в деяких гейтів.

Інший варіант – використання пов'язаних гейтів втрати та входу. Замість того, щоб окремо вирішувати, що забути, а до чого додати нову інформацію, ми ухвалюємо ці рішення одночасно. Ми забуваємо інформацію лише тоді, коли потрібно помістити щось нове на тому самому місці. Нові значення вносяться в стан лише тоді, коли ми забуваємо щось старіше (рис. 6).

Є багато інших варіантів LSTM, таких як PHC з гейтом глибини (Yao et al., 2015). Відомий також зовсім інший підхід до вирішення довгострокових залежностей, наведений у (Koutník et al., 2014).

Порівняння 4 моделей нейронних мереж при вирішенні задачі розпізнавання настроїв. У цьому розділі ми порівняємо 4 різні моделі нейронних мереж при вирішенні задачі розпізнавання (класифікації) настроїв. Ми будемо використовувати мову програмування Python та пакети Keras та TensorFlow. TensorFlow – це відкрите програмне забезпечення для машинного навчання та глибокого навчання, розроблене Google. TensorFlow зручний для використання в різних областях, включаючи великі обчислення, обробку природних мов, комп'ютерний зір і багато інших. Keras – це високорівневий інтерфейс для розробки нейронних мереж, який працює поверх TensorFlow та інших бібліотек машинного навчання.

Ми розглянемо практично розглянемо роботу таких шарів нейронних мереж як Flatten, LSTM, GRU, Convolution. Шари LSTM та GRU стосуються рекурентних нейронних мереж, тобто мереж для яких важливий порядок входів. Вони вже розглядались нами у розділі 2. Шар Flatten використовується для перетворення вхідних даних, які можуть мати багатовимірну форму, у одномірний вектор. Це особливо корисно, коли ви працюєте зі згортковими нейронними мережами (CNN), де виходи згорткових шарів можуть бути тривимірними (висота, ширина, глибина). В основі згорткових шарів (Convolution layer) нейронної мережі лежить операція згортки (рис. 7). Згортка – це процес додавання кожного елемента зображення до його сусідів, зважених ядром.

Одна з задач де порядок входів має значення, тобто бажане застосування рекурентних нейронних мереж, це задача побудови моделі, яка буде розрізняти почуття, навіть якщо слова, використані в двох реченнях, однакові.

1: Моїм друзям подобається фільм, але мені ні. --> негативний відгук

2: Моїм друзям не подобається фільм, але мені подобається. --> позитивний відгук

Ми розглянемо саме цю задачу і порівняємо поведінку 4 зазначених вище типів мереж при вирішенні цієї задачі.

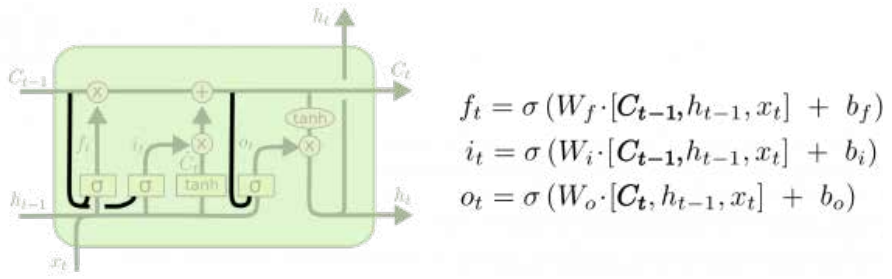


Рис. 5. Варіанті LSTM, де шари гейтів можуть переглядати стан комірки

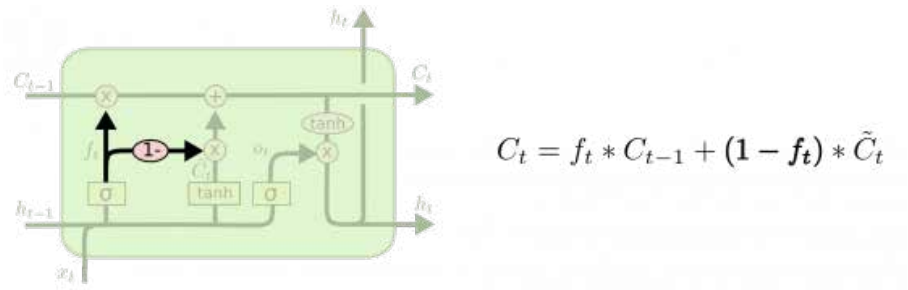


Рис. 6. Варіант LSTM з використанням пов'язаних гейтів втрати та входу

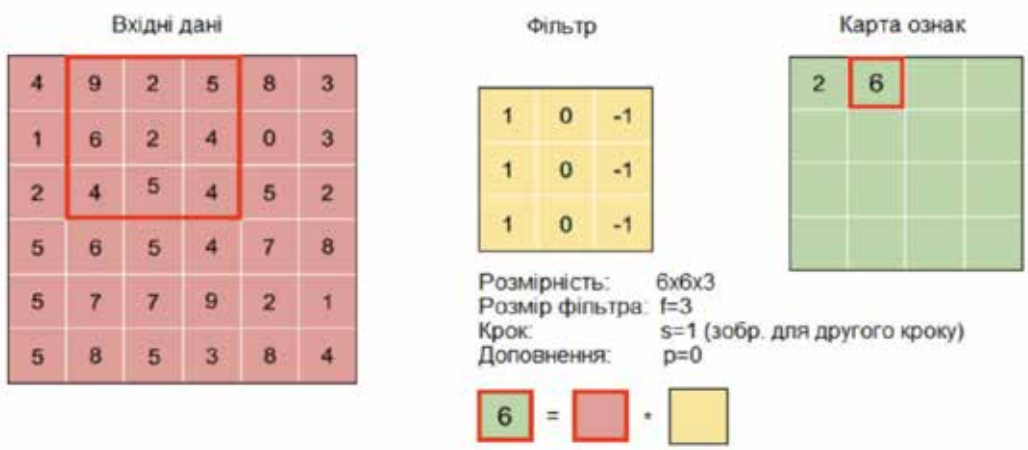


Рис. 7. Операція згортки

```
In 1
import tensorflow_datasets as tfds #4.9.2
# Download the plain text dataset
imdb, info = tfds.load('imdb_reviews', split='train', with_info=True)

df = tfds.as_dataframe(imdb, info)
df.head()
```

Executed at 2023.12.22 10:21:43 in 4s 340ms

```
WARNING:tensorflow:From D:\IdeaProjects\machine-learning\ml_venv\lib\site-packages\keras\src\losses.py:2976: The name tf
```

Out 1

5 rows × 2 columns pd.DataFrame

	label	text
0	0	b"This was an absolutely terrible movie. Don't be lured in by Christopher Walken or Michael Ironside. Both are great…
1	0	b'I have been known to fall asleep during films, but this is usually due to a combination of things including, reall…
2	0	b'Mann photographs the Alberta Rocky Mountains in a superb fashion, and Jimmy Stewart and Walter Brennan give enjoya…
3	1	b'This is the kind of film for a snowy Sunday afternoon when the rest of the world can go ahead with its own busines…
4	1	b'As others have mentioned, all the women that go nude in this film are mostly absolutely gorgeous. The plot very ab…

Рис. 8. Приклад зразків даних з цього датасету

В якості вхідних даних ми будемо використовувати вбудований у Tensorflow набір даних imdb_reviews. Це великий набір даних оглядів фільмів. Він саме створений для бінарної класифікації настроїв, що містить значно більше даних, ніж попередні еталонні набори даних. У набір входять 25 000 оглядів полярних оглядів фільмів для навчання та 25 000 для тестування. Існують додаткові немарковані дані для використання (рис. 8).

Перед усім імпортуємо необхідні залежності. Загрузимо датасет та виконаємо підготовку даних. Далі нам буде потрібно створити словник з нуля та сгенерувати доповнені послідовності. Ми це зробимо за допомогою класу Tokenizer і методу pad_sequences() (рис. 9).

Основною перевагою першої моделі є її простота. Інформацію щодо цієї моделі зображено на рис. 10.

На рис. 11 наведено результати навчання та валідації (тестування) моделі.

Модель LSTM є найбільш перспективною при вирішенні даної задачі. Інформацію щодо цієї моделі зображено на рис. 12.

На рис. 13 наведено результати навчання та валідації (тестування) моделі LSTM.

Модель GRU є спрощеним варіантом попередньої моделі та обчислення повинні займати менше часу. Інформацію щодо цієї моделі зображено на рис. 14.

На рис. 15 наведено результати навчання та валідації (тестування) моделі GRU.

Модель Convolution наведено на рис. 16. Ця модель є спрощеним варіантом попередньої моделі та обчислення повинні займати менше часу. Інформацію щодо цієї моделі зображено на рис. 17.

На рис. 18 наведено результати навчання та валідації (тестування) моделі Convolution.

```
# Parameters
vocab_size = 10000
max_length = 120
trunc_type='post'
oov_tok = "<OOV>"

# Initialize the Tokenizer class
tokenizer = Tokenizer(num_words = vocab_size, oov_token=oov_tok)

# Generate the word index dictionary for the training sentences
tokenizer.fit_on_texts(training_sentences)
word_index = tokenizer.word_index

# Generate and pad the training sequences
sequences = tokenizer.texts_to_sequences(training_sentences)
padded = pad_sequences(sequences,maxlen=max_length, truncating=trunc_type)

# Generate and pad the test sequences
testing_sequences = tokenizer.texts_to_sequences(testing_sentences)
testing_padded = pad_sequences(testing_sequences,maxlen=max_length)
```

Рис. 9. Створення словника та генерування доповнених послідовностей

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 120, 16)           160000

 flatten (Flatten)           (None, 1920)              0

 dense (Dense)               (None, 6)                 11526

 dense_1 (Dense)             (None, 1)                 7

=================================================================
Total params: 171533 (670.05 KB)
Trainable params: 171533 (670.05 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Рис. 10. Детальна інформація про модель Flatten

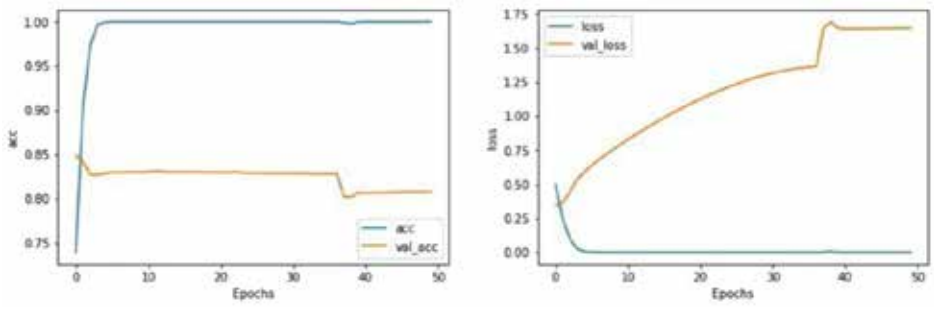


Рис. 11. Результати навчання та тестування моделі Flatten

Отже, тепер, коли ми отримали результати усіх 4 моделей, порівняємо їх. У першій моделі ми використовували слої embedding та flatten, за якими ми використовували, як і в інших моделях повнозв'язні слої. Модель містить 171 533 параметрами. Гарна точність перевірки (~80%), але явне перенавчання. Тренування займає лише близько 5 секунд на епоху.

При використанні моделі LSTM ми маємо 172941 параметр. При цьому навчання займає приблизно 43 секунди на епоху. Точність

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 120, 16)           160000

 bidirectional (Bidirection  (None, 64)                12544
 al)

 dense_2 (Dense)             (None, 6)                 390

 dense_3 (Dense)             (None, 1)                 7

=================================================================
Total params: 172941 (675.55 KB)
Trainable params: 172941 (675.55 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Рис. 12. Детальна інформація про модель LSTM

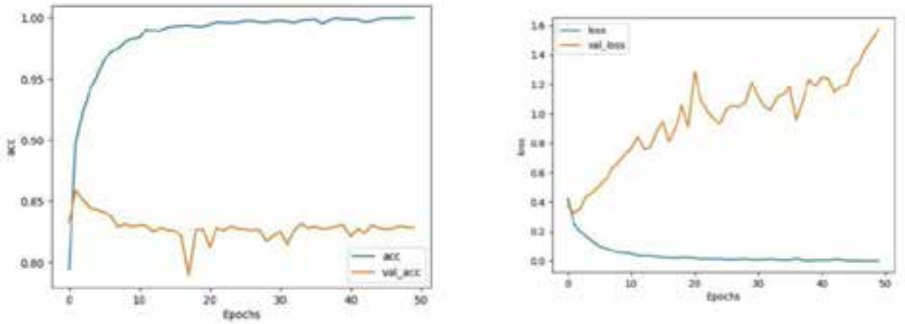


Рис. 13. Результати навчання та тестування моделі LSTM

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_2 (Embedding)     (None, 120, 16)           160000

 bidirectional_1 (Bidirecti  (None, 64)                9600
 onal)

 dense_4 (Dense)             (None, 6)                 390

 dense_5 (Dense)             (None, 1)                 7

=================================================================
Total params: 169997 (664.05 KB)
Trainable params: 169997 (664.05 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Рис. 14. Детальна інформація про модель GRU

перевірки краща (~83%), але є ще деяке перенавчання. При використанні двонаправленого GRU мережа має 169997 параметрів. Час навчання зменшиться до 20 секунд на епоху, і точність знову дуже хороша, під час тренувань і не надто погана під час перевірки (~82%), але мережа знову ж таки демонструє деяке перенавчання. Зі згортковою мережею ми маємо 171149 параметрів, і час навчання складає близько шести секунд на епоху, щоб наблизитися до 100-відсоткової точності під час навчання та близько 83 відсотків під час перевірки, але знову ж таки маємо перенавчання.

Висновки. Розглянто архітектури RNN, LSTM і GRU. Модель LSTM описуються складнішим набором рівнянь у порівнянні з простими нейронними мережами і є великим кроком у розвитку РНМ. Хоча наші моделі показали майже однакову точність, при цьому час навчання при використанні рекурентних нейронних мереж виявився значно більшим, тим не менш вони

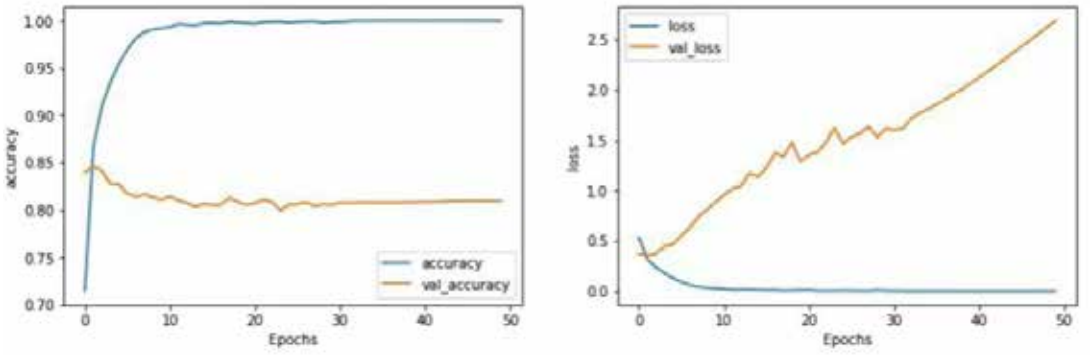


Рис. 15. Результати навчання та тестування моделі GRU

```
# Parameters
embedding_dim = 16
filters = 128
kernel_size = 5
dense_dim = 6

# Model Definition with Conv1D
model_conv = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim,
input_length=max_length),
    tf.keras.layers.Conv1D(filters, kernel_size, activation='relu'),
    tf.keras.layers.GlobalAveragePooling1D(),
    tf.keras.layers.Dense(dense_dim, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

# Set the training parameters
model_conv.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accurac
y'])

# Print the model summary
model_conv.summary()

NUM_EPOCHS = 10
BATCH_SIZE = 128

# Train the model
history_conv = model_conv.fit(padded, training_labels_final,
batch_size=BATCH_SIZE, epochs=NUM_EPOCHS, validation_data=(testing_padded,
testing_labels_final))

# Plot the accuracy and loss history
plot_graphs(history_conv, 'accuracy')
plot_graphs(history_conv, 'loss')
```

Рис. 16. Створення та навчання моделі Convolution

показують значно кращі практичні результати ніж звичайні РНМ при вирішенні більш складних завдань машинного перекладу, розпізнавання руху, генерації тексту та ін. Головною перевагою LSTM є їхня здатність уникнути проблеми зниклих та вибуваючих градієнтів, з якою стикаються звичайні РНМ, завдяки введенню спеціальних воріт (воріт забування, воріт входу, воріт виводу), які контролюють потік інформації в моделі.

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_3 (Embedding)     (None, 120, 16)           160000

 conv1d (Conv1D)             (None, 116, 128)          10368

 global_average_pooling1d (  (None, 128)               0
 GlobalAveragePooling1D)

 dense_6 (Dense)             (None, 6)                 774

 dense_7 (Dense)             (None, 1)                 7

=================================================================
Total params: 171149 (668.55 KB)
Trainable params: 171149 (668.55 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Рис. 17. Детальна інформація про модель Convolution

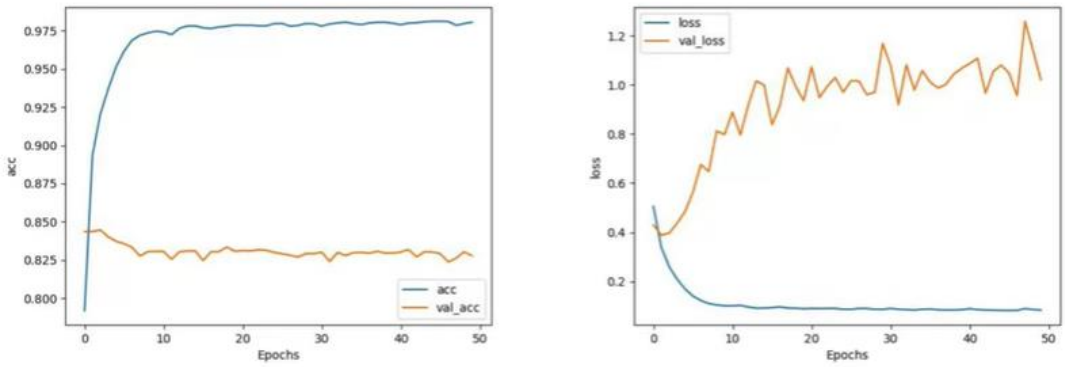


Рис. 18. Результати навчання та тестування моделі Convolution

ЛІТЕРАТУРА:

1. Ісаков С. Рекурентна нейронна мережа (RNN): типи, навчання, приклади. URL: https://neurohive.io/ru/osnovy-data-science/rekurrentnye-nejronnye-seti (дата звернення: 15.08.2024).

2. Глек П. LSTM – мережа довготривалої короткочасної пам'яті. URL: https://neurohive.io/ru/osnovy-data-science/lstm-nejronnaja-set (дата звернення: 15.08.2024).

3. Hochreiter S. Untersuchungen zu dynamischen neuronalen Netzen. Diploma, Technische Universität München, 1991. 31 с.

4. Bengio Y., Simard P., Frasconi P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*. 1994. Vol. 5, № 2. С. 157–166.

5. Hochreiter S., Schmidhuber J. Long Short-Term Memory. *Neural Computation*. 1997. Vol. 9, № 8. С. 1735–1780.

6. Gers F.A., Schmidhuber J. Recurrent nets that time and count. Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium. Como, Italy, 2000. Vol. 3. С. 189–194.

7. Cho K., van Merrienboer B., Gulcehre C., Bougares F., Schwenk H., Bengio Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Conference on Empirical Methods in Natural Language Processing (EMNLP 2014). 2014.

8. Yao K., Cohn T., Vylomova K., Duh K., Dyer C. Depth-gated recurrent neural networks. arXiv, 2015. URL: http://arxiv.org/abs/1508.03790.

9. Koutník J., Greff K., Gomez F., Schmidhuber J. A clockwork RNN. 31st International Conference on Machine Learning, ICML 2014. 2014.

10. Greff K. et al. LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*. 2016. Vol. 28, № 10. С. 2222–2232.

11. Jozefowicz R., Zaremba W., Sutskever I. An Empirical Exploration of Recurrent Network Architectures. Proceedings of the 32nd International Conference on Machine Learning. PMLR 37:2342–2350. 2015.

12. Xu K. et al. Show, attend and tell: Neural image caption generation with visual attention. International conference on machine learning. PMLR, 2015.

REFERENCES:

1. Isakov, S. (n.d.). Recurrent neural network (RNN): Types, training, examples. Neurohive. Retrieved from https://neurohive.io/ru/osnovy-data-science/rekurrentnye-nejronnye-seti [in Ukrainian].

2. Glek, P. (n.d.). LSTM – long short-term memory neural network. Neurohive. Retrieved from https://neurohive.io/ru/osnovy-data-science/lstm-nejronnaja-set [in Ukrainian].

3. Hochreiter, S. (1991). Untersuchungen zu dynamischen neuronalen Netzen (Diploma thesis). Technische Universität München.

4. Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2), 157–166.

5. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780.

6. Gers, F. A., & Schmidhuber, J. (2000). Recurrent nets that time and count. In Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN 2000): Neural Computing: New Challenges and Perspectives for the New Millennium (Vol. 3, pp. 189–194). Como, Italy.

7. Cho, K., van Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014).

8. Yao, K., Cohn, T., Vylomova, K., Duh, K., & Dyer, C. (2015). Depth-gated recurrent neural networks. arXiv preprint arXiv:1508.03790.

9. Koutník, J., Greff, K., Gomez, F., & Schmidhuber, J. (2014). A clockwork RNN. In Proceedings of the 31st International Conference on Machine Learning (ICML 2014) (Vol. 5).

10. Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2016). LSTM: A search space odyssey. IEEE Transactions on Neural Networks and Learning Systems, 28(10), 2222–2232.

11. Jozefowicz, R., Zaremba, W., & Sutskever, I. (2015). An empirical exploration of recurrent network architectures. In Proceedings of the 32nd International Conference on Machine Learning (ICML 2015), PMLR 37, 2342–2350.

12. Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., … & Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In Proceedings of the International Conference on Machine Learning (ICML 2015), PMLR.