Handwriting Autocomplete Pipeline

A technical overview of the four-stage machine learning pipeline required to build a sophisticated, style-preserving handwriting recognition and autocomplete system.

PIPELINE OVERVIEW

Four Stages

1

1. Input Capture & Preprocessing

Normalize and segment raw pen strokes.

2

2. Handwriting Recognition

Convert prepared strokes into digital text (Strokes → Text).

3

3. Next-Word Prediction

Use Language Modeling to suggest the next words (Text → Text).

4

4. Handwriting Style Synthesis

Render the predicted text in the user's specific handwriting style (Text → Strokes/Image).

1. Input Capture and Preprocessing

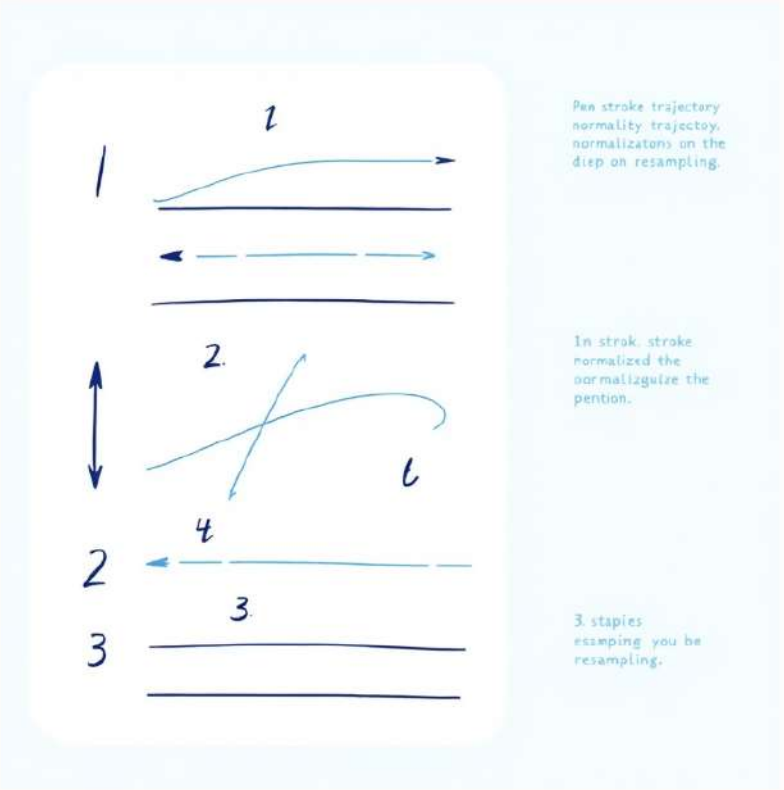
Transforming Raw Data into Normalized Features

The raw input from devices like the iPad is an "online" stream of pen strokes, represented by sequences of (x, y, t) points.

Before feeding this data into the recognition model, it must be normalized and resampled. This ensures the model is invariant to variations in baseline, device resolution, and writing speed.

- Normalize coordinates: Scale (x, y) so the first point is at x=0 and y spans [0,1].
- Resample: Adjust points to fixed time intervals.
- Feature Generation: The network processes the deltas between consecutive points: (\Delta x_i, \Delta y_i, \Delta t_i, p_i, n_i), where p_i and n_i are pen-up/down flags.

Segmentation is a key step, splitting the continuous stroke stream into discrete lines or words for processing.



2. Handwriting Recognition (Strokes → Text)

The Dominant Architecture

The state-of-the-art approach for online Handwriting Recognition (HTR) is the End-to-End Recurrent Network.

- A stack of **Bidirectional LSTMs (BiLSTM)** processes the sequential stroke features.
- A Softmax layer outputs character distributions at each time step.

Training and Decoding

The model is trained using **Connectionist Temporal Classification (CTC) loss** to align the time-step outputs with the ground-truth text, even without a precise character-to-stroke alignment.

Inference utilizes CTC beam search decoding, which typically incorporates a character or word-level language model to improve accuracy and fluency.

All Options For HTR

While BiLSTM-CTC remains the most practical and proven architecture, research is moving toward complex fusion models to handle highly variant styles.

RNN (BiLSTM-CTC)

Processes the raw vector trajectory data. Fast, simple, and effective for "online" data. Currently yields SOTA on benchmarks like IAM-OnDB.

Transformer Fusion (HATCharClassifier)

Encodes both the pen-trace (stroke) and its rendered image (offline) into a shared transformer space. Improves robustness but increases complexity.

CNN-RNN Hybrid (Offline Only)

For pure image snapshots (offline HTR), this approach processes the raster image before the sequential text decoding. Not optimal for the online tablet use case.

3. Next-Word Prediction (Language Modeling)

The role of the Language Model (LM) is to provide fluent and contextually relevant predictions for the user's next word, based on the recognized text so far.

Legacy RNN LMs

Stacked LSTM or GRU networks trained on large text corpora using cross-entropy loss. Simple to implement from scratch and still competitive for simpler tasks.

Pretrained Transformers

State-of-the-art results are achieved using decoder-only Transformer models (e.g., GPT-2/3 styles). These models have learned rich linguistic context, offering highly fluent and accurate suggestions.

Implementation Focus

Utilize pretrained models from Hugging Face for accelerated development. Fine-tuning on domain-specific text (e.g., technical notes) can significantly boost relevance.

4. Handwriting Style Synthesis (Text → Handwriting)

The system must render the predicted text in the user's specific handwriting style. This demands fidelity to the user's unique pen dynamics and flair.

Stroke-Based Synthesis (Graves, 2013)

An autoregressive LSTM is trained to generate the pen stroke sequence conditioned on the target text. It uses a Mixture-of-Gaussians distribution to output the next pen point and pen-up/down probability.

Personalized Style Priming

Crucially, the generative LSTM can be "primed" by feeding it a small sample of the user's real strokes. This captures and mimics the individual writer's style for all subsequent generated text.

Alternative: Image-Based Synthesis

Style Transfer vs. Generation

Recent research views handwriting generation as an image synthesis problem, leveraging advanced deep learning architectures for style transfer.

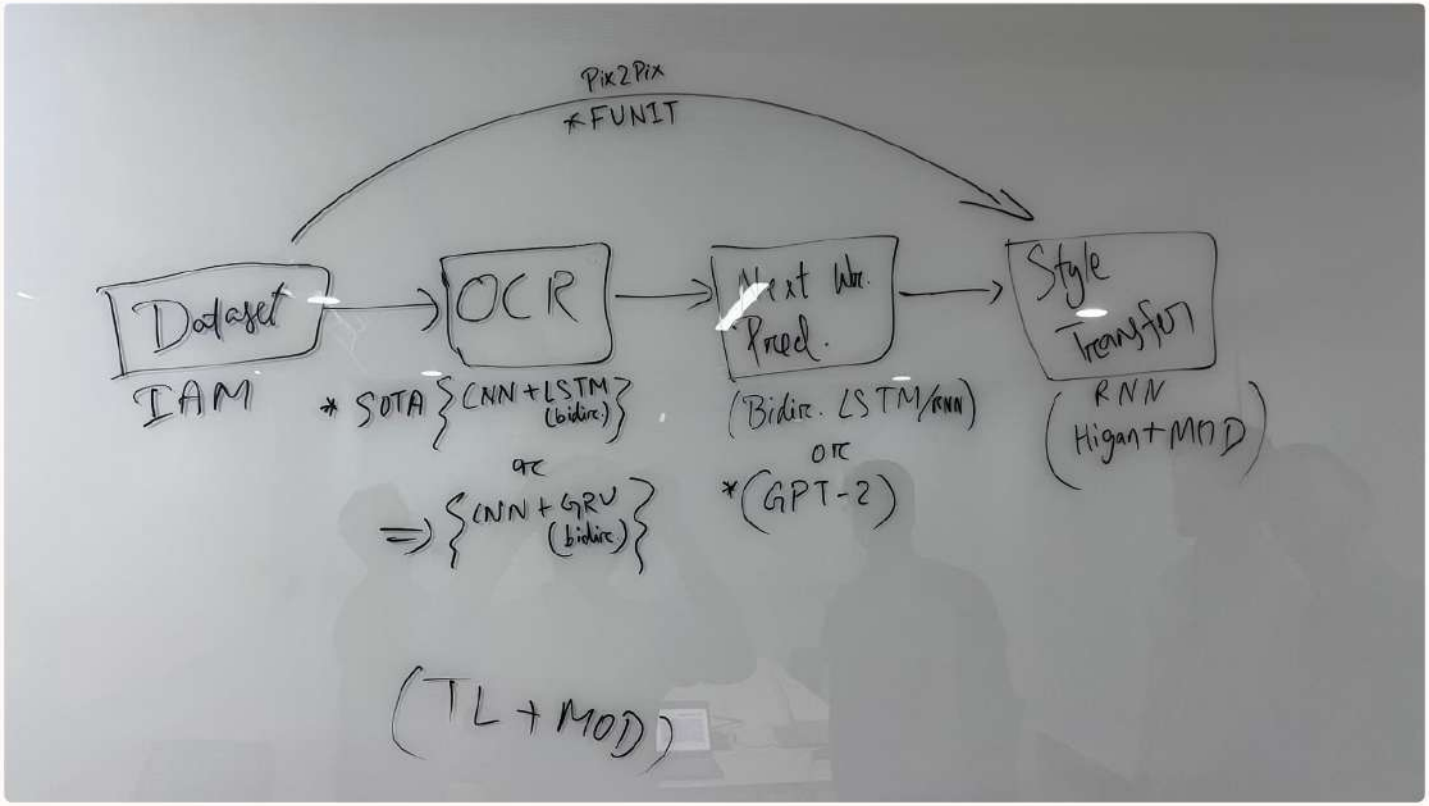
- **VAE/Transformer Models (e.g., Emuru):** Encodes a reference handwriting image into a style embedding, then decodes the target text as a new, styled image. Often zero-shot capable.

While these methods capture texture and thickness, the stroke-based approach remains conceptually cleaner for a pure online system.

Work Done Till Now

- Research on all layers and finding datasets.
- Implementation of Higan+ locally and metrics extraction on Iam dataset.
- OCR and LSTM from scratch as well as sota implementation.
- Implementation of Higan+ from scratch and connecting pipeline.

Brainstorming



Key Takeaways

Recognition

Use a proven BiLSTM+CTC stack for converting strokes to legible text.

Prediction

Deploy a neural Language Model (Transformer or RNN) for contextual next-word suggestions.

Synthesis

Select the stroke-based LSTM generator (Graves, 2013) and implement style "priming" for a highly personalized and editable output.