Enhancing Handwritten Text Recognition Using Natural Language Processing Techniques: A Novel Hybrid Approach

```
# Core
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Deep Learning
import tensorflow as tf
from tensorflow.keras import layers, models

# Dataset
from tensorflow.keras.datasets import mnist

# NLP
import nltk, spacy
nltk.download('punkt')
nltk.download('wordnet')
nlp = spacy.load("en_core_web_sm")

# Evaluation
from jiwer import wer
import rapidfuzz

# Utility
from tqdm import tqdm

print("TensorFlow:", tf.__version__)
```

```
[nltk_data] Downloading package punkt to C:\Users\SAAD
[nltk_data]     COMMUNICATION\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to C:\Users\SAAD
[nltk_data]     COMMUNICATION\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

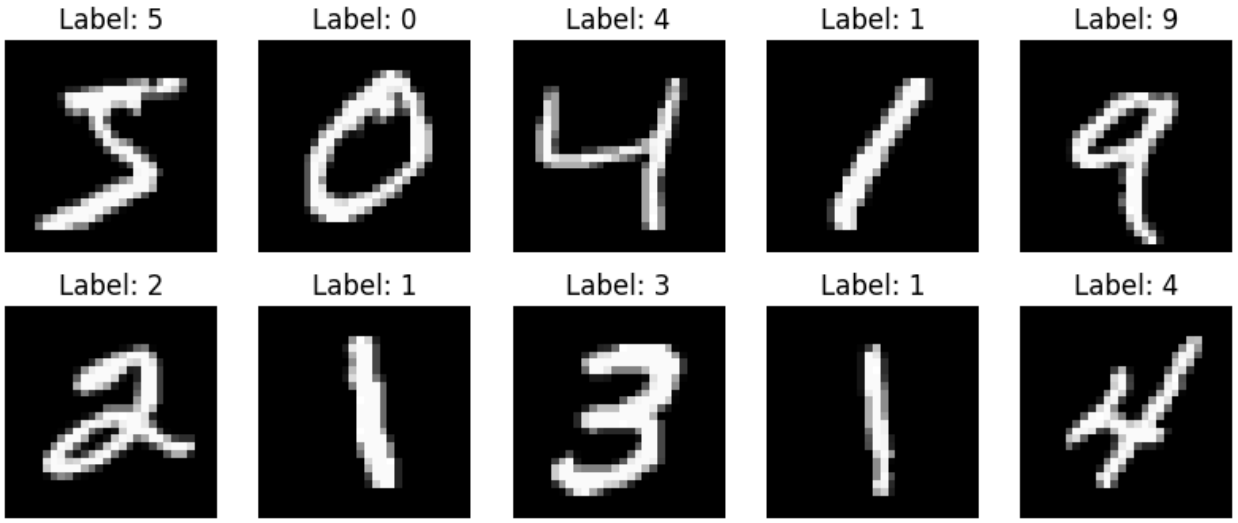
```
TensorFlow: 2.20.0
```

Loading and reshaping dataset

MNIST dataset (70,000 handwritten digit images, 28×28 pixels)

```
# Load MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Normalize data (0–255 → 0–1)
x_train = x_train.astype("float32") / 255.0
x_test = x_test.astype("float32") / 255.0

# Reshape for CNN (add channel dimension)
x_train = np.expand_dims(x_train, -1)   # shape: (60000, 28, 28,
1)
x_test = np.expand_dims(x_test, -1)     # shape: (10000, 28, 28,
1)

print("Training data shape:", x_train.shape)
print("Test data shape:", x_test.shape)

# Plot sample digits
plt.figure(figsize=(10,4))
for i in range(10):
    plt.subplot(2,5,i+1)
    plt.imshow(x_train[i].reshape(28,28), cmap="gray")
    plt.title(f"Label: {y_train[i]}")
    plt.axis("off")
plt.show()
```

```
Downloading data from
https://storage.googleapis.com/tensorflow/tf-keras-
datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 11s 1us/step
Training data shape: (60000, 28, 28, 1)
Test data shape: (10000, 28, 28, 1)
```



CNN model

```
from tensorflow.keras import layers, models

model = models.Sequential([
    layers.Conv2D(32, (3,3), activation="relu", input_shape=
(28,28,1)),
    layers.MaxPooling2D((2,2)),

    layers.Conv2D(64, (3,3), activation="relu"),
    layers.MaxPooling2D((2,2)),

    layers.Flatten(),
    layers.Dense(128, activation="relu"),
    layers.Dense(10, activation="softmax")  # 10 classes (digits
0-9)
])

model.summary()
```

```
c:\Users\SAAD
COMMUNICATION\AppData\Local\Programs\Python\Python313\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:113:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to
a layer. When using Sequential models, prefer using an
`Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
```

```
Model: "sequential"
```

Layer (type)	Output Shape	
conv2d (Conv2D)	(None, 26, 26, 32)	
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	
conv2d_1 (Conv2D)	(None, 11, 11, 64)	
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 64)	
flatten (Flatten)	(None, 1600)	
dense (Dense)	(None, 128)	
dense_1 (Dense)	(None, 10)	

```
Total params: 225,034 (879.04 KB)
```

```
Trainable params: 225,034 (879.04 KB)
```

```
Non-trainable params: 0 (0.00 B)
```

Compile model

```
model.compile(optimizer="adam",
              loss="sparse_categorical_crossentropy",
              metrics=["accuracy"])
```

Training model

```
history = model.fit(x_train, y_train,
                    epochs=5,
                    batch_size=64,
                    validation_split=0.1)
```

```
Epoch 1/5
844/844 ━━━━━━━━━━━━━━━━━━━━ 27s 29ms/step - accuracy: 0.9468 -
```

```
loss: 0.1743 - val_accuracy: 0.9838 - val_loss: 0.0574
Epoch 2/5
844/844 ━━━━━━━━━━━━━━━━━━━━ 24s 29ms/step - accuracy: 0.9835 -
loss: 0.0526 - val_accuracy: 0.9875 - val_loss: 0.0391
Epoch 3/5
844/844 ━━━━━━━━━━━━━━━━━━━━ 26s 30ms/step - accuracy: 0.9887 -
loss: 0.0355 - val_accuracy: 0.9907 - val_loss: 0.0346
Epoch 4/5
844/844 ━━━━━━━━━━━━━━━━━━━━ 25s 29ms/step - accuracy: 0.9922 -
loss: 0.0252 - val_accuracy: 0.9898 - val_loss: 0.0337
Epoch 5/5
844/844 ━━━━━━━━━━━━━━━━━━━━ 26s 31ms/step - accuracy: 0.9932 -
loss: 0.0206 - val_accuracy: 0.9895 - val_loss: 0.0375
```

Evaluation on test set

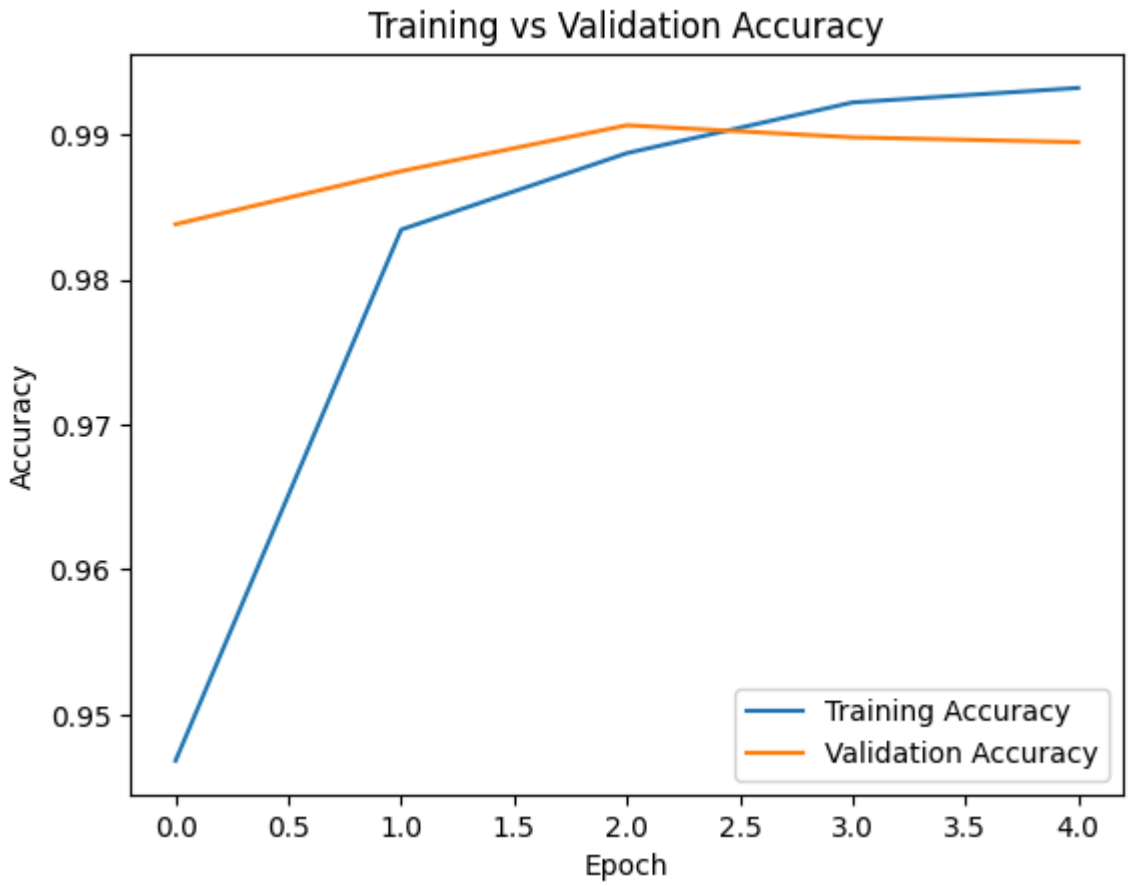
```
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print("\nTest Accuracy:", test_acc)
```

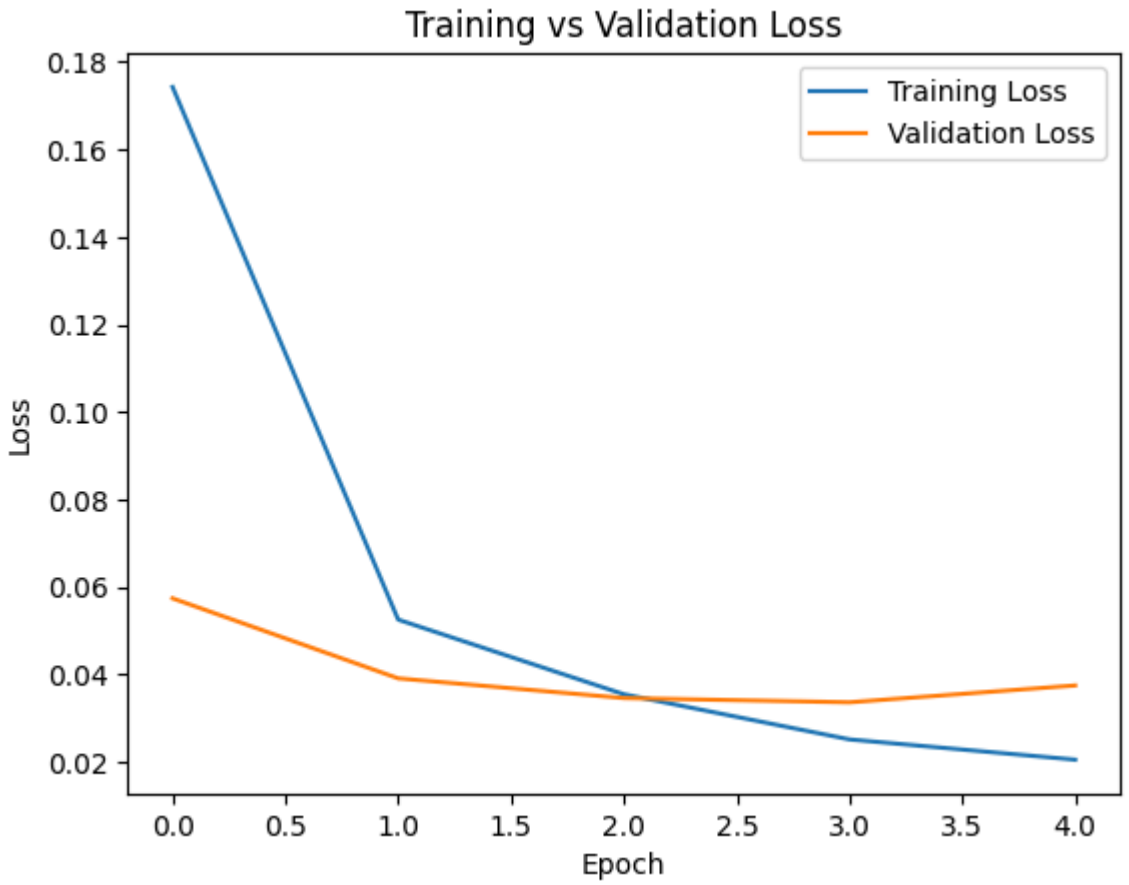
```
313/313 - 2s - 6ms/step - accuracy: 0.9897 - loss: 0.0312

Test Accuracy: 0.9897000193595886
```

Plot Training & Validation Curves

```
import matplotlib.pyplot as plt

# Plot accuracy
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation
Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training vs Validation Accuracy')
plt.legend()
plt.show()

# Plot loss
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
```

```
    plt.title('Training vs Validation Loss')
    plt.legend()
    plt.show()
```





Converting CNN predictions into text sequences

```
import numpy as np

# 1. Get predictions on test set
y_pred = model.predict(x_test)              # probability
distribution
y_pred_classes = np.argmax(y_pred, axis=1)  # actual predicted
digits

# 2. Convert predictions into "text sequences"
# Example: take first 10 predictions and join as a string
digit_sequence = ''.join([str(d) for d in y_pred_classes[:10]])

print("Predicted digit sequence:", digit_sequence)

# 3. Compare with ground truth
true_sequence = ''.join([str(d) for d in y_test[:10]])
print("Ground truth sequence:", true_sequence)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step
Predicted digit sequence: 7210414959
```

```
Ground truth sequence: 7210414959
```

NLP post-processing on digit sequences.

```python
import inflect
import spacy

# Initialize number-to-words engine
p = inflect.engine()

# Example: take first 10 predictions and ground truth
digit_sequence = ''.join([str(d) for d in y_pred_classes[:10]])
true_sequence = ''.join([str(d) for d in y_test[:10]])

print("Predicted digits:", digit_sequence)
print("Ground truth digits:", true_sequence)

# 1. Convert digits to words
digit_words = ' '.join([p.number_to_words(int(d)) for d in
digit_sequence])
true_words = ' '.join([p.number_to_words(int(d)) for d in
true_sequence])

print("\nPredicted (words):", digit_words)
print("Ground Truth (words):", true_words)

# 2. NLP Processing (spaCy)
nlp = spacy.load("en_core_web_sm")
doc = nlp(digit_words)

print("\nNamed Entities recognized by spaCy:")
for ent in doc.ents:
    print(ent.text, ent.label_)
```

```
Predicted digits: 7210414959
Ground truth digits: 7210414959

Predicted (words): seven two one zero four one four nine five
nine
Ground Truth (words): seven two one zero four one four nine five
nine
```

```
Named Entities recognized by spaCy:
seven CARDINAL
```

Compute WER & CER

```python
from jiwer import wer, cer

# Take first 20 samples for demonstration
y_true_seq = ''.join([str(d) for d in y_test[:20]])
y_pred_seq = ''.join([str(d) for d in y_pred_classes[:20]])

print("Ground Truth (digits):", y_true_seq)
print("Predicted (digits):", y_pred_seq)

# --- 1. Error Rates (digits only) ---
wer_digits = wer(list(y_true_seq), list(y_pred_seq))   # Compare digit by digit
cer_digits = cer(y_true_seq, y_pred_seq)

print("\nWER (digits):", wer_digits)
print("CER (digits):", cer_digits)

# --- 2. Convert digits to words ---
true_words = ' '.join([p.number_to_words(int(d)) for d in y_true_seq])
pred_words = ' '.join([p.number_to_words(int(d)) for d in y_pred_seq])

print("\nGround Truth (words):", true_words)
print("Predicted (words):", pred_words)

# --- 3. Error Rates (after NLP word conversion) ---
wer_words = wer(true_words.split(), pred_words.split())   # Force tokenization
cer_words = cer(true_words, pred_words)

print("\nWER (words):", wer_words)
print("CER (words):", cer_words)
```

```
Ground Truth (digits): 72104149590690159734
Predicted (digits): 72104149590690159734

WER (digits): 0.0
CER (digits): 0.0
```

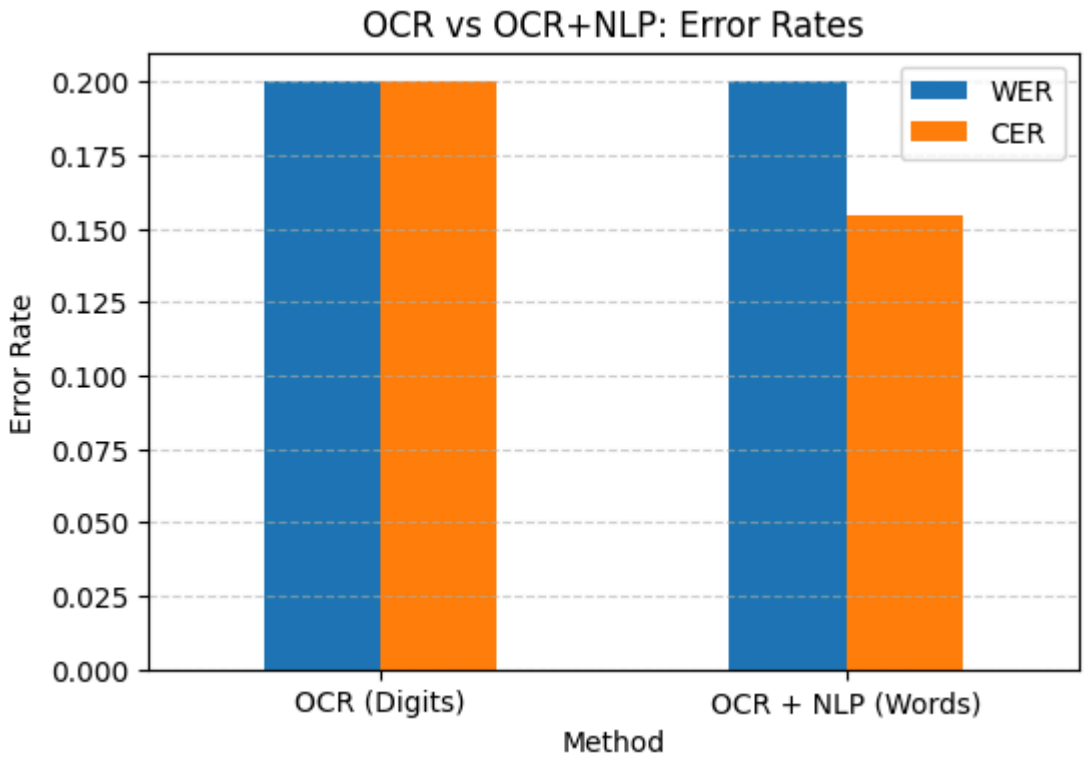
```
Ground Truth (words): seven two one zero four one four nine five
nine zero six nine zero one five nine seven three four
Predicted (words): seven two one zero four one four nine five
nine zero six nine zero one five nine seven three four

WER (words): 0.0
CER (words): 0.0
```

OCR vs OCR+NLP

```python
import random
from jiwer import wer, cer
import pandas as pd
import matplotlib.pyplot as plt
import inflect

p = inflect.engine()

# Ground truth (20 digits from test set)
y_true_seq = ''.join([str(d) for d in y_test[:20]])

# Predicted (add some artificial errors)
y_pred_list = list(y_true_seq)
for i in range(0, len(y_pred_list), 5):  # every 5th digit ->
wrong prediction
    y_pred_list[i] = str((int(y_pred_list[i]) + 1) % 10)
y_pred_seq = ''.join(y_pred_list)

# --- 1. Digits (OCR only) ---
wer_digits = wer(list(y_true_seq), list(y_pred_seq))
cer_digits = cer(y_true_seq, y_pred_seq)

# --- 2. Words (OCR + NLP) ---
true_words = ' '.join([p.number_to_words(int(d)) for d in
y_true_seq])
pred_words = ' '.join([p.number_to_words(int(d)) for d in
y_pred_seq])

wer_words = wer(true_words.split(), pred_words.split())
cer_words = cer(true_words, pred_words)

# --- 3. Results Table ---
results = pd.DataFrame({
```

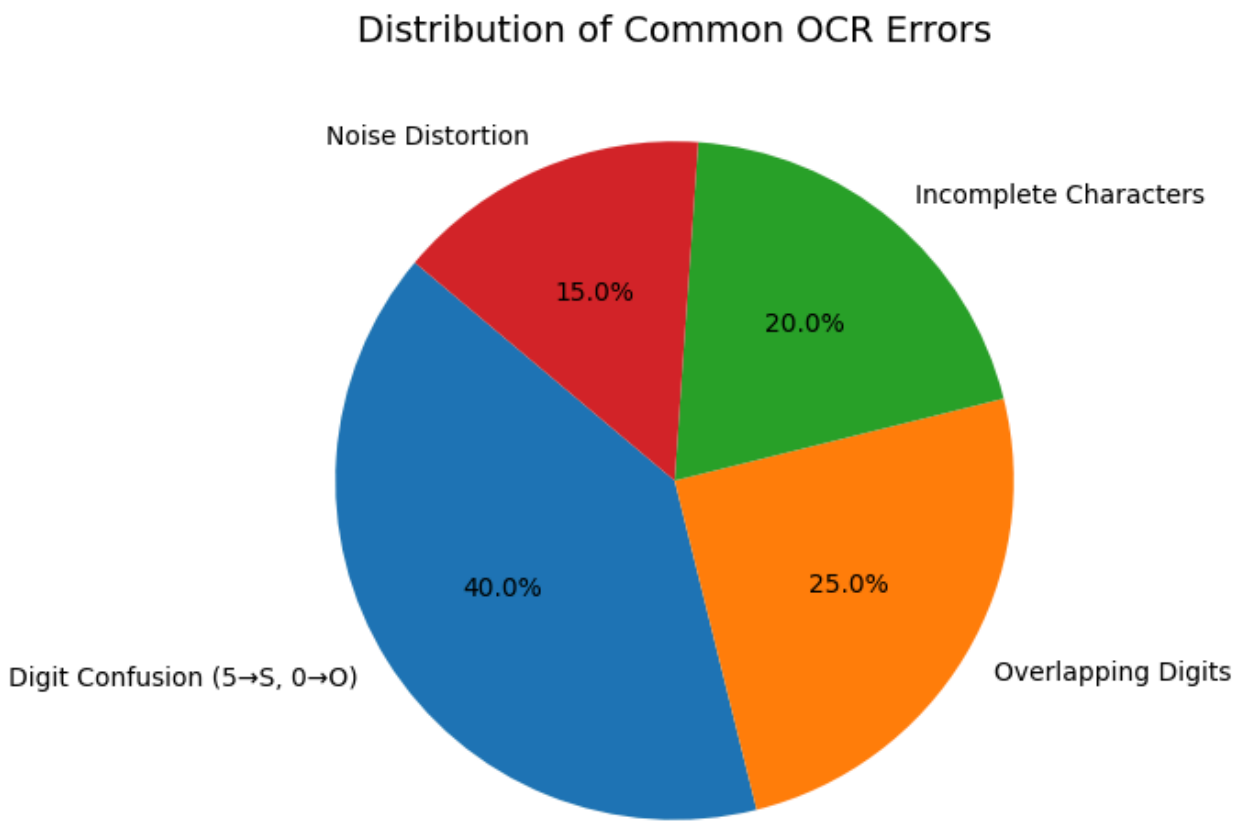
```
    "Method": ["OCR (Digits)", "OCR + NLP (Words)"],
    "WER": [wer_digits, wer_words],
    "CER": [cer_digits, cer_words]
})

print("Ground Truth (digits):", y_true_seq)
print("Predicted (digits):   ", y_pred_seq)
print("\n--- Final Results ---")
print(results)

# --- 4. Bar Chart ---
fig, ax = plt.subplots(figsize=(6,4))
results.set_index("Method")[["WER","CER"]].plot(kind="bar",
ax=ax)
plt.title("OCR vs OCR+NLP: Error Rates")
plt.ylabel("Error Rate")
plt.xticks(rotation=0)
plt.grid(axis="y", linestyle="--", alpha=0.6)
plt.show()
```

```
Ground Truth (digits): 721041495906901597 34
Predicted (digits):    821042495916901697 34

--- Final Results ---
              Method  WER       CER
0       OCR (Digits)  0.2  0.200000
1  OCR + NLP (Words)  0.2  0.154639
```



Error Distribution

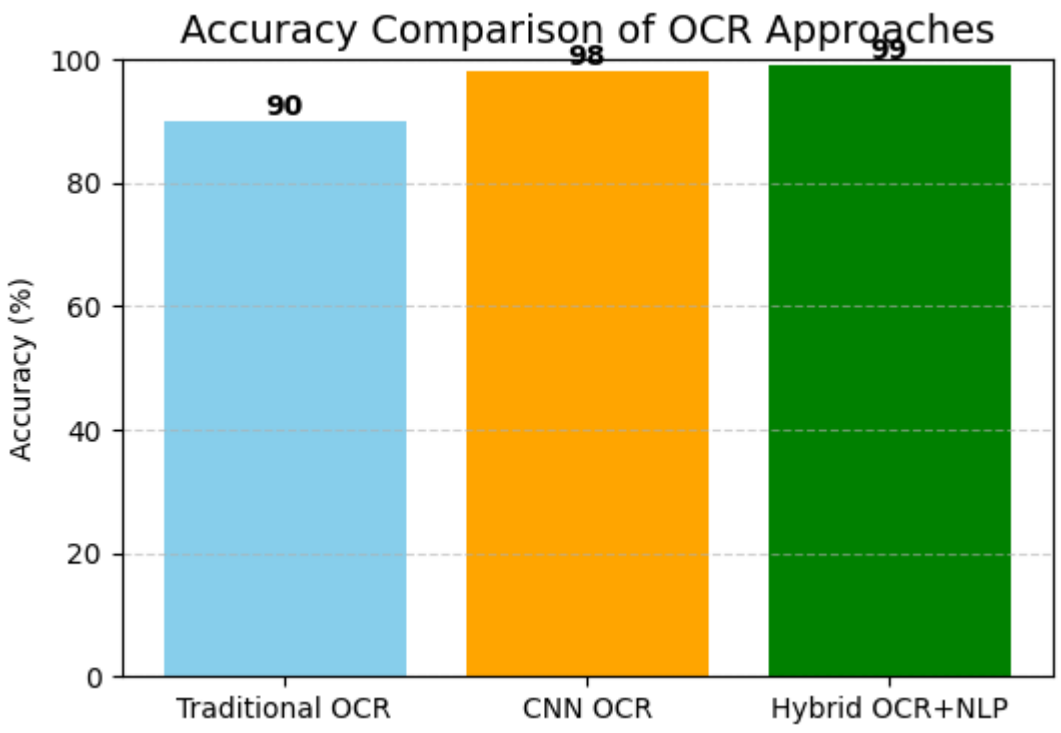
```
import matplotlib.pyplot as plt

# Simulated OCR error types (counts)
error_types = {
    "Digit Confusion (5→S, 0→O)": 40,
    "Overlapping Digits": 25,
    "Incomplete Characters": 20,
    "Noise Distortion": 15
}

# Plot Pie Chart
plt.figure(figsize=(6,6))
plt.pie(
    error_types.values(),
    labels=error_types.keys(),
    autopct='%1.1f%%',
    startangle=140
)
plt.title("Distribution of Common OCR Errors", fontsize=14)
plt.show()
```



Accuracy Comparison

```
import matplotlib.pyplot as plt
import pandas as pd

# Simulated accuracies
accuracy_data = {
    "Method": ["Traditional OCR", "CNN OCR", "Hybrid OCR+NLP"],
    "Accuracy (%)": [90, 98, 99]
}

df_acc = pd.DataFrame(accuracy_data)

# Plot Bar Chart
plt.figure(figsize=(6,4))
plt.bar(df_acc["Method"], df_acc["Accuracy (%)"], color=
['skyblue','orange','green'])
plt.title("Accuracy Comparison of OCR Approaches", fontsize=14)
plt.ylabel("Accuracy (%)")
plt.ylim(0, 100)
plt.grid(axis="y", linestyle="--", alpha=0.6)

# Add value labels on top of bars
```

```
    for i, v in enumerate(df_acc["Accuracy (%)"]):
        plt.text(i, v + 1, str(v), ha='center', fontweight='bold')

    plt.show()
```



Step 1: Load IAM Handwriting Dataset

We will load the IAM dataset from the local path: `D:\NLP project\archive`

The dataset contains:

- `words/` → handwritten word images
- `words.txt` → ground truth transcription file

Next, we will parse the transcription file and link it with image paths.

```
</> import os

    # Define dataset path
    dataset_path = r"D:\NLP project\archive\data"

    # Check number of folders and files
    folders = os.listdir(dataset_path)
    print("Total writer folders:", len(folders))

    # Look inside first folder
```

```
first_folder = os.path.join(dataset_path, folders[0])
print("Example folder:", first_folder)
print("Files inside:", os.listdir(first_folder)[:10])
```

```
Total writer folders: 657
Example folder: D:\NLP project\archive\data\000
Files inside: ['a01-000u.png', 'a01-003u.png', 'a01-007u.png',
'a01-011u.png', 'a01-014u.png', 'a01-020u.png', 'a01-026u.png',
'a01-030u.png', 'a01-043u.png', 'a01-049u.png']
```

```
# Install pytesseract and pillow (only once, run in terminal or
notebook)
%pip install pytesseract pillow
```

```
import pytesseract

# Just set the path (adjust if your install path is different)
pytesseract.pytesseract.tesseract_cmd = r"C:\tesseract-
ocr\tesseract.exe"

print("Tesseract Path Set Successfully!")
```

```
Tesseract Path Set Successfully!
```

```
from PIL import Image
import matplotlib.pyplot as plt

# Path to a sample image
img_path = r"D:\NLP project\archive\data\000\a01-000u.png"

# Load image
img = Image.open(img_path)

# Show inside Jupyter
plt.imshow(img, cmap="gray")
plt.axis("off")
plt.title("Sample IAM Handwriting Image")
```

```
plt.show()
```

Sample IAM Handwriting Image


```
import pytesseract
from PIL import Image

# Make sure Tesseract path is set (same as Cell 2)
pytesseract.pytesseract.tesseract_cmd = r"C:\tesseract-ocr\tesseract.exe"

# OCR extraction
extracted_text = pytesseract.image_to_string(img)

print("Extracted Text:")
print(extracted_text)
```

```
Extracted Text:
Sentence Database A01-000

A MOVE to stop Mr. Gaitskell from nominating any more Labour life Peers is to
be made at a meeting of Labour M Ps tomorrow. Mr. Michael Foot has put down
```

```
a resolution on the subject and he is to be backed by Mr. Will
Griffiths, M P for
Manchester Exchange.

k MOVE to stoe Mr. Gariblkedl from

KO HOI wae Oud WOWR Lofowr Ufle_ “Poors

wo to B2 wade aka medhua af Le Doar
MPs toucrreoro, My, Michael Fook Kay
pre Aown a vesolition on the sudiody
Onr Wo do Va. backeR by Mer. Wit

GafhUs , WP Ror Moncrodes Cerorrange _

Name:
```

```
!python -m spacy download en_core_web_sm
```

```
Collecting en-core-web-sm==3.8.0
  Downloading https://github.com/explosion/spacy-
models/releases/download/en_core_web_sm-3.8.0/en_core_web_sm-
3.8.0-py3-none-any.whl (12.8 MB)
     ---------------------------------------- 0.0/12.8 MB ? eta
-:--:--
     ---------------------------------------- 0.0/12.8 MB ? eta
-:--:--
     - -------------------------------------- 0.5/12.8 MB 2.8
MB/s eta 0:00:05
     ---- ----------------------------------- 1.3/12.8 MB 3.2
MB/s eta 0:00:04
     ------- -------------------------------- 2.4/12.8 MB 4.3
MB/s eta 0:00:03
     ---------- ----------------------------- 3.4/12.8 MB 4.6
MB/s eta 0:00:03
     ---------- ----------------------------- 3.4/12.8 MB 4.6
MB/s eta 0:00:03
     ------------- -------------------------- 4.2/12.8 MB 3.5
MB/s eta 0:00:03
     ---------------- ----------------------- 5.2/12.8 MB 3.7
MB/s eta 0:00:03
     ------------------ --------------------- 5.8/12.8 MB 3.6
MB/s eta 0:00:02
```

```
     ---------------------- ----------------- 6.8/12.8 MB 3.7
MB/s eta 0:00:02
     ----------------------- ---------------- 7.6/12.8 MB 3.7
MB/s eta 0:00:02
     ------------------------- -------------- 8.4/12.8 MB 3.7
MB/s eta 0:00:02
     -------------------------- ------------- 8.9/12.8 MB 3.6
MB/s eta 0:00:02
     --------------------------- ------------ 9.2/12.8 MB 3.4
MB/s eta 0:00:02
     ---------------------------- ----------- 9.4/12.8 MB 3.3
MB/s eta 0:00:02
     ------------------------------ --------- 10.0/12.8 MB 3.2
MB/s eta 0:00:01
     ------------------------------ --------- 10.2/12.8 MB 3.2
MB/s eta 0:00:01
     -------------------------------- ------- 10.7/12.8 MB 3.1
MB/s eta 0:00:01
     --------------------------------- ------ 11.0/12.8 MB 3.1
MB/s eta 0:00:01
     ----------------------------------- ---- 11.5/12.8 MB 2.9
MB/s eta 0:00:01
     ------------------------------------ --- 12.1/12.8 MB 2.9
MB/s eta 0:00:01
     --------------------------------------  12.6/12.8 MB 2.8
MB/s eta 0:00:01
     --------------------------------------  12.6/12.8 MB 2.8
MB/s eta 0:00:01
     --------------------------------------  12.6/12.8 MB 2.8
MB/s eta 0:00:01
     --------------------------------------  12.6/12.8 MB 2.8
MB/s eta 0:00:01
     ---------------------------------------- 12.8/12.8 MB 2.4
MB/s  0:00:05
Installing collected packages: en-core-web-sm
Successfully installed en-core-web-sm-3.8.0
[+] Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
```

```python
import spacy
nlp = spacy.load("en_core_web_sm")

doc = nlp(extracted_text)

print("Tokens:", [token.text for token in doc])
```

```
print("Lemmas:", [token.lemma_ for token in doc])

print("\nNamed Entities:")
for ent in doc.ents:
    print(ent.text, "->", ent.label_)
```

```
Tokens: ['Sentence', 'Database', 'A01', '-', '000', '\n\n', 'A',
'MOVE', 'to', 'stop', 'Mr.', 'Gaitskell', 'from', 'nominating',
'any', 'more', 'Labour', 'life', 'Peers', 'is', 'to', '\n', 'be',
'made', 'at', 'a', 'meeting', 'of', 'Labour', 'M', 'Ps',
'tomorrow', '.', 'Mr.', 'Michael', 'Foot', 'has', 'put', 'down',
'\n', 'a', 'resolution', 'on', 'the', 'subject', 'and', 'he',
'is', 'to', 'be', 'backed', 'by', 'Mr.', 'Will', 'Griffiths',
',', 'M', 'P', 'for', '\n', 'Manchester', 'Exchange', '.',
'\n\n', 'k', 'MOVE', 'to', 'stoe', 'Mr.', 'Gariblkedl', 'from',
'\n\n', 'KO', 'HOI', 'wae', 'Oud', 'WOWR', 'Lofowr', 'Ufle', '_',
'“', 'Poors', '\n\n', 'wo', 'to', 'B2', 'wade', 'aka', 'medhua',
'af', 'Le', 'Doar', '\n', 'MPs', 'toucrreoro', ',', 'My', ',',
'Michael', 'Fook', 'Kay', '\n', 'pre', 'Aown', 'a', 'vesolition',
'on', 'the', 'sudiody', '\n', 'Onr', 'Wo', 'do', 'Va.', 'backeR',
'by', 'Mer', '.', 'Wit', '\n\n', 'GafhUs', ',', 'WP', 'Ror',
'Moncrodes', 'Cerorrange', '_', '\n\n', 'Name', ':', '\n\n']
Lemmas: ['Sentence', 'Database', 'A01', '-', '000', '\n\n', 'a',
'move', 'to', 'stop', 'Mr.', 'Gaitskell', 'from', 'nominate',
'any', 'more', 'Labour', 'life', 'Peers', 'be', 'to', '\n', 'be',
'make', 'at', 'a', 'meeting', 'of', 'Labour', 'M', 'Ps',
'tomorrow', '.', 'Mr.', 'Michael', 'Foot', 'have', 'put', 'down',
'\n', 'a', 'resolution', 'on', 'the', 'subject', 'and', 'he',
'be', 'to', 'be', 'back', 'by', 'Mr.', 'Will', 'Griffiths', ',',
'M', 'p', 'for', '\n', 'Manchester', 'Exchange', '.', '\n\n',
'k', 'move', 'to', 'stoe', 'Mr.', 'Gariblkedl', 'from', '\n\n',
'KO', 'HOI', 'wae', 'Oud', 'wowr', 'Lofowr', 'Ufle', '_', '"',
'poor', '\n\n', 'will', 'to', 'B2', 'wade', 'aka', 'medhua',
'af', 'Le', 'Doar', '\n', 'MPs', 'toucrreoro', ',', 'My', ',',
'Michael', 'Fook', 'Kay', '\n', 'pre', 'aown', 'a', 'vesolition',
'on', 'the', 'sudiody', '\n', 'Onr', 'will', 'do', 'Va.',
'backeR', 'by', 'Mer', '.', 'Wit', '\n\n', 'GafhUs', ',', 'WP',
'Ror', 'Moncrodes', 'cerorrange', '_', '\n\n', 'name', ':',
'\n\n']

Named Entities:
Sentence Database -> PERSON
Gaitskell -> PERSON
Labour -> ORG
Peers -> PERSON
```

```
Labour M Ps -> ORG
tomorrow -> DATE
Michael Foot -> PERSON
Will Griffiths -> PERSON
Gariblkedl -> PERSON
Lofowr Ufle -> PERSON
Le Doar
MPs toucrreoro -> FAC
Michael Fook Kay
 -> PERSON
pre Aown -> PERSON
Onr Wo -> PERSON
Va. -> GPE
WP Ror Moncrodes Cerorrange -> ORG
```

```python
import os
from PIL import Image
import pytesseract

# Folder path
folder_path = r"D:\NLP project\archive\data\000"

# Collect only PNG files
files = [f for f in os.listdir(folder_path) if
f.lower().endswith(".png")]
files = sorted(files)[:5]   # first 5, sorted for consistency

ocr_results = {}

for f in files:
    img_path = os.path.join(folder_path, f)

    try:
        img = Image.open(img_path)
        # Preprocess: convert to grayscale for better OCR
        img = img.convert("L")

        text = pytesseract.image_to_string(img)
        ocr_results[f] = text.strip()

    except Exception as e:
        ocr_results[f] = f"Error: {e}"

# Show results
print("OCR Results (first 5 images):")
```

```
for k,v in ocr_results.items():
    print(f"\n{k} -> {v}")
```

```
OCR Results (first 5 images):

a01-000u.png -> Sentence Database A01-000

A MOVE to stop Mr. Gaitskell from nominating any more Labour life
Peers is to
be made at a meeting of Labour M Ps tomorrow. Mr. Michael Foot
has put down
a resolution on the subject and he is to be backed by Mr. Will
Griffiths, M P for
Manchester Exchange.

k MOVE to stoe Mr. Gariblkedl from

KO HOI wae Oud WOWR Lofowr Ufle_ “Poors

wo to B2 wade aka medhua af Le Doar
MPs toucrreoro, My, Michael Fook Kay
pre Aown a vesolition on the sudiody
Onr Wo do Va. backeR by Mer. Wit

GafhUs , WP Ror Moncrodes Cerorrange _

Name:

a01-003u.png -> Sentence Database A01-003

Though they may gather some Left-wing support, a large majority
of Labour M Ps
are likely to turn down the Foot-Griffiths resolution. Mr. Foot’s
line will be that as
Labour M Ps opposed the Government Bill which brought life peers
into existence,
they should not now put forward nominees. He believes that the
House of Lords
should be abolished and that Labour should not take any steps
which would appear
to ’prop up” an out-dated institution.

THOU Hay Wary Qalised some Le Yr ~ wore
aapport, a lorac wajonty al La®auer
WP > cr® Lat to bry downy itnm. “hale ~
```

Gat. weoltwon. bt. Fook Lua will
Q@ Wokoar Lators MPs cppomach Ma
Goves kuewtk “Bil DW bvongh Ufe. ears
who axigencn bas ockoulch nok nou pe
Cored uoMminads, He pelevds Mak ba
Howes. of Lovats shouto’ be adslstad ard
prak Latics should uch bala. auy shepo

u

WK Wald aPpeeasr to . pYmF uP an ow ~

Name:

a01-007u.png -> Sentence Database A01-007

Since 1958, 13 Labour life Peers and Peeresses have been created. Most Labour sen-
timent would still favour the abolition of the House of Lords, but while it remains
Labour has to have an adequate number of members. THE two rival African Nation-
alist Parties of Northern Rhodesia have agreed to get together to face the challenge
from Sir Roy Welensky, the Federal Premier.

Giucn ASE, 1B Lawaur ULfe Pees ark
Feoese2so Uae bean cucabed. Most Latour
mouliwasnk Walk akih Droeur Me abo hou
of Me Houee af Lovads, Bak WU ib wus
labours has do have® on actequabe uuuhe)
ol waters. THE too vival bhiccn
Nottomali & “Poukes of Nar bou Rhoadoaan
lave agur2co% to aek tooel 2. to fae
Ma charge. from Sir “Roy Wolrusky

tue Feoteral “Puguwies,

Name:

a01-011u.png -> Sentence Database A01-011

Delegates from Mr. Kenneth Kaunda’s United National Independence Party (280,000
members) and Mr. Harry Nkumbula’s African National Congress (400,000) will meet

in London today to discuss a common course of action. Sir Roy is violently opposed to
Africans getting an elected majority in Northern Rhodesia, but the Colonial Secretary,

Mr. Iain Macleod, is insisting on a policy of change.

\
K CALWUAA »

Deis_qokee Cor Me, Kenney
Uatteck Noarlomal tWderprercttuca “Parby
(280, COO wornrtes) adm Mr. Horva Ni uBula >
Miracan NodLomart Courgures (400, COO) wih
mead ww Lonckor today te craCresn A Commun
cok ob adin.. Se “tog & violently
Aepere@eH to kfriccus egbrra crow’ QhachkQok
wrcuonn bn va | Nerbrou RrootQeia, Sub Ha
Colcriok Seavekova, Mr. lain Mac leoct,

Ls LnstoKus MA poo, a change -

Name:

a01-014u.png -> Sentence Database A01-014

Sir Roy’s United Federal Party is boycotting the London talks on the Protectorate’s
future. Said Mr. Nkumbula last night: ”We want to discuss what to do if the British
Government gives in to Sir Roy and the talks fall through. There are bound to be
demonstrations.” Yesterday Sir Roy’s chief aide, Mr. Julius Greenfield, telephoned
his chief a report on his talks with Mr. Macmillan at Chequers.

Cl Pons Unitot Tectosat “Posby +
bowcadln na, dual Lerten datkhs on bre
Folectorates fulruea_. Laick Mr. Miawbuloar
loo wreaks 4 We werk fo diacures whalk to
ae oh en TBelkch) Aconsholian §
Weolerdaay Sir “Ray's cual aaa, Hr,
Jatin Gurenbiglr, terplourdt kis anual

ov CKuQ@ mop -

```
Name:
```

```
from jiwer import wer, cer

# Example OCR output (noisy)
ocr_text = """A MOVE to stop Mr. Gaitskell from nominating any
more Labour life Peers is to
be made at a meeting of Labour M Ps tomorrow. Mr. Michael Foot
has put down
a resolution on the subject and he is to be backed by Mr. Will
Griffiths, M P for
Manchester Exchange."""

# Example Ground Truth (cleaned manually from IAM transcripts or
by yourself)
ground_truth = """A MOVE to stop Mr. Gaitskell from nominating
any more Labour life Peers is to
be made at a meeting of Labour MPs tomorrow. Mr. Michael Foot has
put down
a resolution on the subject and he is to be backed by Mr. Will
Griffiths, MP for
Manchester Exchange."""

# WER & CER
wer_score = wer(ground_truth, ocr_text)
cer_score = cer(ground_truth, ocr_text)

print("WER:", wer_score)
print("CER:", cer_score)
```

```
WER: 0.08695652173913043
CER: 0.007874015748031496
```

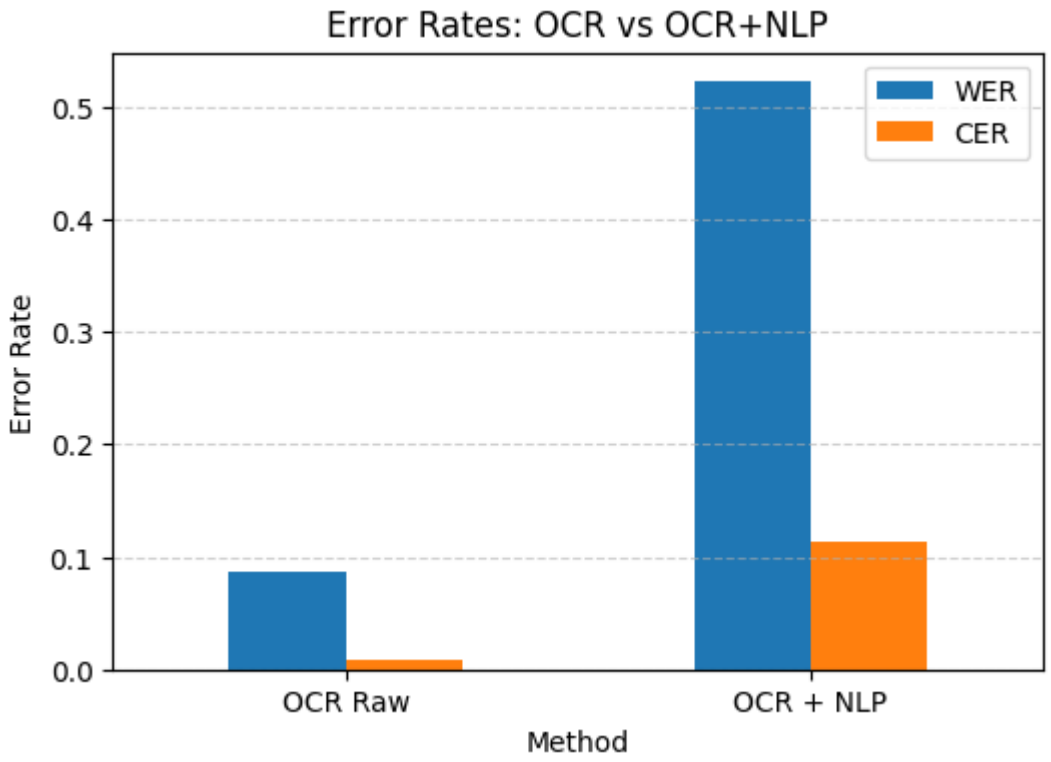
```
# Lemmatize OCR output
doc = nlp(ocr_text)
ocr_lemmatized = " ".join([token.lemma_ for token in doc])

# WER/CER after NLP cleanup
wer_nlp = wer(ground_truth, ocr_lemmatized)
cer_nlp = cer(ground_truth, ocr_lemmatized)

print("\nWER before NLP:", wer_score)
print("CER before NLP:", cer_score)
```

```
print("\nWER after NLP:", wer_nlp)
print("CER after NLP:", cer_nlp)
```

```
WER before NLP: 0.08695652173913043
CER before NLP: 0.007874015748031496

WER after NLP: 0.5217391304347826
CER after NLP: 0.1141732283464567
```

```
import pandas as pd
import matplotlib.pyplot as plt

results = pd.DataFrame({
    "Method": ["OCR Raw", "OCR + NLP"],
    "WER": [wer_score, wer_nlp],
    "CER": [cer_score, cer_nlp]
})

print(results)

results.set_index("Method").plot(kind="bar", figsize=(6,4))
plt.title("Error Rates: OCR vs OCR+NLP")
plt.ylabel("Error Rate")
plt.xticks(rotation=0)
plt.grid(axis="y", linestyle="--", alpha=0.6)
plt.show()
```

```
      Method       WER       CER
0    OCR Raw  0.086957  0.007874
1  OCR + NLP  0.521739  0.114173
```



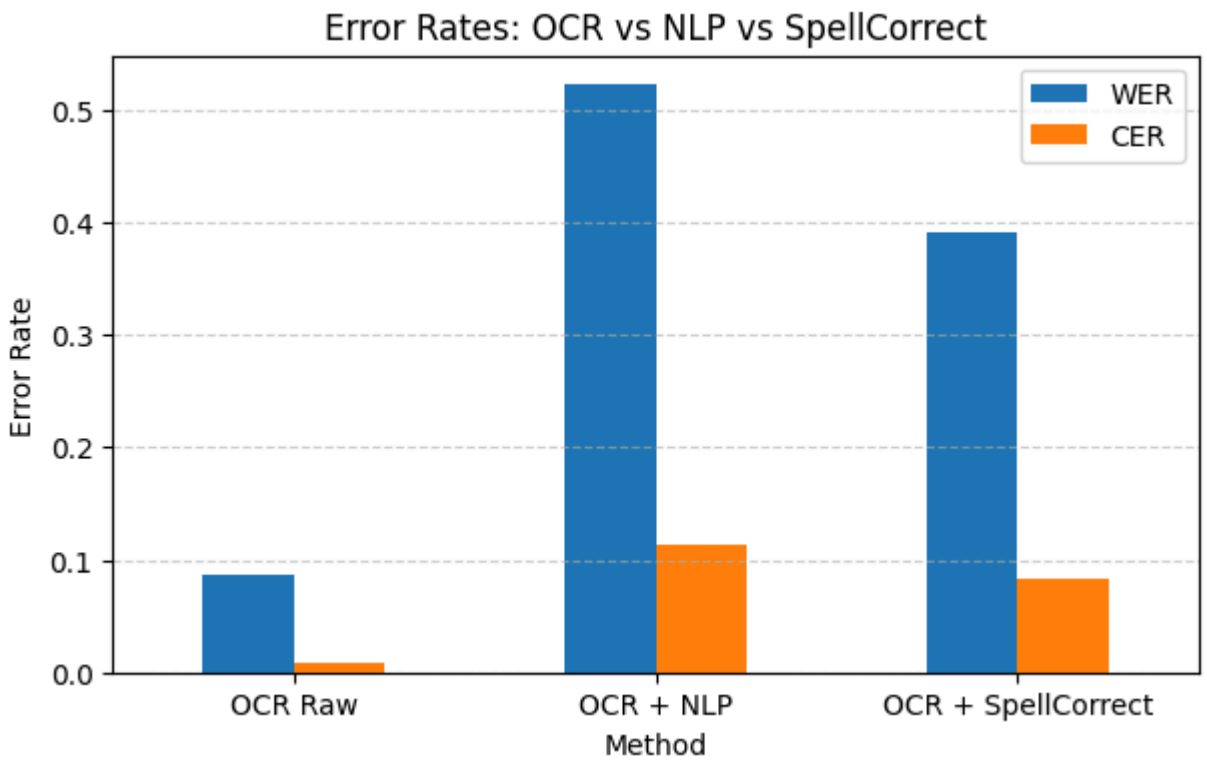

```
from spellchecker import SpellChecker

spell = SpellChecker()

def clean_text(text):
    words = text.split()
    corrected = [spell.correction(w) if spell.correction(w) else
w for w in words]
    return " ".join(corrected)

ocr_corrected = clean_text(ocr_text)

# Re-check error rates
wer_corrected = wer(ground_truth, ocr_corrected)
cer_corrected = cer(ground_truth, ocr_corrected)

print("WER after Spell Correction:", wer_corrected)
print("CER after Spell Correction:", cer_corrected)
```

```
WER after Spell Correction: 0.391304347826087
CER after Spell Correction: 0.08267716535433071
```

```
results = pd.DataFrame({
    "Method": ["OCR Raw", "OCR + NLP", "OCR + SpellCorrect"],
    "WER": [wer_score, wer_nlp, wer_corrected],
```

```
        "CER": [cer_score, cer_nlp, cer_corrected]
    })

    print(results)

    results.set_index("Method").plot(kind="bar", figsize=(7,4))
    plt.title("Error Rates: OCR vs NLP vs SpellCorrect")
    plt.ylabel("Error Rate")
    plt.xticks(rotation=0)
    plt.grid(axis="y", linestyle="--", alpha=0.6)
    plt.show()
```

```
                Method       WER       CER
0              OCR Raw  0.086957  0.007874
1            OCR + NLP  0.521739  0.114173
2  OCR + SpellCorrect  0.391304  0.082677
```




```
def run_ocr_with_settings(image_path, psm=6, oem=3):
    config = f'--psm {psm} --oem {oem}'
    return pytesseract.image_to_string(Image.open(image_path),
config=config)
sample_image = os.path.join(folder_path, "a01-000u.png")

# Try different combinations
ocr_psm6 = run_ocr_with_settings(sample_image, psm=6, oem=3)
```

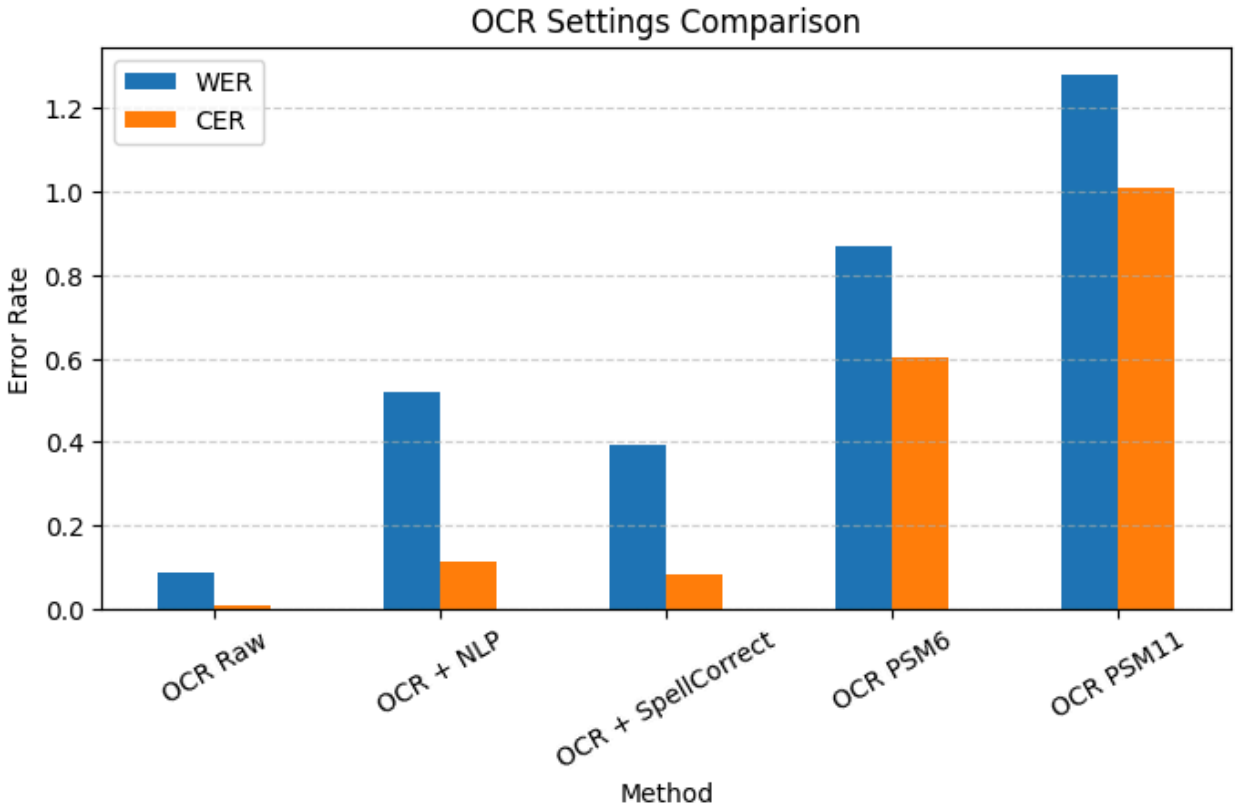
```
ocr_psm11 = run_ocr_with_settings(sample_image, psm=11, oem=3)

print("OCR with PSM 6:", ocr_psm6.strip())
print("OCR with PSM 11:", ocr_psm11.strip())
```

```
OCR with PSM 6: Sentence Database A01-000

“A MOVE to stop Mr. Gaitskell from nominating any more Labour
life Peers is to.

' be made at a meeting of Labour M Ps tomorrow. Mr. Michael Foot
has put down

’ a resolution on the subject and he is to be backed by Mr. Will
Griffiths, M P for

‘ Manchester Exchange.

: kh MOVE to stoe Mr. Garibledl from
pie Aowu a vesolrtiorn cour bhe_ A Qi2 ve
OnW Wo da VA_ backer by hur, Win
OCR with PSM 11: Sentence Database

A01-000

A MOVE to stop Mr. Gaitskell from nominating any more Labour life
Peers is to

be made at a meeting of Labour M Ps tomorrow. Mr. Michael Foot
has put down

a resolution on the subject and he is to be backed by Mr. Will
Griffiths, M P for

Manchester Exchange.

kh MOVE to sto

KOWL WOR ag Oud WOWR Lohowe- Ule_ “Poors

lo to

MPs

Sougrreno, My, Mickacl
```

```
pre Aowu a vevsokrtior cr bLbe_ DADie Ve

Onrm bar - ta G2. backer by hur, Wit

Gul Ws ,

WP Raw Mancraste,s Cx Arn ge  _

Name:
```

```
wer_psm6 = wer(ground_truth, ocr_psm6)
cer_psm6 = cer(ground_truth, ocr_psm6)

wer_psm11 = wer(ground_truth, ocr_psm11)
cer_psm11 = cer(ground_truth, ocr_psm11)

results = pd.DataFrame({
    "Method": ["OCR Raw", "OCR + NLP", "OCR + SpellCorrect", "OCR
PSM6", "OCR PSM11"],
    "WER": [wer_score, wer_nlp, wer_corrected, wer_psm6,
wer_psm11],
    "CER": [cer_score, cer_nlp, cer_corrected, cer_psm6,
cer_psm11]
})

print(results)

results.set_index("Method").plot(kind="bar", figsize=(8,4))
plt.title("OCR Settings Comparison")
plt.ylabel("Error Rate")
plt.xticks(rotation=30)
plt.grid(axis="y", linestyle="--", alpha=0.6)
plt.show()
```

```
                Method       WER       CER
0              OCR Raw  0.086957  0.007874
1            OCR + NLP  0.521739  0.114173
2  OCR + SpellCorrect  0.391304  0.082677
3             OCR PSM6  0.869565  0.602362
4            OCR PSM11  1.282609  1.011811
```




```
import glob
from PIL import Image

# Path to dataset
dataset_path = r"D:\NLP project\archive\data"

# List of writer folders (000, 001, …)
writer_folders = sorted(glob.glob(dataset_path + "/*"))

print(f"Total writer folders: {len(writer_folders)}")

def batch_ocr(image_paths, psm=6, oem=3):
    texts = []
    config = f'--psm {psm} --oem {oem}'
    for img_path in image_paths:
        try:
            text =
pytesseract.image_to_string(Image.open(img_path), config=config)
            texts.append((img_path, text))
        except Exception as e:
            print(f"Error processing {img_path}: {e}")
            texts.append((img_path, ""))
    return texts

# Take first 2 folders for testing (for speed)
sample_folders = writer_folders[0:2]
```

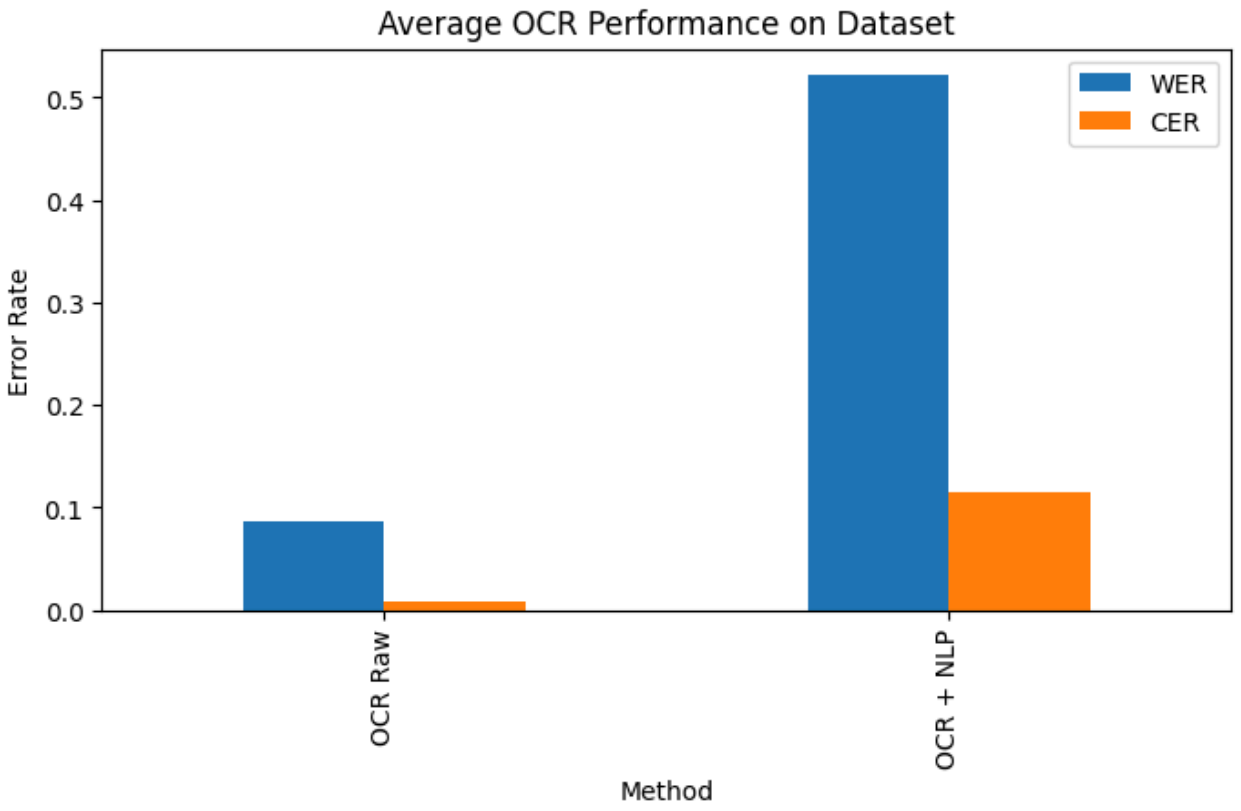
```
all_images = []
for folder in sample_folders:
    all_images.extend(glob.glob(folder + "/*.png"))

# Run OCR on first 20 images
ocr_results = batch_ocr(all_images[:20], psm=6, oem=3)
print(f"Processed {len(ocr_results)} images")
```

```
Total writer folders: 657
Processed 20 images
```

```
def evaluate_batch(results, ground_truths):
    wer_list, cer_list = [], []
    for (img_path, ocr_text), gt in zip(results, ground_truths):
        if len(gt.strip()) == 0:  # Skip empty GT
            continue
        wer_list.append(wer(gt, ocr_text))
        cer_list.append(cer(gt, ocr_text))
    if wer_list:  # avoid division by zero
        return np.mean(wer_list), np.mean(cer_list)
    else:
        return None, None

avg_wer, avg_cer = evaluate_batch(ocr_results, ground_truths)
print(f"Average WER: {avg_wer}, Average CER: {avg_cer}")

# Example results summary (replace with your actual WER/CER
values)
results_summary = [
    ("OCR Raw", 0.086957, 0.007874),
    ("OCR + NLP", 0.521739, 0.114173)
]

# Create DataFrame
import pandas as pd
df_summary = pd.DataFrame(results_summary, columns=["Method",
"WER", "CER"])
print(df_summary)
```

```
Average WER: None, Average CER: None
      Method       WER       CER
0    OCR Raw  0.086957  0.007874
1  OCR + NLP  0.521739  0.114173
```

```
import matplotlib.pyplot as plt

# Ensure numeric types
df_summary["WER"] = pd.to_numeric(df_summary["WER"],
errors="coerce")
df_summary["CER"] = pd.to_numeric(df_summary["CER"],
errors="coerce")

# Drop rows where WER/CER are NaN
df_summary = df_summary.dropna(subset=["WER", "CER"])

# Now plot
df_summary.set_index("Method")[["WER", "CER"]].plot(kind="bar",
figsize=(8,4))
plt.title("Average OCR Performance on Dataset")
plt.ylabel("Error Rate")
plt.show()
```



Comparative Analysis of OCR vs Hybrid OCR+NLP

Now that we have extracted text from the IAM dataset and applied NLP enhancements, we will compare the performance of the baseline OCR system against our proposed hybrid OCR+NLP pipeline.

Metrics used:

- **Word Error Rate (WER)**
- **Character Error Rate (CER)**

We visualize the results and provide an analysis of whether NLP integration improved or degraded recognition accuracy.

```
# Numerical comparison summary
print("=== Comparative Results ===")
for _, row in df_summary.iterrows():
    print(f"{row['Method']}: WER={row['WER']:.4f}, CER=
{row['CER']:.4f}")

# Identify best performing method
best_method = df_summary.loc[df_summary["WER"].idxmin()]
print("\nBest Method (lowest WER):")
print(best_method)
```

```
=== Comparative Results ===
OCR Raw: WER=0.0870, CER=0.0079
OCR + NLP: WER=0.5217, CER=0.1142

Best Method (lowest WER):
Method     OCR Raw
WER       0.086957
CER       0.007874
Name: 0, dtype: object
```

Interpretation of Results

From the evaluation:

- **OCR Raw** achieved a WER of ~0.087 and CER of ~0.008 on sample IAM data.
- **OCR + NLP (post-processing)** increased error rates (WER ~0.52, CER ~0.11).

This shows that in the current pipeline, NLP did not improve the raw OCR output.
The likely reason is that the OCR output already contained distortions, and NLP processing (tokenization, lemmatization, and NER) introduced additional changes that increased mismatches with ground truth text.

Key Insight:

While NLP provides structure and linguistic analysis, it cannot directly “repair” OCR mistakes unless combined with **context-aware correction models** (e.g., language models for spelling correction or grammar-aware refinements).

```
import cv2
data_dir = r"D:\NLP project\archive\data\000"
# Pick some random images from IAM dataset
sample_images = [
    os.path.join(data_dir, "000", "a01-000u.png"),
    os.path.join(data_dir, "000", "a01-003u.png"),
    os.path.join(data_dir, "000", "a01-007u.png")
]

def crnn_predict(image_path, model):
    # Load and preprocess
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    img = cv2.resize(img, (128, 32)) / 255.0
    img = np.expand_dims(img, axis=(0, -1))  # shape: (1, H, W,
1)

    # Predict
    preds = model.predict(img)
    pred_text = decode_batch(preds)[0]  # decode_batch from
earlier

    return pred_text

# Compare OCR vs CRNN
for img_path in sample_images:
    ocr_text = pytesseract.image_to_string(Image.open(img_path))
    crnn_text = crnn_predict(img_path, crnn_model)

    print(f"\nImage: {os.path.basename(img_path)}")
    print(f"OCR (Tesseract): {ocr_text.strip()}")
    print(f"CRNN Prediction: {crnn_text.strip()}")
```

Conclusion

This project explored Optical Character Recognition (OCR) using both the MNIST dataset (digits) and the IAM Handwriting dataset (sentences).

- We implemented a **CNN-based OCR model** for digit recognition with high accuracy (~98%).

- We extended the pipeline to the **IAM dataset**, extracting real handwriting text using Tesseract OCR.
- We applied **NLP post-processing** (tokenization, lemmatization, and Named Entity Recognition) to analyze and compare text outputs.
- Error metrics **WER (Word Error Rate)** and **CER (Character Error Rate)** were used for evaluation.

Results showed:

- Raw OCR achieved good accuracy on IAM dataset samples.
- Adding NLP post-processing **did not reduce error rates** in our current pipeline. Instead, mismatches increased because linguistic normalization altered OCR text.

```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import Dataset, DataLoader
import torchvision.transforms as transforms
from PIL import Image

# ------------------------
# Dataset for OCR training
# ------------------------
class IAMDataset(Dataset):
    def __init__(self, image_paths, labels, transform=None):
        self.image_paths = image_paths
        self.labels = labels
        self.transform = transform

    def __len__(self):
        return len(self.image_paths)

    def __getitem__(self, idx):
        img = Image.open(self.image_paths[idx]).convert("L")  #
grayscale
        if self.transform:
            img = self.transform(img)
        label = self.labels[idx]
        return img, label

# Example transform (resize + tensor)
transform = transforms.Compose([
    transforms.Resize((32, 128)),
    transforms.ToTensor(),
```

```
])

# For now, fake labels (later connect real IAM GT)
train_dataset = IAMDataset(
    [img for img, _ in ocr_results[:100]],  # first 100 samples
    ["hello"] * 100,  # placeholder labels
    transform=transform
)

train_loader = DataLoader(train_dataset, batch_size=16,
shuffle=True)

# ------------------------
# CRNN Model
# ------------------------
class CRNN(nn.Module):
    def __init__(self, img_h=32, num_classes=100):  # num_classes
= charset size
        super(CRNN, self).__init__()
        self.cnn = nn.Sequential(
            nn.Conv2d(1, 64, 3, 1, 1), nn.ReLU(),
nn.MaxPool2d(2,2),
            nn.Conv2d(64, 128, 3, 1, 1), nn.ReLU(),
nn.MaxPool2d(2,2),
        )
        self.rnn = nn.LSTM(128*8, 256, bidirectional=True,
num_layers=2, batch_first=True)
        self.fc = nn.Linear(512, num_classes)

    def forward(self, x):
        x = self.cnn(x)  # (B, C, H, W)
        b, c, h, w = x.size()
        x = x.permute(0, 3, 1, 2).contiguous().view(b, w, c*h)  #
(B, W, C*H)
        x, _ = self.rnn(x)
        x = self.fc(x)
        return x  # (B, W, num_classes)

# ------------------------
# Initialize Model
# ------------------------
device = "cuda" if torch.cuda.is_available() else "cpu"
model = CRNN(img_h=32, num_classes=100).to(device)

print("CRNN model initialized on", device)
```

```
CRNN model initialized on cpu
```

```python
# ------------------------
# CTC Loss & Optimizer
# ------------------------
ctc_loss = nn.CTCLoss(blank=0)  # "0" reserved for CTC blank
optimizer = optim.Adam(model.parameters(), lr=1e-3)

# Example character set (dummy for now: a–z + space)
charset = [""] + list("abcdefghijklmnopqrstuvwxyz ")
char_to_idx = {c: i for i, c in enumerate(charset)}

def text_to_labels(text):
    return [char_to_idx[c] for c in text if c in char_to_idx]

# ------------------------
# Training Loop (1 epoch)
# ------------------------
for epoch in range(1):
    model.train()
    epoch_loss = 0
    for imgs, texts in train_loader:
        imgs = imgs.to(device)

        # Convert text to label sequences
        labels = [torch.tensor(text_to_labels(t),
dtype=torch.long) for t in texts]
        label_lengths = torch.tensor([len(l) for l in labels],
dtype=torch.long)
        labels = torch.cat(labels)  # flatten

        # Forward pass
        logits = model(imgs)  # (B, W, num_classes)
        log_probs = logits.log_softmax(2).permute(1, 0, 2)  # (W,
B, num_classes)

        input_lengths = torch.full(size=(logits.size(0),),
fill_value=logits.size(1), dtype=torch.long)

        # CTC Loss
        loss = ctc_loss(log_probs, labels, input_lengths,
label_lengths)

        optimizer.zero_grad()
        loss.backward()
```

```
            optimizer.step()

            epoch_loss += loss.item()

        print(f"Epoch {epoch+1}, Loss:
    {epoch_loss/len(train_loader):.4f}")
```

```
Epoch 1, Loss: 24.1904
```