GRU for Text Generation

Gated Recurrent Unit (GRU) is a type of Recurrent Neural Network designed to improve upon vanilla RNNs while being simpler than LSTM. GRU uses update and reset gates to control the flow of information and maintain long-term dependencies in sequences. In text generation, a GRU learns patterns from sequential text data and predicts the next character or word using both current input and past context, producing more coherent text than basic RNNs with lower computational complexity than LSTM.

```
# =========================
# GRU BASED TEXT GENERATION
# =========================

import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import GRU, Dense, Embedding

# -------------------------
# DATASET (SAME AS BEFORE)
# -------------------------
text = """
artificial intelligence is transforming modern society.
it is used in healthcare finance education and transportation.
machine learning allows systems to improve automatically with experience.
data plays a critical role in training intelligent systems.
large datasets help models learn complex patterns.
deep learning uses multi layer neural networks.
neural networks are inspired by biological neurons.
each neuron processes input and produces an output.
training a neural network requires optimization techniques.
gradient descent minimizes the loss function.

natural language processing helps computers understand human language.
text generation is a key task in nlp.
language models predict the next word or character.
recurrent neural networks handle sequential data.
lstm and gru models address long term dependency problems.
however rnn based models are slow for long sequences.

transformer models changed the field of nlp.
they rely on self attention mechanisms.
attention allows the model to focus on relevant context.
transformers process data in parallel.
this makes training faster and more efficient.
modern language models are based on transformers.

education is being improved using artificial intelligence.
intelligent tutoring systems personalize learning.
automated grading saves time for teachers.
online education platforms use recommendation systems.
technology enhances the quality of learning experiences.

ethical considerations are important in artificial intelligence.
fairness transparency and accountability must be ensured.
ai systems should be designed responsibly.
data privacy and security are major concerns.
researchers continue to improve ai safety.

text generation models can create stories poems and articles.
they are used in chatbots virtual assistants and content creation.
generated text should be meaningful and coherent.
evaluation of text generation is challenging.
human judgement is often required.

continuous learning is essential in the field of ai.
research and innovation drive technological progress.
students should build strong foundations in mathematics.
programming skills are important for ai engineers.
practical experimentation enhances understanding.
"""


text = text.lower().replace("\n", " ")

# -------------------------
# CHARACTER TOKENIZATION
# -------------------------
chars = sorted(list(set(text)))
char_to_index = {c: i for i, c in enumerate(chars)}
```

```
char_to_index = {c: i for i, c in enumerate(chars)}
index_to_char = {i: c for c, i in char_to_index.items()}

vocab_size = len(chars)
seq_length = 40

# ------------------------
# CREATE INPUT-OUTPUT SEQUENCES
# ------------------------
X, y = [], []

for i in range(len(text) - seq_length):
    X.append([char_to_index[c] for c in text[i:i+seq_length]])
    y.append(char_to_index[text[i+seq_length]])

X = np.array(X)
y = np.array(y)

# ------------------------
# BUILD GRU MODEL
# ------------------------
model = Sequential([
    Embedding(vocab_size, 64),
    GRU(128),
    Dense(vocab_size, activation='softmax')
])

model.compile(
    loss='sparse_categorical_crossentropy',
    optimizer='adam'
)

# ------------------------
# TRAIN MODEL
# ------------------------
model.fit(X, y, epochs=15, batch_size=64)

# ------------------------
# TEXT GENERATION FUNCTION
# ------------------------
def generate_text(seed_text, length=300):
    output = seed_text
    for _ in range(length):
        seq = np.array([[char_to_index[c] for c in output[-seq_length:]]])
        prediction = model.predict(seq, verbose=0)
        next_char = index_to_char[np.argmax(prediction)]
        output += next_char
    return output

# ------------------------
# GENERATE TEXT
# ------------------------
seed = "artificial intelligence "
generated_text = generate_text(seed)

print("Generated Text using GRU:\n")
print(generated_text)
```

```
Epoch 1/15
34/34 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 3.2153
Epoch 2/15
34/34 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 2.8862
Epoch 3/15
34/34 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 2.7275
Epoch 4/15
34/34 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 2.5949
Epoch 5/15
34/34 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 2.4868
Epoch 6/15
34/34 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 2.3954
Epoch 7/15
34/34 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 2.3598
Epoch 8/15
34/34 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 2.3043
Epoch 9/15
34/34 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 2.2097
Epoch 10/15
34/34 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 2.1788
Epoch 11/15
34/34 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 2.1352
Epoch 12/15
34/34 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 2.0808
Epoch 13/15
34/34 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1.9832
```

```
Epoch 14/15
34/34 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1.9646
Epoch 15/15
34/34 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1.8692
Generated Text using GRU:

artificial intelligence models and moders and moders and moders and moders and moders and moders and moders and moders and m
```

Limitations of GRU

- GRU models are still computationally expensive compared to simple RNNs.
- Performance depends heavily on the size and diversity of the training dataset.
- GRUs process sequences sequentially, limiting parallel computation.
- Generated text may still suffer from repetition when greedy decoding is used.
- Transformer-based models outperform GRUs on large-scale text generation tasks.