Journal of Data Mining and Digital Humanities

EPI sciences overlay journals

Toward Automatic Typography Analysis: Serif Classification and Font Similarities

Syed Talal Wasim[1,2], Romain Collaud[3], Lara Défayes[3], Nicolas Henchoz[3], Mathieu Salzmann[1], Delphine Ribes[3]

[1]Computer Vision Lab, Ecole Polytechnique fédérale de Lausanne, Lausanne, Switzerland
[2]Department of Computer Vision, Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi, United Arab Emirates
[3]EPFL+ECAL Lab, Ecole Polytechnique fédérale de Lausanne, Lausanne, Switzerland

Corresponding author: Delphine Ribes , delphine.ribes@epfl.ch

Abstract

Whether a document is of historical or contemporary significance, typography plays a crucial role in its composition. From the early days of modern printing, typographic techniques have evolved and transformed, resulting in changes to the features of typography. By analyzing these features, we can gain insights into specific time periods, geographical locations, and messages conveyed through typography. Therefore, in this paper, we aim to investigate the feasibility of training a model to classify serif types without knowledge of the font and character. We also investigate how to train a vectorial-based image model able to group together fonts with similar features. Specifically, we compare the use of state-of-the-art image classification methods, such as the EfficientNet-B2 and the Vision Transformer Base model with different patch sizes, and the state-of-the-art fine-grained image classification method, TransFG, on the serif classification task. We also evaluate the use of the DeepSVG model to learn to group fonts with similar features. Our investigation reveals that fine-grained image classification methods are better suited for the serif classification tasks and that leveraging the character labels helps to learn more meaningful font similarities.

Keywords

Machine learning; Digital humanities; Typography; Classification models

I INTRODUCTION

Typography is everywhere, from street signs, advertising boards, product labels, and even the text on our clothes carrying a piece of information, a message, a tone, and even an emotion. The importance of typography has been recognized for centuries, and the history of typography reflects the evolution of technology and design as described by Jury [2007] and Tselentis et al. [2012]. Over time, typographic features have evolved according to technological advances and cultural shifts. Studying typographic features can help us understand the cultural and technological area it was designed for and the message it wishes to convey. Analyzing font characteristics can also assist designers and typographers in comprehending the relationships between typefaces and how they can be used together. Additionally, it can shed light on the origins of fonts and how they influence each other. For example, in the 1950s, Max Miedinger and Eduard Hoffmann were commissioned to create the most harmonious font they can. They were inspired by one of the most widely used sans serif fonts of the time, Akzidenz-Grotesk,

introduced by the Berthold foundry in 1898. They made a subtle interpretation of it making it simpler and more readable. The result presented, originally called Neue Haas Grotesk, then Helvetica, is a very legible and widely used font. This work of reinterpretation is strongly linked to the arrival of new techniques (passage from molten metal to phototype, then to digitization) which allow better control and greater freedom of the letter drawing. Nowadays, a new version of Helvetica (Helvetica Now) has been designed for optimal readability on screens, especially at a reduced size. The use of tools, ideally incorporated directly into the typographer's software, to emphasize these modifications can be helpful in recognizing resemblances and discrepancies. The paper presented here concentrates on examining a specific aspect of typography, namely the serif, and exploring the similarities among different fonts.

Font features range from but are not limited to, font weight, font width, font contrast, x-height, serifs and sans, roman and italics (see Figure 1 and Figure 2). One of the most noticeable and distinctive font features is the serif. Serifs are the short appendixes found at the end of the main stems of some characters, as explained in Monotype [2020]. Intuitively, we expect to be able to distinguish the different types of serifs to help non-experts tune into the details of typographical elements and the various styles that exist. Furthermore, classifying a font based on its serif type helps designers understand the context in which the font was created, as well as the role it plays in effective and aesthetically pleasing communication. In this study, we, therefore, investigate the use of different machine learning algorithms to automatically classify serif fonts. Specifically, we evaluate both general and fine-grained image classification models. To provide a context, reference, and to highlight the specific challenges linked with serif classification, we used the same models to perform font classification, a well-known task from the literature.

The second feature we investigate in this study is font similarities. This characteristic serves a multitude of purposes. Firstly, it allows us to identify connections between fonts that were created during different time periods, yet adhere to the same principles. Additionally, it provides insight into the evolution of typeface design over time, as well as its various applications in print and screen contexts. Furthermore, it can assist designers in avoiding the use of font variants that are overly similar when developing a collection of fonts for a poster or a book while allowing for certain features that bind them together. In our context, font similarity is defined as the capacity of an algorithm to find fonts that apply similar features (e.g., similar serif, similar width, similar contrast). While finding similar fonts based on raster images and conditioned on both the character and the font has been studied in the past (see section II), here, we follow a different approach and develop a method based on Scalable Vector Graphics (SVG) data, as vector files are the file format primarily used by typographers when creating a font. To this end, we propose and evaluate an SVG-based Variational Auto-Encoder to measure font similarity. We also evaluate the feasibility of learning similarity without knowing the character label (e.g., if it is a A, or a B). In order to facilitate comparisons, we trained a convolutional variational autoencoder on raster images and used the same evaluation methods.

To assess our algorithms (serif and similarities), we created an annotated dataset detailed in the method section and available as supplementary material. Our results indicate that fine-grained recognition models provide the most effective framework for serif classification and that knowing the character label helps in learning font similarities.

II RELATED WORK

In this section, we review the literature from three domains: algorithms for typographic analysis, deep learning algorithms for image classification as they can be directly applied to the studied

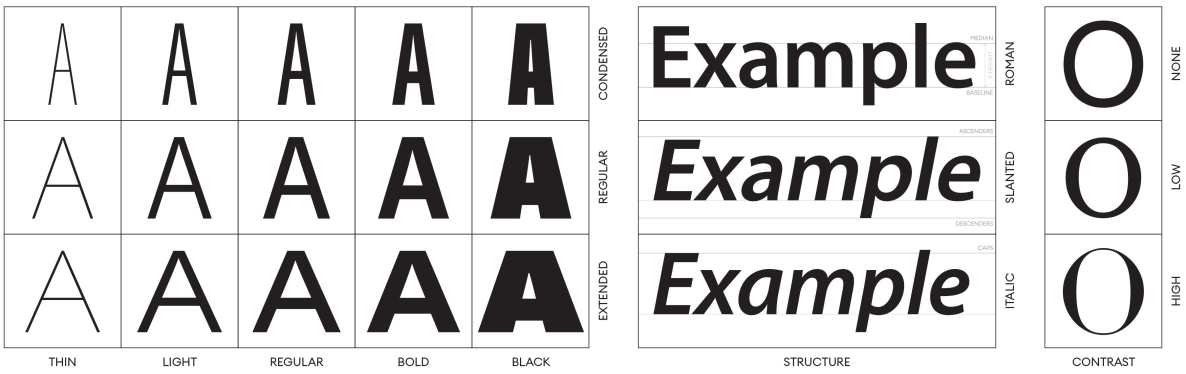

Figure 1: Font features examples: weight, baseline, contrast, structure, ... Font features are numerous, we present here a non-exhaustive list to understand their varieties. More details can be found in *The Geometry of Type* and *Letter Fountain* written by Coles [2016] and Pohlen [2011] respectively.

cases, and deep learning algorithms on SVG images.

2.1 Typographic Analysis

To the best of our knowledge, few attempts have been made to automatically analyze typographic features. Existing methods use Variational Auto-Encoders (VAE) such as Kingma and Welling [2014] to build a latent space representation of the font structure as in Srivatsan et al. [2019]. In this case, the auto-encoder is conditioned on both the character and the font. Using this method, it is possible to build a latent space that groups together characters from similar fonts. Other existing methods, such as Adobe DeepFont Wang et al. [2015] or Javed et al. [2014], Zhu et al. [2001], Ioffe and Szegedy [2015], Zramdini and Ingold [1998], Bataineh et al. [2012], Gao et al. [2008], focus on font classification and font recognition rather than font feature analysis.

In addition, some methods have been proposed to extract font characteristics such as the text line described by Murdock et al. [2015] and the baseline explained in Diem et al. [2017]. Another method developed by Shinahara et al. [2019] attempted to build a feature representation of fonts but focused on categories like “historical” and “fancy” rather than typographic features.

2.2 Deep Learning for Image Classification

This section describes standard image classification methods that have been used on the Imagenet benchmark from Russakovsky et al. [2015] and can be applied to our context. These methods were selected as being the state-of-the-art algorithms at the start of the project. They include EfficientNets from Tan and Le [2019] (and its variants as Xie et al. [2019]), and Vision Transformers such as Dosovitskiy et al. [2020]. In addition, we describe methods to perform *fine-grained* image classification, targeting the recognition of visually-similar categories, such as different species of birds. Our choice of such methods was motivated by the fact that serif features are small and difficult to distinguish.

EfficientNet described by Tan and Le [2019] uses a network architecture scaling method that scales all the feature dimensions (depth/width/resolution) using a *compound coefficient*. Unlike conventional practice, *compound coefficient* applies an arbitrary scaling. The model is based on the inverted bottleneck and linear residual blocks of the MobilenetV2 from Sandler et al. [2019] while adding squeeze and excitation layers.

The vision transformer explained in Dosovitskiy et al. [2020], also an image classification model, is inspired by the Natural Language Processing domain state-of-the-art Transformer model from

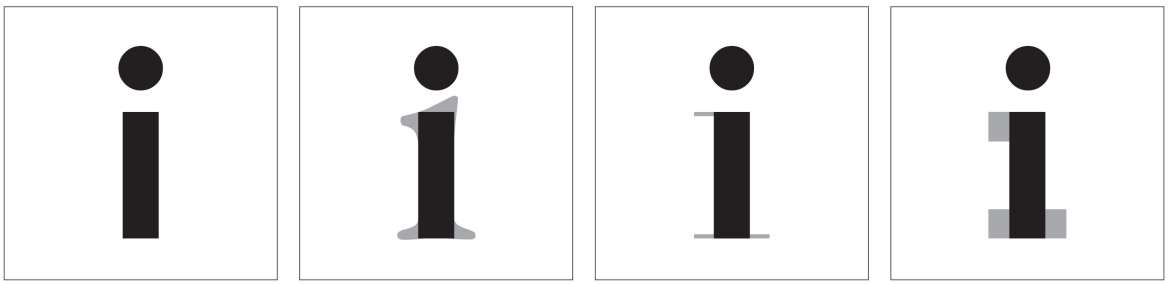

Figure 2: Examples of serifs and sans serifs types. In the context of the project, we grouped them into three categories, triangular, linear and slab.

Vaswani et al. [2017]. It is applied to image patches. The model was the first example of a fully transformer-based model applied to a vision task without any convolution layers. It achieved state-of-the-art accuracy at the time it was published on the Imagenet dataset and still is very competitively close to the current state-of-the-art.

The state-of-the-art in fine-grained image classification at the start of the project was the TransFG model from He et al. [2021], which builds on top of the vision transformer from Dosovitskiy et al. [2020] by employing a "parts selection module" to localize the regions with the most distinctive features. It then uses a contrastive loss to improve the discrimination between regions. TransFG and other fine-grained recognition models are interesting to consider for learning to localize specific parts of the image that are most relevant to the classification task. This is specifically interesting for typography-related tasks like classification of the serif type, as serifs are found in specific positions on different characters and are quite small compared to the overall letter.

The above methods were proposed for general image recognition, or fine-grained recognition on standard benchmarks, none of which depict typographic content. Here, we therefore evaluate them for the task of serif recognition.

2.3 Deep Learning for SVG Images

Unlike raster images, deep learning on vector graphics is a domain that has not received extensive attention. Using vectorial formats like SVG instead of the raster format is relevant for the typography domain as they are commonly used by typographers and are practical for their scalability. To measure the feasibility of integrating similar features within typographer's software, we focus the study here on the SVG format. Mostly focused on the generation of vectorized sketches, the SketchRNN described by Ha and Eck [2017] used a Long Short-Term Memory (LSTM) as in Hochreiter and Schmidhuber [1997] based VAE from Kingma and Welling [2014]. Recently, the Sketchformer developed by Sampaio Ferraz Ribeiro et al. [2020] replaced the LSTM-based model with a Transformer as in Vaswani et al. [2017] based architecture, resulting in better graphic generation due the Transformer's ability to better represent long temporal dependencies. These methods worked with datasets of SVG icons from various themes, focusing on the tasks of image reconstruction and latent space operations.

One of the first methods that could generate full vector graphics with straight lines and Bezier curves was SVG-VAE from Lopes et al. [2019], which used a one-stage autoregressive approach to generate path commands. By contrast, DeepSVG from Carlier et al. [2020] proposed a two-stage hierarchical Transformer based architecture, which instead uses a feed-forward approach to predict path components in a non-autoregressive manner. This method qualitatively showed improvement on the task of SVG generation and interpolation compared to the previous methods.

2.4 Conclusion

Typographic analysis is not an area that has yet received extensive attention. While there have been some attempts at typographic similarity on raster images, they have been done using conditions on fonts and characters. Therefore, while these methods work well in tasks where all font metadata is known, they fail to generalize well to other situations.

Similarly, for deep learning on SVG images, DeepSVG comes out as the best solution among a limited number of attempts in the area. Moreover, all found methods focus on the task of icon generation and not on typographic analysis. This is what we achieve here, leveraging the fact that SVG-based deep-learning methods can be particularly interesting in this field considering that they inherently encode the geometry of the font, and hence could be a good source to differentiate specific typographic features.

III METHODOLOGY

We seek to answer the following questions:

- Given a dataset of images of characters of different serifs types (sans-serif, triangular, linear, and slab), can a model be built to accurately predict the serif type?
- Given a dataset of images of different fonts without font labels, can a latent representation be built that learns to cluster together characters of similar features?

3.1 Classifying Serif Type

While serifs are quite a distinctive feature, they are small compared to the overall font image. The serif may also appear in different positions depending on the character under consideration. Hence, it is unclear whether the problem is better formulated as an image classification scenario or a fine-grained classification scenario. We therefore decided to evaluate both kinds of models on this task. To this end, we trained the EfficientNet-B2 and the Vision Transformer Base model (ViT B) with 12 transformer layers. Also, two variants of the ViT B were trained as defined in the ViT article by Dosovitskiy et al. [2020], with patch sizes of 16×16 (ViT B/16) and 32×32 (ViT B/32), respectively. The method and the model used, as well as the obtained weights for this task, are available as supplementary material.

To test the accuracy of the models, we built a dataset with ground-truth font type and serif category. There are several typologies of serifs, and their distinction is not always obvious and can be linked to an interpretation of their drawing. In our context, we focus on their perceived shape and have grouped the different types of serifs into three different categories (see Figure 2): triangular (including oldstyle, transitional, wedge, etc.), lineal (including hairline, didone, modern, etc.) and slab (quadrangular). A variant of a typeface that does incorporate serifs is called a "serif font", while one which doesn't is called "sans", or "sans serifs". Training and validation datasets were built by first selecting a range of fonts from the Google font database (including variations) for the four categories (sans-serif, triangular, linear, and slab). This was done in collaboration with a type designer. Using this approach, we generated a comprehensive dataset comprising 126 666 font samples derived from 2 043 fonts, including various font variants. the dataset encompasses upper- and lower-case versions of the Latin alphabet (Aa - Zz) as well as digits (0 - 9). The created database is available as supplementary material. The results were evaluated using accuracy measures across the three sets, along with precision, recall, and F1-score on the test set. To provide context to the results, we also trained the same models to perform font classification and used the same accuracy measures.

3.2 Typographic Similarity

Typographic similarity aims to group together characters sharing the same kind of typographic features (belonging to the same font family, having the same serif type, etc.), regardless of the structure of the character itself. This is a challenging task because any model trained in an unsupervised manner (like a VAE) would tend to "memorize" the more obvious character structure and group those together rather than learn the more subtle underlying features unique to each font family. A workaround to this is to use conditions on the character, font family, and font variations, as proposed by Srivatsan et al. [2019].

This approach enables the model to learn some specific features like the serif. However, this method poses a challenge, the necessary amount of font labeled data required (>10K fonts). Such a large amount of font-labeled data is not available. Moreover, there is a growing and exponential number of fonts that do not allow for an up-to-date and open-source database. Therefore, the aim here was to learn a useful latent representation using either no labels or only character labels.

The existing methods employing VAEs to build a latent representation of fonts use raster images with various label conditions. However, no existing work in the literature achieves this using an SVG representation of the fonts. We therefore propose to use a VAE with and without character labels to build a latent representation inspired by the DeepSVG Hierarchical Generative Network presented by Carlier et al. [2020] and discussed in section II.

We used the same training dataset for this task as for the serif classification one but in the format of SVG, and trained two models, one knowing the character label and one without knowing it. We also trained a convolutional variational autoencoder (CNN-VAE) on raster images to provide a comparison (Nellas et al. [2021]). The Davies-Bouldin (Davies and Bouldin [1979]) index is utilized to evaluate all the models. To conduct this evaluation, we employ the k-means clustering technique on the embeddings generated from the middle layer's latent dimension in the models. The determination of the optimal number of clusters was accomplished through the elbow method (Thorndike [1953]). Specifically, during the serif clustering process, we obtain a minimum value of 4, which aligns with the number of classes present. Similarly, when clustering for fonts, we achieve a minimum of 28 clusters, corresponding to the number of fonts in the testing set.

IV RESULTS

4.1 Classifying Font and Serif Type

The accuracy of the serif and the font classification models are calculated on the training, validation, and the font-independent test set described in section III. The results are shown in Table 1 and in Table 2.

With respect to the serif task, the general image classification model (EfficientNet-B2) performs best on the training sets (accuracy of 0.93). However, it shows a significant accuracy degradation on the font-independent test (0.79). In contrast, the fine-grained image classification model, TransFG, performs comparatively worse on the training sets but generalizes very well on the test set (0.91 and 0.89, respectively, for the VIT B/32 backend). The model with the ViT B/32 backend performs much better than the ViT B/16 one, with the larger patch size improving accuracy across all three dataset splits. For further analysis, detailed results are calculated on the font-independent test set, including metrics for precision, recall, which are then used to calculate the F1-score (see Table 1b). We also display examples on the test set in Figure 3. We showcase

Model + Backend	Accuracies	
	Train	Test
TransFG ViT B/16 Backend	0.88	0.84
TransFG ViT B/32 Backend	0.91	0.89
EfficientNet-B2	0.93	0.79

(a) Training and test results for serif classification

Model + Backend	Results on Test Set		
	Precision	Recall	F1-Score
TransFG ViT B/16 Backend	0.92	0.84	0.88
TransFG ViT B/32 Backend	0.93	0.89	0.91
EfficientNet-B2	0.84	0.79	0.78

(b) Precision, recall and F1-Score on the serif classification test set

Table 1: Results for the serif classification task on the training and font-independent test datasets.

the outcomes for letters with obvious serifs (such as "u" and "v"), as well as for letters that present challenges in observing the serif (like "t") or are very difficult (such as "o").

Upon examining the results of the font classification task (Table 2), EfficientNet-B2 model performs better than both of the TransFG models (higher accuracies and better results on the test set).

4.2 Typographic Similarity

The David-Bouldin index for SVG-based and raster based models with and without conditioning on the character labels are provided in Table 3. In short, these indexes show that conditioning on the character label helps in learning meaningful representations for font similarities. Furthermore, we observe better clustering performances for the SVG-based model compared to the raster-based one.

V DISCUSSION AND CONCLUSION

5.1 Serif Classification

For the serif classification, Table 1, the fine-grained image classification model, TransFG, gave accurate results, reaching a font-independent test accuracy of 89.4% and an F1-score of 0.91, using the ViT B/32 backend. While its counterpart with the ViT B/16 backend was not as accurate, it still generalized well on the test set, yielding an accuracy of 84.0% and an F1-score of 0.88. By contrast, the general image classification model, EfficientNet-B2, got an accuracy of 79.2% on the test set. This shows that this model tends to "memorize" the fonts rather than picking up the more subtle underlying serif features. This is also confirmed by the results of the font classification task. As a consequence, the classification of serif types is better modeled as a fine-grained classification task than as a general image classification problem, given the subtle nature of the serif features.

Examining the results in Figure 3 reveals that the majority of errors for the TransFG models occur when dealing with fonts that have extremely subtle serifs or when encountering specific instances of serif fonts where the serifs are absent, such as in the characters "t" and "o". The type

Model + Backend		Accuracies	
		Train	Test
TransFG	ViT B/16 Backend	0.91	0.87
TransFG	ViT B/32 Backend	0.93	0.90
EfficientNet-B2		0.96	0.91

(a) Training and test results for font classification

Model + Backend		Results on Test Set		
		Precision	Recall	F1-Score
TransFG	ViT B/16 Backend	0.83	0.87	0.85
TransFG	ViT B/32 Backend	0.86	0.90	0.88
EfficientNet-B2		0.89	0.91	0.90

(b) Precision, recall and F1-Score on the font classification test set

Table 2: Results for the font classification task on the training and test datasets.

of contrast and the angle of the axis used when drawing a character can provide clues about the type of serif used in a font, even if the serif itself is not visible in a specific letter (e.g. "o"). For instance, when a character has a visible contrast and a tilted axis, it may correspond to a triangular serif font. Conversely, a character with high contrast and a straight axis tends to be associated with linear serif fonts. On the other hand, slab-serif fonts usually lack contrast and can be seen as grotesk sans-serif fonts with added serifs (e.g. Helvetica). These variations in how typographers draw letters have a historical background. Before the advent of digital fonts, letters were crafted using tools like reeds and quills, and the form of these instruments impacted the structure of the letterforms. This impact is still apparent in digital typefaces, albeit in a modified manner.

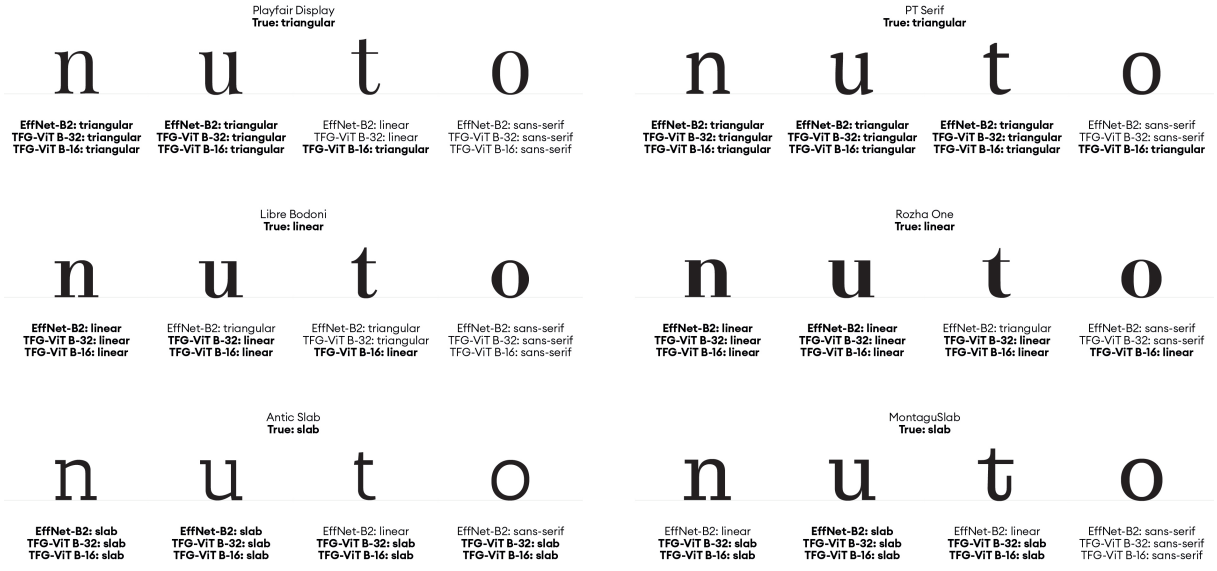

Figure 3: Predicted serif on the serif classification test set. Bold corresponds to correct predictions. Compared to TransFG (TFG), EfficientNet-B2 (EffNet) fails to classify serifs in several visually-obvious cases.

Model	Font Clustering	Serif Clustering
CNN-VAE (Unconditioned)	0.3779	0.4460
CNN-VAE (Conditioned)	0.2932	0.4047
DeepSVG (Unconditioned)	0.3233	0.4105
DeepSVG (Conditioned)	0.2518	0.3338

Table 3: The Davies-Bouldin index was computed on the results of the K-means clustering process for both the SVG-based DeepSVG model and the raster-based CNN-VAE model in the font and serif tasks. Results are presented for both Unconditioned and Conditioned (on character labels) training. Note that the VAE latent dimension is set to $z = 64$ in all cases. In addition, it should be noted that a lower score indicates better clustering.

Classical digital fonts often feature triangular serifs, reminiscent of pen-and-ink writing (e.g., the Garalde font family). On the other hand, modern fonts are often characterized by simpler geometrical shapes. When looking at Figure 3, the triangular fonts *Playfair Display* and *PT Serif*, are both contrasted and with a tilted axis. However, the shape of the letter "t" for *Playfair Display* is very close to what one finds in a Didone linear serif, hence the possible classification error given by EfficientNet-B2 and TransFG ViT B/16. Similarly, the almost straight axis of the letter "o" for *Playfair Display* can be seen as a typical feature of a linear serif making it difficult for the classification task even for an expert. When looking at the misclassified linear serif for *Libre Bodoni*, it is difficult to understand why EfficientNet-B2 predicted a triangular serif instead of a linear serif for the letter "u" as all linear font characteristics are met. The same goes for the letter "o". However, the beak of the letter "t" (top left of the letter) can be interpreted as a triangular serif explaining the confusion of the EfficientNet-B2 and TransFG ViT B/16. Finally, it is hard to explain why EfficientNet-B2 misclassifies the letter "t" for *Antic Slab* and *Montagu Slab* as Slab serif criteria are met. Still the contrast of letter "o" for *Montagu Slab* can explain the wrong predictions of all the models. The visual analysis of the results confirms that the classification of serif types is better modeled as a fine-grained classification task than as a general image classification problem. However, as modern fonts become increasingly hybridized, it is becoming more difficult to make clear classifications. In order to gain a deeper understanding of the fine-grained classification model's decision-making process, one could employ back-propagation strategies to determine the most significant part of the letter that influences the classification outcome. It will also be interesting to compare the classification errors made by the models with the predictions of experts in the field.

One important point to note is that the TransFG model is significantly more expensive to train. On a single Titan RTX GPU with 24GB of CUDA Memory, the EfficientNet-B2 model took 4 hours to train for 30 epochs. By contrast, the TransFG model took approximately 13 hours to train for the same number of epochs. This is primarily because of the Vision Transformer backbone used in the TransFG model (for which see Dosovitskiy et al. [2020]), which is computationally more expensive than the EfficientNet backbone.

Regarding the font classification task, all the models performed equally better (Table 2) than the serif task with slightly better results for the EfficientNet-B2 model. According to this, font recognition is better modeled as a general image classification problem and serif recognition is a more difficult task.

5.2 Font Similarities

According to the Davies-Bouldin index results, both the unconditioned and conditioned SVG-based models demonstrated better performance compared to the raster-based image model. The decision to employ the CNN-VAE model was driven by the desire to establish a fair comparison with the DeepSVG model, as they share similar architectural characteristics. Nevertheless, it is likely that more recent models, such as those proposed by Radford et al. [2021], Caron et al. [2021], and He et al. [2022], would have exhibited superior performance. Thus, we propose considering these models as the next iteration for both raster-based and SVG-based tasks. However, it should be noted that the implementation of the latter will require modifications to ensure compatibility with the vectorial format. Considering DeepSVG, future work in this direction could focus on improving the design of the autoencoder to make it more suitable for font SVGs. In particular, the autoencoder used in this work exploits the hierarchical nature of an SVG, which contains multiple paths, and multiple points in each path. However, they fail in cases where the entire font SVG is formed of a single path, which is the case with some characters. This could be addressed by designing a monolithic architecture that does not explicitly focus on the hierarchy.

5.3 Conclusions

This paper has explored the application of various machine learning algorithms for classifying serif fonts and matching fonts based on their feature similarities. Our experiments have revealed that fine-grained image classification yields the most accurate results for serif classification, although classification may fail in cases where serif shape is difficult to distinguish. In addition, we have found that character labels greatly enhance similarity-matching performance. While model weights can be improved, we believe that the value of such algorithms lies in their ability to capture variations and similarities.

References

Bilal Bataineh, Siti Norul Huda Sheikh Abdullah, and Khairuddin Omar. A novel statistical feature extraction method for textual images: Optical font recognition. *Expert Systems with Applications*, 39(5):5470–5477, April 2012. ISSN 0957-4174. doi:10.1016/j.eswa.2011.11.078. URL https://www.sciencedirect.com/science/article/pii/S0957417411016241.

Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. Deepsvg: A hierarchical generative network for vector graphics animation, 2020.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.

Stephen Coles. *The Geometry of type*. Thames & Hudson, 2016.

David L. Davies and Donald W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, 1979. doi:10.1109/TPAMI.1979.4766909.

Markus Diem, Florian Kleber, Stefan Fiel, Tobias Grüning, and Basilis Gatos. cbad: Icdar2017 competition on baseline detection. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 1355–1360, 2017. doi:10.1109/ICDAR.2017.222.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

Liangcai Gao, Zhi Tang, Xiaofan Lin, and Ruiheng Qiu. Comprehensive Global Typography Extraction System for Electronic Book Documents. In *2008 The Eighth IAPR International Workshop on Document Analysis Systems*, pages 615–621, September 2008. doi:10.1109/DAS.2008.30.

David Ha and Douglas Eck. A neural representation of sketch drawings, 2017.

Ju He, Jieneng Chen, Shuai Liu, Adam Kortylewski, Cheng Yang, Yutong Bai, Changhu Wang, and Alan Yuille. Transfg: A transformer architecture for fine-grained recognition. *arXiv preprint arXiv:2103.07976*, 2021.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi:10.1162/neco.1997.9.8.1735.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 448–456. JMLR.org, 2015.

Mohammed Javed, P. Nagabhushan, and B. B. Chaudhuri. Automatic detection of font size straight from run length compressed text documents. *arXiv:1402.4388*, 2014. doi:10.48550/ARXIV.1402.4388. URL https://arxiv.org/abs/1402.4388.

David Jury. History of Typography. In *Handbook of Research on Writing*. Routledge, 2007. ISBN 978-1-4106-1647-0. Num Pages: 36.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6114.

Raphael Gontijo Lopes, David Ha, Douglas Eck, and Jonathon Shlens. A learned representation for scalable vector graphics. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7929–7938, 2019. doi:10.1109/ICCV.2019.00802.

Monotype. Typography terms and definitions., January 2020. URL https://www.monotype.com/resources/studio/typography-terms. Section: Expertise.

Michael Murdock, Shawn Reid, Blaine Hamilton, and Jackson Reese. Icdar 2015 competition on text line detection in historical documents. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1171–1175, 2015. doi:10.1109/ICDAR.2015.7333945.

Ioannis A. Nellas, Sotiris K. Tasoulis, and Vassilis P. Plagianakos. Convolutional variational autoencoders for image clustering. In *2021 International Conference on Data Mining Workshops (ICDMW)*, pages 695–702, 2021. doi:10.1109/ICDMW53433.2021.00091.

Joep Pohlen. *Letter fountain : (on printing types)*. Taschen, 2011.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi:10.1007/s11263-015-0816-y.

Leo Sampaio Ferraz Ribeiro, Tu Bui, John Collomosse, and Moacir Ponti. Sketchformer: Transformer-based representation for sketched structure. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14141–14150, 2020. doi:10.1109/CVPR42600.2020.01416.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.

Yuto Shinahara, Takuro Karamatsu, Daisuke Harada, Kota Yamaguchi, and Seiichi Uchida. Serif or sans: Visual font analytics on book covers and online advertisements, 2019. URL https://arxiv.org/abs/1906.10269.

Nikita Srivatsan, Jonathan T. Barron, Dan Klein, and Taylor Berg-Kirkpatrick. A deep factorization of style and structure in fonts, 2019. URL https://arxiv.org/abs/1910.00748.

Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv:1905.11946*, 2019. doi:10.48550/ARXIV.1905.11946. URL https://arxiv.org/abs/1905.11946.

Robert L. Thorndike. Who belongs in the family? *Psychometrika*, 18(4):267–276, December 1953. ISSN 1860-0980. doi:10.1007/BF02289263. URL https://doi.org/10.1007/BF02289263.

Jason Tselentis, Allan Haley, Richard Poulin, Tony Seddon, Gerry Leonidas, Ina Saltz, Kathryn Henderson, and Tyler Alterman. *Typography, Referenced: A Comprehensive Visual Guide to the Language, History, and Practice of Typography*. Rockport Publishers, February 2012. ISBN 978-1-61058-205-6.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Zhangyang Wang, Jianchao Yang, Hailin Jin, Eli Shechtman, Aseem Agarwala, Jonathan Brandt, and Thomas S. Huang. Deepfont: Identify your font from an image, 2015. URL https://arxiv.org/abs/1507.03196.

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V. Le. Self-training with noisy student improves imagenet classification, 2019. URL https://arxiv.org/abs/1911.04252.

Yong Zhu, Tieniu Tan, and Yunhong Wang. Font recognition based on global texture analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10):1192–1200, October 2001. ISSN 1939-3539. doi:10.1109/34.954608. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

A. Zramdini and R. Ingold. Optical font recognition using typographical features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):877–882, August 1998. ISSN 1939-3539. doi:10.1109/34.709616. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.