Font Similarities with Artificial Neural Networks

Edy Garfinkiel
Department of Computer Science - University of Toronto
Bahen Centre 40 St. George Street, Room 4242 Toronto ON, M5S 2E4, Canada
egarfink@dgp.toronto.edu

Abstract

In this paper, I attempt to map images of characters of multiple fonts from image space to a latent font space. This is experimented by using three different neural network architectures. The networks attempt at performing both character and font recognition with a subset of hidden nodes responsible for each task. I evaluate this approach by testing the classification performance of the neural network on test data as well as on the classification attempt of the network on unseen fonts. The results on test data are also evaluated in the latent space by k-nearest neighbors (k-NN) and neighbourhood components analysis (NCA) with k-NN. The results show lower misclassification errors in the latent space showing that there is a better space to represent fonts. Interestingly, in the latent space, characters of the same font tend to cluster together suggesting possible similarities in the fonts as well as similarities in unseen fonts.

1 Introduction and Related Work

Styles are present in line drawings, paintings, fonts and almost any form of art. It is interesting to note the difference between the style of a piece of work and its underlying content. A particular piece of art, say a line drawing, may be drawn by an artist that has her own style and a collection of her works would be easily recognized as having the same style. Similarly, most characters in a particular font are part of it since they possess the prominent features of that font, or what can be thought of as style. If a set of characters from varying fonts are mixed together, it is certainly (at least in the case of very different styled fonts) possible to classify the characters by font due to the features that make the characters fit together. That is, there is an underlying style in a character that is separate from its content (the letter or digit). The perceptual problem of decoupling style from content is one that the human brain can perform. It has been studied and is part of ongoing research [3]. Learning an artist's style so as to synthesize the style in new strokes is of interest in computer graphics [2].

I approach the question of whether it is possible to learn a mapping from the original image space so that it may be represented in a new 'font' space, whereby characters of a particular font would cluster or be close by some distance metric. I choose a set of training fonts that possess a broad range of styles and that are differentiated easily. The mapping can then be used on new, unseen fonts with the motivation that similar-looking fonts would be neighbours in the latent font space. With this idea in mind, similar fonts may be chosen over others and similarities between fonts can be learned. That is, the network will try to classify a character to a known font and the motivation is that the known font should be similar in style to the font of the input character. I create three different architectures of neural networks and evaluate their performance on both the held out data and on the final classification attempt.

2 Description of Dataset

A total of 21 fonts of broad styles were chosen as the base training fonts. These are freely available fonts already installed on most computers. Each font consists of 62 characters, namely uppercase, lowercase and ten digits. The characters are centered on a 32x32 pixel grayscale image rendered at the maximal point size that allows for each of the characters to fit within the borders. To train the neural network, 12 random characters from each font were held out for pseudo-validation[1] and 50 for training. Varying splits were experimented with as well. In Figure 1, the held out characters for testing are shown for a subset of the final fonts used. In the implementation, the characters are actually white on a black background so that the 'ink' is in white.

Due to some fonts having peculiarities such as some characters coming out of the bounding box when rendered, some characters have minor parts cut out. The majority of the character was still captured and this fortunately was only the case for some of the characters of only a subset of fonts. This is regarded as a minimal level of noise in the data that still allows for the prominent features of the font's style to be captured.

A particular challenge in dealing with fonts is that of a limited data set. This would not be an issue with line drawings as a larger training set of strokes within the same style can be created and plenty of unseen strokes could be used for validation and model selection [1]. In the case of fonts, the data set is quite a closed one and limits the amount of learning that can take place. Regardless, the results proved that there is something worth investigating.

The following are the training fonts used: AGaramondPro-Regular, AmericanTypewriter, Apple Chancery, AppleCasual, BickhamScriptPro-Regular, BrushScriptStd, Chalkduster, CharlemagneStd-Bold, Comic Sans MS, Copperplate, EccentricStd, Euclid Fraktur, Giddyup-Std, Handwriting - Dakota, Herculanum, HoboStd, LithosPro-Regular, NanumScript, Noteworthy, OCRAStd, Papyrus, Pilgiche, Zapfino.

The following are the testing fonts, unseen to any of the networks: Andada, Chalkboard, Chunky, DroidSansMono-Regular, GEInspira, Harrington, KomikaText, Monaco, Zalamander, Tagesschrift.

Figure 1: Sample of Font Dataset

3 Neural Network Architectures

Three different architectures are used to approach the problem of mapping a font to a latent font space. As discussed, the image of a particular character possesses both content and style information

[1] Since the data set is limited and the intent of the method was to learn similarities between fonts, the testing performance of the model was based on the fitness of the representation of the fonts in the latent space rather than on the generalization performance on the remaining 12 characters, although both were used. The held out characters can be thought more of as a validation set, while the new font space representation as a form of testing.

and we wish to somehow factorize these two pieces of information. As such, we would like the network to specialize and learn important features of this information and for it to factorize these tasks. There is one basic direct approach to learning a mapping and two slightly more complicated architectures that attempt at factorizing the tasks. The final number of hidden nodes were chosen in such a way so as to achieve good testing error on the held out characters and for getting sensible mapping of unseen fonts to the latent space. This is rather subjective as there is not a direct distance measure between two fonts or styles.

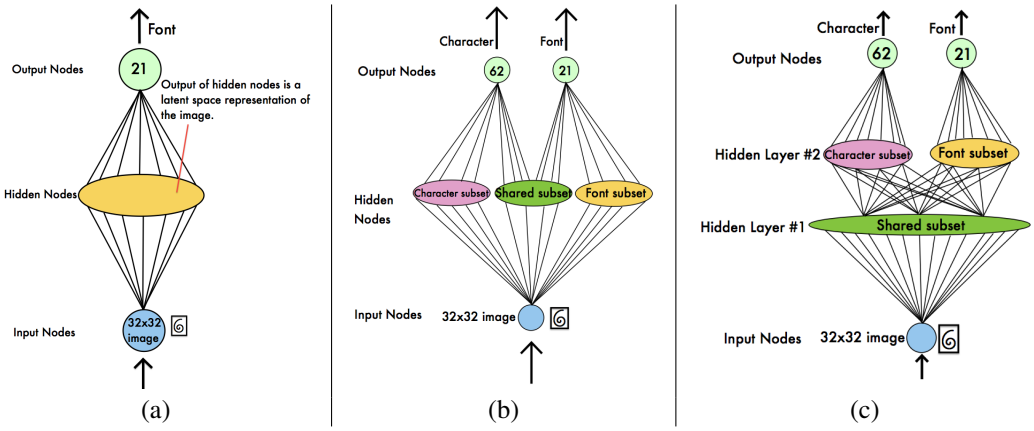

Figure 2: Neural Network Architectures

3.1 Single Hidden Layer with Output of Font Class

This architecture is set up with one hidden layer and one softmax activated output layer representing the fonts used for learning. In this architecture, the hidden nodes take on the task of representing the fonts directly into a font space. This is the simplest of all architectures attempted. The output of the hidden nodes is a representation of an input image in a latent font space. Refer to Figure 2(a) for a diagram of the network.

Final choice of nodes: 10 hidden nodes

3.2 Single Subdivided Hidden Layer with Outputs of Font and Character Classes

This architecture contains one hidden layer and two softmax activated outputs. One of the outputs is a softmax over all characters while the other is a softmax over all fonts used in the training set of fonts. This network performs both character and font recognition. The hidden layer is split into three subsets, each responsible for a task. The left subset is fully connected to the character output nodes and similarly the right subset is fully connected to the font output nodes. The middle subset is a shared subset that is fully connected to both character and font output nodes. Refer to Figure 2(b) for a diagram of the network.

Final choice of nodes: 10 character nodes, 15 font nodes, 25 shared nodes

3.3 Two Hidden Layers with Outputs of Font and Character Classes

The final architecture is similar to the second with the exception that the shared nodes are dedicated to a complete hidden layer. As such, the first hidden layer is fully connected to the second hidden layer that is composed of two subsets: a subset of nodes responsible for the character (content) part of the image and a subset of nodes responsible for the font (style) part of the image. Once again, each separate subset is fully connected to its respective output (character subset fully connected to character softmax output and font subset fully connected to font softmax output). Refer to Figure 2(c) for a diagram of the network.

Final choice of nodes: 30 character nodes, 15 font nodes, 10 shared nodes

4 Experimental Results

4.1 k-NN in original image space

As a base test, k-NN was experimented with for font classification in the original image space. Since at this point the characters are simply represented by their pixel intensities, the results are expectedly poor. The desire is to have characters of a particular font (style) be neighbours to one another. However in the original image space, it is much more likely for characters of the same content to be neighbours, since most characters of equal content but of different style possess a similar overall sweep in the image. For example, the character 'A' in one font is likely to be close, in image space, to the same character 'A' in another font. The results obtained are:

K = 1 Test error: 0.77
K = 3 Test error: 0.78
K = 5 Test error: 0.77

4.2 Neural Network Experiments

4.2.1 Training of Networks and Baseline Tests

The neural network architectures were experimented with varying numbers of hidden nodes for the different subsets until a satisfactory level of training and testing error performances were achieved on the held out characters. As well, the performance of the network on classifying unseen fonts was considered in varying the number of nodes. Training and testing curves are shown for architecture 2 in Figure 3 which used a learning rate of 0.001 and a momentum coefficient of 0.1.

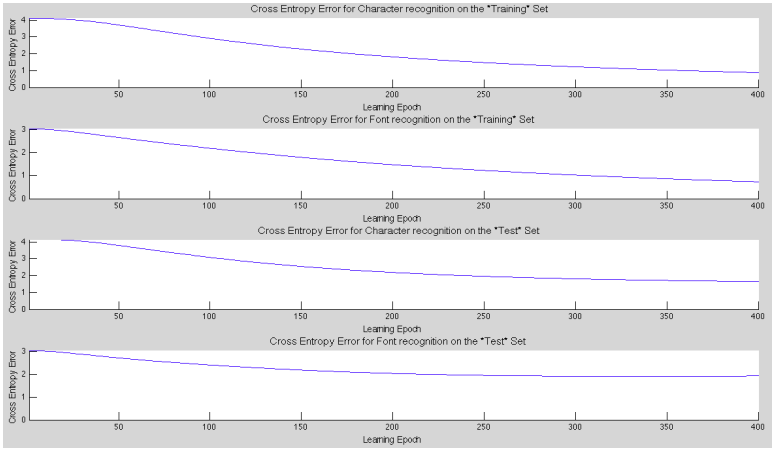

Figure 3: Training and testing curves for architecture 2

The following are the tests performed on the left out characters on each of the networks:
1) k-NN classification in latent font space
2) Classification with learned neural network weights
3) NCA in latent font space with k-NN.

The test error results for test 1) for each architecture:

K = 1 Test error: 0.59	K = 1 Test error: 0.62	K = 1 Test error: 0.75
K = 3 Test error: 0.55	K = 3 Test error: 0.62	K = 3 Test error: 0.77
K = 5 Test error: 0.55	K = 5 Test error: 0.63	K = 5 Test error: 0.78

The test error results for test 2) for each architecture:

Architecture 1, misclassification error on test characters: 0.56
Architecture 2, misclassification error on test characters: 0.62
Architecture 3, misclassification error on test characters: 0.85

Lastly, the results using NCA performed rather poorly on all architectures. The NCA algorithm was run in the latent font space output by the hidden layers of the neural network. The test error results for test 3) for each architecture:

K = 1 Test error: 0.62	K = 1 Test error: 0.67	K = 1 Test error: 0.83
K = 3 Test error: 0.62	K = 3 Test error: 0.67	K = 3 Test error: 0.83
K = 5 Test error: 0.63	K = 5 Test error: 0.67	K = 5 Test error: 0.85

4.2.2 Using Trained Networks to Find Similar Fonts

With the trained networks on the 21 fonts, a set of 10 unseen fonts by any of the network were used. The testing fonts were chosen so that some fonts have similar appearances to one or more of the training fonts. It is expected that the network would attempt to classify a character of a given font in the testing set with a similar-looking font in the training set. This is a difficult measure, however a simple test used and hypothesized was a comic-like font being mapped to another comic-like font. This was in fact the case with one of the networks.

The 62 characters in the testing font were input to the network and classified by the network. Since the network had never seen the font, the classification result can be viewed as the network's best attempt at classifying the character to the training font with most similar features. However, each of the 62 characters are classified into a font which may be different for some characters. In order to pick a suitable font class for the entire testing font, the font most classified for individual characters was used. For the experiments, the second neural network architecture was used producing desirable results. The first architecture was not far off in results. Some interesting mappings on three of the fonts are reported. The two highest choices for fonts are shown in decreasing order:

1) Chalkboard mapped as Comic Sans and Futura
2) Chunky mapped as HoboStd and Copperplate
3) Harrington mapped as Courier New and Handwriting - Dakota

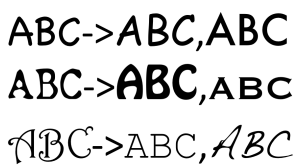

Figure 4: Top two predictions for the above three outlined test fonts

As well, for visualization purposes, the mapped data from the latent font space was reduced to two dimensions using PCA in Figure 6. Interestingly, when viewed in two dimensions, characters of the same font tended to form clusters. All networks produced similar outputs.

5 Conclusions and Future Extensions

Overall, the networks achieved a better classification rate in the learned font space and helped to show that there is something interesting to be learned about fonts. This was seen from the two dimensional visualization where obvious clusters of fonts formed which helped merit the proposed hypothesis. This was also seen upon the classification of the font 'Chalkboard' as 'Comic Sans', both very similar comic-style fonts. This was certainly not the case for all fonts, however some desirable results were found and interesting style properties can be learned with neural networks.

There is room in continuing work on this project by using a much larger training font set to increase the variance of styles to be learned from. As well, different image sizes might prove to be effective in capturing more details of the style as some is lost due to discretization. Furthermore, this approach can be used and extended to studying further the style and content in line drawings where a much larger data set exists. Line drawings of the same style would hopefully follow the same trend of clustering in a latent space, depending on the features chosen to represent them.

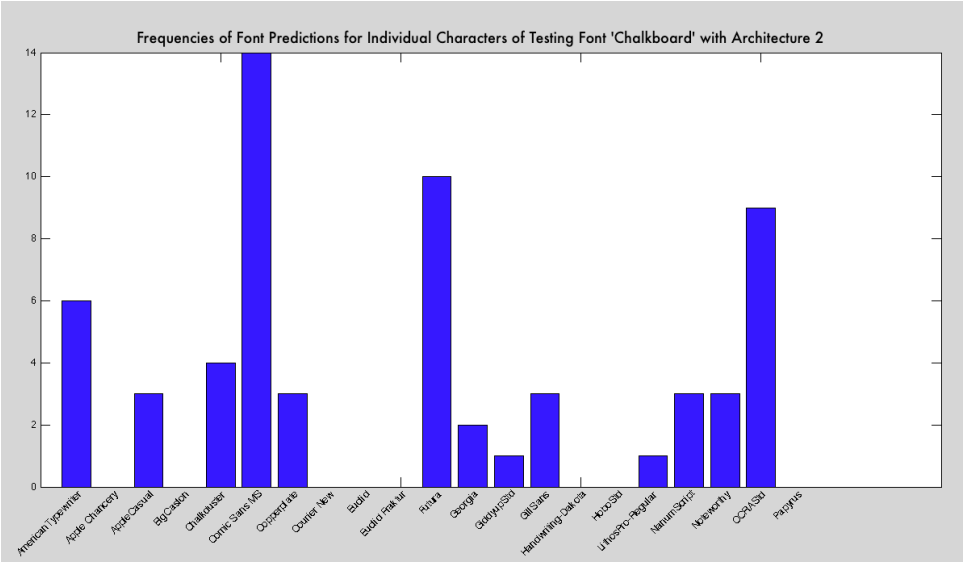

Figure 5: Frequencies of predictions for individual characters for font 'Chalkboard'

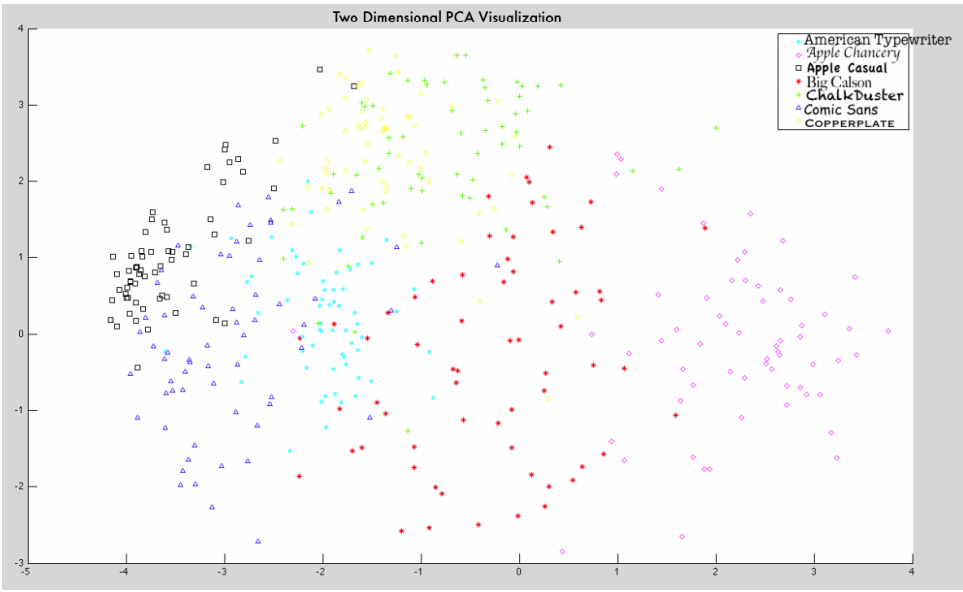

Figure 6: Latent space data from neural network architecture visualized in two dimensions. Characters in the same font family form clusters

References

[1] William T. Freeman, Joshua B. Tenenbaum, and Egon C. Pasztor. Learning style translation for the lines of a drawing. *ACM Trans. Graph.*, 22(1):33–46, January 2003.

[2] Jingwan Lu, Fisher Yu, Adam Finkelstein, and Stephen DiVerdi. HelpingHand: Example-based stroke stylization. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, August 2012.

[3] Joshua B. Tenenbaum and William T. Freeman. Separating style and content with bilinear models. *Neural Comput.*, 12(6):1247–1283, June 2000.