УДК 004.932.2:004.8 DOI https://doi.org/10.35546/kntu2078-4481.2025.2.2.8

Т. А. ВАКАЛЮК
доктор педагогічних наук, професор,
завідувач кафедри інженерії програмного забезпечення
Державний університет «Житомирська політехніка»
ORCID: 0000-0001-6825-4697

Д. В. ФУРІХАТА
старший викладач кафедри комп'ютерних наук
Державний університет «Житомирська політехніка»
ORCID: 0000-0002-6093-664X

В. М. ЯНЧУК
кандидат технічних наук, доцент,
доцент кафедри робототехніки, електроенергетики
та автоматизації ім. проф. Б. Б. Самотокіна
Державний університет «Житомирська політехніка»
ORCID: 0000-0002-6715-4667

Н. В. ЛЕУТ
студент
Державний університет «Житомирська політехніка»
ORCID: 0009-0009-4317-0898

РОЗРОБКА ТА ОЦІНКА ЕФЕКТИВНОСТІ МОДЕЛІ РОЗПІЗНАВАННЯ РУКОПИСНОГО ТЕКСТУ НА ОСНОВІ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ

Стаття присвячена розробці та оцінці ефективності моделі розпізнавання рукописного тексту на основі згорткових нейронних мереж з використанням набору даних IAM Sentences. В роботі детально розглянуто процес створення моделі, яка поєднує сучасні методи глибокого навчання для вирішення складної задачі перетворення рукописної інформації в цифровий формат. Проведено комплексний аналіз наукових досліджень у галузі розпізнавання тексту, що підтверджує актуальність застосування нейронних мереж.

Методологія дослідження включає детальну попередню обробку набору даних IAM Sentences, що містить зображення рукописних речень з відповідними текстовими мітками. Описано процес підготовки даних, включаючи читання метаданих, фільтрацію помилкових записів, нормалізацію зображень та створення словника унікальних символів. Особливу увагу приділено методам доповнення даних з використанням випадкових змін яскравості, масштабування для підвищення стійкості моделі.

Архітектура розробленої моделі базується на поєднанні згорткових шарів для виділення просторових ознак зображень та LSTM-шарів для захоплення послідовних залежностей між символами. Використання функції втрат Connectionist Temporal Classification (CTC) дозволяє моделі прогнозувати послідовності символів без явного вирівнювання між входом та виходом, що є критично важливим для обробки рукописного тексту змінної довжини.

Результати експериментів демонструють високу ефективність розробленої системи з досягненням CER на рівні 11.04 % після навчання протягом 67 епох. Цей показник свідчить про високу точність розпізнавання символів, що є конкурентоспроможним результатом для задач розпізнавання рукописного тексту. Аналіз кривих навчання через TensorBoard показав стабільне покращення метрик з незначними флуктуаціями, що підтверджує коректність обраної архітектури та параметрів навчання.

Ключові слова: *розпізнавання рукописного тексту, згорткові нейронні мережі, LSTM, глибоке навчання, обробка зображень, набір даних IAM Sentences, TensorFlow.*

T. A. VAKALIUK
Doctor of Pedagogical Sciences, Professor,
Head of the Department of Software Engineering
Zhytomyr Polytechnic State University
ORCID: 0000-0001-6825-4697

D. V. FURIKHATA
Senior Lecturer at the Department of Computer Science
Zhytomyr Polytechnic State University
ORCID: 0000-0002-6093-664X

V. M. YANCHUK
Candidate of Technical Sciences, Associate Professor,
Associate Professor at the Department of Robotics, Electrical Engineering
and Automation named after Prof. B. B. Samotokin
Zhytomyr Polytechnic State University
ORCID: 0000-0002-6715-4667

N. V. LEUT
Student
Zhytomyr Polytechnic State University
ORCID: 0009-0009-4317-0898

DEVELOPMENT AND EVALUATION OF THE EFFECTIVENESS OF A HANDWRITTEN TEXT RECOGNITION MODEL BASED ON CONVOLUTIONAL NEURAL NETWORKS

The article is devoted to developing and evaluating the effectiveness of a handwritten text recognition model based on convolutional neural networks using the IAM Sentences dataset. The paper details the process of creating a model that combines modern deep learning methods to solve the complex task of converting handwritten information into digital format. A comprehensive analysis of scientific research in text recognition confirms the relevance of neural networks.

The research methodology includes detailed pre-processing of the IAM Sentences dataset, which contains images of handwritten sentences with corresponding text labels. The data preparation includes metadata reading, filtering of false entries, image normalisation, and creating a dictionary of unique characters. Particular attention is paid to data augmentation methods using random brightness changes and scaling to improve model robustness.

The architecture of the developed model is based on a combination of convolutional layers to extract spatial features from images and LSTM layers to capture sequential dependencies between characters. The Connectionist Temporal Classification (CTC) loss function allows the model to predict symbol sequences without explicit alignment between input and output, critical for processing variable-length handwritten text.

The experiments' results demonstrate the developed system's high efficiency, achieving a CER of 11.04 % after training for 67 epochs. This indicator shows high character recognition accuracy, a competitive result for handwritten text recognition tasks. Analysis of the training curves via TensorBoard showed a steady improvement in metrics with minor fluctuations, confirming the correctness of the chosen architecture and training parameters.

Key words: *handwritten text recognition, convolutional neural networks, LSTM, deep learning, image processing, IAM Sentences dataset, TensorFlow.*

Постановка проблеми

В сучасному цифровому світі зростає потреба в автоматизації процесів обробки рукописної інформації. Незважаючи на широке поширення цифрових технологій, рукописний текст залишається важливим джерелом інформації в освіті, медицині, юриспруденції та багатьох інших галузях. Однак процес перетворення рукописного тексту в цифровий формат досі залишається трудомістким і часозатратним завданням.

Традиційні методи оптичного розпізнавання символів (OCR) демонструють високу ефективність при роботі з друкованим текстом, проте їх застосування для розпізнавання рукописного тексту обмежене через значну варіативність почерків, різноманітність стилів написання, нечіткість символів та можливі спотворення зображень. Ці фактори призводять до низької точності розпізнавання та високого відсотка помилок.

Існуючі системи розпізнавання рукописного тексту часто потребують значних обчислювальних ресурсів, складного налаштування та не завжди забезпечують прийнятну точність для практичного використання. Крім того, більшість комерційних рішень є дорогими та не адаптованими для специфічних потреб користувачів.

Сучасні досягнення в галузі глибокого навчання, зокрема застосування згорткових нейронних мереж та архітектур LSTM, відкривають нові можливості для вирішення задач розпізнавання рукописного тексту. Однак питання оптимального поєднання цих технологій, вибору відповідного набору даних для навчання та досягнення високої точності розпізнавання при збереженні ефективності обчислень залишається актуальним.

Таким чином, існує потреба в розробці ефективної системи розпізнавання рукописного тексту, яка б поєднувала сучасні методи глибокого навчання з практичними вимогами до точності та швидкості обробки інформації.

Аналіз останніх досліджень і публікацій

Аналіз літератури демонструє активний розвиток технологій розпізнавання рукописного тексту з використанням методів глибокого навчання. Дослідження в цій галузі зосереджуються на удосконаленні архітектур нейронних мереж та підвищенні точності розпізнавання.

Nebauer [8] досліджував застосування згорткових нейронних мереж для візуального розпізнавання. Автор проаналізував ефективність CNN у задачах комп'ютерного зору та встановив, що згорткові шари здатні ефективно виділяти локальні ознаки зображень, що є критично важливим для розпізнавання символів. Дослідження показало, що архітектура CNN забезпечує інваріантність до зсувів та деформацій, що особливо актуально для обробки рукописного тексту з його природною варіативністю.

Yamashita та ін. [9] представили комплексний огляд згорткових нейронних мереж та їх застосування в радіології, що розширює розуміння можливостей CNN у медичних застосуваннях. Автори детально проаналізували архітектурні особливості CNN та продемонстрували їх ефективність у обробці медичних зображень. Особливо важливим є висновок про здатність глибоких згорткових мереж автоматично виучувати ієрархічні представлення даних, що дозволяє виділяти як низькорівневі, так і високорівневі ознаки зображень. Це дослідження підкреслює універсальність CNN для різних доменів візуального розпізнавання.

Mienye, Swart та Obaido [10] опублікували актуальний огляд рекурентних нейронних мереж, їх архітектур, варіантів та застосувань. Автори систематично проаналізували розвиток RNN-технологій, включаючи LSTM та GRU архітектури, та їх ефективність у обробці послідовних даних. Дослідження підкреслює важливість LSTM-шарів для захоплення довгострокових залежностей у послідовностях, що є критично важливим для розпізнавання тексту, де контекст між символами впливає на точність розпізнавання. Автори також розглянули сучасні тенденції в комбінуванні RNN з іншими архітектурами для покращення продуктивності.

Аналіз літератури показує, що поєднання згорткових та рекурентних архітектур є перспективним напрямком для розпізнавання рукописного тексту. Згорткові шари забезпечують ефективне виділення просторових ознак, тоді як LSTM-компоненти дозволяють моделювати послідовні залежності між символами. Однак існує потреба в подальшому дослідженні оптимальних конфігурацій таких гібридних архітектур та методів їх навчання для досягнення максимальної точності розпізнавання.

Формулювання мети дослідження

Метою статті є розробка та оцінка ефективності системи розпізнавання рукописного тексту на основі згорткових нейронних мереж з використанням набору даних IAM Sentences.

Викладення основного матеріалу дослідження

Для розробки системи розпізнавання рукописного тексту було використано набір даних IAM Sentences [1], який містить зображення тексту речень, написаних від руки, з відповідними текстовими мітками. Цей набір даних є доволі популярним у галузі і перевірений часом, часто використовується для оцінки ефективності моделей розпізнавання тексту.

Перед початком навчання моделі необхідно провести аналіз та попередню обробку даних. Кожен зразок у цьому наборі даних складається із зображення рукописного тексту та відповідного зображенню рядка тексту. Файл sentences.txt містить метадані для кожного зразка зображення в наборі даних, включаючи шлях до файлу зображення та відповідну текстову мітку (рис. 1).

Створимо змінні dataset, кожен елемент якого міститиме шлях до файлу зображення та текстову мітку, що дозволяє легко звертатися до даних під час навчання моделі, vocab – набір, що міститиме унікальні символи міток та max_length – найбільшу довжину міток:

```
dataset, vocab, max_length = [], set(), 0
```

Розглянемо кожен рядок файлу sentences.txt. Якщо рядок починається з «#» – пропускаємо його:

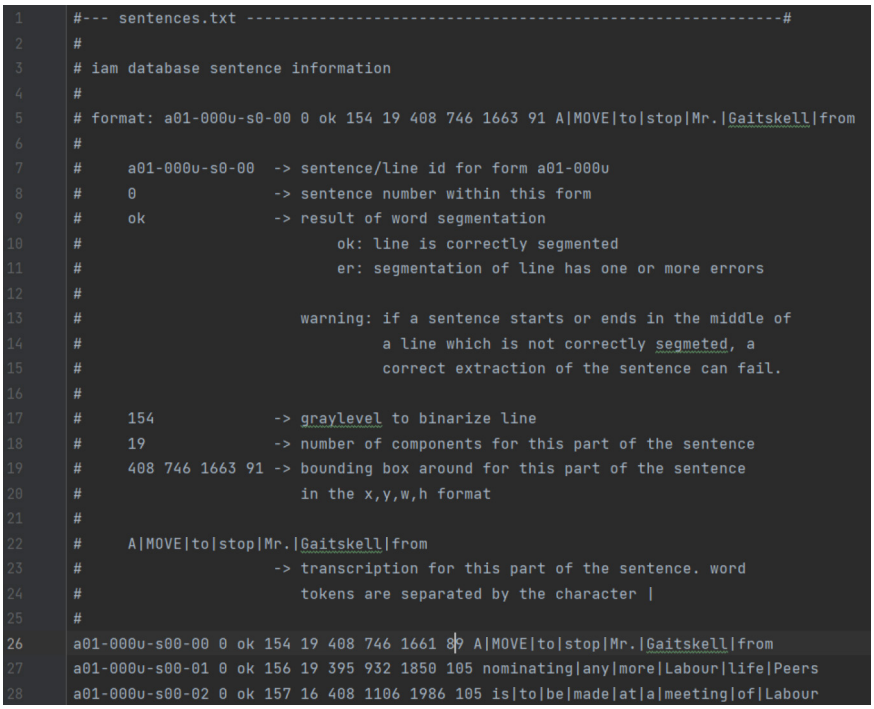
```
words = open(sentences_txt_path, "r").readlines()
for line in tqdm(words):
   if line.startswith("#"):
      continue
```

Після цього отримуємо елементи рядка розділені пробілами та розглядаємо далі, якщо третій елемент не «err» (в наборі помічені дані):

```
line_split = line.split(" ")
if line_split[2] == "err":
   continue
```

Отримуємо назву файлів, взявши перший елемент рядка:

```
file_name = line_split[0] + ".png"
```



```
#--- sentences.txt ----------------------------------------------------------#
#
# iam database sentence information
#
# format: a01-000u-s0-00 0 ok 154 19 408 746 1663 91 A|MOVE|to|stop|Mr.|Gaitskell|from
#
#     a01-000u-s0-00  -> sentence/line id for form a01-000u
#     0               -> sentence number within this form
#     ok              -> result of word segmentation
#                            ok: line is correctly segmented
#                            er: segmentation of line has one or more errors
#
#                        warning: if a sentence starts or ends in the middle of
#                                 a line which is not correctly segmeted, a
#                                 correct extraction of the sentence can fail.
#
#     154             -> graylevel to binarize line
#     19              -> number of components for this part of the sentence
#     408 746 1663 91 -> bounding box around for this part of the sentence
#                        in the x,y,w,h format
#
#     A|MOVE|to|stop|Mr.|Gaitskell|from
#                     -> transcription for this part of the sentence. word
#                        tokens are separated by the character |
#
a01-000u-s00-00 0 ok 154 19 408 746 1661 89 A|MOVE|to|stop|Mr.|Gaitskell|from
a01-000u-s00-01 0 ok 156 19 395 932 1850 105 nominating|any|more|Labour|life|Peers
a01-000u-s00-02 0 ok 157 16 408 1106 1986 105 is|to|be|made|at|a|meeting|of|Labour
```

Рис. 1. Структура набору даних

Замінимо всі символи «|» на пробіли, для отримання тексту рядків:

```
label = line_split[-1].rstrip("\n").replace("|", " ")
```

Створюємо змінну шляху до файлу. Якщо шляху до файлу не існує – пропускаємо цей рядок:

```
rel_path = os.path.join(sentences_folder_path, file_name)
if not os.path.exists(rel_path):
   print(f"Файл не знайдено: {rel_path}")
   continue
```

Якщо такий шлях є – додаємо шлях до dataset та мітку до списку наборів даних, оновлюємо змінну vocab символами з мітки та присвоюємо max_length максимальне значення між поточним значенням змінної та довжиною мітки:

```
dataset.append([rel_path, label])
vocab.update(list(label))
max_length = max(max_length, len(label))
```

Для навчання моделі необхідно мати набір усіх унікальних символів, які зустрічаються у текстових мітках, що дозволяє моделі працювати з обмеженим набором символів і спрощує процес навчання. Максимальна довжина мітки використовується для фіксування розміру вихідного масиву, що генерується моделлю і потрібна для коректної роботи алгоритму Connectionist Temporal Classification.

Після попередньої обробки даних наступним кроком є створення та навчання моделі розпізнавання рукописного тексту. Спершу заповнимо конфігураційний файл configs.py та створимо об'єкт ModelConfigs на його основі для зберігання конфігурацій моделі, задамо словник vocab і максимальну довжину max_length тексту як атрибути цього об'єкта. Цей конфігураційний файл знадобиться при запуску виведення на навченій моделі:

```
configs = ModelConfigs()
configs.vocab = "".join(sorted(vocab))
configs.max_text_length = max_length
configs.save()
```

Потім розділимо набір даних у співвідношенні 90 % для навчання і 10 % для перевірки.

```
data_df = pd.DataFrame(dataset, columns=["image_path", "label"])
train_df, val_df = train_test_split(data_df, test_size=0.2, random_state=42)
train_df.to_csv(os.path.join(configs.model_path, "train.csv"), index=False)
val_df.to_csv(os.path.join(configs.model_path, "val.csv"), index=False)
```

Крім цього, створимо об’єкти, застосуємо випадкові яскравість, роздільну здатність та різкість для поповнення даних.

```
train_data_gen = ImageDataGenerator(
   brightness_range=[0.8, 1.2],
   zoom_range=0.2,
   preprocessing_function=lambda x: cv2.GaussianBlur(x, (5, 5), 0)
)
```

Завантажимо та підготуємо для використання у тренуванні зображення:

```
train_images, train_labels = [], []

for index, row in train_df.iterrows():
   image_path, label = row[“image_path”], row[“label”]
   try:
      image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
      if image is None:
         print(f“Could not read image: {image_path}”)
         continue
      image = cv2.resize(image, (configs.width, configs.height))
      image = image / 255.0 # Нормалізація
      train_images.append(np.expand_dims(image, axis=-1))
      train_labels.append(text_to_sequence(label))
   except Exception as e:
      print(f“Error processing image {image_path}: {e}”)
train_images = np.array(train_images)
train_labels = np.array(train_labels)
```

Перетворимо текстові мітки у числовий формат з падинґом для використання функції втрат далі.

```
train_labels = [text_to_sequence(label) for label in train_df[“label”]]
val_labels = [text_to_sequence(label) for label in val_df[“label”]]
# Застосування падинґу
train_labels = pad_sequences(train_labels, maxlen=configs.max_text_length, padding=“post”,
value=-1)
val_labels = pad_sequences(val_labels, maxlen=configs.max_text_length, padding=“post”,
value=-1)
# Підготовка наборів даних
train_dataset = tf.data.Dataset.from_tensor_slices((train_images, train_labels))
train_dataset = train_dataset.batch(configs.batch_size).prefetch(tf.data.AUTOTUNE)
val_dataset = tf.data.Dataset.from_tensor_slices((val_images, val_labels))
val_dataset = val_dataset.batch(configs.batch_size).prefetch(tf.data.AUTOTUNE)
```

Модель розпізнавання рукописного тексту базується на згортковій нейронній мережі. Модель отримує зображення на вході і створює послідовність міток (символьних позначок) для кожного зображення на виході.

Шари згортки використовуються для виділення ознак із зображень. LSTM-шари (long short-term memory) [2] фіксують зв’язки між символами в мітках, що дозволяє краще розуміти контекст та порядок символів. Вихідні дані з шарів LSTM проходять через щільний шар з активацією softmax [3], який створює розподіл ймовірностей для символів у словнику для кожного часового кроку, що дозволяє передбачити правильну мітку для кожного символу на вхідному зображенні.

Процес навчання моделі базується на використанні функції втрат CTC, яка дозволяє моделі прогнозувати послідовності символів без явного вирівнювання між входом і виходом. Цей підхід є ключовим для обробки даних, де довжина вхідних і вихідних послідовностей може відрізнятися.

Оптимізація параметрів здійснюється за допомогою алгоритму Adam [4], який автоматично налаштовує швидкість навчання, використовуючи градієнти першого та другого порядків. Це забезпечує ефективну збіжність та покращує стабільність навчання.

Оцінка ефективності моделі виконується за допомогою метрик CWER та WER [5]. CWER визначає відсоток помилок на рівні символів, а WER оцінює точність розпізнавання на рівні слів у реченнях. Комбінація цих метрик дозволяє точно виміряти продуктивність моделі як на символічному, так і на словесному рівнях.

Визначимо зворотні виклики, які будуть використовуватися під час навчання:

```
callbacks = [
   EarlyStopping(patience=10, restore_best_weights=True),
   ModelCheckpoint(os.path.join(configs.model_path,“best_model.h5”),save_best_only=True),
   ReduceLROnPlateau(patience=5, factor=0.5, verbose=1),
   TensorBoard(log_dir=os.path.join(configs.model_path, “logs”)),
   MetricsLogger(val_data=val_dataset, char_list=configs.vocab)
]
```

EarlyStopping [6] – зупиняє навчання, якщо продуктивність на валідації не покращується протягом 10 епох (patience=10). Параметр restore_best_weights=True відновлює ваги з найкращої епохи;

ModelCheckpoint [6] – зберігає ваги моделі у файл наприкінці кожної епохи в разі покращення продуктивності на тестовому наборі;

ReduceLROnPlateau [6] – зменшує швидкість навчання оптимізатора вдвічі, якщо продуктивність моделі на валідаційному наборі не покращується протягом заданих 5 епох. Повідомлення про зміну швидкості виводиться в консоль;

TensorBoard [7] – записує метрики навчання та валідації в лог-файли, які потім можна переглянути за допомогою веб-інструмента TensorBoard для візуалізації та аналізу;

MetricsLogger – вимірює та виводить CWER та WER під час тренування моделі для оцінки її продуктивності, допомагає контролювати якість моделі під час тренувального процесу.

Після навчання модель зберігається для подальшого використання.

Тренування моделі:

```
model.fit(
   train_dataset,
   validation_data=val_dataset,
   epochs=configs.train_epochs,
   callbacks=callbacks
)
```

Процес тренування можна побачити у терміналі під час запуску програми (рис. 2).

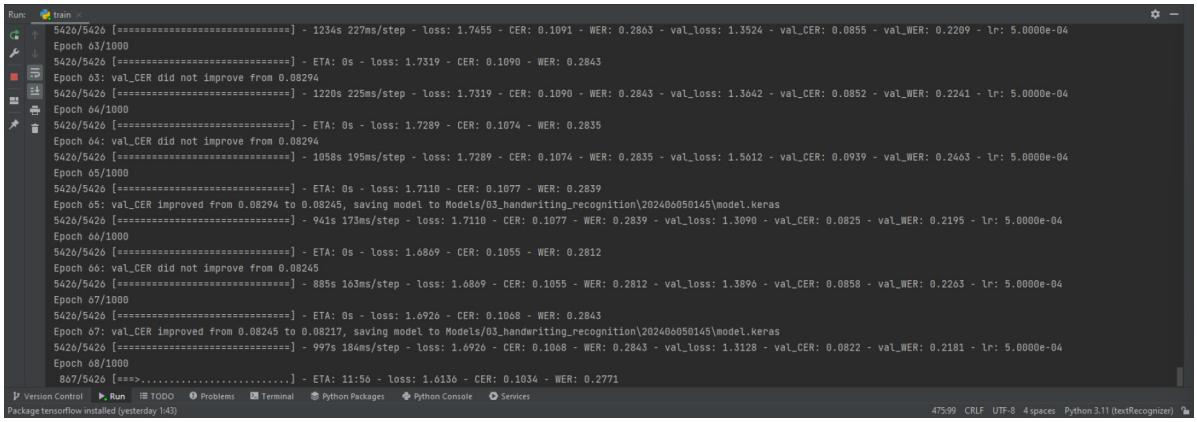


Рис. 2. Процес тренування моделі

Проаналізуємо як навчилася модель, відкривши TensorBoard і переглянувши на метрики оцінки Character Error Rate (рис. 3) і Word Error Rate (рис. 4). З графіків бачимо поступове покращення передбачень моделі, яке в результаті навчання впродовж 67 епох досягла помилки в символах 0.1104 що означає що лише у 11.04 % випадків символ передбачено неправильно. Цей результат є доволі високим.

Помилки ж на рівні слів близько 30 %. Крім цього, можна зауважити аномальні стрибки значень графіку, причиною яких скоріше за все є присутність розділових знаків у наборі даних. Після завершення процесу навчання проведемо тестування моделі на валідаційному наборі даних, щоб оцінити її продуктивність на даних, яких вона не бачила під час навчання.

Для цього використаємо ті ж метрики, які показує відсоток символів та слів, які модель прогнозує неправильно. Показник CER був обраний, оскільки він найбільш точно відображає точність розпізнавання на рівні

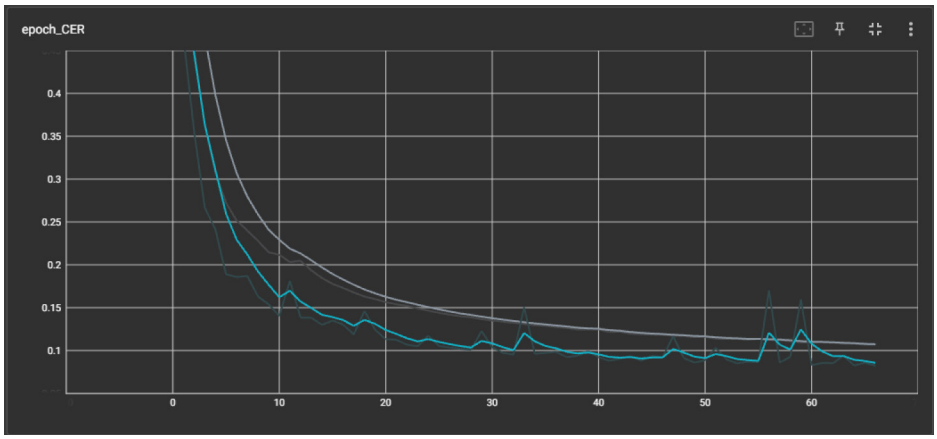


Рис. 3. Character Error Rate

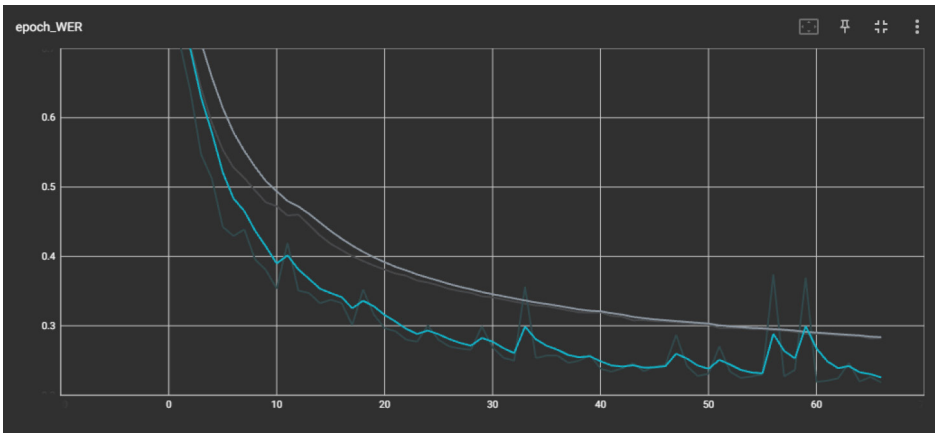


Рис. 4. Word Error Rate

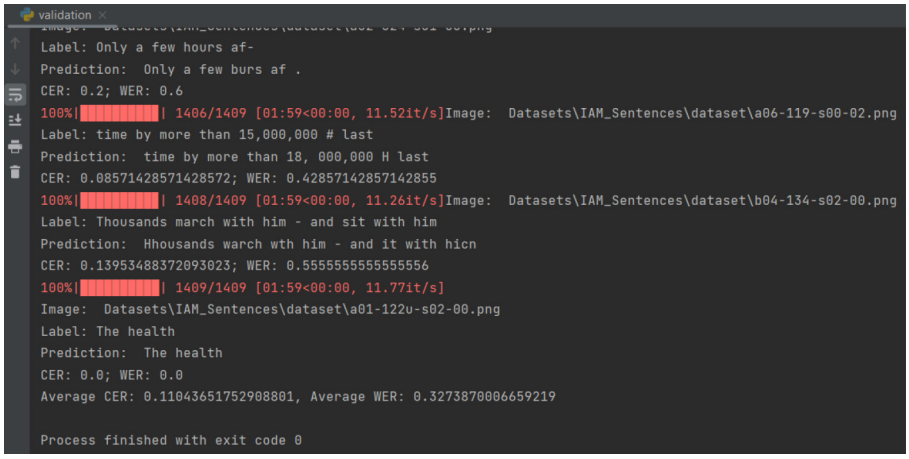


Рис. 5. Визначення точності моделі

Рис. 6. Приклад розпізнавання тексту зображення

символів. Процес перевірки на валідаційних даних з IAM Sentences та на зображеннях тексту на рисунках 5 та 6 відповідно.

Цей результат є досить хорошим для задачі розпізнавання рукописного тексту, враховуючи складність розпізнавання різних почерків та можливих варіацій у написанні символів. Проте модель має потенціал для подальшого покращення шляхом оптимізації даних та налаштувань параметрів.

Висновки

Було детально розглянуто процес підготовки набору даних IAM Sentences для навчання моделі розпізнавання рукописного тексту, включаючи читання та розбір даних, побудову та навчання моделі на основі згорткових нейронних мереж. Навчання моделі проводилося з використанням функції втрат CTC та оптимізатора Adam та дозволило досягти високої точності розпізнавання. Оцінка якості моделі проводилася за допомогою метрики CER та інструментарію TensorBoard та показала ефективність розробленої системи.

Було проаналізовано результати роботи моделі розпізнавання рукописного тексту, в результаті тестування на валідаційному наборі даних. Було встановлено, що середній CER у 11.04 % свідчить про високу точність моделі, проте має потенціал для подальшого покращення. Розглянуто можливі напрямки для подальшого розвитку та оптимізації системи, що допоможе покращити її продуктивність та розширити спектр застосувань

Навчання моделі на більшому або більш різноманітному наборі даних може покращити її здатність до узагальнення. Додавання додаткових шарів або параметрів, а також використання інших оптимізаторів може покращити продуктивність моделі. Використання методів доповнення, таких як додавання шуму або перетворення зображень, допоможе моделі краще узагальнювати дані. Крім цього, в майбутньому модель можна модифікувати для розпізнавання рукописів з інших мов.

В подальшому буде розроблено функціонал API, яке прийматиме фотографію рукописного тексту і надсилатиме у відповідь розпізнаний текст та сам мобільний застосунок, який взаємодіятиме з серверною частиною у вигляді API.

Список використаної літератури

1. IAM Handwriting Database. URL: https://fki.tic.heia-fr.ch/databases/iam-handwriting-database (дата звернення: 07.06.2025).
2. Discover LSTM. NVIDIA Developer. 2024. URL: https://developer.nvidia.com/discover/lstm (дата звернення: 07.06.2025).
3. The Role of Softmax in Neural Networks: Detailed Explanation and Applications. GeeksforGeeks. 2024. URL: https://www.geeksforgeeks.org/the-role-of-softmax-in-neural-networks-detailed-explanation-and-applications/ (дата звернення: 07.06.2025).
4. Adam Optimizer. Keras. 2024. URL: https://keras.io/api/optimizers/adam/ (дата звернення: 07.06.2025).
5. WER, CER, MER Metrics. Kolena. 2024. URL: https://docs.kolena.com/metrics/wer-cer-mer/ (дата звернення: 07.06.2025).
6. Keras Callbacks API. TensorFlow. 2024. URL: https://www.tensorflow.org/api_docs/python/tf/keras/callbacks (дата звернення: 07.06.2025).
7. TensorBoard. TensorFlow. 2024. URL: https://www.tensorflow.org/tensorboard (дата звернення: 07.06.2025).
8. Nebauer C. Evaluation of convolutional neural networks for visual recognition. IEEE Transactions on Neural Networks. 1998. Vol. 9, no. 4. P. 685–696. DOI: 10.1109/72.701181.
9. Yamashita R., Nishio M., Do R. K. G. et al. Convolutional neural networks: an overview and application in radiology. Insights Imaging. 2018. Vol. 9. P. 611–629. DOI: 10.1007/s13244-018-0639-9.
10. Mienye I. D., Swart T. G., Obaido G. Recurrent Neural Networks: A Comprehensive Review of Architectures, Variants, and Applications. Information. 2024. Vol. 15, no. 9. P. 517. DOI: 10.3390/info15090517.

References

1. IAM Handwriting Database. URL: https://fki.tic.heia-fr.ch/databases/iam-handwriting-database. [in English].
2. Discover LSTM. NVIDIA Developer. 2024. URL: https://developer.nvidia.com/discover/lstm. [in English].
3. The Role of Softmax in Neural Networks: Detailed Explanation and Applications. GeeksforGeeks. 2024. URL: https://www.geeksforgeeks.org/the-role-of-softmax-in-neural-networks-detailed-explanation-and-applications/ [in English].
4. Adam Optimizer. Keras. 2024. URL: https://keras.io/api/optimizers/adam/. [in English].
5. WER, CER, MER Metrics. Kolena. 2024. URL: https://docs.kolena.com/metrics/wer-cer-mer/. [in English].
6. Keras Callbacks API. TensorFlow. 2024. URL: https://www.tensorflow.org/api_docs/python/tf/keras/callbacks [in English].

7. TensorBoard. TensorFlow. 2024. URL: https://www.tensorflow.org/tensorboard [in English].

8. Nebauer C. Evaluation of convolutional neural networks for visual recognition. IEEE Transactions on Neural Networks. 1998. Vol. 9, no. 4. P. 685–696. DOI: 10.1109/72.701181 [in English].

9. Yamashita R., Nishio M., Do R. K. G. et al. Convolutional neural networks: an overview and application in radiology. Insights Imaging. 2018. Vol. 9. P. 611–629. DOI: 10.1007/s13244-018-0639-9 [in English].

10. Mienye I. D., Swart T. G., Obaido G. Recurrent Neural Networks: A Comprehensive Review of Architectures, Variants, and Applications. Information. 2024. Vol. 15, no. 9. P. 517. DOI: 10.3390/info15090517 [in English].