RAJALAKSHMI ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

Department of Computer Science and Engineering

FONT SET BLENDER AND GENERATOR

Mrs. Divya M. M.E.,
Project Mentor

J Jessielyn Jenisha
220701901

Problem Statement and Motivation

- Designing fonts for multiple languages is time-consuming and requires manual effort from expert designers.
- Existing tools lack the capability to easily blend styles or generate customized, multilingual fonts.
- Users have no simple way to create personalized fonts, such as mixing their handwriting with existing styles.
- There is a need for an automated, user-friendly system to generate and customize fonts across scripts.

Problem Statement and Motivation

- Rising demand for creative, personalized, and multilingual typography in digital platforms.
- Designers and users seek efficient tools to explore new font styles without technical barriers.
- Advances in deep learning offer potential to automate and simplify the font creation process.
- This project aims to empower users to blend fonts, create unique styles, and extend designs across languages.

Existing System

- Font creation today relies heavily on manual design using tools like Adobe Illustrator, FontForge.
- Multilingual fonts are designed separately for each script, leading to inconsistent styles.
- Some AI models exist (e.g., MC-GAN [4], Semantic Typeface Transfer [6]):
  - Focused on single-script transfer (like Latin only).
  - Require large datasets and are not user-driven.
- No existing tool allows easy font blending and personalization using user input.

Objectives

- To develop an AI-powered system for font generation and blending.
- Enable users to mix multiple font styles to create unique designs.
- Provide an option to include user handwriting in the generated font.
- Make the system user-friendly and efficient, requiring no design expertise.

Abstract

- This project presents an AI-driven Font Generation and Blending System that enables users to create unique, cross-lingual fonts.
- Using a **deep autoencoder model**, the system learns to represent and reconstruct font glyphs.
- Users can **blend multiple font styles** by **adjusting mixing ratios** and even incorporate their own handwriting samples.
- The generated fonts can be previewed and downloaded, empowering both designers and non-designers to create customized typefaces effortlessly.

Proposed System

- Uses a Convolutional Autoencoder trained on diverse fonts to learn compressed representations.
- Font Blending Module:
  - Takes user-selected fonts and blends their encoded features based on chosen ratios.
  - Decodes the blended encoding to generate new glyphs.
- Supports User Handwriting Upload:
  - Users can upload handwritten samples for personalized font generation.
- Enables Cross-Lingual Style Transfer:
  - The blended style can be applied to other scripts (e.g., Tamil, Hindi, Japanese).
- Outputs a preview and downloadable font pack for immediate use.

System Architecture

- Font Input: User selects multiple fonts (TTF files) or uploads handwriting samples.
- Preprocessing Module:
  - Extracts and renders glyph images from fonts.
  - Normalizes and resizes them (e.g., 64x64 pixels).
- Encoder (Feature Extraction): A convolutional encoder compresses glyph images into compact style embeddings.

System Architecture

- Blending Module:
  - Takes multiple font embeddings and blends them using user-defined ratios.
  - Supports dynamic style mixing (e.g., 40% Font A + 60% Font B).
- Decoder (Glyph Generation):The blended encoding is decoded to reconstruct the glyph images with new styles.
- Output: Users can preview generated fonts and download them as font packs.

List of Modules

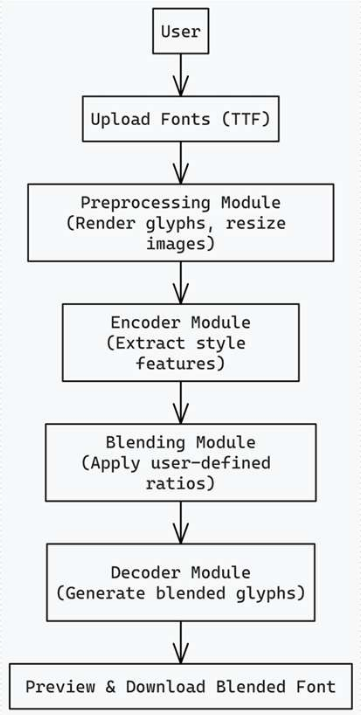
- Font Preprocessing Module
- Encoder
- Blending Module
- Decoder
- Preview and Download

Functional Description for each modules with DFD and Activity Diagram

- Font Preprocessing Module - Renders glyphs from TTF/OTF files and standardizes image sizes.
- Encoder - Extracts compact style features from font glyphs (CNN-based).
- Blending Module - Mixes multiple style features based on user-specified ratios.
- Decoder - Reconstructs new glyph images from blended features.
- Preview & Download – Displays generated font samples and provides a download option.

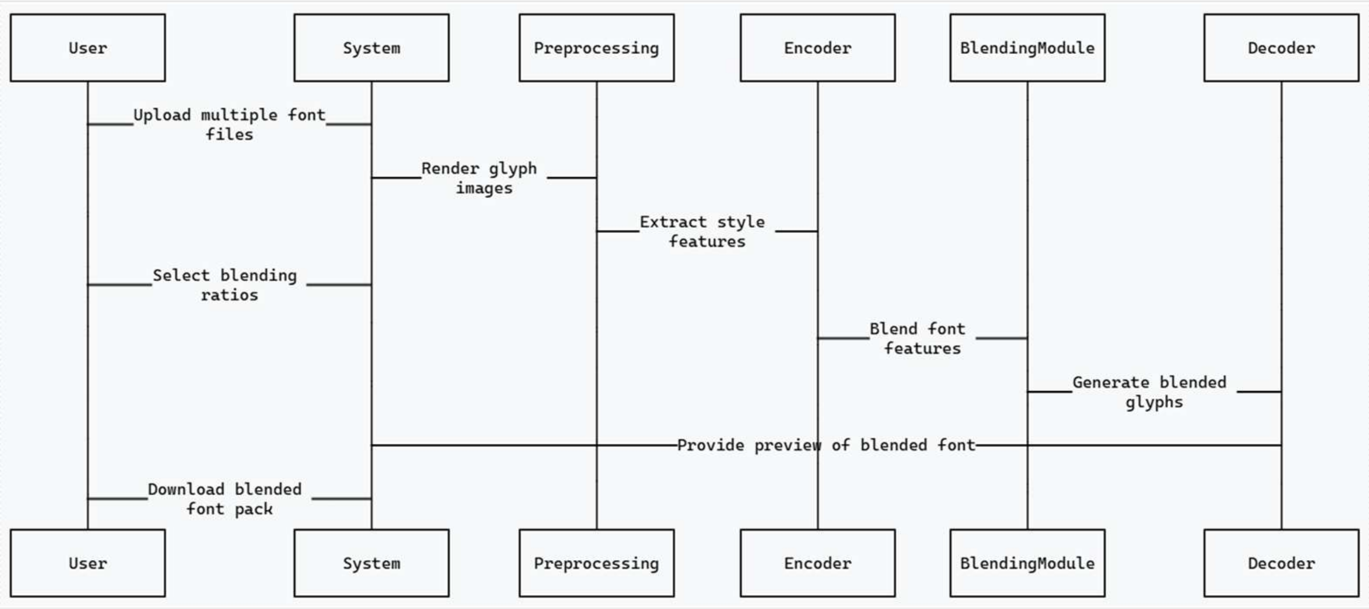
Functional Description for each modules with DFD and Activity Diagram

- Data Flow Diagram



Functional Description for each modules with DFD and Activity Diagram

- Activity Diagram



Implementation & Results of Module

- Tech Stack : Python, TensorFlow/Keras, NumPy, PIL, Matplotlib
- Model Used : Convolutional Autoencoder trained on glyph images
- Blending Mechanism : Encoded glyphs from multiple fonts are combined using weighted averages (ratios set by user)
- Frontend : Uses Matplotlib for preview; outputs images and downloadable blended font pack
- Modules :
  - Glyph Renderer (TTF to image)
  - Encoder & Decoder (Autoencoder-based)
  - Font Blender (Weighted encoding blending)
  - Output Generator (Blended glyph previews)

Implementation & Results of Module

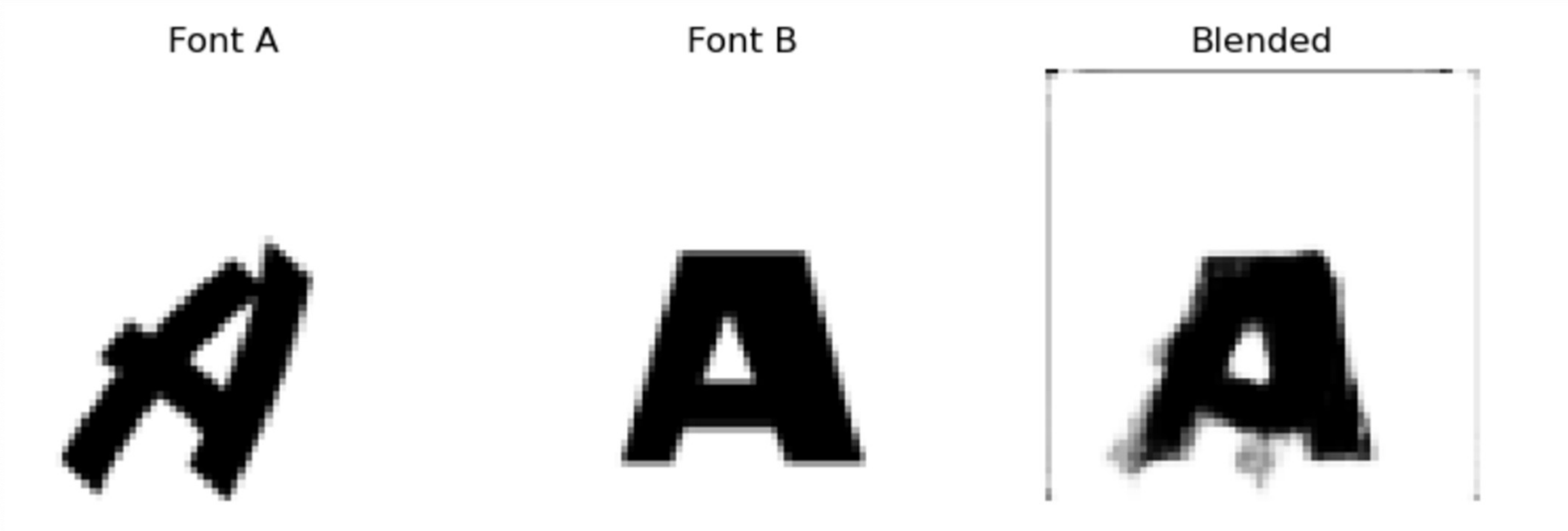
- Blending Logic

```
# Blend encodings
blended_encoding = np.zeros_like(encoded_fonts[0])
for encoded, ratio in zip(encoded_fonts, ratios):
    blended_encoding += encoded * ratio

# Decode to generate blended glyphs
reconstructed_fonts = decoder.predict(blended_encoding)
```

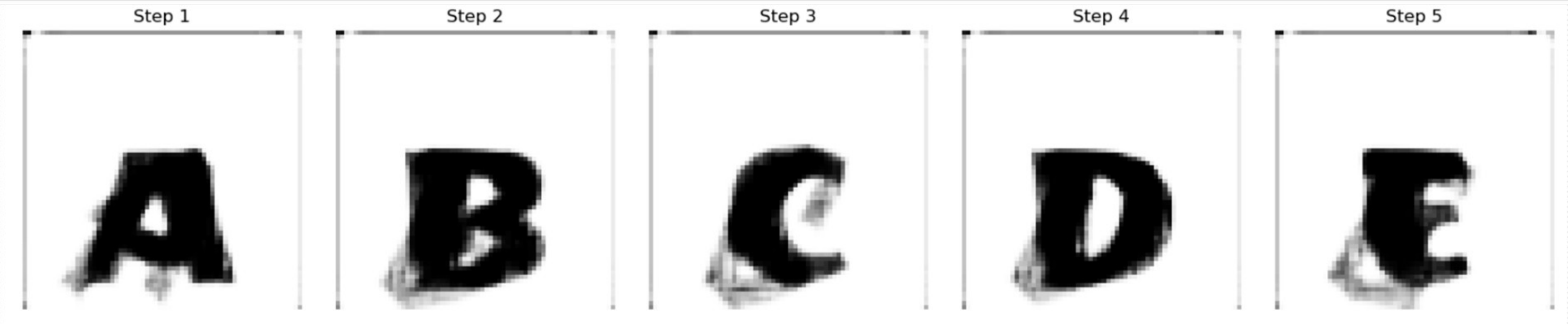
Implementation & Results of Module

- Generated Blended Fonts



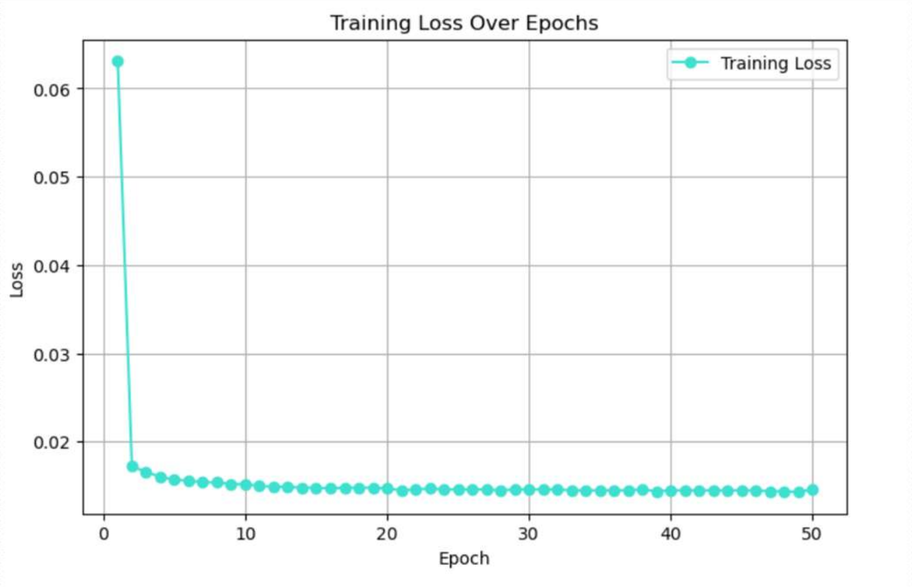
Implementation & Results of Module

- Interpolation Samples



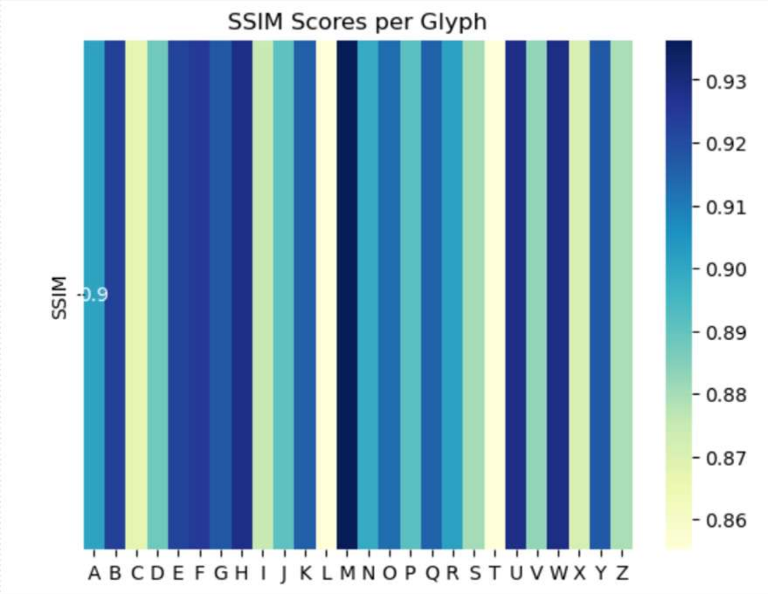
Implementation & Results of Module

- Quality Metrics
  - Reconstruction loss during training



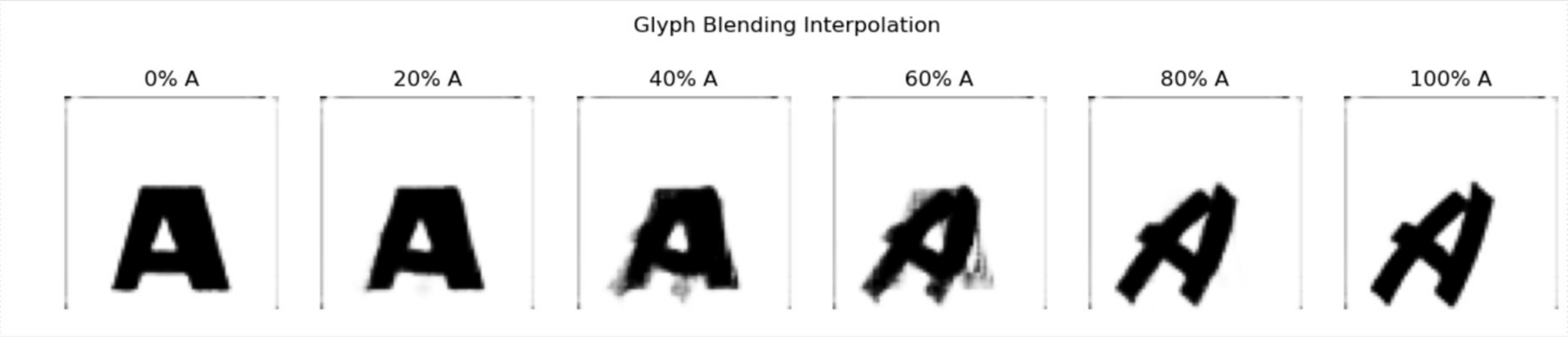
Implementation & Results of Module

- Quality Metrics
  - SSIM scores per glyph



Implementation & Results of Module

- Ratio Control



Conclusion & Future Work

- Successfully implemented a font blending system using deep learning-based convolutional autoencoders.
- Allows users to upload multiple fonts and blend them using adjustable weight ratios.
- Generated new blended fonts that inherit features from the selected source fonts.
- Achieved good visual quality and structural similarity using SSIM scores for evaluation.
- Demonstrated an automated, flexible method for creative font generation without manual design work.

Conclusion & Future Work

- Multilingual Support - Extend the system to handle scripts like Devanagari, Arabic and Chinese for global font blending.
- User-Controlled Attribute Blending - Let users adjust specific features like thickness, roundness, or serif presence during blending.
- On-Device Deployment - Optimize for mobile and desktop applications to enable offline font generation.
- Interactive Preview Interface - Build a real-time system where users can adjust blending and preview results instantly.
- Dataset Expansion - Add more fonts and scripts to improve system generalization and versatility.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, et al., "Generative Adversarial Nets," Advances in Neural Information Processing Systems, vol. 27, pp. 2672–2680, 2014.

[2] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution," European Conference on Computer Vision, pp. 694–711, 2016.

[3] T. Karras, S. Laine, and T. Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks," IEEE Conference on Computer Vision and Pattern Recognition, pp. 4401–4410, 2019.

[4] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," arXiv preprint arXiv:1511.06434, 2015.

[5] H. Lian and C. Meng, "Neural Font Style Transfer with Content Preservation," IEEE Transactions on Image Processing, vol. 30, pp. 2706–2718, 2021.

References

[6] Y. Azadi, M. Fisher, V. G. Kim, et al., "Multi-Content GAN for Few-Shot Font Style Transfer," IEEE Conference on Computer Vision and Pattern Recognition, pp. 7564–7573, 2018.

[7] Y. Kang, K. R. M. Gupta, and J. Lee, "Font2Font: A Dual Learning Framework for Font Style Transfer," Pattern Recognition, vol. 125, 108533, 2022.

[8] Y. Li, L. Liang, H. Qin, et al., "Few-Shot Font Generation with Dual Memory," ACM Transactions on Graphics, vol. 40, no. 6, pp. 1–13, 2021.

[9] Y. Yang, J. Liu, and X. Sun, "Controllable Font Generation via Complex Decomposition," IEEE Access, vol. 8, pp. 155930–155940, 2020.

[10] Z. Chen, C. Chen, J. Xu, and M. Sun, "AI-Aided Typeface Design: A Review," IEEE Transactions on Visualization and Computer Graphics, vol. 29, no. 1, pp. 458–475, 2023.

Paper Publication Status

- Not Published

Thank You