A Deep Factorization of Style and Structure in Fonts

Nikita Srivatsan , Jonathan T. Barron , Dan Klein, Taylor Berg-Kirkpatrick

Language Technologies Institute, Carnegie Mellon University
Google Research
Computer Science Division, University of California, Berkeley
Computer Science and Engineering, University of California, San Diego

EMNLP 2019

DATE: 2021/09/01

SPEAKER: YANG, YI-TING

Outline

- Introduction
- Font Reconstruction
- Model
- Learning and Inference
- Experiments and Result
- Conclusion

Introduction

- **Probabilistic latent variable model**
  - Disentangling the stylistic features of fonts from the underlying structure of each character.
  - The style of each font --> A vector-valued latent variable
    - Each style latent variable is shared by all characters within a font.
  - The structure of each character --> A learned embedding
    - Each character embeddings are shared by characters of the same type across all fonts.

Figure 1: Example fonts from the Capitals64 dataset. The task of font reconstruction involves generating missing glyphs from partially observed novel fonts.

Introduction

- **Probabilistic latent variable model**
  - We parameterize the distribution that combines style and structure in order to generate glyph images as a **transpose convolutional** neural decoder**.**
  - Further, the decoder is fed character embeddings early on in the process, while the font latent variables directly parameterize the convolution filters.

Font Reconstruction

- We may wish to have models that can infer these missing glyphs, a task referred to as font reconstruction.
  - We can view a collection of fonts as a matrix, $\boldsymbol{X}$, where each column corresponds to a particular character type, and each row corresponds to a specific font. Each entry in the matrix, x_{ij} , is an image of the glyph for character i in the style of a font j, which we observe as a 64×64 grayscale image.

Figure 1: Example fonts from the Capitals64 dataset. The task of font reconstruction involves generating missing glyphs from partially observed novel fonts.

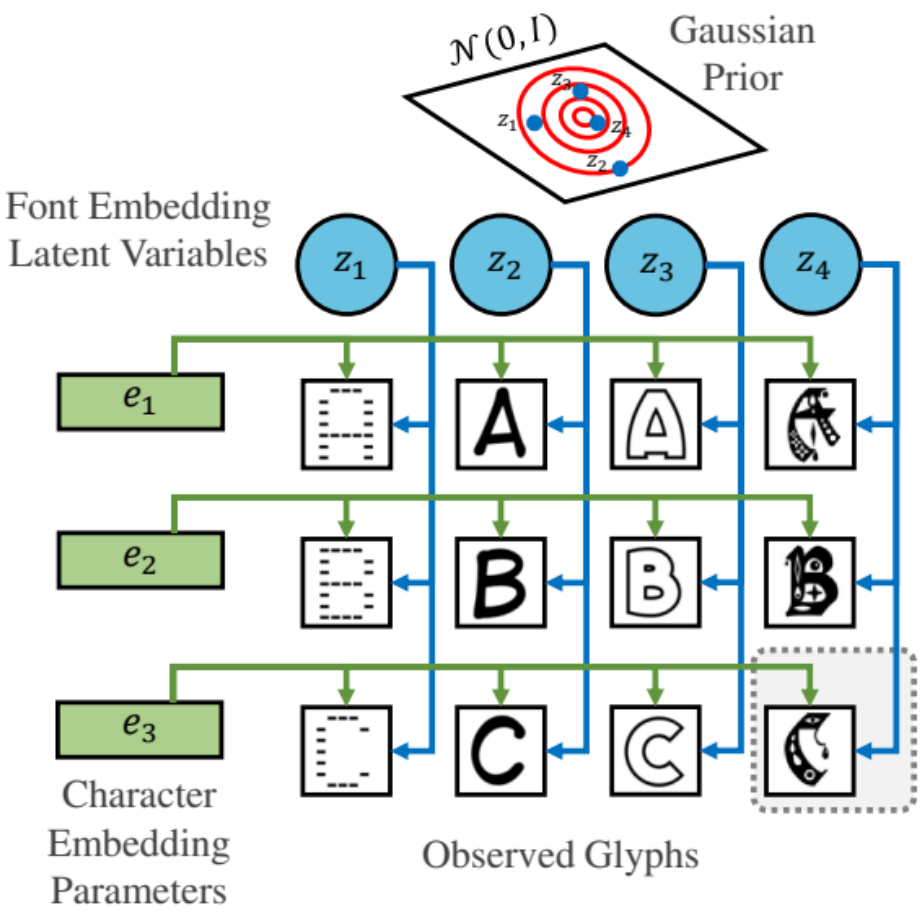
Font Reconstruction

- Past work on font reconstruction has focused on discriminative techniques.
  - Use an adversarial network to directly predict held out glyphs conditioned on observed glyphs.

- We propose a generative approach using a deep latent variable model.
  - Under our approach fonts are generated based on an unobserved style embedding, which we can perform inference over given any number of observations.

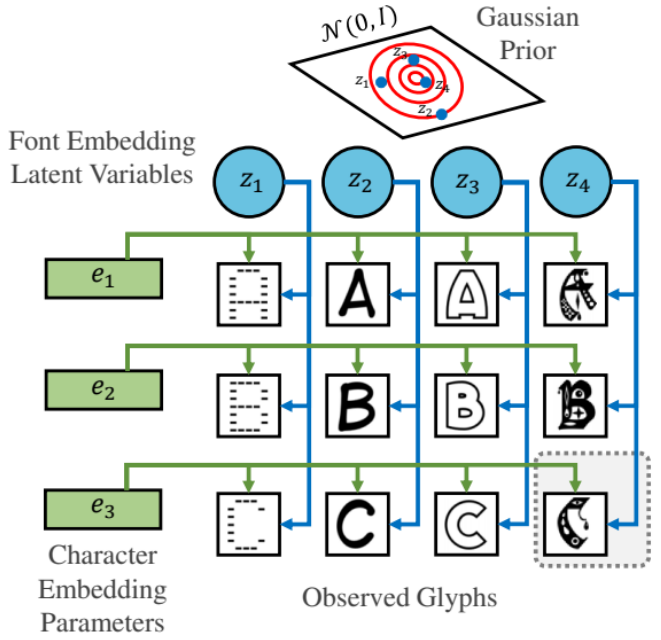
Model

- Our model hypothesizes a separation of character-specific structural attributes and font-specific stylistic attributes into two different representations.
  - Character-specific structural attributes --> An embedding vector
    - All characters are observed in at least one font.
  - Font-specific stylistic attributes --> A vector-valued latent variable
    - Only a subset of fonts is observed during training and our model will be expected to generalize to reconstructing unseen fonts at test time.



Model

- Each glyph image, x_{ij} , is generated independently, conditioned on the corresponding font embedding variable, $z_j \in \mathbb{R}^k$, which is sampled from a fixed multivariate Gaussian prior, $p(z_j) = \mathcal{N}(0,\ I_k)$, and a character specific parameter vector, $e_i \in \mathbb{R}^k$, which we refer to as a character embedding.
  - Glyphs of the same character type share a character embedding.
  - Glyphs of the same font share a font variable.



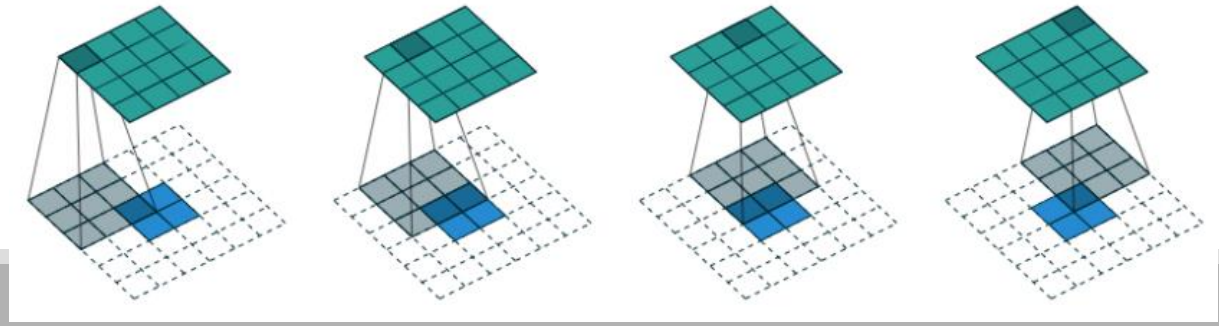
Model

- We denote this decoder distribution as $p(x_{ij}|z_j; e_i, \emptyset)$, and let $\emptyset$ represent parameters, shared by all glyphs, that govern how font variables and character embeddings combine to produce glyph images.
  - The character embedding parameters, e_i , which feed into the decoder, are only shared by glyphs of the same character type.
  - The font variables, z_j , are unobserved during training and will be inferred at test time.
- The joint probability under our model is given by:

$$p(X, Z; E, \phi) = \prod_{i,j} p(x_{ij}|z_j; e_i, \phi)p(z_j)$$

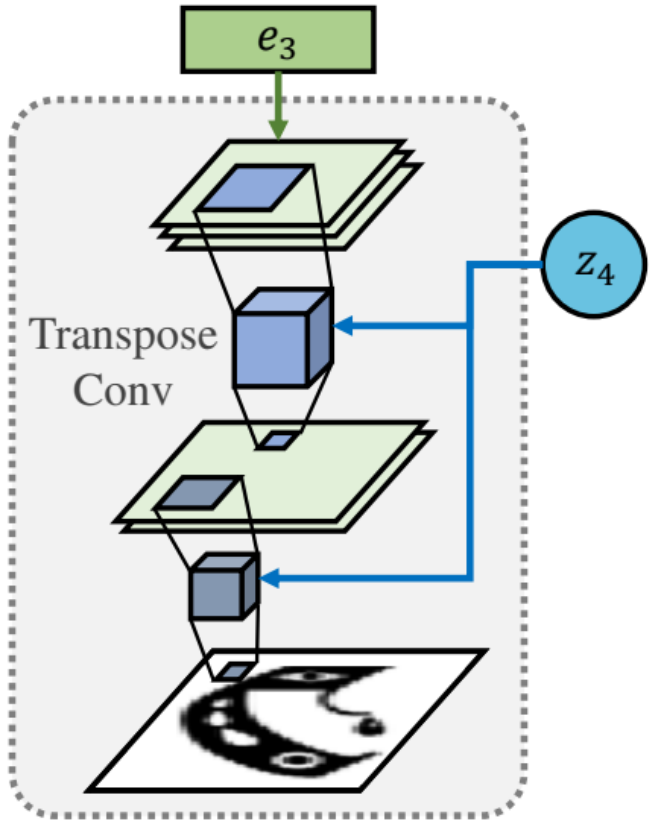
Model

- **Decoder Architecture**
  - We can think of the character type as specifying the overall shape of the image, and the font style as influencing the finer details.
  - We hypothesize that a glyph can be modeled in terms of a low-resolution character representation to which a complex operator specific to that font has been applied.
- **Transpose convolution**
  - A transpose convolution is a convolution performed on an undecimated input (i.e. with zeros inserted in between pixels in alternating rows and columns), resulting in an upscaled output.



Model

- We form our decoder as follows:
  - The character embedding is projected to a low-resolution matrix with a large number of channels.
  - We apply several transpose convolutional layers which increase the resolution, and reduce the number of channels.
    - Critically, the convolutional filter at each step is not a learned parameter of the model, but rather the output of a small multilayer perceptron whose input is the font latent variable Z.
  - Between these transpose convolutions, we insert vanilla convolutional layers to fine-tune following the increase in resolution.

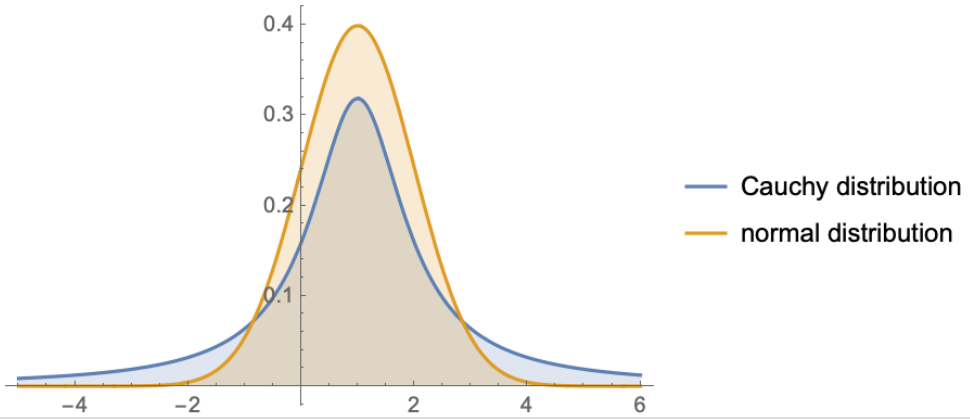


Decoder Architecture

Model

[1] David J. Field. 1987. Relations between the statistics of natural images and the response properties of cortical cells. JOSA A.
[2] Tamara Melmer, Seyed Ali Amirshahi, Michael Koch, Joachim Denzler, and Christoph Redies. 2013. From regular text to artistic writing and artworks: Fourier statistics of images with low and high aesthetic appeal. Frontiers in Human Neuroscience.
[3] Jonathan T. Barron. 2019. A general and adaptive robust loss function. CVPR.

- **Projected Loss**
  - Our reconstruction loss we use a heavy-tailed (leptokurtic) distribution placed on a transformed representation of the image, similar to the approach of Barron [3].
    - Natural images are not well-described in terms of statistically independent pixels, but are instead better modeled in terms of edges [1].
    - Images of text have similar statistical distributions as natural images [2].
  - Modeling the statistics of font glyphs in this fashion results in sharper samples, while modeling independent pixels with a Gaussian distribution results in blurry, oversmoothed results.



[4] Nasir Ahmed, T Natarajan, and Kamisetty R Rao. 1974. Discrete cosine transform. IEEE Transactions on Computers.

Model

- **2-Dimensional Discrete Cosine Transform [4]**
  - We transform both the observed glyph image and the corresponding grid or parameters produced by our decoder before computing the observation's likelihood.

[3] Jonathan T. Barron. 2019. A general and adaptive robust loss function. CVPR.

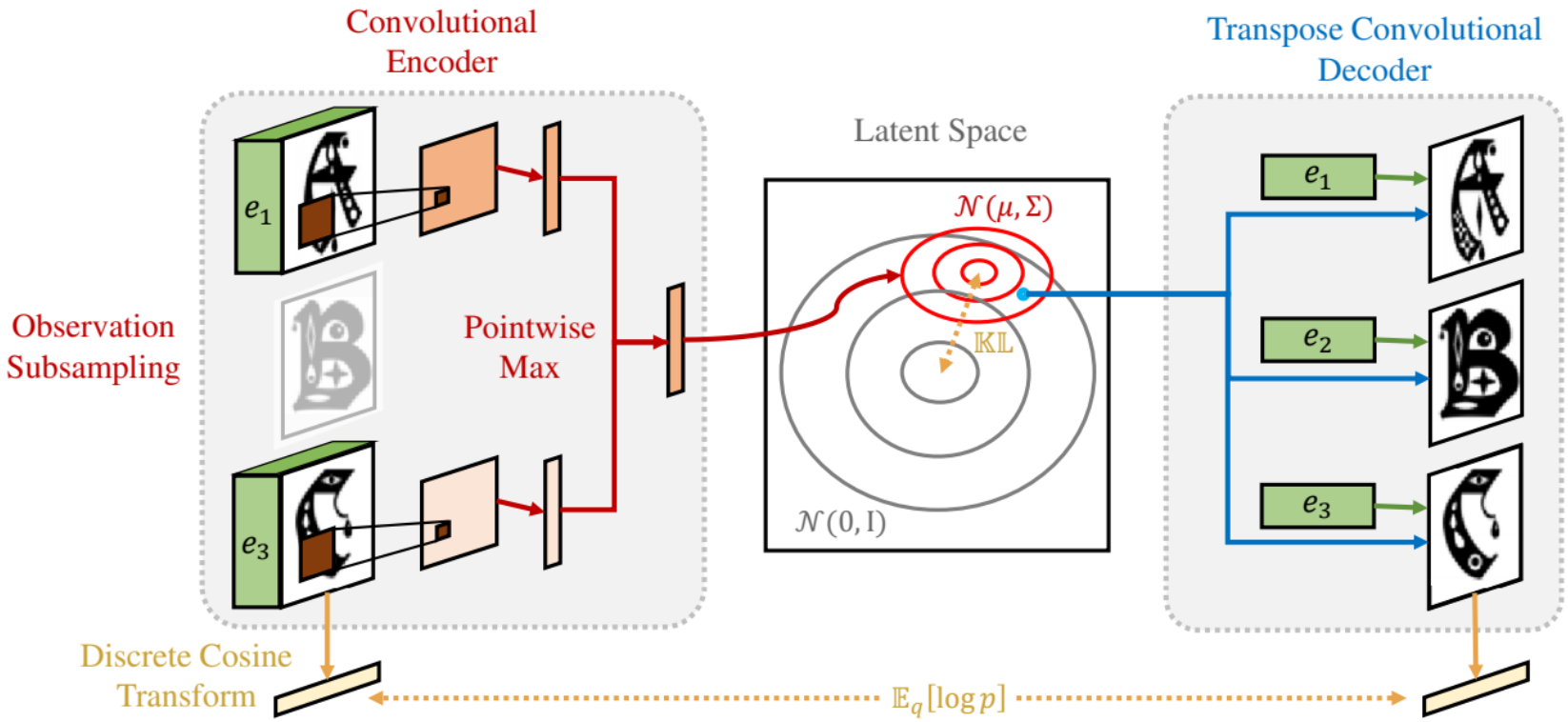
Model

- **Cauchy Distribution [3]**
  - The Cauchy distribution accurately captures the heavy-tailed structure of the edges in natural images.
  - Computing this heavy-tailed loss over the frequency decomposition provided by the DCT-II instead of the raw pixel values encourages the decoder to generate sharper images without needing either an adversarial discriminator or a vectorized representation of the characters during training.

Learning and Inference

- A key property which we desire is the ability to perform consistent inference over $\mathcal{Z}$ given a variable number of observed glyphs in a font.
  - **Observation Subsampling**
    - We mask out a randomly selected subset of the characters before passing them to the encoder.
    - This incentivizes the encoder to produce reasonable estimates without becoming too reliant on the features extracted from any one particular character, which more closely matches the setup at test time.
  - **Encoder Architecture**
    - We use a convolutional neural network which takes in a batch of characters from a single font, concatenated with their respective character type embedding.
    - Following the final convolutional layer, we perform an elementwise max operation across the batch, reducing to a single vector representation for the entire font which we pass through further fully-connected layers to obtain the output parameters.

Learning and Inference



[5] Samaneh Azadi, Matthew Fisher, Vladimir G Kim, Zhaowen Wang, Eli Shechtman, and Trevor Darrell. 2018. Multi-content GAN for few-shot font style transfer. CVPR.

Experiments

- **Datasets - Capitals64 [5]**
  - 26 capital letters of the English alphabet as grayscale 64×64 pixel images across 10, 682 fonts.
  - Training: 7649, Dev:1473, Test: 1560
  - Because several fonts have an almost visually indistinguishable nearest neighbor, these datapoints are less informative . Specifically, we choose the 10% of test fonts that have maximal L_2distance from their closest equivalent in the training set, which we call "**Test Hard**".
- **Baselines**
  - Nearest neighbors algorithm (NN)
  - GlyphNet model [5]

Results

- **Automatic Evaluation**

	Test Full				Test Hard			
Observations	1	2	4	8	1	2	4	8
NN	483.13	424.49	386.81	363.97	880.22	814.67	761.29	735.18
GlyphNet	669.36	533.97	455.23	416.65	935.01	813.50	718.02	653.57
Ours (FC)	353.63	316.47	293.67	281.89	**596.57**	556.21	527.50	513.25
Ours (Conv)	**352.07**	**300.46**	**271.03**	**254.92**	615.87	**556.03**	**511.05**	**489.58**

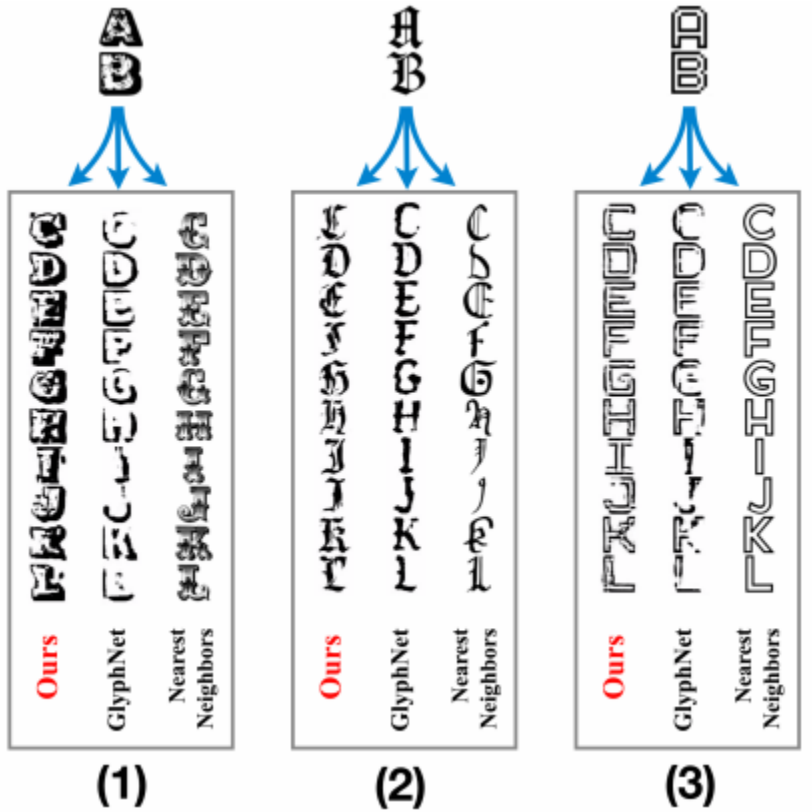
Table 1: L_2 reconstruction per glyph by number of observed characters. "Full" includes the entire test set while "Hard" is measured only over the 10% of test fonts with the highest L_2 distance from the closest font in train.

- **Human Evaluation**
  - Turkers selected a ranking based on which reconstructed font best matched the style of the observed character, and a separate ranking based on which was more realistic.
  - On the full test set (1560 fonts, with 5 turkers assigned to each font), humans preferred our system over GlyphNet 81.3% and 81.8% of the time for style and realism respectively.

Results

Figure 4: Reconstructions of partially observed fonts in the hard subset from our model, GlyphNet, and nearest neighbors. Given images of glyphs for 'A' and 'B' in each font, we visualize reconstructions of the remaining characters. Fonts are chosen such that the L_2 loss of our model on these examples closely matches the average loss over the full evaluation set.

- **Qualitative Analysis**
  - Nearest neighbors will necessarily output well-formed glyphs, but with lower fidelity to the style, particularly on more unique fonts.
  - GlyphNet does pick up on some subtle features.
  - Our model tends to produce the most coherent output on harder fonts.



Results

- **Analysis of Learned Manifold**
  - We take two fonts from the same font family, which differ along one property, and pass them through our encoder to obtain the latent font variable for each. We then interpolate between these values, passing the result at various steps into our decoder to produce new fonts that exists in between the observations.

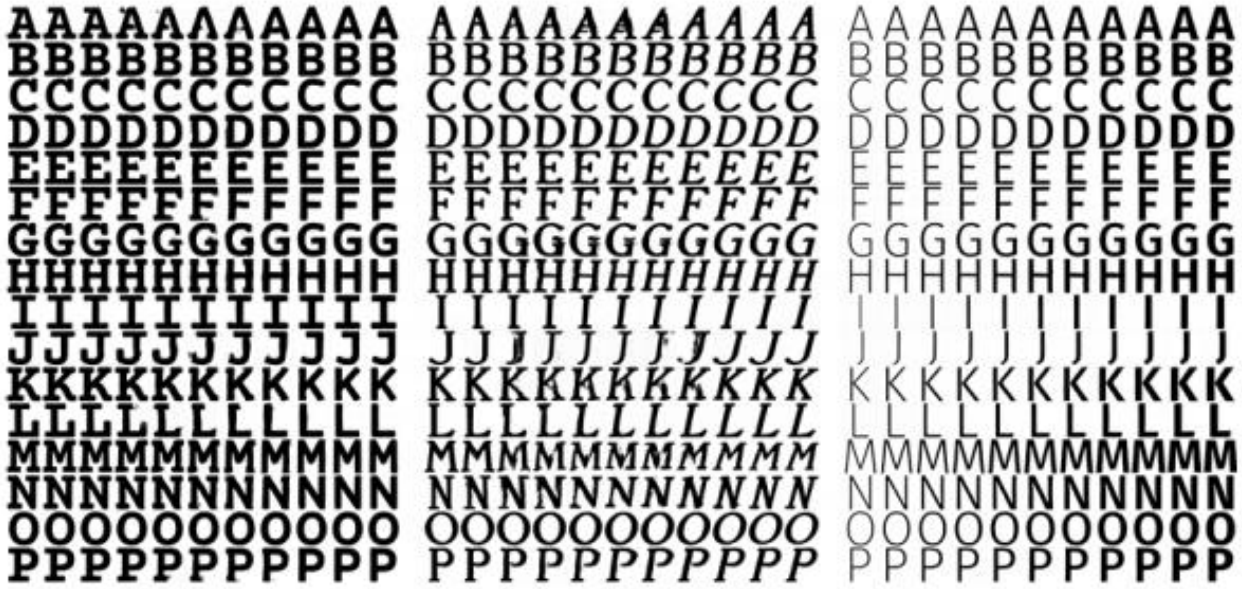


Figure 6: Interpolation between font variants from the same font family, showing smoothness of the latent manifold. Linear combinations of the embedded fonts correspond to outputs that lie intuitively "in between".

[6] Laurens Van Der Maaten. 2013. Barnes-Hut-SNE. arXiv preprint arXiv:1301.3342.

Results

- **Analysis of Learned Manifold**
  - We use our encoder to infer latent font variables $\mathcal{Z}$ for each of the fonts in our training data, and use a t-SNE projection [6] to plot them in 2D.
  - We perform a k-means clusterin with k = 10 on the high-dimensional representations to visualize these high-level groupings. Additionally, we display a sample glyph for the centroid for each cluster.

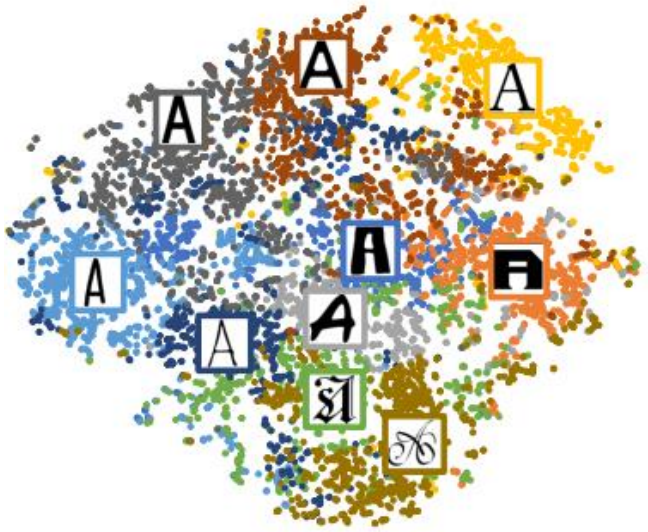


Figure 5: t-SNE projection of latent font variables inferred for the full training set, colored by k-means clustering with $k = 10$. The glyph for the letter "A" for each centroid is shown in overlay.

[6] Laurens Van Der Maaten. 2013. Barnes-Hut-SNE. arXiv preprint arXiv:1301.3342.

Results

- **Analysis of Learned Manifold**
  - To analyze how well our latent embeddings correspond to human defined notions of font style, we also run our system on the Google Fonts dataset which despite containing fewer fonts and less diversity in style, lists metadata including numerical weight and category.

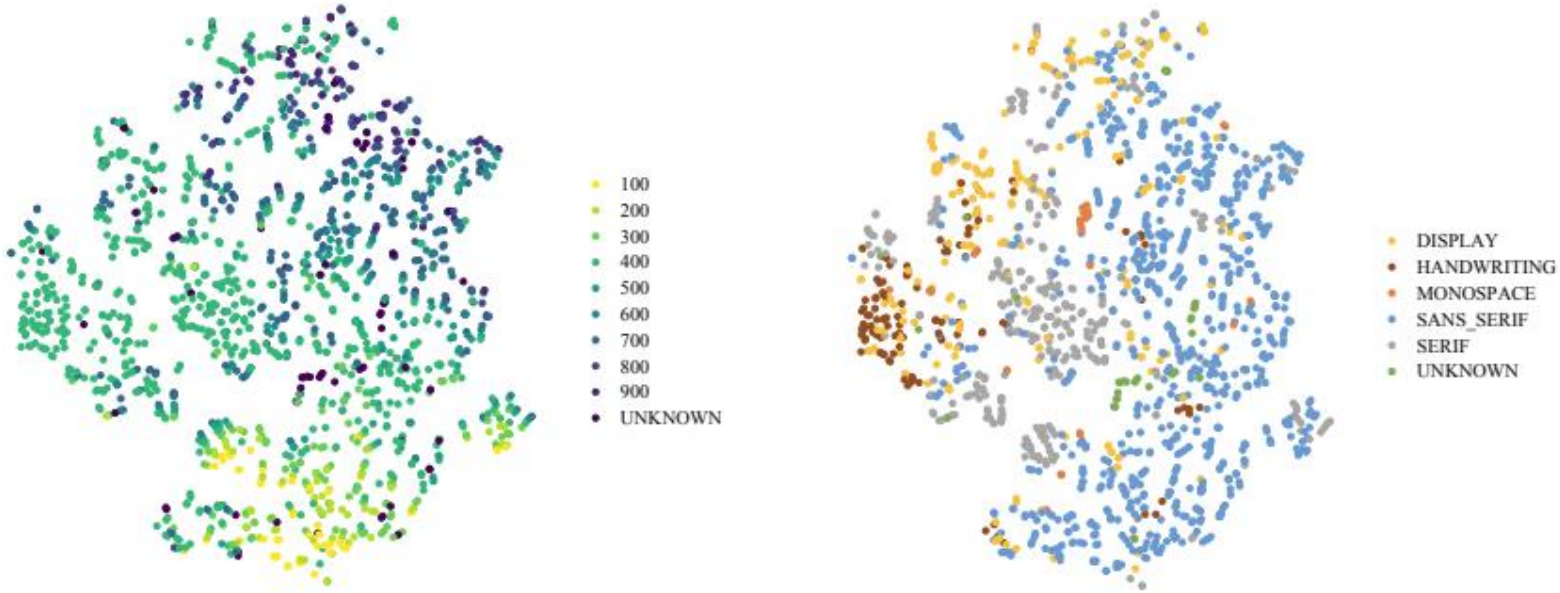


Figure 7: t-SNE projection of latent font variables inferred on Google Fonts, colored by weight and category.

Conclusion

- We presented a latent variable model of glyphs which learns disentangled representations of the structural properties of underlying characters from stylistic features of the font.

- We evaluated our model on the task of font reconstruction and showed that it outperformed both a strong nearest neighbors baseline and prior work based on GANs especially for fonts highly dissimilar to any instance in the training set.

- In future work, it may be worth extending this model to learn a latent manifold on content as well as style.

Progress Report

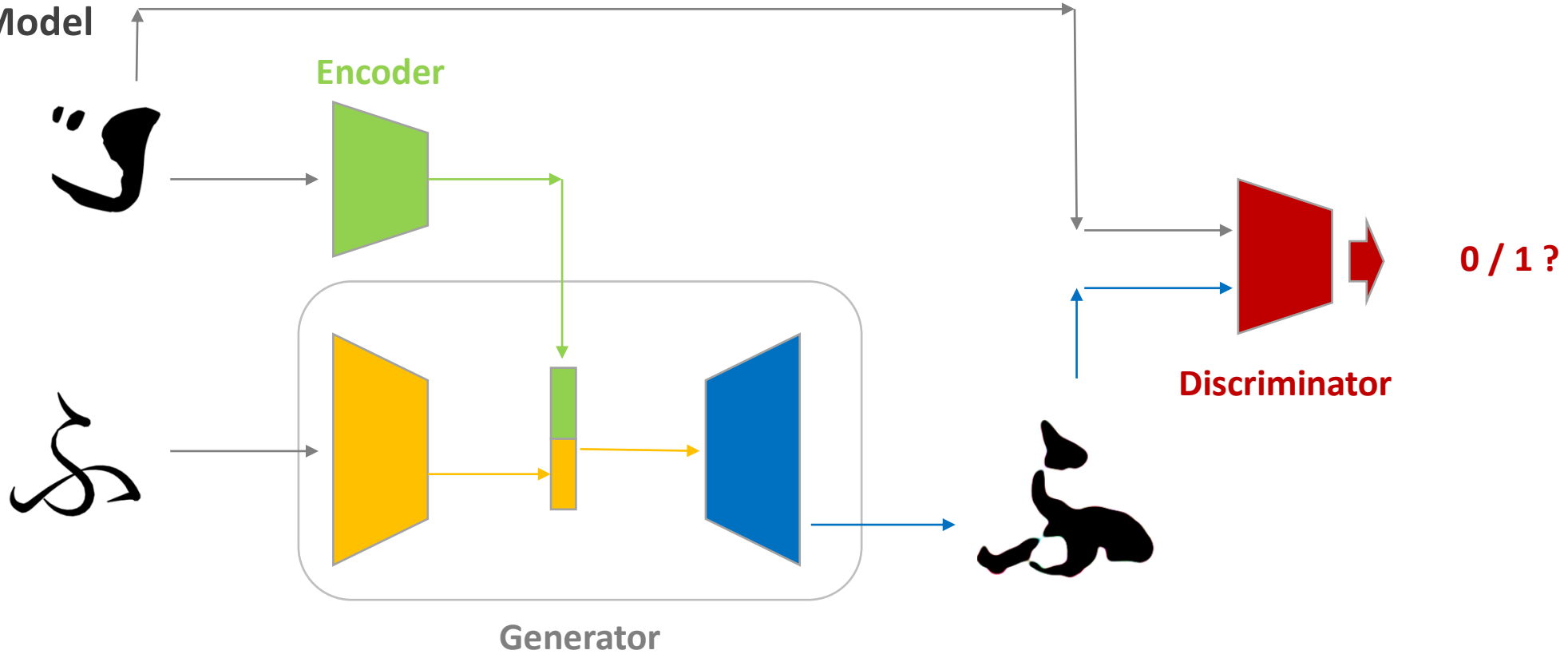
Progress Report

- **Manga Font Generation**
  - To generate the font which is like the ornomatopoeia in manga.

- **Dataset**
  - 15 Font.
  - 142 characters in each font, 113 for training and 29 for testing.
- **Method**
  - GAN

Progress Report

- Model



END