HIGH LEVEL DESIGN DOCUMENT

Style-Consistent Kannada Font Generation

UE17CS490A - Capstone Project Phase - 1

Submitted by:

Saahil B Jain	**PES1201700241**
Jeevana R Hegde	**PES1201700633**

Under the guidance of

Prof. K.S Srinivas
Associate Professor
PES University

August - December 2020

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru - 560 085, Karnataka, India

TABLE OF CONTENTS

1. Introduction

Indian languages currently have a limited set of fonts available and Generation of new fonts is a manual process where designers use tools to draw letters in different ways. No current work exist in automating the generation of fonts for Indian languages like Kannada. This brings the need for our project which aims to automate the generation of new fonts for the Kannada scripts while simultaneously ensuring generation of maatras.

Currently fonts are generated manually by designers. These designers use software drawing tools to draw and create each letter of the alphabet set. Not only is this process hard because the designers have to ensure style consistency among every letter, but it is time consuming as well.

Recent research have attempted to generate fonts for the English language, however they require a large amount of pre-existing fonts to start generating new fonts. This amount of data doesn't exist for Indian languages like Kannada, which have only about 4-5 available fonts.

Hence a different approach is required to generate fonts for Indian languages so that fonts can be generated without the need for pre-existing fonts. This is where our solution comes into picture where we can generate new style consistent fonts without the need for pre-existing fonts.

Our project could finally be used by a website which would offer users multiple auto-generated fonts to choose from for the Kannada script. Each option would be available separately where users can browse through all options and choose and download the ones they like most.

2. Current System

Currently fonts are generated by manually by designers. These designers use software drawing tools to draw and create each letter of the alphabet set. Not only is this process hard because the designers must ensure style consistency among every letter, but it is time consuming as well.

Recent research have attempted to generate fonts for the English language, however they require around 6500 pre-existing fonts to start generating new fonts. This amount of data doesn't exist for Indian languages like Kannada, which have only about 4-5 available fonts.

3. Design Details

3.1. Novelty

The novelty of our project lies not only in the fact that automation of font generation is new for Indian languages like Kannada, but also the use of technologies like Neural Cellular Automata for generation problems. Neural Cellular Automata has never been used for generation problems, Our project is the first use case of Neural Cellular Automata as a generative model.

3.2. Innovativeness

Neural Cellular Automata has often been used for recreational problems, but never for generation problems. Our project is the first use case of Neural Cellular Automata as a generative model. Our project is also unique because it is able to generate new fonts without any data requirements.

3.3. Architecture Choices

To automate the generation of fonts, we had three options available. These are as follows :

A. Neural Networks:

The first step is to create a "font vector" that is a vector in latent space that "defines" a certain font. That way we embed all fonts in a space where similar fonts have similar vectors. Then a neural network is trained to generate the fonts from these "font vectors". Any change in the "font vector" would then result in new fonts.

Pros:

a). Simple and easy to build architecture.

b). Enough data is present. Hence, slight modifications lead to the generation of significantly large number of fonts with high legibility.

c). Style consistency is reasonable.

Cons:

a). There is an extremely high data dependency.

b). Diversity is quite low.

B. GANs:

We can train GANs which can be trained over a large dataset of fonts. Once trained the GAN could easily generate better fonts than the neural network.

Pros:

a). Better legibility compared to deep-fonts.

b). Style consistency is the same.

c). More diversity in available in the fonts compared to neural network.

Cons:

a). High data dependency is present, which is unavailable for Indian languages like Kannada.

C. Neural Cellular Automata:
We first train a simple NCA model for each letter and separate model for each letter with different maatras as well. Once all models are created we only need to change the filter used to perceive the neighbouring cells. With slightly different views of the neighbours, the model generates completely different fonts.
Pros:
a). Data requirement is very low. Just a single font is required to generate significantly different fonts.
b). Style consistency is accurate.
c). Models are very small.
Cons:
a). Every letter requires a different model.
b). Different model required for every letter with each maatra.

Having read through all our options we chose the Neural Cellular Automata as this is the only method which can generate fonts without the requirement for additions data. Which is an essential requirement when we try to generate fonts for Indian languages like Kannada which have a very small amount of available fonts.

3.4. Performance

To generate fonts we need to create separate models for each letter and separate model for each letter with different maatras as well. This makes generation of base models challenging as well as time consuming. Processing power required is considerably low. But increase in image size increases model size exponentially. Maintainability

3.5. Constraints

To generate fonts we need to create separate models for each letter and separate model for each letter with different maatras as well. This makes generation of base models challenging as well as time consuming.
Processing power required is considerably low. But increase in image size increases model size exponentially.

3.6. Dependencies

1 set of alphabets with all maatras.

3.7. Assumptions

Compute resource powerful enough to generate the base models in reasonable amount of time.

3.8. Interoperability

Once the fonts are generated as images and converted to fonts usable by the system, they can be downloaded on the system on which the application requires them. Once downloaded they can be used on literally any system as regular fonts are used.

3.9. Portability

Our method can easily be used to generate new fonts for other languages like Telugu or Hindi.
It can also be used for many generative tasks as well.
For example: this method can also be used for data augmentation purposes for images which would be very helpful in case of low clarity images.

3.10. Legacy to modernization

Currently fonts are generated manually by designers. These designers use software drawing tools to draw and create each letter of the alphabet set. Not only is this process hard because the designers have to ensure style consistency among every letter, but it is time consuming as well.
Recent research have attempted to generate fonts for the English language, however they require a large amount of pre-existing fonts to start generating new fonts. This amount of data doesn't exist for Indian languages like Kannada, which have only about 4-5 availablefonts.
Hence a different approach is required to generate fonts for Indian languages so that fonts can be generated without the need for pre-existing fonts. This is where our solution comes into picture where we can generate new style consistent fonts without the need for pre-existing fonts.

3.11. Reusability

Not only can the given method be used to generate new fonts for other languages like Telugu, but can also be used for many generative tasks as well.
This method can also be used for data augmentation purposes for images which would be very helpful in case of low clarity images.

3.12. Application compatibility

Once the fonts are generated as images they could easily be converted as fonts and used along with literally any application that requires these fonts.
They would however need to be converted to fonts usable by the system and downloaded on the system on which the application requires them.

Appendix A: Definitions, Acronyms and Abbreviations

1. GAN - Generative Adversarial Networks
2. NCA - Neural Cellular Automata

Appendix B: References

Article in Online Encyclopedia

[1] Alexander Mordvintsev, Ettore Randazzo, Eyvind Niklasson, Michael Levin, "Growing, Neural Cellular Automata", Feb. 11, 2020. [Online], Available: https://distill.pub/2020/growing-ca/ [Accessed Sep. 02, 2020]

Journal Article Abstract

[2] Koen Janssens, "Network Cellular Automata—A Development for the Future?", *Computational Materials Engineering*, 2007. [Online],Available: https://www.sciencedirect.com/topics/computerscience/cellularautomata [Accessed Sep. 03, 2020]

Research paper

[3] Hideaki Hayashia, Kohtaro Abea, Seiichi Uchidaa, GlyphGAN, *Department of Advanced Information Technology,* May 30, 2019. Available: https://arxiv.org/pdf/1905.12502.pdf [Accessed Sep. 04, 2020]

Research paper

[4] Eric Bernhardsson, "Deep-fonts", Jan. 21,2016. Available: https://erikbern.com/2016/01/21/analyzing-50k-fonts-using-deep-neural networks.html [Accessed Oct. 04, 2020]

Appendix C: Record of Change History

#	Date	Document Version No.	Change Description	Reason for Change

Appendix D: Traceability Matrix

Project Requirement Specification Reference Section No. and Name.	DESIGN / HLD Reference Section No. and Name.