Beautifying handwriting using neural networks

Pankesh Bamotra
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213
pbamotra@cs.cmu.edu

Sanchit Agarwal
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213
sagarwa1@cs.cmu.edu

1 Introduction

1.1 Overview of project idea

Though we live in the age of emails and SMSs, there is no denying the importance of handwritten messages to help connect with someone at much more personalized level. However, handwritten text might impact the legibility and comprehensibility of the content. Our idea is to beautify the handwritten text by superimposing it with features from an existing typography. This way we retain the handwriting style while enhancing it with stylistic features of a known font. The idea is similar in spirit to the popular phone application, Prisma, that creates artistic images by recombining style extracted from one image and content from another.

1.2 Literature Survey

1.2.1 Learning Typographic Style - S. Baluja, Google Research[1]

Baluja has shown that it is possible to learn the style features of the fonts from their corresponding images. There are two tasks in the paper – first one is about classifying if a given character exhibits the characteristics of font by using only four characters from that font as features. The other task is a multi-task learning problem where system tries to learn and generate the other characters of the same font again using only a few characters as basis. Thus, a very small subset of characters is sufficient to learn both discriminative and generative models for representing typographic style. This pair of simultaneously learning the discriminative/generative model allows the author to evaluate the generative model. The accuracy of the discriminative model is 92.1%. The neural network architecture uses convolutional layers, pooling layers, and fully connected layers.

1.2.2 Separating Style and Content with Bilinear Models - J.B. Tenenbaum, MIT; W. T. Freeman, MERL[2]

This paper presents a framework for solving two-factor tasks using bilinear models for sensory inputs of lesser complexity like handwriting using efficient algorithms based on SVD and EM. Bilinear models are able to incorporate mathematical separability while allowing multiplicative interactions between style and content. In general, these two factors can be any two independent factors from the perspective of a perceptual observation.

1.2.3 A Neural Algorithm of Artistic Style - L.A. Gatys, A.S. Ecker, M. Bethge - University of Tubingen, Germany[3]

This paper proposes a view that a DNN can model the complex interplay between the content and style of images. The task was to create artistic images with high perceptual quality. The authors have used CNN based approach to obtain a representation of the style image along with feature space representations to capture texture information. As n the previous paper, this paper brings out the fact

10807 Class Project (2016).

that representations of content and style in a CNN are separable. This fact can be used to manipulate both representations independently to produce artistic styled images. An important fact the authors mention is that the visually most appealing images are created by matching the style representation up to the highest layers in the network.

2 Methods

We follow the similar methodology as employed by Gatys et. al. in [3]. The central idea is to use a pre-trained convolutional neural net (CNN) to extract content and style feature vectors from images. Using these vectors, the total loss is computed which is the sum of content loss and style loss and a regularization term. At runtime, the system starts with a white noise image and iteratively updates it by minimizing the total loss using the standard backpropagation method.

More specifically, let $\vec{h}$ and $\vec{x}$ be the handwriting image and the output images and H^l and X^l be their feature representations in layer l respectively. The content loss w.r.t layer l is then defined as:

$$L_{content}(\vec{h}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} \left(X^l_{ij} - H^l_{ij}\right)^2 \tag{1}$$

In order to calculate the style loss, a style representation is defined by building the gram matrix to capture the feature correlations. Let $\vec{f}$ be the font image and A^l be its style representation from layer l. Let B^l be the style representation of output image from layer l. The style loss w.r.t layer l is then given as,

$$L_{style}(\vec{f}, \vec{x}, l) = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(B^l_{ij} - A^l_{ij}\right)^2 \tag{2}$$

Usually, multiple layers are used to get the complete style representation and the total style loss is a weighted sum of style loss from each layer.

The total loss is then computed as a weighted sum of content loss, style loss and total variational regularization term.

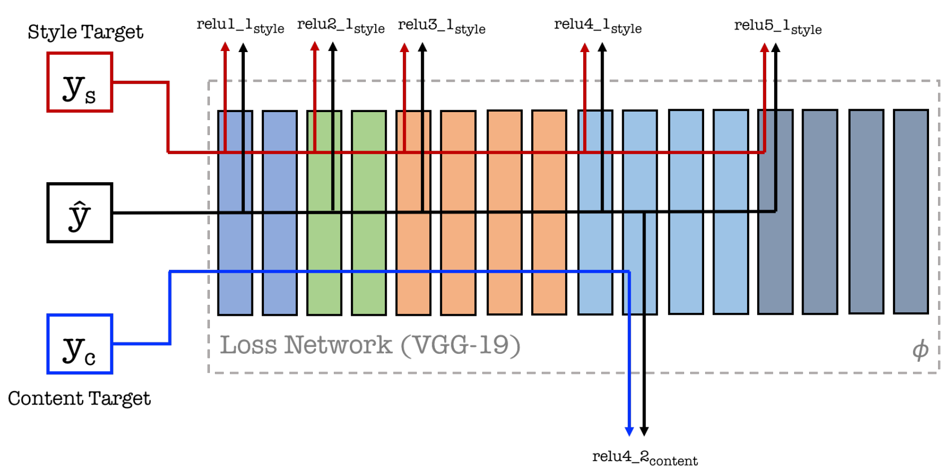


Figure 1: System overview

In our context, the content (extracted from an image with handwritten text) loosely refers to the textual content along with the particular handwriting style e.g, degree of slant, angle of strokes etc. in that image. The style (extracted from a font image) loosely refers to distinct stylistic features e.g. boldness in the font image.

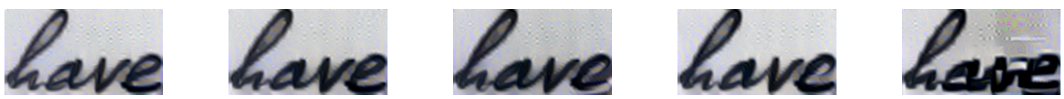
For our initial experiments, we use VGG-19 as the CNN for style and content extraction. As seen in Figure 1 (adapted from [4]), we used *relu4_2* as the content layer and *relu1_1*, *relu2_1*, *relu3_1*, *relu4_1* and *relu5_1* as the style layers.Since VGG is trained with 3-channel images, we had to convert our grayscale images into 3-channel format by replicating the 1-channel input.

3 Preliminary experiments

(a) "have" (b) "B"

Figure 2: Content Images

Figure 3: Style Images



(a) (b) (c) (d) (e)

Figure 4: Style transfer on Figure 2(a), content weight = 0.8, style weight = 0.2

Figure 5: Style transfer on Figure 2(b), content weight = 0.8, style weight = 0.2

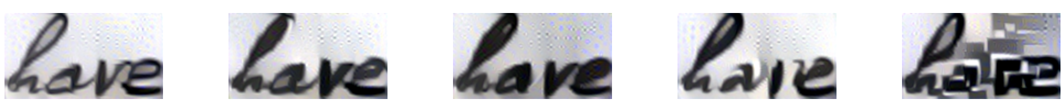


Figure 6: Style transfer on Figure 2(a), content weight = 0.5, style weight = 0.5

Figure 7: Style transfer on Figure 2(b), content weight = 0.5, style weight = 0.5

Figure 8: Style transfer on Figure 2(a), content weight = 0.5, style weight = 0.5, additional style layers

Figure 9: Style transfer on Figure 2(b), content weight = 0.5, style weight = 0.5, additional style layers

We conducted the following set of experiments with our current baseline system in place. In each experiment, we used two handwritten images, one of a single character and the other of a word. We used five different fonts to extract style.

In the first set of experiments, we set the content weight to 0.8 and style weight to 0.2. The results are visualized in figure 4 and figure 5. Though the output images are different from the starting input, the style features are not quite distinctly seen. This is probably because we placed too much weight on the content. It was done so as to preserve the legibility of input text. In Figure 4(e), however, we see some font characteristics in terms of horizontal and vertical lines besides the bolder font weight. For the single character image, not much difference is seen except the boldness.

In the next set of experiments, we give equal weights to content and style, i.e. 0.5. The results can be seen in figures 6 and 7. As expected, we see more stylistic features in the output images, however, now we lose some legibility as expected. The performance and transferred style at character level is visually less appealing and less prominent.

In the final set of experiments, we used two additional layers *relu1_2* and *relu2_2* to extract style from fonts. The results are presented in figures 8 and 9. As seen, the results are more appealing than before, though, legibility is badly hurt in 8(e). Similar to previous experiments, it is hard to visualize the effect with single character image.

4 Final plan

- Milestone 1: Experiment with different loss function for e.g., per pixel loss for content or use kernel to compute gram matrix for style.
- Milestone 2: Experiment with different layers for extracting content and style representations from VGG net.
- Milestone 3: Use a deep variational Autoencoder trained to reproduce for e.g. MNIST digits to extract style and content representations.
- Milestone 4: Build an evaluation model as a discriminative classifier to test how well the style was transferred to the input. This is important because currently we rely on visual perception of the output to decide if style transfer worked well or not.
- Milestone 5: Work towards final project report.

5 Additional notes

5.1 Dataset

After comprehensive survey of literature and discussion with the project TA, we are now using the following datasets for our experiments.

- Processed fonts data [5] by Eric Bernhardsson
- IAM Handwriting Database (http://www.fki.inf.unibe.ch/databases/iam-handwriting-database)
- NIST special database 19 [6] handprinted forms and characters database

References

[1] Shumeet Baluja. *Learning Typographic Style*, arXiv (2016).

[2] Tenenbaum, Joshua B., and William T. Freeman. *Separating style and content with bilinear models*, Neural computation 12.6 (2000): 1247-1283.

[3] Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. *Neural algorithm of artistic style*, arXiv preprint arXiv:1508.06576 (2015).

[4] Johnson, Justin, Alexandre Alahi, and Li Fei-Fei. *Perceptual losses for real-time style transfer and super-resolution arXiv preprint arXiv:1603.08155 (2016).*

[5] Eric Bernhardsson, Analyzing 50k fonts using deep neural networks

[6] Grother, Patrick J. "NIST special database 19 handprinted forms and characters database." National Institute of Standards and Technology (1995).