Bhadra Dilip Gada

TYIT

23BIT018

Handwriting Synthesis

Title of the project:

Handwriting Synthesis using RNNs and Reinforcement Learning

Problem Statement:

Handwriting synthesis is the task of generating written text in a style that appears hand-written by a human. Achieving truly natural-looking handwriting through a computer program is challenging due to the complex, sequential nature of writing and the subtle personal variations in style. Traditional pattern-generation techniques struggle to capture the temporal dependencies and stylistic nuances of human penmanship. This project addresses the problem of creating a system that can learn from real handwriting samples and produce new handwritten text that is legible and mimics human writing styles. By leveraging Recurrent Neural Networks (RNNs) for sequence modeling and Reinforcement Learning for feedback-driven refinement, the aim is to overcome limitations of earlier methods and generate handwriting that is both realistic and varied, closely approximating the way a person would write.

Why this topic is chosen:

This topic is chosen because it sits at the intersection of two influential areas in modern AI: deep learning (specifically sequence modeling with RNNs) and reinforcement learning. Handwriting synthesis is a compelling problem for a final-year IT project as it involves advanced concepts like neural networks learning temporal patterns and decision-making algorithms improving outcomes through trial and error. The problem has practical relevance – solutions could be applied in digital note-taking apps, automated form filling, personalized font generation, or aiding individuals who cannot physically write. Academically, the project provides an opportunity to explore state-of-the-art techniques and research (such as learning from sequence data and policy optimization) in a manageable scope. In essence, the challenge of teaching a machine to “write” like a human is both innovative and educational, allowing the student to deepen their knowledge of machine learning by applying it to a real-world inspired task.

Objective:

- **Develop a handwriting generator** that transforms any input text into convincingly human-like script.
- **Leverage existing handwriting data** so the model learns real stroke dynamics, spacing, and stylistic variation.
- **Design and train an RNN (LSTM) model** that outputs sequential stroke coordinates or equivalent pixel paths for drawing letters.
- **Apply reinforcement learning** to fine-tune the pretrained RNN, rewarding higher legibility and natural flow.
- **Maintain low text-reproduction error** so generated writing faithfully spells the input while retaining human variability.
- **Validate performance** by comparing synthetic handwriting to genuine samples through visual checks and OCR accuracy.
- **Deliver a working prototype** that demonstrates the added value of combining RNNs with reinforcement learning over a purely supervised approach.

Scope:

- Review key RNN + reinforcement-learning research on handwriting synthesis.
- Preprocess a public English-handwriting dataset (Latin alphabet only).
- Train an RNN model and fine-tune it with RL for added realism.
- Evaluate on unseen sentences to verify generalization.
- Build a simple demo that converts user text into a rendered handwriting image.
- Exclude non-Latin scripts, advanced UI, and large multi-style libraries to keep the scope manageable.

Methodology:

- **Data collection & preprocessing**
  - Use a public handwriting dataset (images or stroke sequences).
  - Normalize stroke-by-stroke data; if only images, apply vectorization/skeletonization to extract stroke paths.

- **Baseline model design**
    - Build an LSTM/GRU-based RNN with a Mixture Density Network output layer to predict next-stroke coordinates plus pen-up/pen-down state.
    - Train supervised to mimic real sequences given prior strokes and text context.
- **Reinforcement-learning refinement**
    - Treat the trained RNN as an agent that generates handwriting, then score each sample.
    - Reward criteria: OCR readability and stroke naturalness (statistical similarity to real writing).
    - Update via policy-gradient or actor–critic methods so the model iteratively improves beyond pure supervision.
- **Evaluation & iterative tuning**
    - Continuously validate on held-out samples, monitor quality, and adjust architecture, rewards, or hyperparameters if outputs become illegible or repetitive.
    - Explore learning-rate schedules, model size, and reward weighting to stabilize both supervised and RL phases.
  - **End goal**
      - Deliver a model that converts arbitrary user text into stroke sequences or rendered images that closely resemble authentic human handwriting.

Modules:

1. **Data Preprocessing Module**: Loads real handwritten datasets, cleans stroke data, normalizes coordinates, and prepares sequences for training.
2. **RNN-Based Stroke Generation Module**: Uses an LSTM or GRU model to learn handwriting patterns and generate sequential pen strokes from text input.
3. **Style Conditioning Module**: Adds style embeddings to control the appearance of handwriting, allowing the generator to imitate different writing styles.
4. **Reinforcement Learning Fine-Tuning Module**: Improves the baseline RNN by rewarding smoother, more natural, and more readable handwriting during generation.

5. **Rendering Module**: Converts predicted stroke sequences into human-readable handwriting images or SVG paths for visualization.
6. **Evaluation and Testing Module**: Assesses handwriting quality using OCR-based legibility tests, visual comparisons, and stroke-level metrics.
7. **User Interface and Demo Module**: Offers a simple interface for users to input text, select a style, and view or download synthesized handwriting.

Hardware & Software Requirements:

Hardware Requirements

- **Baseline Machine:** Standard PC with multi-core CPU.
- **GPU (Recommended):** NVIDIA CUDA-compatible card for faster RNN training.
- **Memory:** $\geq$ 8 GB RAM to load data and large model tensors.
- **Storage:** A few GB free space for dataset and checkpoints.
- **Optional:** Cloud or university GPU resources for heavier experiments.

Software Requirements

- **Language & Core Stack:** Python 3.x with TensorFlow (+ Keras) or PyTorch for LSTM/GRU implementation and automatic differentiation.
- **RL Add-ons:** Custom policy-gradient code; reference algorithms or snippets can be adapted from RL libraries (Gym not directly needed).
- **Data / Image Utils:** NumPy, Pandas for preprocessing; OpenCV or Pillow for rendering and analysis.
- **Dev Environment:** Jupyter Notebook or a Python IDE for iterative experiments; Git for version control and reproducibility.
- **Frontend:** A lightweight Nextjs Application can be handy for users to access the model

Testing Technology:

- **Unit checks:** unittest ensures preprocessing shapes, RNN outputs, and RL updates behave as expected.
- **Legibility:** Run generated text through OCR; high match-rate ⇒ readable handwriting.
- **Realism stats:** Compare stroke-level metrics (curvature, pen-lift rate) against real samples.
- **Quick Turing test:** Peers try to spot real vs. generated handwriting; note confusion rate.
- **Visual debug:** Plot stroke paths to catch gaps, jitters, or loops that don’t close; tweak accordingly.

What contribution would the project make?

- **Academic value:** Demonstrates how reinforcement learning can enhance sequence models beyond games, offering a clear case study for future students on reward-design trade-offs in creative generation.
- **Practical potential:** Lays groundwork for tools that mimic a person’s handwriting, enable personalized fonts, and augment OCR datasets with realistic synthetic samples.

- **Community impact:** Provides an open, documented prototype others can extend—pushing AI creativity in calligraphy much like neural art and music generators.

Conclusion:

- Combining an LSTM/GRU with RL can produce handwriting that is both readable and stylistically natural.
- Development tackles challenges in sequence generation, reward shaping and system integration, yielding a working demo that turns typed text into convincing script.
- Results will show strong progress yet still highlight a creativity gap between machine-made and truly human handwriting, pointing to clear avenues for future research.

Limitations:

- **Data-bound style:** Output diversity limited by the training set's range.
- **RL fragility:** Reward misalignment or hyper-parameter instability may produce odd-looking strokes.
- **Compute & time:** Full hyper-parameter tuning is constrained by GPU budget and project deadlines.
- **Script scope:** Only English; adaptation to non-Latin scripts left for future work.
- **Subjective evaluation:** Definitively proving "human-likeness" remains partly qualitative.

References:

1. Graves, A. (2013). *Generating Sequences with Recurrent Neural Networks.*
2. Sutton, R. S. & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.).
3. Williams, R. J. (1992). "Simple Statistical Gradient-Following Algorithms for Connectionist RL." *Machine Learning*, 8.
4. Ha, D. & Eck, D. (2018). *A Neural Representation of Sketch Drawings.*