# Import necessary libraries

import numpy as np

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense

# Load and preprocess the dataset

# Replace 'your\_dataset.txt' with the actual file containing handwritten text examples

with open('your\_dataset.txt', 'r', encoding='utf-8') as file:

text = file.read()

# Create a mapping between characters and numerical indices

char\_to\_index = {char: i for i, char in enumerate(sorted(set(text)))}

index\_to\_char = {i: char for i, char in enumerate(sorted(set(text)))}

# Convert the text to numerical representation

text\_as\_int = np.array([char\_to\_index[char] for char in text])

# Create input and target sequences

seq\_length = 100 # Length of input sequences

examples\_per\_epoch = len(text)//(seq\_length+1)

char\_dataset = tf.data.Dataset.from\_tensor\_slices(text\_as\_int)

sequences = char\_dataset.batch(seq\_length+1, drop\_remainder=True)

def split\_input\_target(chunk):

input\_text = chunk[:-1]

target\_text = chunk[1:]

return input\_text, target\_text

dataset = sequences.map(split\_input\_target)

# Batch and shuffle the data

BATCH\_SIZE = 64

BUFFER\_SIZE = 10000

dataset = dataset.shuffle(BUFFER\_SIZE).batch(BATCH\_SIZE, drop\_remainder=True)

# Build the model

vocab\_size = len(char\_to\_index)

embedding\_dim = 256

rnn\_units = 1024

model = Sequential([

tf.keras.layers.Embedding(vocab\_size, embedding\_dim, batch\_input\_shape=[BATCH\_SIZE, None]),

tf.keras.layers.LSTM(rnn\_units, return\_sequences=True, stateful=True, recurrent\_initializer='glorot\_uniform'),

tf.keras.layers.Dense(vocab\_size)

])

# Compile the model

model.compile(optimizer='adam', loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True))

# Train the model

EPOCHS = 10

history = model.fit(dataset, epochs=EPOCHS)

# Generate text using the trained model

def generate\_text(model, start\_string, temperature=1.0):

num\_generate = 1000

input\_eval = [char\_to\_index[s] for s in start\_string]

input\_eval = tf.expand\_dims(input\_eval, 0)

text\_generated = []

model.reset\_states()

for i in range(num\_generate):

predictions = model(input\_eval)

predictions = tf.squeeze(predictions, 0) / temperature

predicted\_id = tf.random.categorical(predictions, num\_samples=1)[-1, 0].numpy()

input\_eval = tf.expand\_dims([predicted\_id], 0)

text\_generated.append(index\_to\_char[predicted\_id])

return start\_string + ''.join(text\_generated)

# Generate text using the model with a given seed string

generated\_text = generate\_text(model, start\_string="Hello", temperature=0.8)

print(generated\_text)