ANALYZING POSTER COLLECTIONS USING AUTOMATIC SERIF CLASSIFICATION AND FONT SIMILARITIES

A PREPRINT

Syed Talal Wasim
Computer Vision Lab
Ecole Polytechnique fédérale de Lausanne
Lausanne, Switzerland
Department of Computer Vision
Mohamed bin Zayed University of Artificial Intelligence
syed.wasim@mbzuai.ac.ae

Romain Collaud
EPFL+ECAL Lab
Ecole Polytechnique fédérale de Lausanne
Lausanne, Switzerland
romain.collaud@epfl.ch

Lara Défayes
EPFL+ECAL Lab
Ecole Polytechnique fédérale de Lausanne
Lausanne, Switzerland
lara.defayes@epfl.ch

Nicolas Henchoz
EPFL+ECAL Lab
Ecole Polytechnique fédérale de Lausanne
Lausanne, Switzerland
nicolas.henchoz@epfl.ch

Mathieu Salzmann
Computer Vision Lab
Ecole Polytechnique fédérale de Lausanne
Lausanne, Switzerland
mathieu.salzmann@epfl.ch

Delphine Ribes
EPFL+ECAL Lab
Ecole Polytechnique fédérale de Lausanne
Lausanne, Switzerland
delphine.ribes@epfl.ch

ABSTRACT

Using a digital collection composed of more than 52'000 posters from different years, designers, topics, and clients, we study the feasibility of comparing posters based on their typographic features. To this end, we explore the possibilities of training a model to classify serif types without knowing the font and the 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

1 Introduction

Creating an interactive installation for a large digital poster collection, such as the one of the museum für Gestaltung in Zurich, containing tens of thousands of items from different years, cultures and themes, is a challenging task. To achieve this in an attractive and sustainable manner, we proposed, tested and validated the following hypothesis: Highlighting poster features by presenting the visitor a small number of visually or historically similar posters together increases emotional and cognitive connection towards the collection [11]. More specifically, the idea was to group and display posters according to certain features allowing a comparison.

In this work, we study this from the perspective of font characteristics. They spawn from, but are not limited to, font weight, font width, font contrast, x-height, serifs and sans, roman and italics (see figure Figure 2a). For the use case of the poster collection, we had access to a digital collection of 52'000 posters, with metadata such as the designer, the year and the client. However, no information with regards to the fonts was available. Moreover, posters are a very specific medium in which designers create and play with graphical elements, including fonts, to catch attention making the created font specific to a topic, a time or a designer. In this context, we therefore investigated the use of algorithms able to extract font characteristics from figurative content, i.e., posters. We focus this study on two particular font features, namely serifs and font similarities.

Figure 1: Samples from the poster collection in chronological order. The text and the associated typography have been chosen, positioned, created to catch people attention and to underline poster messages.

Serifs are the short appendixes found at the end of the main stems of some characters [17]. Serif fonts are descendant of humanist typefaces. Italian Humanists developed a style which mimic the strokes of a pen that enabled them to copy manuscripts more quickly than they could before. There are several typologies of serifs, whose evolution in form and use is linked to the history of type design and the resulting categories of fonts. Their distinction is thus not always obvious, and sometimes linked to an interpretation of their drawing. In our context, we focus mainly on their perceived shape and have grouped the different types of serifs into three different groups (see figure Figure 2b): 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". 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.

The second feature we investigate in this study is font similarities. It is defined here as the capacity of an algorithm to find fonts which apply the same 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 2), 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 therefore propose and evaluate an SVG-based Variational Auto-Encoder to measure font similarity. We also evaluate the feasibility of learninig similarity without knowing the character label (e.g., if it is a A, or a B).

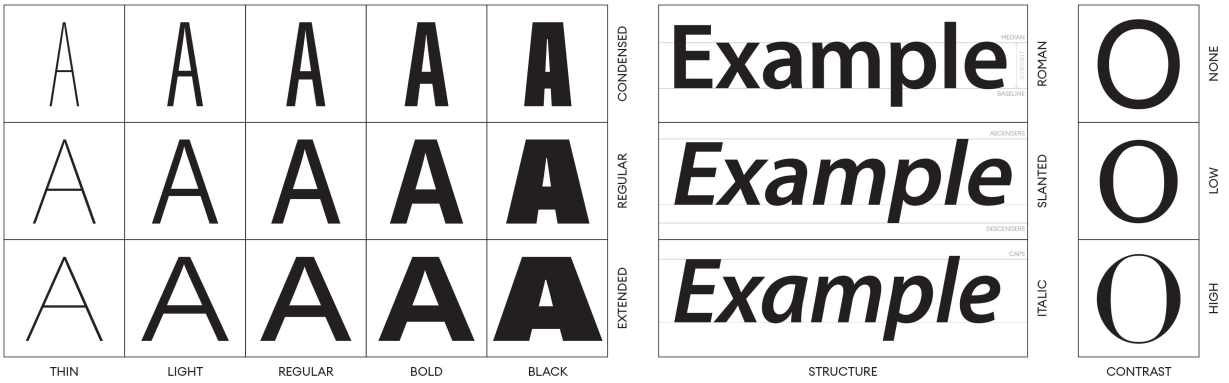
To evaluate our algorithms (serif and similarities), we created an annotated dataset composed of 24'366 images. Our results indicate that fine-grained recognition models provide the most effective framework for serif classification, and that knowing the character label helps in learning font similarities.

2 Related Work

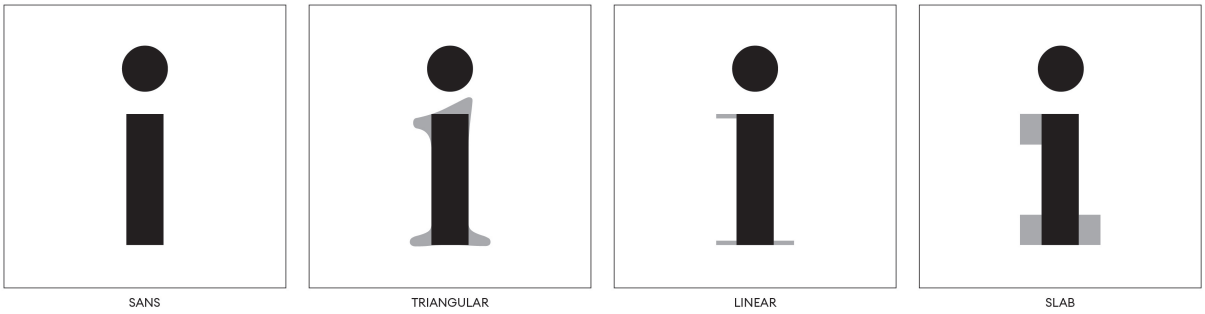
In this section, we review the literature from three domains: algorithms for typographic analysis, deep learning algorithms for image classification as they can be directly applied to the studied cases, and deep learning algorithms on SVG images.

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) [15] to build a latent space representation of the font structure as in [24]. 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 [27] or [14, 29, 13, 30, 1, 7], focus on font classification and font recognition rather than font feature analysis.



(a) Font features examples: weight, baseline, contrast, structure, ...



(b) Examples of serifs and sans serifs types. In the context of the project, we grouped them into three categories, triangular, linear and slab.

Figure 2: 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* [4] and *Letter Fountain* [19].

In addition, some methods have been proposed to extract font characteristics such as the text line [18] and the baseline [5]. Another method [23] 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 [20] benchmark 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 [25] (and its variants [28]), and Vision Transformers [6]. In addition, we describe methods to perform *fine-grained* image classification, targeting the recongition of visually-similar categories, such as different species of birds. Our choice of such methods was motivatied by the fact that serif features are small and difficult to distinguish.

EfficientNet [25] uses a network architecture scaling method which 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 [22] while adding squeeze and excitation layers.

The vision transformer [6], also an image classification model, is inspired from the Natural Language Processing domain state-of-the-art Transformer [26] model. 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 [9], which builds on top of the vision transformer [6] 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 the SVG format instead of raster format is particularly interesting in the typography domain as vector files are the format used by typographers. Mostly focused on the generation of vectorized sketches, the SketchRNN [8] used a Long Short-Term Memory (LSTM) [12] based VAE [15]. Recently, the Sketchformer [21] replaced the LSTM-based model with a Transformer [26] 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 [16], which used a one-stage autoregressive approach to generate path commands. By contrast, DeepSVG [3] 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. The DeepSVG architecture is shown in Figure 3.

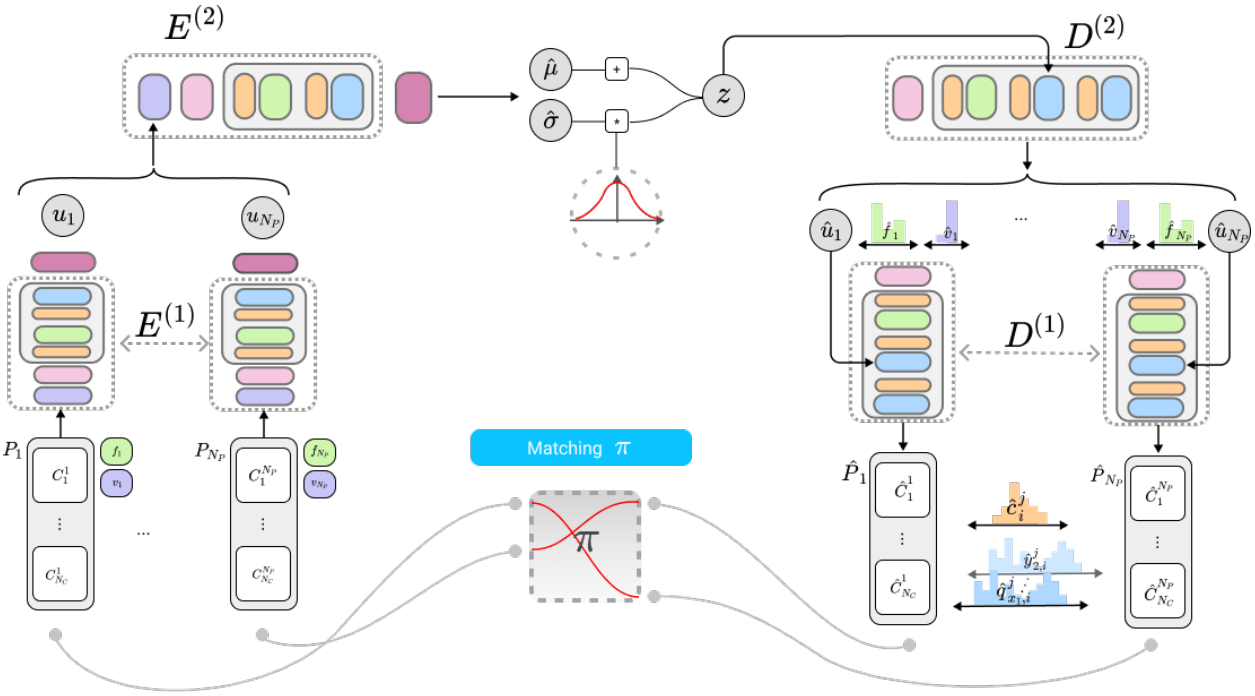


Figure 3: DeepSVG (Hierarchical Generative Network): Model overview. The input paths $\{P_i\}_1^{N_p}$ are encoded separately using the path encoder $E_{(1)}$. The encoded vectors are then aggregated using the second encoder $E_{(2)}$, which produces the latent vector z. The decoder $D_{(2)}$ outputs the path representations along with their fill and visibility attributes $\{(\hat{u}_i, \hat{f}_i, \hat{v}_i)\}_1^{N_p}$. Finally $\{\hat{u}_i\}_1^{N_p}$ are decoded independently using the path decoder $D_{(1)}$, which outputs the actual draw commands and arguments [3].

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. However, the font labels are not available for our use case of analyzing a large poster collection. 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.

3 Methodology

To associate posters according to some of their characteristics, 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. Therefore, 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 [6], with patch sizes of 16×16 (ViT B/16) and 32×32 (ViT B/32), respectively.

To test the accuracy of the models, we built a dataset with ground-truth font type and serif category. Our ultimate objective nonetheless is to use the resulting models on a poster collection where no annotated data on the font features are available. Training and validation datasets were built by first selecting a range of common fonts (including variations) for the four categories (sans-serif, triangular, linear, and slab). This was done in collaboration with a type designer. Afterwards, for each font variation, 62 images (26 uppercase, 26 lowercase, 10 digits) were created, making a total dataset of 24'366 images. 80% (19'492) of these images were used for training and 20% (4'874) for validation. To ensure that the model was actually learning the serif features and not over-fitting and memorizing font families, a font-independent test set was created where each font was a single variation from a unique font family. A total of 28 font variations were created, yielding a total of 1'736 images in the test set. The results were evaluated using accuracy measures across the three sets, along with precision, recall, and F1-score on the test set.

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 [24]. It enables the model to learn some specific features like the serif. However, this method lends a challenge, the necessary amount of font labeled data required (>10'000K fonts). Such a large amount of font labeled data was not available in our use case. 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 inspired by the DeepSVG [3] Hierarchical Generative Network discussed in section 2 with and without character labels to build a latent representation.

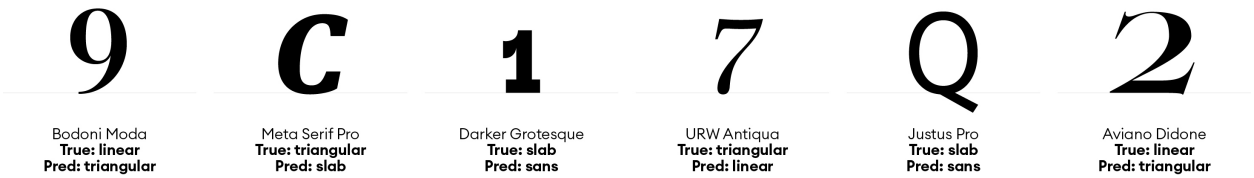
We used the same training dataset for this task as for the serif classification one, and trained two models, one knowing the character label and one without knowing it. The evaluation of the models are done based on their ability to group characters of the same serif type and of the same font family. To this end, we randomly sampled 50 of the same characters from the dataset for each serif type (sans-serif, linear, triangular, slab) and generated a t-distributed stochastic neighbor embedding (t-SNE) plot for each model from the latent vectors encoding of the characters.

4 Results

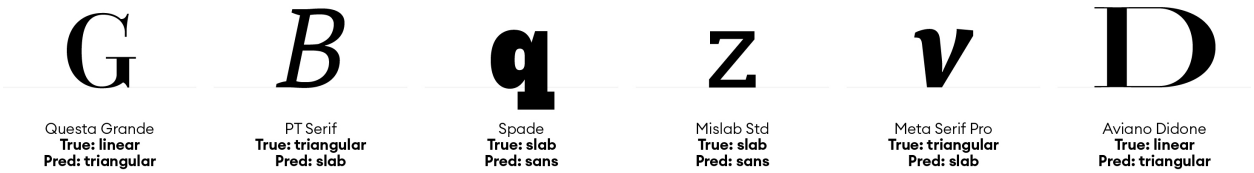
4.1 Classifying Serif Type

The accuracy of the serif models are calculated on the training, validation, and the font-independent test set described in section 3. The results are shown in Table 1a.

(a) Sample failure cases on the serif classification test set with TransFG (ViT B/16 backend)



(b) Sample failure cases on the serif classification test set with TransFG (ViT B/32 backend)



(c) Sample failure cases on the serif classification test set with EfficientNet-B2

Figure 4: Compared to TransFG, EfficientNet-B2 fails to classify serifs in several visually-obvious cases.

Model + Backend	Accuracies		
	Train	Validation	Test
TransFG ViT B/16 Backend	0.88	0.74	0.79
TransFG ViT B/32 Backend	0.97	0.75	0.89
EfficientNet-B2	0.98	0.96	0.69

(a) Training, validation and test results for serif classification

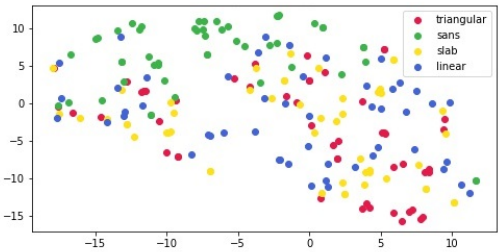
Model + Backend	Results on Test Set		
	Precision	Recall	F1-Score
TransFG ViT B/16 Backend	0.91	0.80	0.83
TransFG ViT B/32 Backend	0.93	0.89	0.91
EfficientNet-B2	0.80	0.69	0.73

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

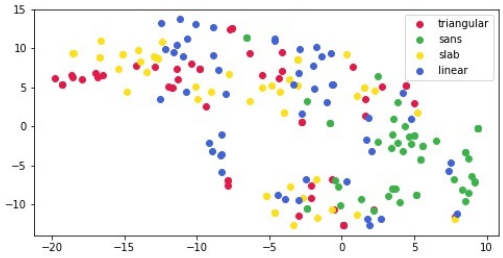
Table 1: Results for the serif classification task on the training, validation and font-independent test datasets.

The general image classification model (EfficientNet-B2) performs best on both the training and validation sets (accuracy of 0.98 and 0.96, respectively). However, it shows a significant accuracy degradation on the font-independent test (0.69). In contrast, the fine-grained image classification model, TransFG, performs comparatively worse on the training and validation sets but generalizes very well on the test set (0.97, 0.75 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).

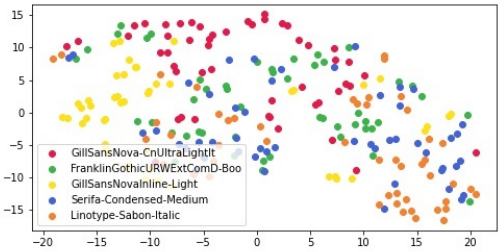
We also display failure examples on the test set in Figure 4a, Figure 4b and Figure 4c for the ViT B/16, ViT B/32, and EfficientNet-B2 models, respectively.



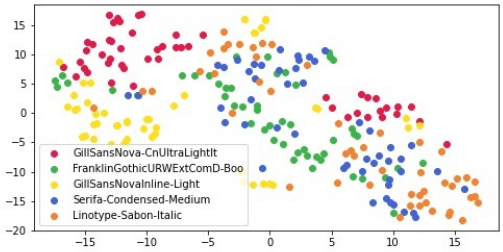
(a) t-SNE plots for 50 characters across different serif types, with unconditioned VAE trained for latent dimension size of $z = 64$.



(b) t-SNE plots for 50 characters across different serif types, with character-label conditioned VAE trained for latent dimension size of $z = 64$.



(c) t-SNE plots for 50 characters across different fonts selected randomly, with unconditioned VAE trained for latent dimension size of $z = 64$.



(d) t-SNE plots for 50 characters across different fonts selected randomly, with character-label conditioned VAE trained for latent dimension size of $z = 64$.

Figure 5: Results of the trained VAE on the SVG representation with and without character labels. The models, unconditioned and conditioned, are able to group sans serif and serif fonts but not to group the different types of serifs (see (a) and (b)). In (d), two GillSansNova variants are grouped next to each other which is less obvious in (c). Even if there is no clear clusters corresponding to each typeface, some groups appear especially in (d).

4.2 Typographic Similarity

The t-SNE plots for a latent dimension of 64 with and without conditioning on the character labels are provided in Figure 5d and Figure 5c, respectively. In short, these plots show that conditioning on the character label helps in learning meaningful representations for font similarities. We postpone the detailed discussion of these results to section 5

5 Discussion and Conclusion

5.1 Serif Classification

For the serif classification, 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 80.0% and an F1-score of 0.83. By contrast, the general image classification model, EfficientNet-B2, did not generalize well to the test set, with a font-independent test accuracy of 69.2% compared to a validation accuracy of 96.7%. This shows that this model tends to "memorize" the fonts rather than picking up the more subtle underlying serif features.

Looking at the failure cases (Figure 4) shows that most of the mistakes for the TransFG models occur when the serifs are either very subtle, or are special cases of serif font instances where the serif does not appear, as in the characters "e" and "o", and in the digits "0", "8" and "9". Other cases depict situations where the geometry of the serif and its surroundings

can cause confusion. For example, with the letter G on the left in Figure 4c, the design of the bottom of the letter may be related to a triangular serif. This is also the case for the letter D on the far right in Figure 4c. The linear serif is normally drawn straight but the slight slant is probably interpreted as a triangle. Nevertheless, observing the results evidences that the EfficientNet-B2 model fails on more visually obvious cases than TransFG. As a consequence, the classification of serif types is better modelled as a fine-grained classification task than as a general image classification problem, given the subtle nature of the serif features. 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 [6] backbone used in the TransFG model, which is computationally more expensive than the EfficientNet backbone.

5.2 Font Similarities

When it comes to exploring font similarities, inspecting Figure 5b reveals that the model conditioned on the character label can distinguish between the sans-serif and serif fonts, with all the sans-serifs grouped closely together compared to the others. However, the model is unable to differentiate between the various types of serifs. On the other hand, without character-label conditioning (Figure 5a), the differentiation is even less obvious with most of the sans-serif samples mixed in with the rest, and not appearing as a separate group. This shows that using the character labels is helpful to better differentiate between some serif types, but not sufficient to separate the more subtle differences.

In Figure 5 c) and d), we visualize different fonts (as opposed to different serifs in the previous plots) using t-SNE for the unconditioned and conditioned models, respectively. Both models learn some features and group characters from similar fonts together. For example, two GillSansNova variants are grouped close to each other when using the model trained with labels. The same can be observed in the unlabelled case, albeit less obvious. Altogether, although the t-SNE projections do not lead to clear clusters corresponding to the individual fonts, some groups nonetheless emerge, particularly with the features extracted from the character-conditioned model.

Figure 6: Letter segmentation failure case examples. In those cases, letters are embedded into the graphic design making the comparison based on typography not feasible.

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. Considering the font classification results, drawing inspiration from the fine-grained recognition literature could also be an interesting area for future research.

5.3 Conclusions

Although our experiments have led to promising results, much works remains to be done to be able to deploy our methods on the full poster collection for a public exhibition. In particular, one of the main difficulties we have faced is the inaccurate extraction of the letters on the posters and the lack of a graphical feature visualisation understandable by end-users [11]. For example, when looking at the posters in Figure 6, some letters are nested within the design making their segmentation very challenging. Moreover, posters can be composed of many fonts, increasing the difficulty to create understandable interactions within the context of the installation.

Nevertheless, the results of the fine-grained classification model allow, when letter segmentation is possible, to study in an automatic manner the evolution of typography on the criteria of the serif. In order to study in greater depth the influence of a period, a theme or a usage on the choice of typographer, it would be interesting to be able to make comparisons on a multitude of features separately. In addition to the one studied here, we think the following list is of greatest interest: weight, proportion, distortion, contrast and axis. Each of them being useful to understand the effect of an era or technology on its design [1] [2].

[1] Letter Fountain `https://www.letterfountain.com/extras/Timeline_typography_and_art.pdf`

[2] Calligraphy `http://calligraphy-expo.com/en/cognitivecalligraphy/tools-and-materials`

References

[1] B. Bataineh, S. N. H. S. Abdullah, and K. Omar. A novel statistical feature extraction method for textual images: Optical font recognition. *Expert Systems with Applications*, 39(5):5470–5477, Apr. 2012. ISSN 0957-4174. doi:10.1016/j.eswa.2011.11.078. URL https://www.sciencedirect.com/science/article/pii/S0957417411016241.

[2] A. Brock, S. De, S. L. Smith, and K. Simonyan. High-performance large-scale image recognition without normalization, 2021. URL https://arxiv.org/abs/2102.06171.

[3] A. Carlier, M. Danelljan, A. Alahi, and R. Timofte. Deepsvg: A hierarchical generative network for vector graphics animation, 2020.

[4] S. Coles. *The Geometry of type*. Thames & Hudson, 2016.

[5] M. Diem, F. Kleber, S. Fiel, T. Grüning, and B. 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.

[6] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

[7] L. Gao, Z. Tang, X. Lin, and R. Qiu. Comprehensive Global Typography Extraction System for Electronic Book Documents. In *2008 The Eighth IAPR International Workshop on Document Analysis Systems*, pages 615–621, Sept. 2008. doi:10.1109/DAS.2008.30.

[8] D. Ha and D. Eck. A neural representation of sketch drawings, 2017.

[9] J. He, J. Chen, S. Liu, A. Kortylewski, C. Yang, Y. Bai, C. Wang, and A. Yuille. Transfg: A transformer architecture for fine-grained recognition. *arXiv preprint arXiv:2103.07976*, 2021.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[11] N. Henchoz, A. Andrade, L. Défayes, A. Schneider, E. Groves, M. Salzmann, and D. Ribes. Poster world: bespoke ai meets curator expertise for public engagement. *Pages on arts and design, in Press, expected in December 2022*, 23, 2022.

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

[13] S. Ioffe and C. 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.

[14] M. 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.

[15] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In Y. Bengio and Y. 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.

[16] R. G. Lopes, D. Ha, D. Eck, and J. 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.

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

[18] M. Murdock, S. Reid, B. Hamilton, and J. 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.

[19] J. Pohlen. *Letter fountain : (on printing types)*. Taschen, 2011.

[20] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. 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.

[21] L. Sampaio Ferraz Ribeiro, T. Bui, J. Collomosse, and M. 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.

[22] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.

[23] Y. Shinahara, T. Karamatsu, D. Harada, K. Yamaguchi, and S. Uchida. Serif or sans: Visual font analytics on book covers and online advertisements, 2019. URL https://arxiv.org/abs/1906.10269.

[24] N. Srivatsan, J. T. Barron, D. Klein, and T. Berg-Kirkpatrick. A deep factorization of style and structure in fonts, 2019. URL https://arxiv.org/abs/1910.00748.

[25] M. Tan and Q. 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.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, u. Kaiser, and I. 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.

[27] Z. Wang, J. Yang, H. Jin, E. Shechtman, A. Agarwala, J. Brandt, and T. S. Huang. Deepfont: Identify your font from an image, 2015. URL https://arxiv.org/abs/1507.03196.

[28] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le. Self-training with noisy student improves imagenet classification, 2019. URL https://arxiv.org/abs/1911.04252.

[29] Y. Zhu, T. Tan, and Y. Wang. Font recognition based on global texture analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10):1192–1200, Oct. 2001. ISSN 1939-3539. doi:10.1109/34.954608. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

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