Co-funded by the
Erasmus+ Programme
of the European Union

ERASMUS MUNDUS MASTER IN
IMAGE PROCESSING AND COMPUTER VISION

PÁZMÁNY PÉTER CATHOLIC UNIVERSITY

université de BORDEAUX

UAM Universidad Autónoma de Madrid

MSc THESIS

Automatic Typography Analysis on Figurative Content

PRESENTED AT

UNIVERSITE DE BORDEAUX

université de BORDEAUX

Author: WASIM, Syed Talal

Academic Supervisor: DESBARATS, Pascal

Consultant (CVLAB, EPFL): SALZMANN, Mathieu
Consultant (EPFL+ECAL LAB): RIBES LEMAY, Delphine

EPFL

DATE: June, 2021

Master Thesis Proposal Form
(IPCV Programme)

Student data:

Name: Syed Talal Wasim	**Neptun ID:** A91SIO
UAM student ID: 426062	**UBx student ID:** 21924087
Cohort (academic year of starting the IPCV program): 2019	

Internship data:

Academic course: Master's Thesis 2021	
Company/Institution: Ecole Polytechnique Fédérale de Lausanne	
Starting date: February 15, 2021	**Ending date:** June 30, 2021
Internship consultant/responsible at the Company/Institution: • **Name:** Mathieu Salzmann • **Position:** Scientist	
Working place (full address): BC 309 (Bâtiment BC), Station 14, CH-1015 Lausanne, SWITZERLAND	
Workday info: • **Average hours per day:** 40 • **Working schedule (starting/ending hour or flexible):** flexible • **Total internship hours:** 752	
Salary (gross amount €/month): 0/month	

University data:

University[1]: UBx
Academic supervisor: • **Name:** Pascal Desbarats • **Department:** LaBRI • **Position:** Resp. master's in computer science IPCV course
Academic co-supervisor (if any): • **Name:** • **Department:** • **Position:**

Master Thesis description:

Title: Automatic Typography Analysis on Figurative Content
Introduction/Objectives[2]: The Museum of Gestaltung, in Zurich, has one of the biggest poster collections in the World. It has established a design research partnership with the EPFL+ECAL lab about visitors' engagement with this heritage. The collaboration will investigate how artificial intelligence can open new forms of representation, association and perception. We hypothesize that showing the construction structure of the posters will strengthen visitor engagement with the collection. To understand poster creation some key elements can be analysed such as the grid, the lines of force and the typography. In the last years many algorithms have been developed in order to automatically recognize the characters present in a text. However, little is known about automatic typographic analysis. It consists in recognizing the font style (e.g. serif, …), the variation (e.g. bold, italic, …), the decoration (e.g. Capitals. underline, …), the height of the letters and the font name (e.g. Helvetica). The proposed project aims at taking stock on the topic of automatic typographic analysis and at initiating the development of a framework to automatically extract typographic information. The project will start by finding and testing existing algorithms and evaluate their performances then initiating the technological transfer to the physical installation which will reveal the poster collection.
Workplan (list of tasks and their schedule)[2]: ● Task 1: Literature Review ○ Read and summarize literature on: ○ Text detection, segmentation and classification

[1] Underline as appropriate

[2] Expand as required

    - Font classification
    - Deep learning for SVG images
- Task 2: Compose the dataset
    - Extract text from poster collection (CRAFT algorithm)
    - Build a segmentation pipeline to generate binary masks
    - Extract individual characters, image trace and convert to SVG
- Task 3: Build an Encoder/Decoder model
    - Based on DeepSVG
    - For abstract latent space comparison
    - Read and understand the DeepSVG model
    - Modify as required and train the model on the custom dataset
    - Look into the possibility of transfer learning from the letter icons dataset used in the original paper
- Task 4: Identify specific Typography features
    - Read up on Typography features and understand significance of each.
    - Define a set of features to be recognized based on importance.
    - Using traditional image processing techniques, build pipelines to extract those features.

Tentative Schedule:

Task 1: February and March 2021 (Mostly in February. Might continue in March in case of some specific topics coming up later)

Task 2: February and March 2021

Task 3 & 4: March, April and May 2021 (The two tasks will run in parallel considering that the Segmentation/Deep SVG models would take a significant amount of time to train)

Testing of implementation and Consolidation of Results: May 2021

SIGNATURES:

Hereby I undertake to supervise the Master's thesis of the student.

Place and date: Lausanne, February 25, 2021

Signature of Company/Institution consultant or responsible

Hereby I undertake to supervise the Master's thesis of the student.

Place and date: Talence, 25/02/2021 .....

.....

Signature of University supervisor

Hereby I apply for the approval of the topic of my Master's thesis and declare that I wish to take the final exam and defend my thesis at the following University[3]:

Pázmány Péter Catholic University

Universidad Autónoma de Madrid

Université de Bordeaux

Place and date: Bordeaux, France, 25/2/2021

Signature of student

[3] Underline as appropriate

The topic of the Master's thesis has been approved following the rules and agreements required by the Partner University under the scope of the IPCV Programme.

Place and date: Talence, 25/02/2021

.....

IPCV local coordinator

Declaration of Authenticity

I, the undersigned Syed Talal Wasim, student of the Image Processing and Computer Vision MSc program, hereby certify that this thesis has been written without any unauthorized help, solely by me, using only the referenced sources. Every part quoted in whole or in part is indicated clearly with a reference.

I declare that I have not submitted this thesis in any other higher education institution, excluding the partner universities of the IPCV Consortium.

.....

Syed Talal Wasim

Abstract

The automatic detection of typographic features is of immense importance for designers because it can help reveal interesting insights regarding pieces of art such as historical posters. Since typographic features are numerous, this thesis focuses on four of them, namely calculation of contrast, classification of serifs, associating character based on typographic similarity, and developing feature descriptors that can encode the notion of typographic complexity.

Firstly, for contrast calculation, a multi-step algorithm is introduced which exploits the inherent geometric nature and infinite scalability of vector graphics. It achieves highly accurate results on a dataset of varied fonts and characters, with a mean error of 39% and a minimum error of 0.3%. The mean error goes so high because the model fails with $< 600\%$ error in outlier cases where the character boundary is not a simple polygon (which is an assumption of the algorithm).

Secondly, for serif classification, both general and fine-grained image classification models are evaluated. It is found that while the general image classification model (EfficientNet-B2) achieves good accuracy on the train and validations sets ($\sim 96\%$), it fails to generalize to a font independent test (achieving an accuracy of only 69%). While the fine-grained recognition model is comparatively worse on the validation set ($\sim 80\%$), it achieves $\sim 90\%$ accuracy on the test set.

Thirdly, on the topic of similarity, an SVG-based Variational Auto-Encoder is proposed that uses only character labels to learn different font features. The resulting model is able to differentiate sans-serif vs serif fonts and group together fonts of the same family. However, it is not able to differentiate between different kinds of serifs and struggles to differentiate between different families of fonts.

Finally, for typographic complexity, four description vectors are proposed which attempt to aggregate the highly abstract notion of typographic complexity as described by design experts. The feature descriptors use various statistics (like mean, max, std-deviation, a combination of all) on SVG image features (like the number of points and

paths). Among the four descriptors, there is no clear indication for which one is the best because responses from the design experts are mixed. However, some trends are noticed that show that feature descriptors working with maximum and a combination of all statistics tend to be the best while the one working with standard deviation is rejected by all designers.

To conclude, an attempt is made towards the automatic analysis of four typographic features. For the contrast and serif classification, highly accurate results are achieved. For similarity, results look promising, but training and evaluation with larger datasets is clearly required. Finally, the highly subjective nature of complexity makes it difficult to formulate and evaluate. However, an attempt is made to build a global descriptor for typographic complexity.

Acknowledgements

I wish to thank my supervisors Dr. Pascal Desbarats (Université de Bordeaux), Dr. Mathieu Salzmann (CVLAB, EPFL) and Delphine Ribes Lemay (EPFL+ECAL Lab) for their constant guidance and support throughout the project, and their help in reviewing this thesis. Special thanks to Romain Collaud, for his expert advice on different aspects of typography and fonts, along with labelling the ground truth dataset for the font contrast calculation. To Céline Dupuis and André for providing support in generating appropriate results and datasets. And to Yves Kalberer for helping in setting up computation systems at lab for me to access remotely.

Contents

Chapter 1

Introduction

The digitalization and digital transformation taking place in today's world have impacted all aspects of society. One of the most important aspects of this transformation is the concept of limiting usage of paper or, as it is sometimes called, "going paperless" [1]. This has impacted various aspects of society and culture, not the least of which is art [2]. Among those affected by this transformation in art, is the Museum Für Gestaltung in Zürich, which owns a digitized collection of posters that have been collected since the opening of the Museum in 1875 [3].

The Museum has established a partnership with the EPFL+ECAL lab to pursue a design research project on this digitized collection of posters. The overall aim of this research project, known as the **Poster World** project, is to valorize and reveal hidden aspects and treasures in this collection. One important aspect of this project is to form a basis for selecting and presenting posters to a viewer in a meaningful manner. While classic methods focusing on the metadata (designer, year, etc.) already exist, there is a need for more innovative methods that capture the interest of the visitor who views the collection. In that regard, the project aims at extracting various other aspects regarding the content of the posters. One such aspect is the Typography of the written content.

Typography is defined as "the art of arranging type which makes written language readable, legible, and appealing", involving the selection of various aspects such as typefaces, line-spacing, letter-spacing, point-sizes, etc. [4]. The topic of this thesis concerns "Automatic Typographic Analysis" of the text in the poster collection. The aim is to develop tools using methods in Computer Vision and Machine Learning, in general, to extract concrete and abstract typographic features from the posters and then build associations of posters that are semantically meaningful.

1.1. History and Importance of Typography

The word "Typography" comes from the Greek roots of *typos* and *graphia*, which mean "impression" and "writing" respectively [5]. Early examples of typography date back to the Minoan Bronze Age, like the "Phaistos Disk" [6]. However, the modern "Movable Type" was not invented until the 11^{th} century in China during the Song dynasty [7]. The initial versions at that time were manufactured using ceramics and clay. These were followed by wooden and metal ones in the 12^{th} [8] and 13^{th} century [9] respectively. However, the modern lead-based movable type (and later the mechanical printing press) was not invented until 1439 and is attributed to Johannes Gutenberg [10].

In typography, there are three main focus points, namely, *legibility*, *readability*, and *aesthetics*. *Legibility*, as defined by Walter Tracy, is the "the quality of being decipherable and recognizable". It is a question of how decipherable are the *individual characters and letters*. On the other hand, *readability* is defined as how easy the text is to read *overall* [11]. Finally, the consideration of harmonizing typefaces and layouts which produce a tasteful and appealing display is known as the concept of *aesthetics* [10].

The reason why typography is an important consideration is that it has strong associations with different societal, cultural, and historical aspects. Therefore, the analysis of typography has the potential to reveal such hidden aspects of the poster collection. For example, typography can sometimes represent national and cultural identities. Since typography precedes nations, it does not *necessarily* convey national identities. However, because nationalist sentiments developed on the back of print language, different cultures have over time adopted various typographic features [12][13].

1.2. The Poster Collection

The Museum Für Gestaltung in Zurich has a collection of nearly 120,000 posters. A subset of 1500 posters from this collection is made available for analysis in this thesis. Figure 1.1 displays a sample of the collection.

1.3. Summary of Tasks Performed

While typographic features are numerous (as detailed in chapter 2), this thesis focus on four of them, which were selected based on discussions with the designers at EPFL+ECAL

Lab. The four features and the associated tasks are listed below:

- Contrast: Given an image of a character, can the contrast be accurately calculated?
- Recognizing Serifs: Given a dataset of images of characters of different Serifs types (San-Serif, Triangular, Linear, and Slab), can a model be built to accurately predict the Serif type?
- Typographic Similarity: 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?
- Typographic Complexity: Given a dataset of posters with typographic content, can a representation of typographic complexity be built, which could be used to associate posters of similar complexity?

1.3.1 Methods Used

Contrast Calculation: For calculating the contrast, a multi-step geometrical algorithm is proposed that exploits the inherent infinite scalability of Vector graphics. The algorithm is evaluated thanks to a dataset annotated by a type font expert.

Recognizing Serifs: Since this is a classification task, we applied state-of-the-art approaches for classification. The serifs are a subtle feature, both general image classification and fine-grained image classification approaches are evaluated. The models are evaluated on a custom dataset created using common font families.

Typographic Similarity: While work has been done on building latent representations for similarity using raster images, they either fail to capture font features or use conditions on both fonts and characters (Which are not available for datasets of scene text). A Variational Auto-Encoder (VAE) based on Vector graphics is proposed instead which uses only character labels to build meaningful representations with limited data.

Typographic Complexity: Typographic complexity is a highly abstract concept that can vary from one designer to another. Within this project, an attempt is made to capture this concept using feature descriptors for the poster collection that use the shape information from the vector graphic representation in an attempt to encode the notion of typographic complexity.

1.3.2 Datasets Created

Fonts Dataset: The dataset was built by first selecting a range of common fonts by consulting a typography designer at the EPFL+ECAL lab. Afterward, for each font 62 images (26 uppercase, 26 lowercase, 10 digits) were extracted, making a total dataset of size 24366 images, labeled for the character, font, and serif type. All images were produced in both Raster and Vector format. A further font independent test set (for serif classification) was created where each font was a single variation from a unique font family. A total of 28 font variations were used which made a total of 1736 images in the test set.

Poster Dataset: A dataset of characters from the poster set was also created by first segmenting individual characters from the posters, and then converting them to Vector format.

1.4. Outline

Summary of the chapters of the thesis work:

Chapter 2: Typography This chapter presents an overview of common typographic features used by designers when analyzing fonts. The presented features are not all necessarily the focus of this thesis. Instead, it is meant to serve as a comprehensive summary of important typographic features

Chapter 3: Related Works The chapter presents a summary of the related literature to the topic in focus. All relevant literature is covered including typographic analysis, text detection and segmentation, deep learning for image classification, and deep learning on vector graphics.

Chapter 4: Methods This chapter describes the methodology used for each task performed, including the datasets used and the metrics focused on for evaluation.

Chapter 5: Results This chapter presents the results and analysis of the various methods evaluated.

Chapter 6: Summary This chapter summarizes the tasks and the results achieved for each, along with future directions of research.

(a) (b)

(c) (d)

Figure 1.1: Sample posters from the poster collection at the Museum Für Gestaltung in Zurich

Chapter 2

Overview of Typographic Features

This chapter provides an overview of the various typographic features that are used by designers when analyzing a particular font. Note that the features mentioned are not necessarily focused on in the thesis at present, but are mentioned here to ensure a broad and comprehensive overview of typography. The graphics and examples mentioned are obtained from two major publications on the topic, namely the *The Geometry of Type* [14] and *Letter Fountain* [15].

2.1. Typographic Features

This section covers the major typographic features. These can be divided into four major concrete categories (as explained by a type designer at the EPFL+ECAL lab) namely, Construction, Style, Terminal, and Typesetting. Finer details of each category are explained in the sub-sections below. Furthermore, there are a couple of more abstract features that are also considered by designers, which would be also discussed briefly.

2.1.1 Construction

The construction category comprises three main features: Line terms, Structure, and Spacing.

Line Terms

Line terms are imaginary lines that define the positioning of the font. These include the Ascent, Descent, Median, Base, and Cap. The specific position of these lines can be seen in Figure 2.1, with a brief description given below.

- Ascent: The topmost line where the upstroke of the lowercase characters should reach.
- Descent: The lowest point for lowercase characters that go below the baseline.
- Base: The lowest point for all uppercase and most lowercase characters.
- Cap: The height to which the uppercase characters should reach.
- Median: The height to which most lowercase characters should reach (except those like 'd' and 'i' that have a longer upstroke or a tittle).

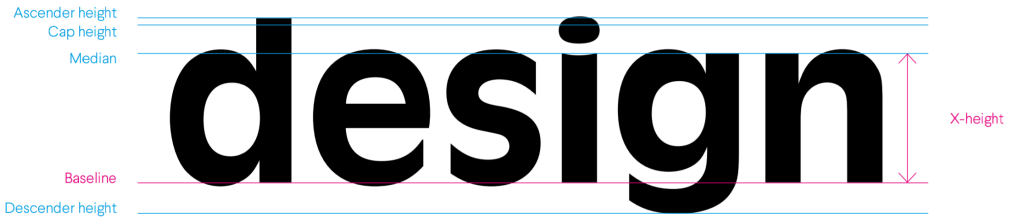


Figure 2.1: Typography Line Terms: Imaginary lines that define font positioning

Structure

The structure of the font categorizes how slanted it is. This can fall into three categories. Romans, Obliques, and Italics. See Figure 2.2for examples along with the details as follows.

- Romans: Straight characters without any slanting.
- Obliques: Forward or back-slanted versions of the Romans
- Italics: These include some simple obliques as well as designs that mimic cursive writing.

Spacing

The spacing between characters can be either Proportional or Monospace as shown in Figure 2.3. Each is briefly described below:

- Proportional: The space occupied by each character is different and spacing is adjusted accordingly
- Monospace: Each character occupies the same fixed space and the character structure is adapted accordingly

Figure 2.2: Typography Structure: From top to bottom Roman, Oblique and Italic structures

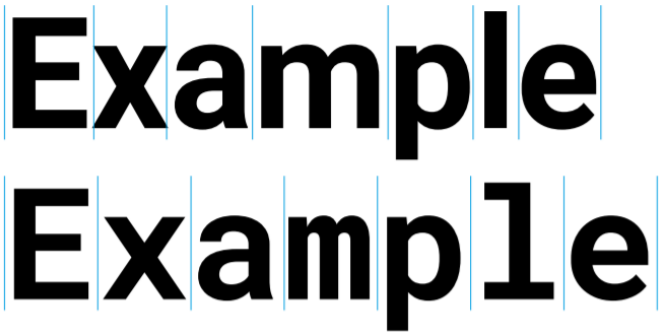


Figure 2.3: Typography Spacing: From top to bottom Proportional and Monospace spacing

2.1.2 Style

The style category comprises the weight, proportion, distortion, contrast, and axis of the font.

Weight

The weight of the font is defined as the thickness of the character outlines relative to the height of the font. These come in various thicknesses with the major ones being (in increasing order of thickness), Thin, Light, Regular, Medium, Bold, and Black, as shown in Figure 2.4.

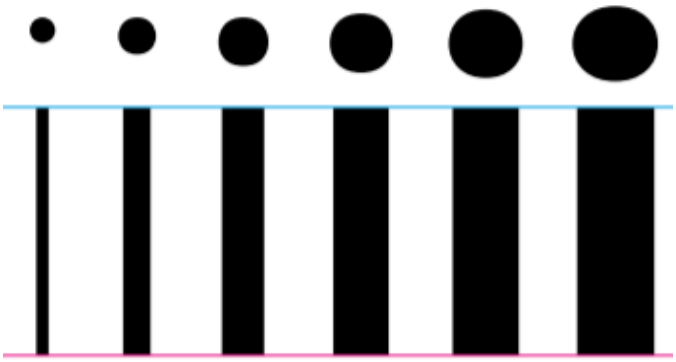


Figure 2.4: Typography Weight: From left to right Thin, Light, Regular, Medium, Bold and Black weights

Proportion

The proportion refers to the width of a character relative to the height. It can be of three types Condensed, Regular and Extended, in the order of increasing width. See Figure 2.5 for a visual example.

Figure 2.5: Typography Proportion: From left to right, Condensed, Regular and Extended proportions

Distortion

The distortion in a font is the variation in the width of the stem relative to the thickness of the rest of the font. Distortion is defined as a percentage, and an example of 80%, 100% (no distortion), and 120% can be seen in Figure 2.6.

Figure 2.6: Typography Proportion: From left to right 80%, 100% (no distortion), and 120% distortion

Contrast

In typography, contrast denotes the ratio between the thickest and thinnest strokes. See Figure 2.7.

Figure 2.7: Typography Contrast: Examples of different levels of contrast

Axis

The axis is an imaginary line drawn from top to bottom in a character which indicates the angle of stress when strokes of various thickness are used. A straight line indicates zero stress while a left or right-angled axis indicates positive or negative stress respectively, as shown in Figure 2.8.

Figure 2.8: Typography Axis: Examples of different levels of contrast

2.1.3 Terminal

The terminal category considers the form of the ending strokes of a font, which can be Sans-Serif or Serif. Sans-Serif fonts don't end in Serif strokes as shown in Figure 2.9.

Figure 2.9: Typography Sans-Serif: Example of a sans-serif font

Fonts ending in a Serif can have three types, namely Triangular, Linear and Slab. Examples can be seen in Figure 2.10.

(a) Triangular

(b) Linear

(c) Slab

Figure 2.10: Typography Serifs: Examples of different Serif types

2.1.4 Typesetting

Typesetting is the art of arranging text to make it ready for printing. It includes the concepts of Letter-spacing, Orientation, and Lowercase/Uppercase representations.

Letter-Spacing

Letter-spacing can be normal, narrow, or wide. Examples of the three can be seen in Figure 2.11.

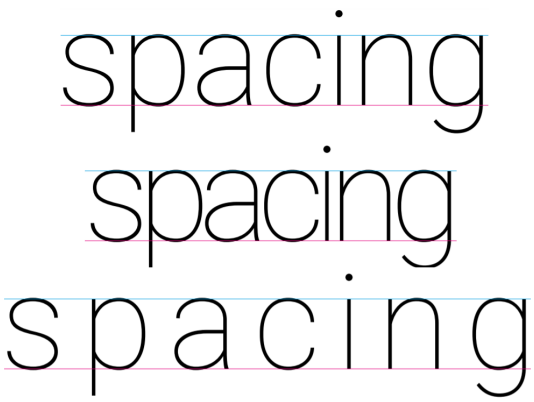


Figure 2.11: Typography Letter-spacing: Top to bottom Normal, Narrow and Wide spacing

Orientation

Orientation can also be of three types, namely horizontal, vertical and curvilinear, as shown in Figure 2.12

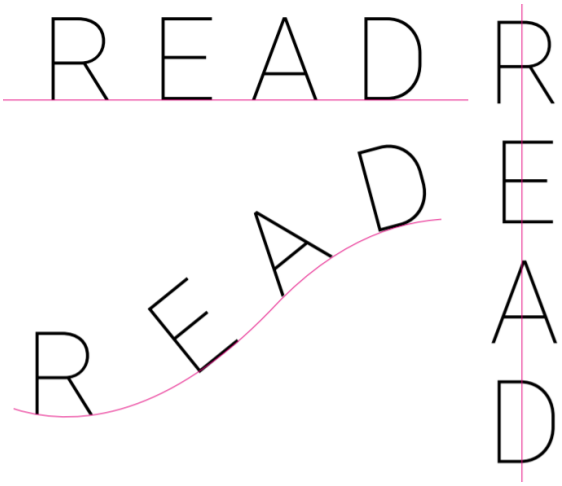


Figure 2.12: Typography Orientation: Horizontal (Left-Top), Curvilinear (Left-Bottom) and Vertical (Right)

Lower and Uppercase

Lower and uppercase characters can be represented in four ways: first letter capitalization, all lowercase, all caps and small caps. These are shown in Figure 2.13

Figure 2.13: Typography Lower and Uppercase: first letter capitalization (Left-Top), all lowercase (Right-Top), all caps (Left-Bottom) and small caps (Right-Bottom)

2.1.5 Abstract Features

Abstract features are those that are neither strictly defined like the Serifs and Capitalization, nor strictly quantifiable like the Contrast and Weight. These features may have a subjective interpretation that varies from one designer to another. Two such features

are covered here, namely “Typographic Similarity” and “Typographic Complexity”.

Typographic Similarity

In some ways, the typographic similarity is the more concrete of the two. The idea being that can a latent representation be defined (whether through hand-crafted features or some other method) in which characters of the same Font (sharing same typographic features) would be closer and those with different features would be further away? Such a representation would be independent of the structure of character, whereby grouping characters of the same “font” together rather than grouping together the same characters regardless of the font.

This was the focus of most of the research up till now in typographic analysis with works like [16], [17] focusing on using Variational Auto-Encoders (both with and without conditions) to learn a latent space representation for the characters.

Typographic Complexity

The complexity of a typeface is a highly abstract concept that can vary according to several parameters. Since this is a subjective concept with no strict definition, the details mentioned here are a summary of the discussions on the topic with a typography design expert at the EPFL+ECAL lab. Basically, the concept of complexity covers both local and global parameters and the variation in both contributes to the overall notion of complexity of a typeface. For example, consider Figure 2.14, which shows the difference in single (low complexity) and double (high complexity) story representations of lowercase characters.

Figure 2.14: Typographic complexity at a local (character) level: single and double story representation of sample lowercase characters [a]

[a] https://www.quora.com/Why-is-the-shape-of-the-letter-%E2%80%9Ca%E2%80%9D-in-computer-fonts-different-from-its-handwritten-version

Similarly, consider Figure 2.15, which gives an example of a more global notion of complexity. The word at the bottom has a visually simpler design compared to the one at the top. However, it should be noted that in the end this is quite an abstract notion. While the examples shown are some simple examples with easily discernible features, the overall notion can be quite subtle, with differences that need an expert to identify.

Figure 2.15: Typographic complexity at a global (word) level: Example of a simple (bottom) and complex (top) representations of the same word[a]

[a] https://www.webdesign.org/where-the-web-typography-trend-is-going-in-2016.22607.html

Chapter 3

Related Works

This chapter focuses on the relevant literature of this thesis. The literature reviewed focuses on four sections. The first is on typographic analysis, looking for any existing methods in this area. The second is on text extraction covering methods focusing on both text detection and segmentation. The third focuses on the current state-of-the-art in image classification, because of its direct applicability towards some aspects of typographic feature classification. The fourth and final section covers literature related to deep learning on SVG images.

3.1. Typographic Analysis

In the domain of automatic typographic analysis, there is a lack of literature. A couple methods focus on a designing a Variational Auto-Encoder [18] to build a latent space representation of the font structure. Favre [16] attempted this on a dataset of characters extracted from posters. They did not have any information about the character or the font type, but instead employed a special loss function imposed on distance between letters of the same word in latent space. Their work failed to achieve any meaningful representation, with latent space grouping characters together, without any notion of similarity in terms of typography. Another work [17], used fonts from the google fonts database. Since they knew both the character and the font, the auto-encoder was conditioned on both parameters. They achieved much better and more meaningful results (with the latent spacing grouping together characters from similar fonts), but had the constraint of requiring both the knowledge of the character and the font type. Other than these two the available literature primarily focuses on analysis of the font and not the typography, with methods focusing on font classification like the Adobe DeepFont

[19] and some works on font size recognition in documents like [20].

Recently, some attempts have been made towards features that are primarily related to document text but can have applications towards typography. Such as text line detection [21] and baseline detection [22]. One method [23] attempted at build a feature representation of fonts, but instead focused on categories more appropriate to print types like "historical" and "fancy" rather than typographic feature categories.

3.2. Text Extraction

This section summarizes the literature regarding text extraction, focusing both on detection/localization and segmentation. The majority of the works found on this topic focus primarily on text detection and localization, with a lack of literature on text segmentation.

3.2.1 Text Detection

Methods in text detection are numerous and can be categorized by their application area [24], namely *long text*, *multi-oriented text* and *irregular text*. The current state-of-the-art for each category is shown in Table 3.1.

The current method that performs best across all three categories is Character Region Awareness for Text Detection (CRAFT) method which has state-of-the-art results across all three major text detection datasets (ICDAR 2013 [34], ICDAR 2015 [35], ICDAR 2017 [36]).

3.2.2 Text Segmentation

When it comes to text segmentation, there is lack of both methods and datasets in the domain. Extensive search for literature revealed three recent works that achieved state-of-the-art results on the available datasets. Two end-to-end deep learning methods and one based refinement method using the CRAFT detector as a pre-processing step.

Al-Rawi et al. [37] proposed a segmentation method based on the DeepLabV3+ [38] model. Their method achieved second best results on the segmentation challenge of the ICDAR 2013 dataset [34] and state-of-the-art on the KAIST dataset [39]. More recently, Xu et al. [40] propose a "Text Refinement Network" (TextRNet), which uses a DeepLabV3+ [38] or HRNet [41] semantic segmentation model as a backbone, and

Method Type	Current State-of-the-Art
Long Text	Detecting Oriented Text in Natural Images by Linking Segments [25] R2CNN: rotational region CNN for orientation robust scene text detection [26] Multi-Oriented Scene Text Detection via Corner Localization and Region Segmentation [27]
Multi-oriented Text	Geometry-Aware Scene Text Detection With Instance Transformation Network [28] Rotation-Sensitive Regression for Oriented Scene Text Detection [29] EAST: An Efficient and Accurate Scene Text Detector [30]
Irregular Text	Character Region Awareness for Text Detection [31] Look More Than Once: An Accurate Detector for Text of Arbitrary Shapes [32] Efficient and Accurate Arbitrary-Shaped Text Detection With Pixel Aggregation Network [33]

Table 3.1: Text detection state-of-the-art

applies key feature pooling and attention to improve the segmentation maps. They achieve state-of-the-art accuracy on ICDAR 2013, COCO_TS [42], MLT_S [43] and their own introduced dataset TextSeg [40].

It is important to note that most of these datasets were introduced many years ago and are very small in size to be sufficient for the latest deep learning methods. ICDAR 2013 segmentation set has only 230 images, while KAIST (released in 2010) has 300 english and 1071 korean text annotated images. While COCO_TS and MLT_S (both released in 2019) provide a much larger set at 14690 and 6896 images respectively. However, their pixel-level annotations are not accurate, having been done automatically through a weakly-supervised method. Finally, the TextSeg dataset, released recently in 2021 contains 2646 training images. Therefore, due to a lack of large, accurately labelled datasets, text segmentation has not seen a lot of applications of modern deep learning

methods.

Another method was proposed by Favre in their undergraduate thesis [16]. They started with text detection results from CRAFT [31], followed by a skew correction and connected component based segmentation step. The results achieved seem reasonable on visual inspection, but fail to segment joint text and very irregular shaped text. The author does not provide accuracy on any benchmark dataset.

3.3. Deep Learning for Image Classification

This section on image classification focuses on two aspects. On one hand it covers the state-of-the-art on the standard Image Classification methods such as those on the Imagenet [44] benchmark. Furthermore, the methods concerning the task of *Fine-Grained* Image Classification are also covered in this section. Some typographic features are hard to distinguish and hence, fine-grained classification methods may be the more appropriate method for them. For image classification the current state-of-the-art mainly include EfficientNets [45] (and its variants [46]), NFNets [47] and Vision Transformers [48].

EfficientNet introduced a network architecture scaling method that scales all the dimensions (depth/width/resolution) with what is called a *compound coefficient*, which is unlike conventional practice of arbitrary scaling. The model is based on inverted bottleneck and linear residual blocks of the MobilenetV2 [49] while adding squeeze and excitation layers. The resulting architecture scaling method is shown in Figure 3.1.

The problem with Effientnets (and most vision models in general) was the usage of the batch normalization [50] layer. Batch normalization has some significant advantages. It eliminates mean-shift in the activations and acts as a regularizer. It also allows for training with larger batch sizes and training rates without overfitting. However, all is not good about batch normalization. Computing batch-level statistics for the normalization operation is a computationally expensive task and breaks the assumption of data independence of the samples in the mini-batch. To counter this issue NFNets were proposed. NFNETs tackled the challenge of training large models without batch normalization which significantly improves training time and computation requirements. To counter the unstable training of models without batch normalization, the authors propose an adaptive gradient clipping strategy and train a set of normalizer-free ResNets [51] that match in performance the EfficientNet-B7 variant. Since it is crucial to suppress the scale of the

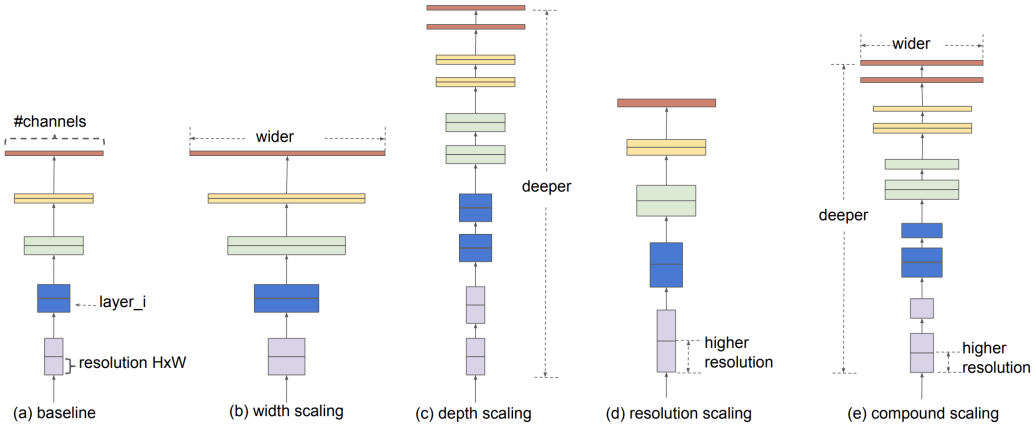


Figure 3.1: EfficientNet Model Scaling. (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio [45].

activations on the residual branches, the authors modify the standard ResNet bottleneck by introducing two scalar parameters α and β to scale activations at the start and end of the bottleneck. The modified block can be seen in Figure 3.2.

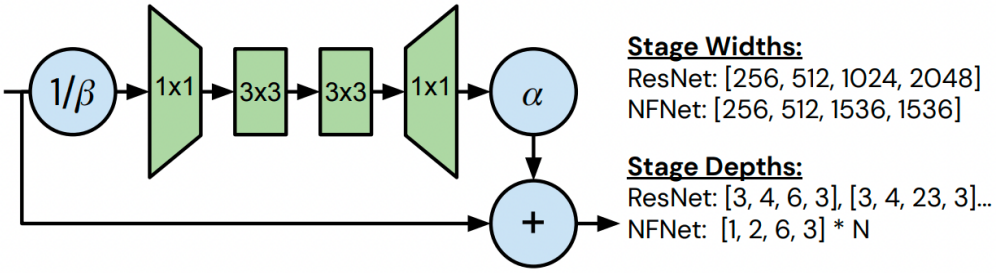


Figure 3.2: Summary of NFNet bottleneck block design and architectural differences. [47].

The vision transformer [48] is an image classification model inspired from the Natural Language Processing domain state-of-the-art Transformer [52] model. It is applied to patches of images. 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. An overview of the model is presented in Figure 3.3.

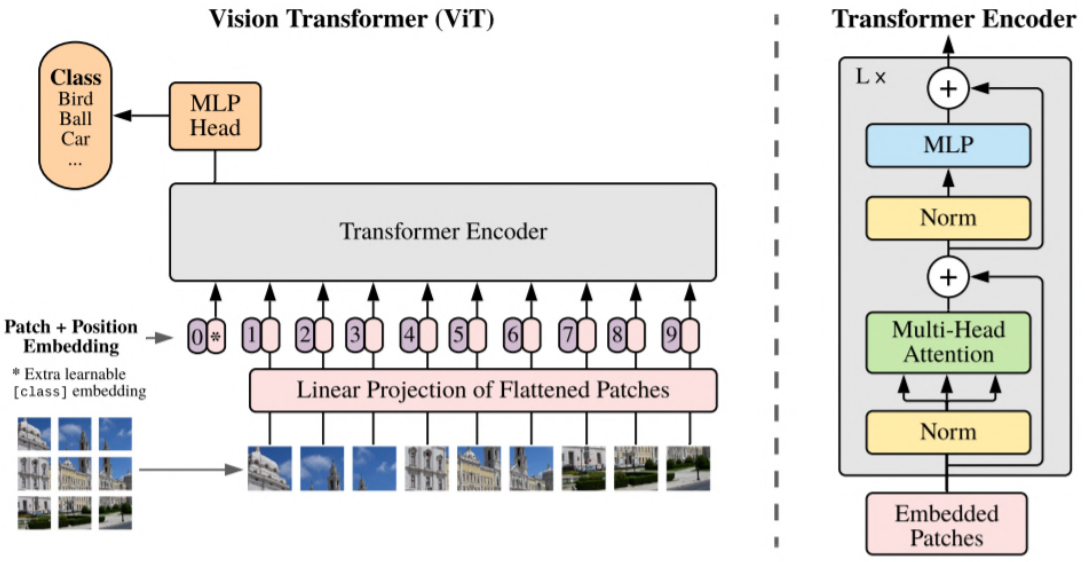


Figure 3.3: Vision Transformer Model overview. An image is split into fixed-size patches, linearly embedded each of them with added position embeddings, and then fed to a standard Transformer encoder. To perform classification, an extra "classification token" is added to the sequence [48].

The current state-of-the-art in Fine-grained image classification is the TransFG model [53] which builds on top of the vision transformer [48] by employing a "parts selection module" to localize the regions with the most distinctive features. It then uses a contrastive loss on top to improve the discrimination between regions. They achieve state-of-the-art accuracy on three major datasets, Caltech-UCSD Birds-200-2011 [54], Stanford Dogs [55], NABirds [56] and currently rank second best on the iNaturalist Species [57]. An overview of the model is presented in Figure 3.4.

The reason why models like TransFG (and other fine-grained recognition models in general) are interesting to consider for the task at hand is that they are able to learn 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, which are found in specific positions on different characters and quite small in size compared to the overall font.

3.4. Deep Learning for SVG Images

Unlike raster images, deep learning on vector graphics is a domain which has not received extensive attention. Mostly focused on the generation of vectorized sketches,

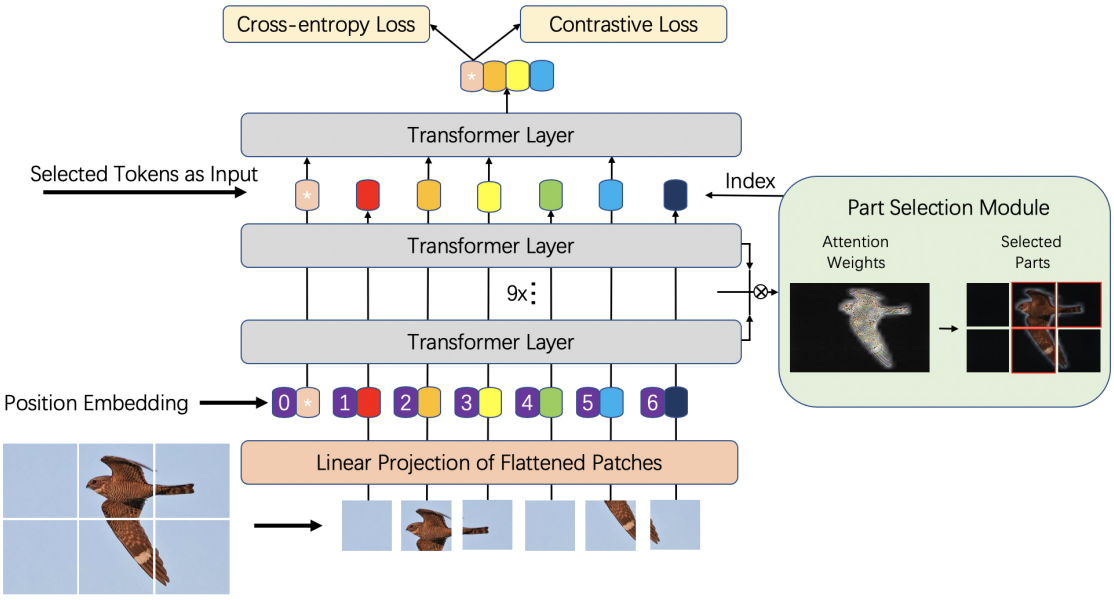


Figure 3.4: TransFG Model overview. Images are split into small patches (a non-overlapping split is shown here) and projected into the embedding space. The input to the Transformer Encoder consists of patch embeddings along with learnable position embeddings. Before the last Transformer Layer, a Part Selection Module (PSM) is applied to select tokens that corresponds to the discriminative image patches and only use these selected tokens as input. Cross-entropy loss and contrastive loss on the final classification token contribute to the training of TransFG [53].

the SketchRNN [58] used a Long Short-Term Memory (LSTM) [59] based VAE [18]. Recently, the Sketchformer [60] replaced the LSTM based model with a Transformer [52] based architecture which resulted 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 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 [61], which used a one-stage autoregressive approach to generate path commands. In contrast DeepSVG [62] 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 architecture of the DeepSVG model is shown in Figure 3.5.

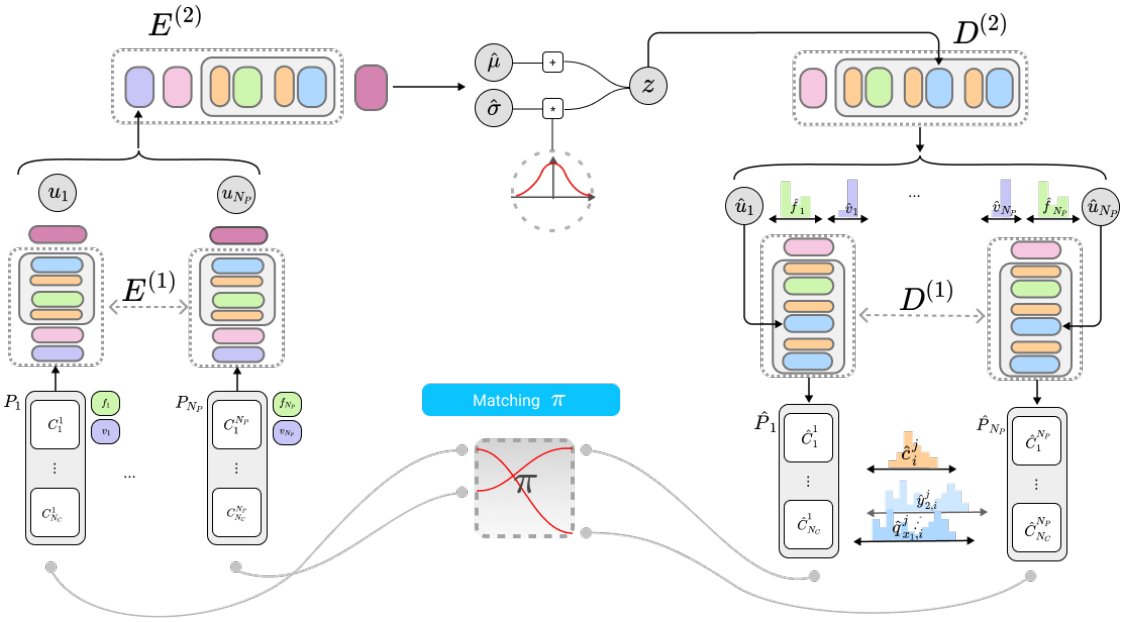


Figure 3.5: DeepSVG Model overview. Hierarchical Generative Network, DeepSVG. Input paths $\{P_i\}_1^{N_p}$ are encoded separately using the path encoder $E_{(1)}$. The encoded vectors are then aggregated using the second encoder $E_{(2)}$, which produces the latent vector z. The decoder $D_{(2)}$ outputs 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 [62].

3.5. Conclusion

From the literature above, it can be seen that typographic analysis is not an area that has received extensive focus. While there have been some attempts at typographic similarity on raster images, those have been done using conditions on fonts and characters, of which the font labels are not available for figurative content. Therefore, while these methods work well in tasks where all font metadata is known, they fail to generalize well to other applications.

However, there are quite a few relevant areas of research that can contribute towards the advancement of automatic typographic analysis. For text, extraction CRAFT seems to be the clear best for detection, while segmentation has not received a lot of attention, primarily due to the lack of data. However, both conventional [16] and deep learning [37], [40] exist for this task. While the deep learning methods easily outperform the conventional ones when enough data is available. The lack of labeled data means that the conventional methods are also quite competitive.

For classification-related tasks, the current state-of-the-art methods include Efficient-

Nets, NFNets, and Vision Transformers, while the task of fine-grained image classification is lead by the TransFG model, which builds on top of the Vision Transformer. Similarly, for deep learning on SVG images, DeepSVG Hierarchical Auto-Encoder based on the Transformer seems to be the best among a limited number of attempts in the area. All methods focus on the task of icon generation and don't focus on typographic analysis. The SVG based deep learning methods can be particularly interesting considering that they inherently encode the geometry of the font, and hence could be a good source to differentiate specific typographic feature.

Based on this literature, chapter 4 will introduce the tasks to be performed and their associated methodologies.

Chapter 4

Methods

This section focuses on outlining the tasks performed, the methodologies evaluated for each task along the associated datasets and metrics used for evaluation. Since typographic features are quite numerous, as mentioned in chapter 2, this thesis focuses on a subset of those. To select a relevant subset of the features, the designers at the EPFL+ECAL[1] lab were consulted to shortlist the most important ones. Based on this discussion four features were shortlisted to be the focus of this thesis, as listed below:

- Contrast: Given an image of a character, can the contrast be accurately calculated?
- Recognizing Serifs: Given a dataset of images of characters of different Serifs types (San-Serif, Triangular, Linear, and Slab), can a model be built to accurately predict the Serif type?
- Typographic Similarity: 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?
- Typographic Complexity: Given a dataset of posters with typographic content, can a representation of typographic complexity be built, which could be used to associate posters of similar complexity?

The methods used to achieve each of these tasks, along with the datasets they are tested on and the metrics used for evaluation are discussed in the sub-sections below

[1] https://epfl-ecal-lab.ch/

4.1. Calculation of Contrast

As discussed in chapter 2, the contrast is the ratio between the thickest and the thinnest strokes. To calculate the contrast, one can either work with the Raster image format or the Support Vector Graphic (SVG) image format. Since contrast is defined as the ratio of the perpendicular width of thickest and the thinnest strokes (see Figure 2.7 in chapter 2), it makes sense to use the geometric representation rather than the raster one. This would also allow for a more generalized solution owing to the infinite scalability of SVGs.

4.1.1 Description of Method

A multi-step algorithm is proposed to solve this task (summarized in Algorithm 1). Consider the SVG image in Figure 4.1. The image has been resolved into individual path segments (indicated by colors and numbers).

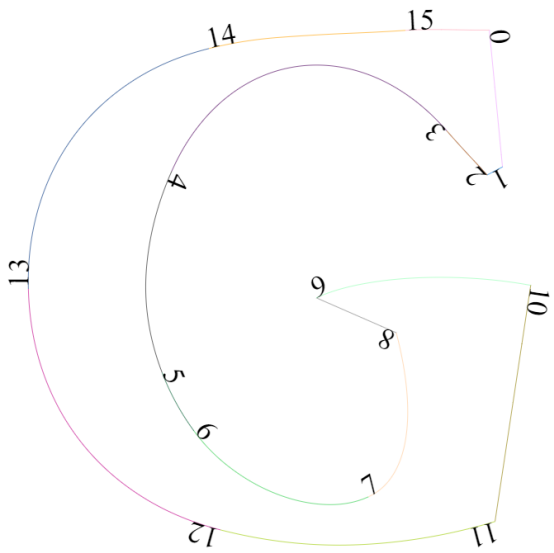
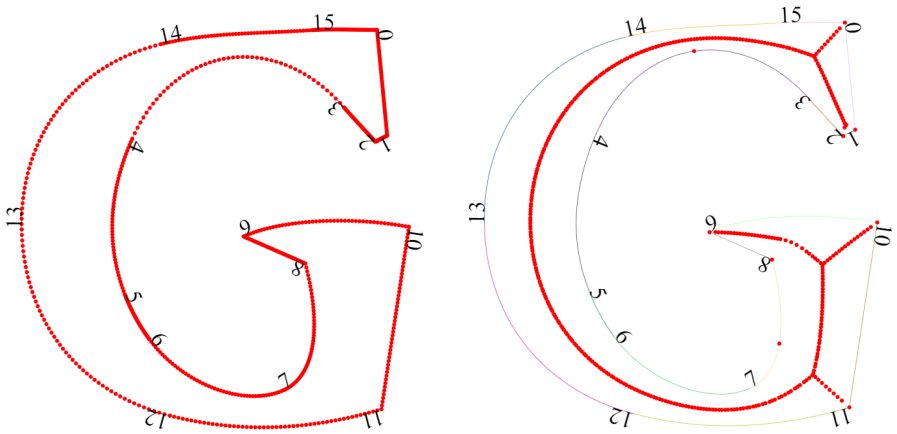


Figure 4.1: Example SVG image, where different colors and numbers indicate separate path segments

The aim is to find lines between paths running parallel to each other, the ratio between the maximum and minimum of those would approximate the contrast. One method would be to calculate perpendicular lines from sample points on each segment to every other segment in the path. However, that will have the disadvantage of calculating lines that would not necessarily be between parallel segments. Since SVGs are defined as a sequence of path segments, there is no notion of which segment is running opposite to another. However, one way this can be countered is to calculate the medial axis [63]. Given the fact that the medial path runs through the center of the image, lines going perpendicular

to the medial path at any point would intersect two opposite segments of the SVG. Using this property, the opposite segments can be recognized.

However, there is one more complication. The medial axis is generally defined for raster images, with the calculation of the same for vector graphics not readily available. To solve this, one can approximate the vector graphic as a simple polygon. The medial axis of a simple polygon is well defined problem with linear time solutions [64]. For this purpose, the SVG is decomposed into path segments defined as 1D parameterized polynomials $S_i(t)$, for segments $i = 0...N-1$ for some parameter $0 \leq t \leq 1$. After defining the polynomials, one can densely sample points on each segment, which can be used to approximate the SVG as a simple polygon. The polygon approximation can in turn be used to calculate the medial axis. This can be seen in Figure 4.2. Note that the points are sampled sparsely for visualization purposes, while the actual algorithm uses an order of magnitude denser sampling.

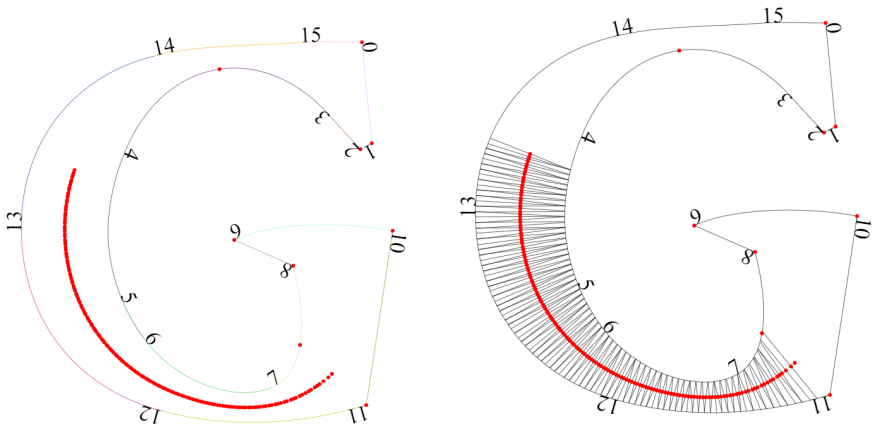


(a) Sampled points defining polygon (b) Calculated medial axis and corner points

Figure 4.2: Polygon approximation of SVG and associated medial axis

Once the medial path has been calculated, it needs to be trimmed to remove the end branches. Otherwise, the perpendicular will be calculated between the corner segments which are not running parallel to each other. To trim the lines, first the corners C are calculated which can also be seen in Figure 4.2. The corners are defined as points between segments with gradient change higher than T_{angle} and center points of arcs with curvature higher than $T_{curvature}$. Once these corners are obtained, the points closest to them are deleted systematically. Since we need to stop deleting points once the branching point is

reached, a pairwise distance threshold ratio $T_{distance\,ratio}$ is defined. The points will be trimmed until the ratio between the next two closest points is larger than $T_{distance\,ratio}$, working under the assumption that points would grow closer with respect to the corners at the branching point. Based on this the branches can be removed, followed by calculating the perpendicular lines through the medial axis intersecting the path segments. The trimmed medial axis and associated perpendicular lines are shown in Figure 4.3.



(a) Trimmed medial axis (b) Associated perpendicular lines

Figure 4.3: Trimmed medial axis and associated perpendicular lines

Once the perpendicular lines are obtained, the maximum and minimum length lines can be selected which would indicate the contrast ratio, as shown in Figure 4.4.

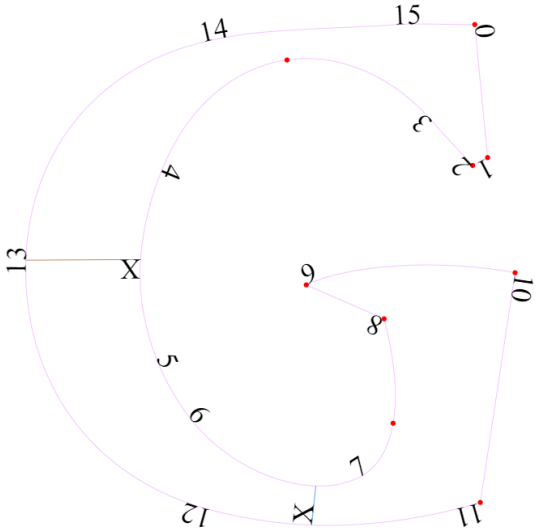


Figure 4.4: Contrast result with the maximum and minimum path widths indicted by lines marked “X”

Based on this discussion the final algorithm can be summarized as shown in Algorithm 1.

Algorithm 1: Contrast Calculation

Result: Calculation of contrast ratio for SVG image of character

initialization: Decompose SVG P into $1D$ parameterized polynomials $S_i(t)$, for segments $i = 0...N-1$ for some parameter $0 \leq t \leq 1$. ;

Define polygon by sampling M points p on each segment for a total of NxM points. ;

Calculate medial axis M of the polygon as defined by [64]. ;

Calculate corners C as the junction points of segments with gradient higher than T_{angle} and center points of arcs with curvature higher than $T_{curvature}$. ;

foreach c *in* C **do**

    Calculate sorted distances D between c and all points $m \in M$;

    Sort M according to D ;

    initialize: $i = 0$, $ratio = \infty$;

    while $i < Size(D) || ratio > T_{distance\,ratio}$ **do**

        ratio = D(i+1)/D(i) delete D(i) delete M(i)

    end

end

foreach m *in* M **do**

    Calculate array A of line lengths perpendicular to the medial axis at m and intersecting P ;

end

Calculate $Contrast = max(A)/min(A)$

4.1.2 Dataset and Metric

The dataset used for evaluation of contrast comprises 90 images composed of six font categories (Linear, Slab, Triangular, Humanistic Sans, Lineal Grotesk, and Geometrical), three fonts per category, and five characters per font. Each was manually marked by a designer at EPFL+ECAL lab and used to manually calculate font contrast. Sample images are shown in Figure 4.5, while the full list of fonts used with associated font categories is shown in Table 4.1.

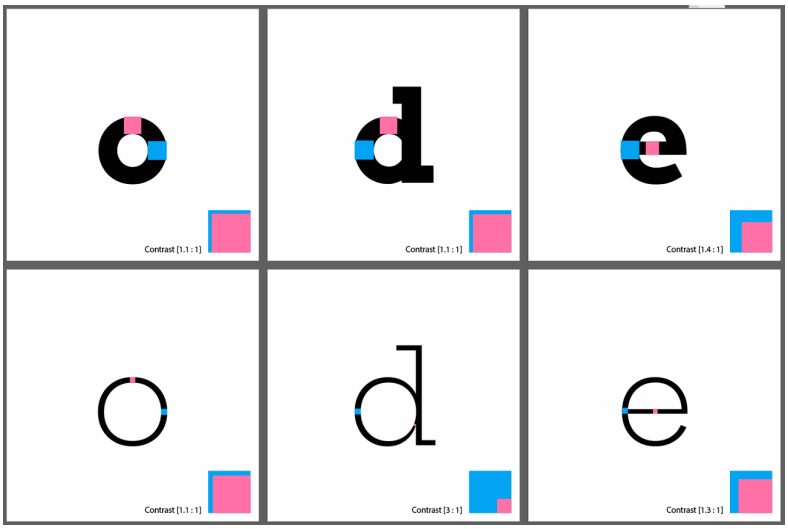


Figure 4.5: Sample test images contrast ratio marked and calculated

4.2. Classifying Serif Type

Since the recognition of the Serif Type is a classification problem, it was decided to use existing classification methods literature. Details of the methods are described in the subsections below.

4.2.1 Description of Method

As discussed in chapter 3, among the current state-of-the-art models in image classification is the EfficientNet [45] and its variants. Similarly, for the fine-grained image classification task, the current best model across various datasets is the TransFG [53] architecture which builds upon the Vision-Transformer [48] image classification model.

While serifs are quite a distinctive feature as shown in chapter 2, they are quite small compared to the overall font image. And the Serif may 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. Given this, it was decided to try both kinds of models to see what worked best in this case. For the EfficientNet, the variant EfficientNet-B2 was the biggest one that could be trained given the resource constraint, while for TransFG, the Vision Transformer Base model (ViT B) with 12 transformer layers was the largest feasible one. Two variants of the ViT B were attempted as defined in the ViT paper [48], with a patch size of 16×16 (ViT B/16) and 32×32 (ViT B/32).

Font Category	Font Family
Triangle Serifs	EB Garamond
	Libre Baskerville
	Libre Caslon Text
Linear Serifs	Abhaya Libre
	Bodoni Moda
	Rozha One
Slab Serifs	Arvo
	Hepta Slab
	Zilla Slab
Humanistic Sans	Alegreya Sans
	Open Sans
	PT Sans
Lineal Grotesk	Darker Grotesque
	Inter
	Space Grotesk
Geometrical	Josefin Sans
	Poppins
	Rajdhani

Table 4.1: List of fonts used along with their associated categories

4.2.2 Dataset and Metrics

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) by consulting a typography designer at the EPFL+ECAL lab. Afterward, for each font variation, 62 images (26 uppercase, 26 lowercase, 10 digits) were extracted, making a total dataset of size 24366 images. 80% (19492) of these images were used in training while 20% (4874) were used 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 used which made a total of 1736 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. A summary of the dataset is provided in Table 4.2.

Dataset	Train	Validation	Test (Font Independent)
No: of Images	19492	4874	1736

Table 4.2: Summary of dataset for serif classification model

4.3. Typographic Similarity

As discussed in chapter 2, typographic similarity aims at grouping 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 can be quite challenging in general 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 (as discussed in chapter 3 in the work by Favre [16]). A workaround to this was to use conditions on the character, font family, and font variations, as proposed by [17], which enabled the model to learn some specific features like the serif classification. However, this method lends a challenge, considering the fact that it was trained on conditions of the font and character both on a large dataset of $<$10000K fonts. Such a large amount of font labeled data is not available for figurative content, which is the eventual use-case and motivation for this research. Therefore, the aim here would be to learn a useful latent representation using either no labels or only character labels.

4.3.1 Description of Method

As already discussed above and in chapter 3, the existing methods employing VAEs to build a latent representation of fonts use raster images with various label conditions. However, there is no existing work in literature that does this task using the SVG representation of fonts. The reason why that should be considered is that it provides an advantage in being invariable to size consideration (due to the property of infinite scalability) and better encoding the geometric details of the font (due to the vector representation). Therefore, it is proposed to use a VAE on the SVG representation with and

without character labels to build a latent representation. The used model is inspired by the DeepSVG [62] Hierarchical Generative Network discussed in chapter 3.

4.3.2 Dataset and Metrics

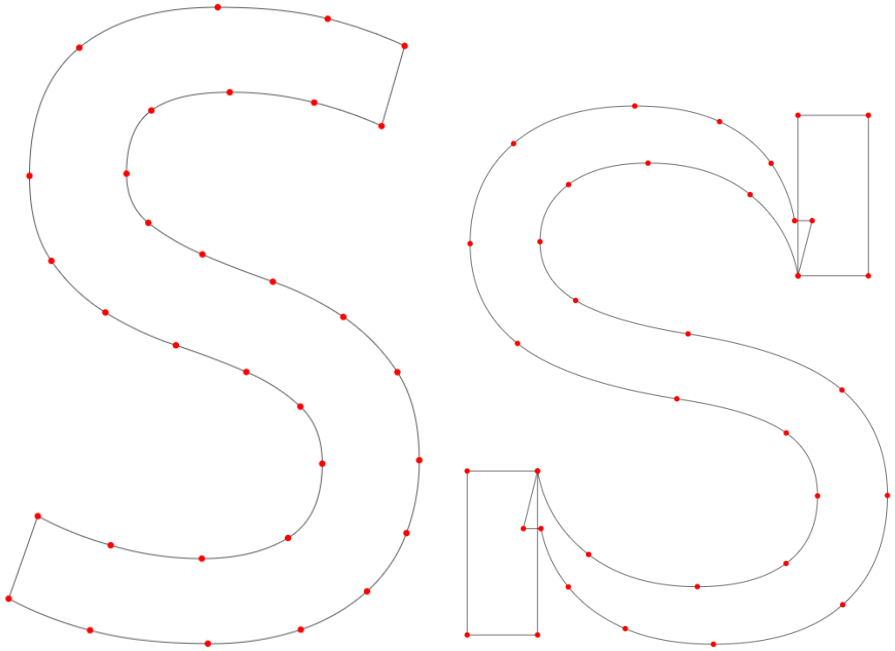
The Training dataset used is the same as the Serif classification task, built by first selecting a range of common fonts (including variations) for the four categories (sans-serif, triangular, linear, and slab) by consulting a typography designer at the EPFL+ECAL lab. Afterward, for each font variation, 62 images (26 uppercase, 26 lowercase, 10 digits) were extracted, making a total dataset of size 24366 images. The models are trained both with and without character conditions on the varying sizes of the latent dimension. The model is evaluated on how well it is able to group characters of the same serif type and belonging to the same font family, as well as the impact of the variation in the size of the latent dimension.

4.4. Typographic Complexity

As previously mentioned in chapter 2, typographic complexity is a pretty abstract concept. To better define the task at hand, the typeface designers at the EPFL+ECAL Lab were consulted to understand what the end goal was for this concept of complexity in the broader context of their projects. The identify objective is to be able to group posters based on their level of typography complexity. Therefore, a method is proposed to make a first attempt towards incorporating the various discussions into a concrete feature description for complexity.

4.4.1 Description of Method

As discussed in chapter 2, the notion of complexity has both a local and a global context. It can incorporate aspects like the complexity of a single character, as well as the overall complexity of a word (which can then also connect to other factors like the number of characters per word and number of words per poster). Therefore, to consider all of these different factors, a basic feature descriptor is proposed which incorporates these features. The number of characters per word and the number of words per poster is easily quantifiable. But the question is then on how to quantify the complexity of individual characters on the posters. One possible answer for this goes back to the SVG representation of an image. An image that is more complex would require more points and segments to define it, as shown in Figure 4.6.



(a) Low Complexity (32 points) (b) High Complexity (38 points)

Figure 4.6: Example of difference in number of points for low and high complexity SVGs.

So as a basic descriptor, one can incorporate the number of paths/points per SVG of the character to better model the approximate level of "complexity" in each. However, the question then comes to how these would be aggregated to make a word level, and then later a poster level descriptor. To do so, one can take a statistical measure of the points/paths, and then use that along with the number of words and characters. Based on this idea, the following basic description vectors are proposed at the poster level:

- Descriptor 1 (Des-AVG):
  - ⋆ Number of words per poster
  - ⋆ Number of characters per poster
  - ⋆ Mean number of characters per word
  - ⋆ Mean number of paths per character
  - ⋆ Mean number of points per path
- Descriptor 2 (Des-MAX):
  - ⋆ Number of words per poster
  - ⋆ Number of characters per poster
  - ⋆ Max number of characters per word

    - ⋆ Max number of paths per character
    - ⋆ Max number of points per path
- Descriptor 3 (Des-STD):
    - ⋆ Number of words per poster
    - ⋆ Number of characters per poster
    - ⋆ Std. Dev. of number of characters per word
    - ⋆ Std. Dev. of number of paths per character
    - ⋆ Std. Dev. of number of points per path
- Descriptor 4 (Des-ALL):
    - ⋆ Number of words per poster
    - ⋆ Number of characters per poster
    - ⋆ Mean number of characters per word
    - ⋆ Max number of characters per word
    - ⋆ Std. Dev. of number of characters per word
    - ⋆ Mean number of paths per character
    - ⋆ Max number of paths per character
    - ⋆ Std. Dev. of number of paths per character
    - ⋆ Mean number of points per path
    - ⋆ Max number of points per path
    - ⋆ Std. Dev. of number of points per path

The first three descriptors take different statistical measures for the various features, while the fourth one aggregates all three. Since the features in the descriptors are correlated, the extracted feature vectors are transformed to a lower dimension space using Principal Component Analysis (PCA) [65].

4.4.2 Dataset and Metrics

Since there is no objective measure for evaluation, one can come up with different feature descriptors and then use them to pull similar posters from the dataset and ask the designers to choose which they think is the most similar in terms of complexity.

This is exactly what was done in this case. First, a poster was randomly selected from the dataset, called the "Reference" poster. Then, for each descriptor, three most similar posters to the "Reference" poster were collected, to form four sets of four posters (1 Reference, and 3 most similar based on the descriptor). These four sets, along with the fifth set of 4 (1 Reference, 3 randomly chosen) added as a control, were presented to multiple designers at the EPFL+ECAL lab in the form of a questionnaire. The designers were asked to rate which set they thought was the most similar in terms of typographic complexity. To better aggregate the results, three such questionnaires were made (each with a different Reference poster randomly selected). The results are used to understand which, if any, of the descriptors are meaningful measures for complexity.

The dataset used was a set of 1500 posters from poster collection at The Museum Für Gestaltung in Zurich, introduced in chapter 1 (refer to sample posters in Figure 1.1). First, the dataset characters were segmented using the method proposed by Favre [16], who worked on the same dataset. Then each image was converted to SVG format before being used in the analysis.

4.5. Conclusion

In this chapter, methods are proposed for solving each task. For contrast calculation, a multi-step algorithm is introduced which exploits the inherent geometric nature and infinite scalability of vector graphics. For the classification of serifs, both general and fine-grained image classification models are suggested (given resource constraints). These would be evaluated to see which is more appropriate for the task at hand. In the case of typographic similarity, an SVG-based VAE is proposed (both with and without label conditions on characters). It aims to exploit the geometric nature of the SVG to build a latent space representation that learns to differentiate between specific font features. Finally, on the topic of typographic complexity, four description vectors are proposed which attempt to aggregate the discussions on complexity with the design experts.

The results for these methods on the associated datasets are presented along with a detailed analysis in chapter 5.

Chapter 5

Results

This chapter presents the results for the methods discussed in chapter 4. For each of the four tasks discussed previously (calculation of contrast, classification of serif types, typographic similarity, typographic complexity), results and analysis are presented in separate sections below.

5.1. Calculation of Contrast

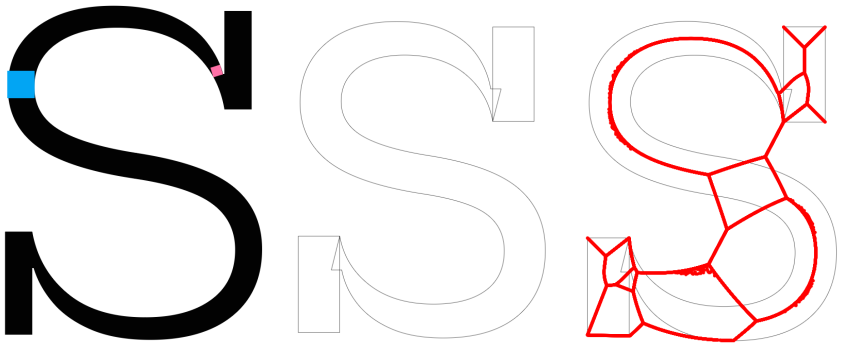
As discussed in the methods section, the contrast calculation is evaluated on a dataset of 90 images labeled by an expert designer. The results are evaluated for different values of the distance ratio threshold parameter and whether arcs with high curvature are included as corners or not. For each evaluation, the minimum, maximum, and mean percentage error across the dataset is calculated. These results are shown in Table 5.1

Distance Ratio Threshold	Include Arcs in Corners	Mean Absolute Error (%)	Max Absolute Error (%)	Min Absolute Error (%)
0.0001	Yes	39.00237	611.36658	0.30460
0.0001	No	43.68014	611.36658	0.30460
0.001	Yes	65.16174	1257.42230	0.01848
0.001	No	72.60139	1257.42230	0.01848

Table 5.1: Mean, Maximum and Minimum absolute percentage error in contrast for different values of distance ratio threshold ($T_{distance\,ratio}$) and whether corners are included as curves or not.

5.1.1 Analysis

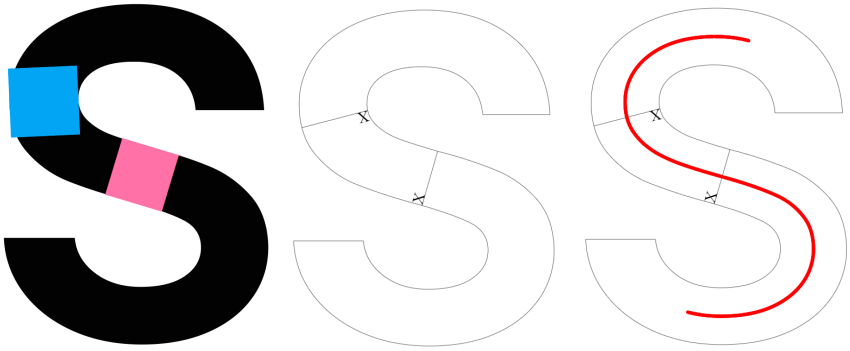
To better understand the behavior of the algorithm, the maximum and minimum error instances are further analyzed, for $T_{distance\,ratio} = 0.0001$ and arcs included as corners. For the maximum error, the ground truth annotation, SVG boundary, and medial path are displayed in Figure 5.1. Notice, how the boundary path is self-intersecting, and therefore cannot be approximated as a simple polygon in this case. This results in the medial path algorithm completely failing, resulting in an eventual 600% error compared to the ground truth.



(a) Ground Truth (b) SVG Boundary (c) Calculated Medial Path

Figure 5.1: Contrast sample test case with maximum error (611.37%)

In contrast, the sample with the lowest error can be seen in Figure 5.2. Notice how the path is not self-intersecting, allowing for the correct calculation of the medial path, resulting in high accuracy of the contrast calculated.



(a) Ground Truth (b) SVG Boundary and Min/Max lines (c) Trimmed Medial Path and Min/Max lines

Figure 5.2: Contrast sample test case with minimum error (0.30%)

To compare how easy or difficult it is to calculate the contrast for a particular letter, font category, or font family, the same statistics (mean, maximum, and minimum absolute percentage errors) are extracted across each character, font category, and font family for $T_{distance\,ratio} = 0.0001$ and arcs included as corners. The mean, maximum and minimum percentage error per character can be seen in Figure 5.3. Notice how the character "s" seems to have the highest maximum and mean errors. However, from Figure 5.1, it is known that the highest error for this character is $> 600\%$. So after ignoring the top 10% outliers, the algorithm is worst performing for the characters "e" and "M" (highest mean, maximum and minimum error).

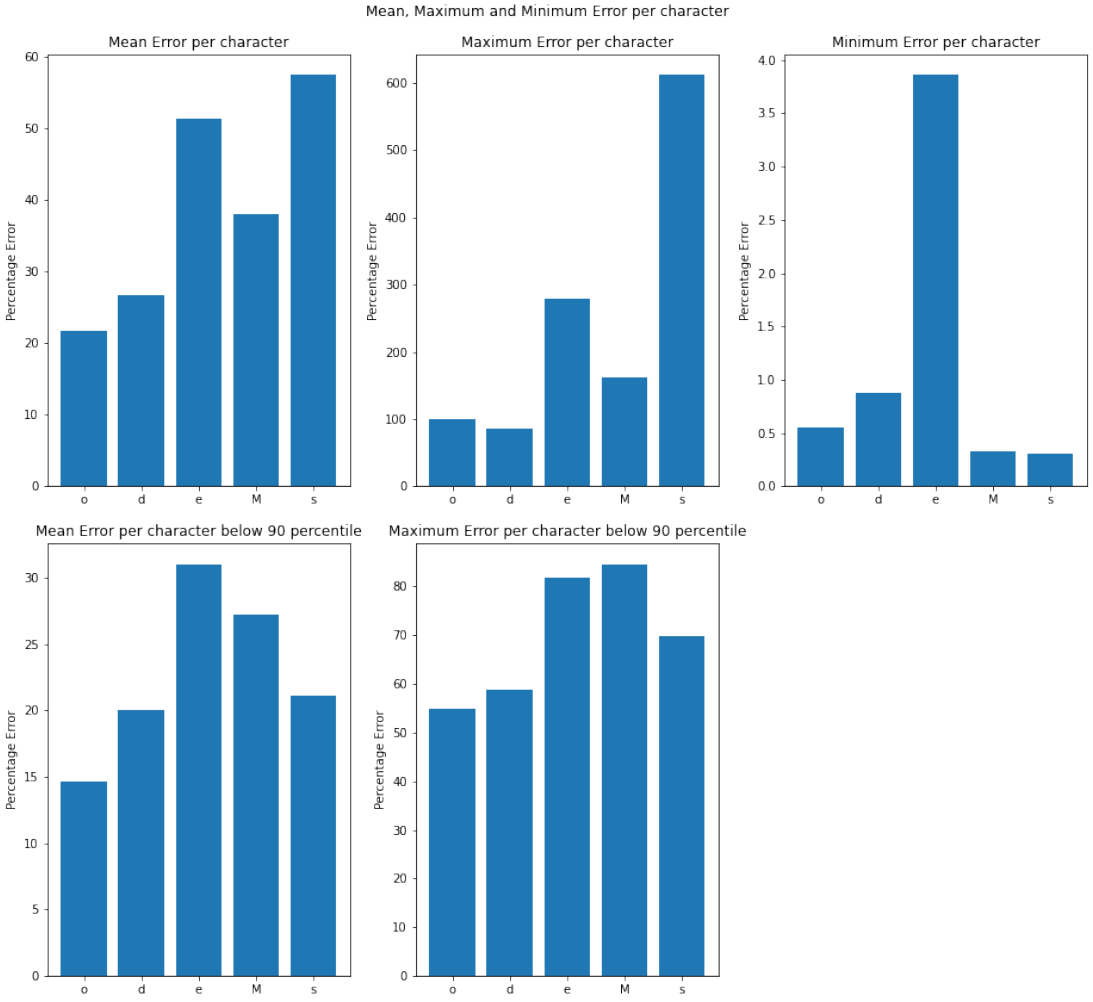


Figure 5.3: Mean, Maximum and Minimum percentage contrast error per character for $T_{distance\,ratio} = 0.0001$ and arcs included as corners. Also shown is the mean and maximum percentage errors of the bottom 90^{th} percentile

Next, the same statistics are obtained for each font category considered in the dataset. The associated plots are shown in Figure 5.4. From these results, it can be seen that the

serif font categories (linear, triangular, and slab) seem to be the most challenging, while the sans-serif one is the easiest in general. On the other hand, geometrical fonts are the most accurately calculated ones, once the top 10% of the outlier error percentages are ignored.

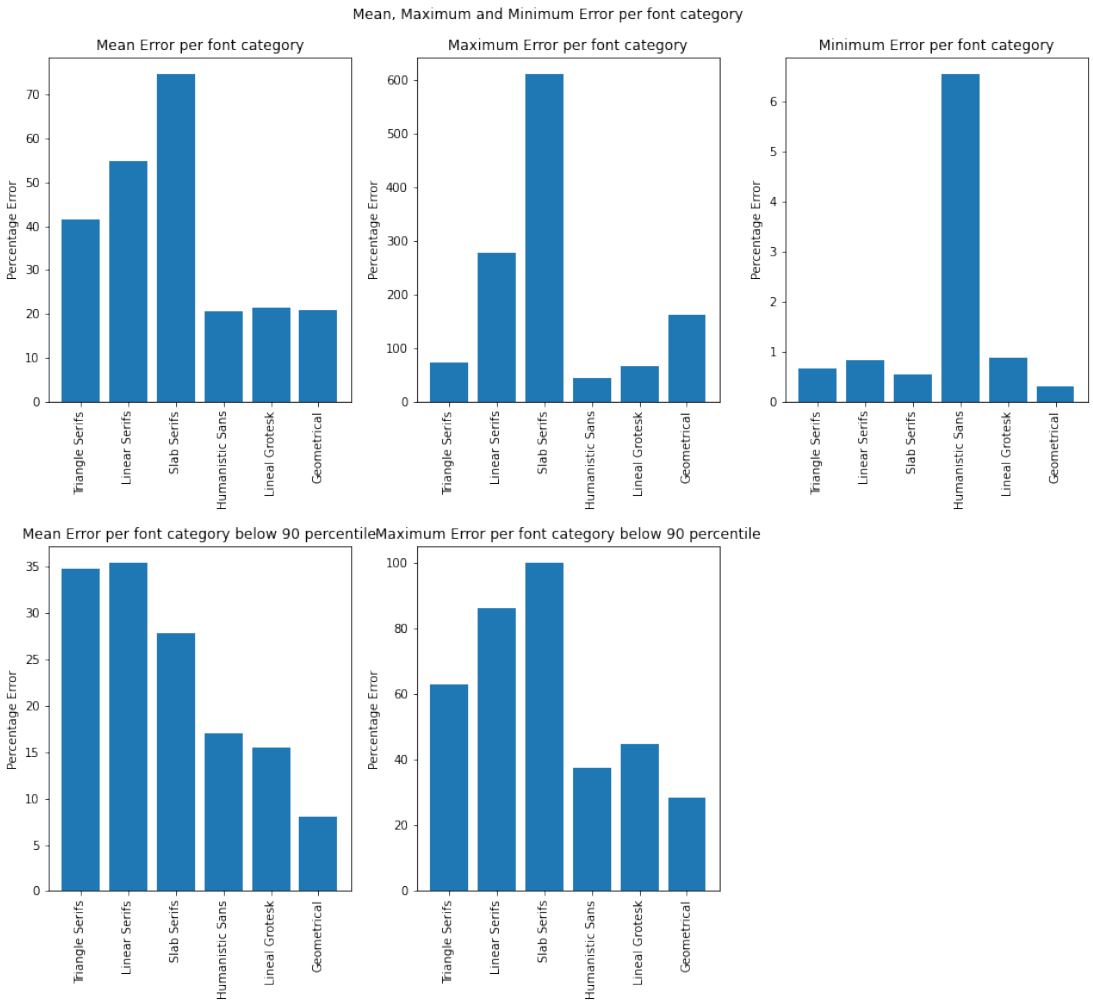


Figure 5.4: Top row: Mean, Maximum and Minimum percentage contrast error per font category for $T_{distance\,ratio} = 0.0001$ and arcs included as corners. Bottom row: mean and maximum percentage errors of below the 90^{th} percentile.

Finally, to understand the distribution of the errors across individual font families, the statistics are extracted for each font family within each font category. The associated plots are shown in Figure 5.5. For details on which font belongs to which category, please refer to Table 4.1 in chapter 4. Notice how the highest mean, minimum, and maximum errors are all for the serif fonts. This shows that serif font poses a much more challenging problem, presenting issues such as the self-intersecting polygon shown in Figure 5.1.

Finally, the histogram distribution of the errors is shown in Figure 5.6. Notice how

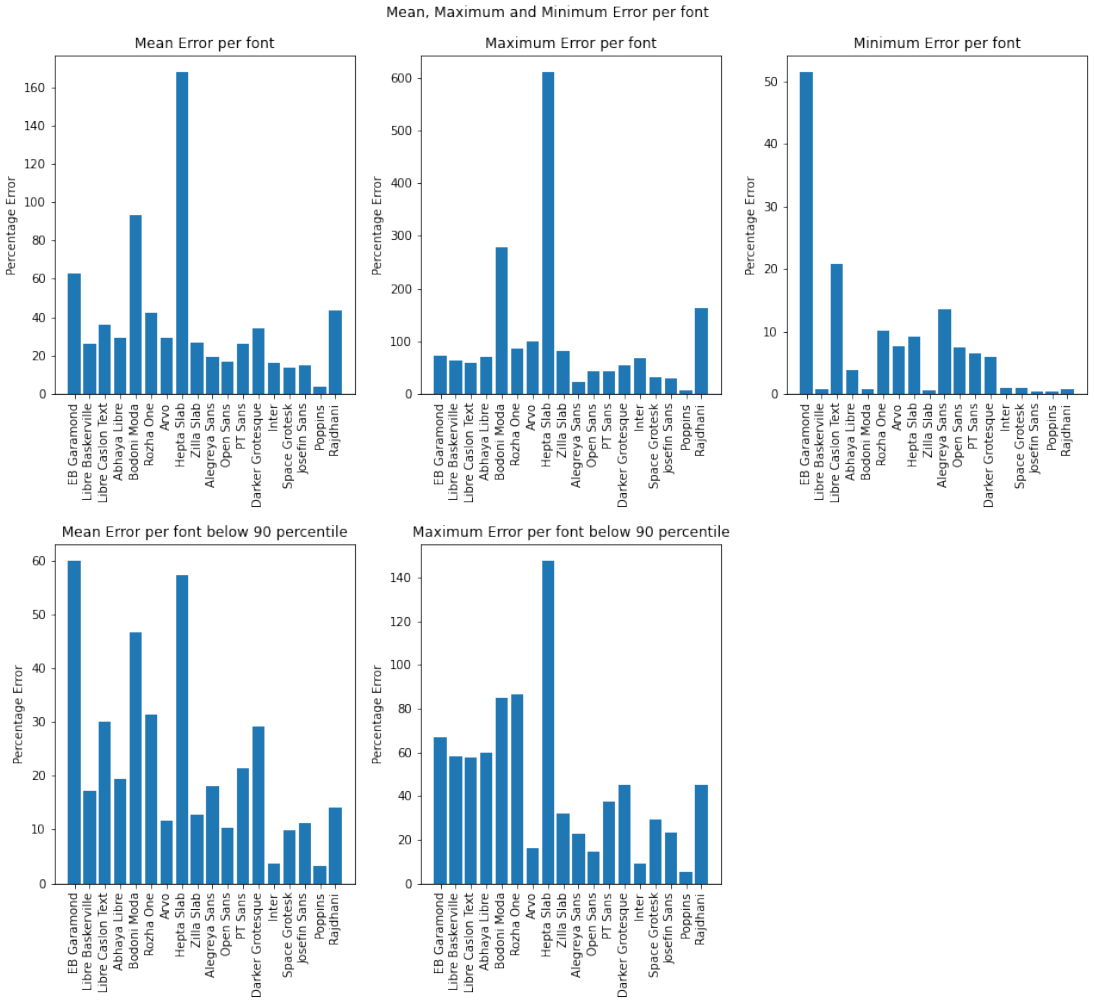


Figure 5.5: Top row: Mean, Maximum and Minimum percentage contrast error per font family for $T_{distance\,ratio} = 0.0001$ and arcs included as corners. Bottom row: Mean and maximum percentage errors of below the 90^{th} percentile.

most of the errors lie in the first few bins, indicating that overall the model is quite accurate.

5.2. Classifying Serif Type

For the serif classification, accuracies are calculated on the training, validation, and a font-independent test set. The resulting accuracy for each trained model on each of the sets is shown in Table 5.2.

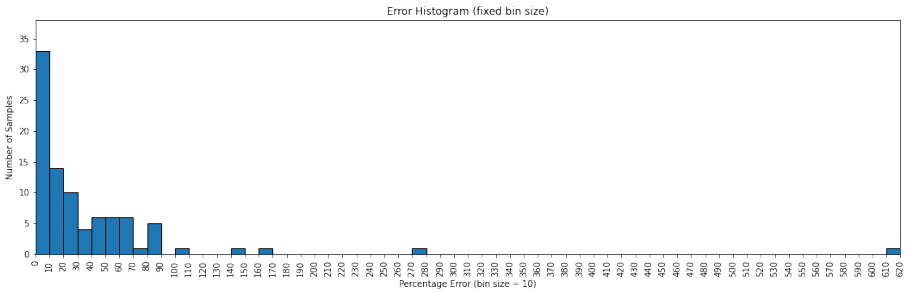


Figure 5.6: Error histogram for percentage contrast errors for $T_{distance\,ratio} = 0.0001$ and arcs included as corners. The axis has a bin size of 10 percentages/bar.

Model + Backend	Accuracies		
	Train	Validation	Test (Font Independent)
TransFG ViT B/16 Backend	0.88954	0.74620	0.79570
TransFG ViT B/32 Backend	0.97958	0.75318	0.89367
EfficientNet-B2	0.98887	0.96635	0.69176

Table 5.2: Train, Validation and Test results for Serif Classification

5.2.1 Analysis

Notice how the general image classification model (EfficientNet-B2) performs best on both the training and validation set. If it had been performing best on just the train set, one could claim that the model was overfitting. However, good performance on both the training and validation sets indicates that the model is not overfitting. However, the model shows significant accuracy degradation on the font-independent test. This indicates that the model is memorizing font features rather than learning the subtle features of the serifs.

In contrast, the fine-grained image classification model, TransFG, performs comparatively worse on the train and validation sets but generalizes very well on the test set. 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 test set, including metrics for precision, recall, which are then used to calculate the F1-score.

The high F1-score for the TransFG models shows that it performs well across all classes and is not biased any specific. While the EfficientNet-B2 models has an F1-score

Model + Backend		Detailed 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

Table 5.3: Precision, Recall and F1-Score for Serif Classification Test Set

which is nearly 0.2 units lower, indicating that it does not generalize well to new fonts. From these results, it can be concluded that the classification of serif types is better modelled as a fine-grained classification task rather than a general image classification problem, given the subtle nature of the serif features.

To understand cases in which the models fail, failure samples are displayed for each on the test set. These samples can be seen in Figure 5.7, Figure 5.8 and Figure 5.9 for the ViT B/16, ViT B/32, and EfficientNet-B2 models respectively.

Notice how most of the mistakes for the TransFG models are those where serifs are either very subtle, or are special cases of serif font instances where the serif does not appear like the characters “e” and “o”, and digits “0”, “8” and “9”. While the EfficientNet-B2 model fails on more visually obvious cases, further confirming that it fails to generalize well on the font independent test set.

5.3. Typographic Similarity

The typographic similarity models are evaluated in two cases. Their ability to associate characters of the same serif type and the characters of the same font type in the latent space, with and without character label condition for varying sizes of the latent dimension z. In that regard, models are trained on the dataset discussed in chapter 4 for three different latent space dimensions (64, 128, and 256), and two different label conditions (with character labels and without character labels), making a total of six models. Then each model is tested on two cases (serifs and fonts). For serifs, 50 of the same characters are randomly sampled from the dataset for each serif type (sans-serif, linear, triangular, slab) and a t-distributed stochastic neighbor embedding (TSNE) plot is generated for each model from latent vector encoding for the character. The resulting plots of various latent dimension sizes without labels can be seen in Figure 5.10 and with labels in Figure 5.11.

Figure 5.7: Sample failure cases on Serif Classification Test Set with TransFG (ViT B/16 Backend)

Similarly for the fonts, 5 different fonts are randomly selected and then 50 characters are randomly sampled for each font for each model. The associated plots without labels and with labels can be seen in Figure 5.12 and Figure 5.13 respectively.

5.3.1 Analysis

Consider first the serifs case with Figure 5.10 and Figure 5.11, without labels and with labels respectively. Notice, how in the labeled case in Figure 5.11, the model is able to make a distinction 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, if the unlabelled models are

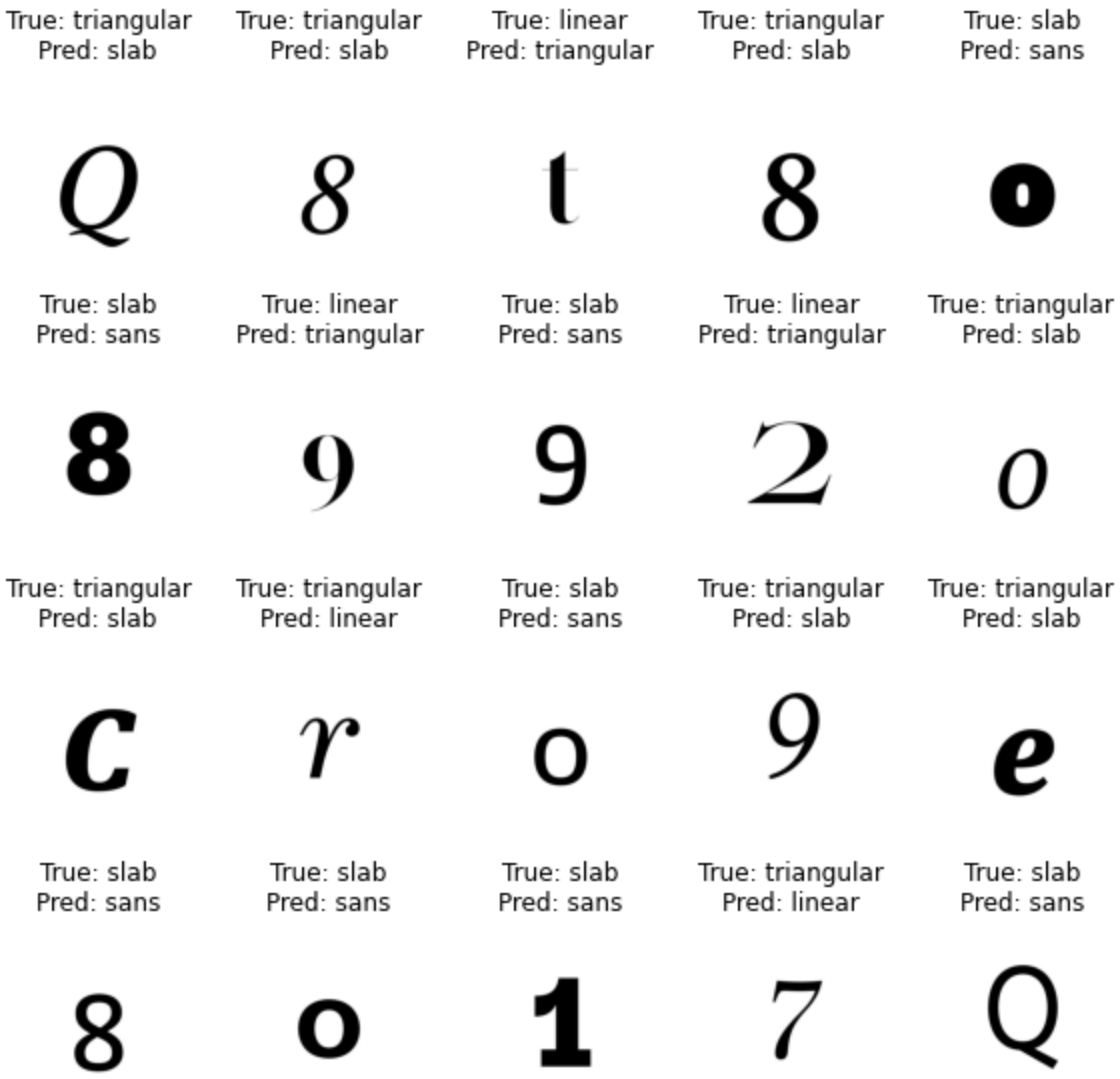


Figure 5.8: Sample failure cases on Serif Classification Test Set with TransFG (ViT B/32 Backend)

considered in Figure 5.10, the differentiation is even less obvious with most of the sans-serif samples also mixed in with the rest, and not appearing as a separate group. This shows that the character labels help the model better differentiate between the serif types, but not well enough to differentiate the more subtle difference between the serif types. It can be further noted that changing the latent dimension size does not seem to have any obvious impact. This could be because the dataset is relatively small for even a latent dimension of size $z = 64$ to be a bottleneck.

Next consider the case of fonts, with the unlabelled and labeled TSNE plots shown in Figure 5.12 and Figure 5.13 respectively. Once more the model is able to learn some features and group characters from similar fonts together. Notice how the two GillSansNova

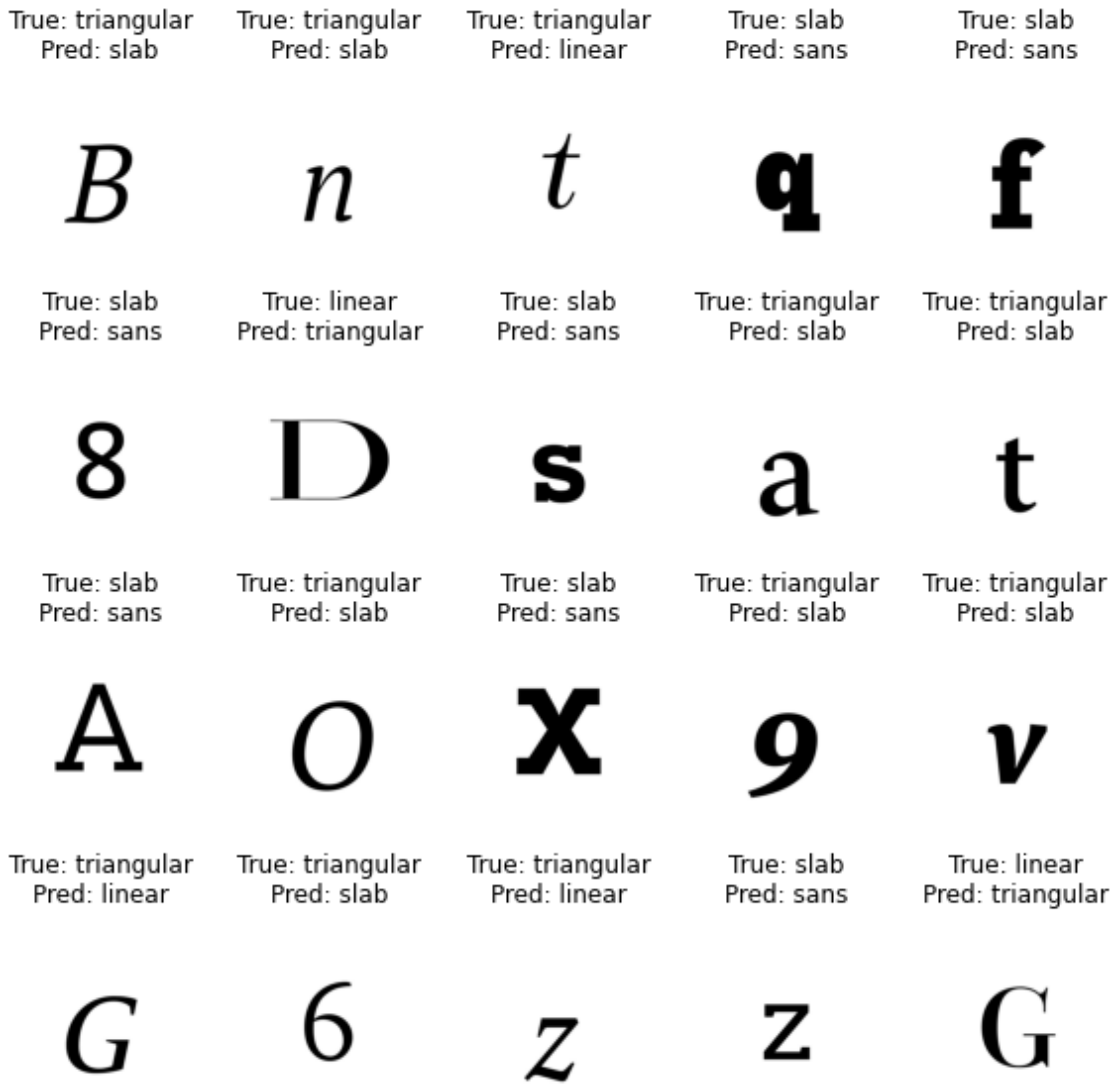


Figure 5.9: Sample failure cases on Serif Classification Test Set with EfficientNet-B2

variants are grouped close to each other by the labeled models in Figure 5.13. The same can be observed in the unlabelled case in Figure 5.12, but is much less obvious. Interestingly, there is a subtle visual difference when using different sizes of the latent dimension z, with the smaller size grouping together the two similar fonts much better than the larger ones. However, the model is unable to make a clear distinction between the three other fonts which are different from each other and should ideally be in three separate clusters.

5.4. Typographic Complexity

For typographic complexity, four different descriptors (Des-AVG, Des-MAX, Des-STD, Des-ALL) had been defined as described in chapter 4. Afterward, three questionnaires

were made, each with one Reference poster (randomly chosen) and three most similar posters for each metric, along with three posters chosen randomly for control. These were sent to different design experts who were asked to choose which set in each questionnaire was the most similar in terms of complexity and which was most dissimilar. A total of seven responses were received, and the user responses can be seen in Figure 5.14.

5.4.1 Analysis

Based on the results it can be seen that the opinions are quite mixed. For q1 and q3, the clear best is Des-ALL (Set4), but it ends up being nominated as the most dissimilar for q2. While not in the majority in q2 and q3 (and not appearing at all in q1), Des-MAX (set2) is the only one that was mentioned as one of the most similar and not mentioned as one of the least similar. Surprisingly, Des-STD (set3) got selected as worst more than the random (set5), with a unanimous agreement on Dissimilarity for q3.

Based on these results, it seems that the two best descriptors among the test ones are Des-MAX and Des-ALL, while Des-STD ends up being nominated as the worst.

5.5. Conclusion

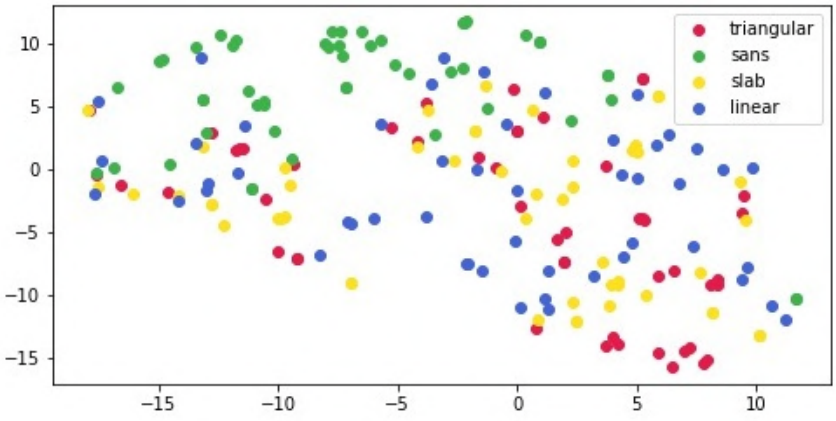
Results are presented for the four tasks in focus namely contrast calculation, serif classification, typographic similarity, and typographic complexity. For the task serif classification, highly accurate results are achieved, with the fine-grained classification model (TransFG) achieving ($\sim 96\%$) accuracy on a font-independent test set. It is also found out that a general image classification model (EfficientNet-B2) does not generalize well to the test set, although performing well both on the train and validation sets. This indicates that the subtle features of serifs are more suited to a fine grain recognition task.

For contrast calculation, the proposed algorithm achieves a mean error of 39% across the dataset, with a minimum error of $< 0.3\%$. However, the algorithm fails completely on outlier cases where the boundary of the character is not a simple polygon (a fundamental assumption of the algorithm). It is also important to note that the algorithm would be quite computationally expensive for large datasets considering the large number of loops that are used in different parts of it. But, in general, as the first attempt on this task, this algorithm works well across most types of fonts, fast enough for small datasets.

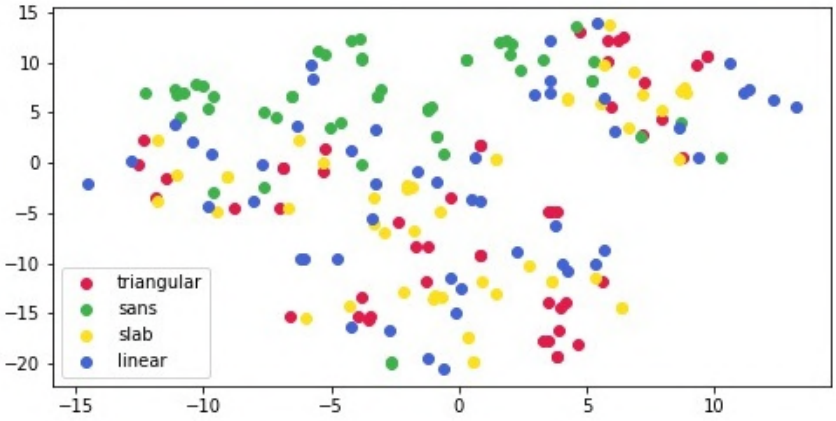
For the typographic similarity, the proposed SVG VAE is able to differentiate sans-serif and serif fonts, and group together characters from the same font family. But it fails to differentiate between serif types and fonts of different families. These results may seem

worse compared to those presented in [17], but it should be noted that they are achieved with a dataset of a small number of fonts without font label condition.

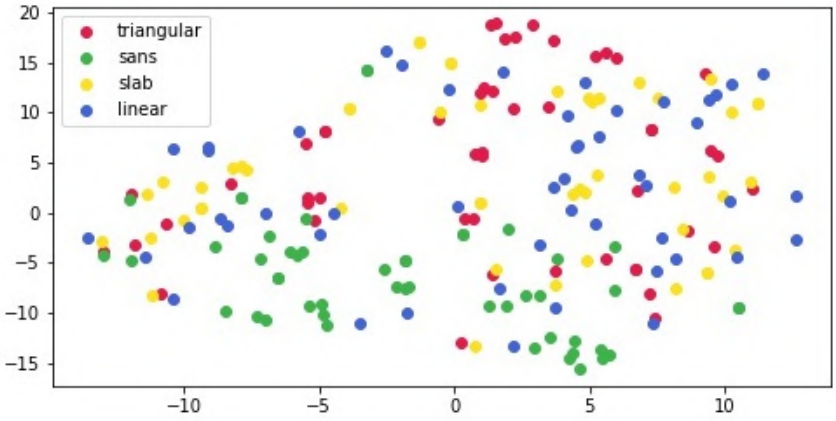
Finally, for typographic complexity, four description vectors are proposed which attempt to aggregate the highly abstract notion of typographic complexity as described by design experts. Among the four descriptors, there is no clear indication for which one is the best, considering that the responses from the design experts are mixed. However, some trends can be seen. The designers seem to lean towards Des-MAX and Des-ALL and seem to reject Des-STD. But the current sample size is not statistically large enough to draw solid conclusions, and therefore requires further investigation.



(a) Unlabelled (64)

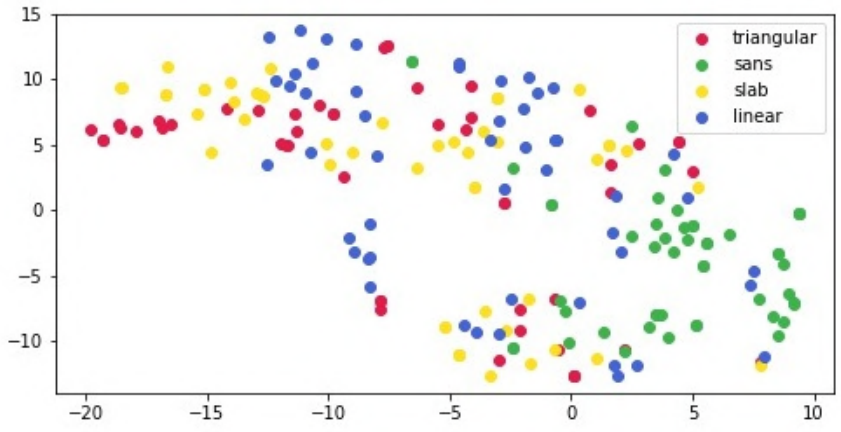


(b) Unlabelled (128)

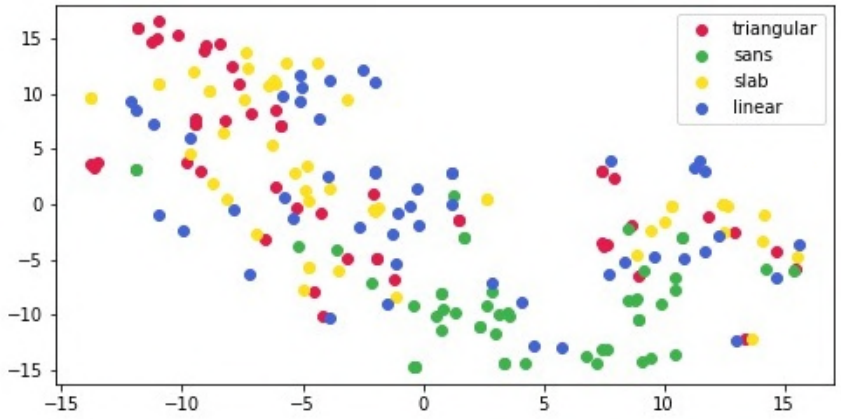


(c) Unlabelled (256)

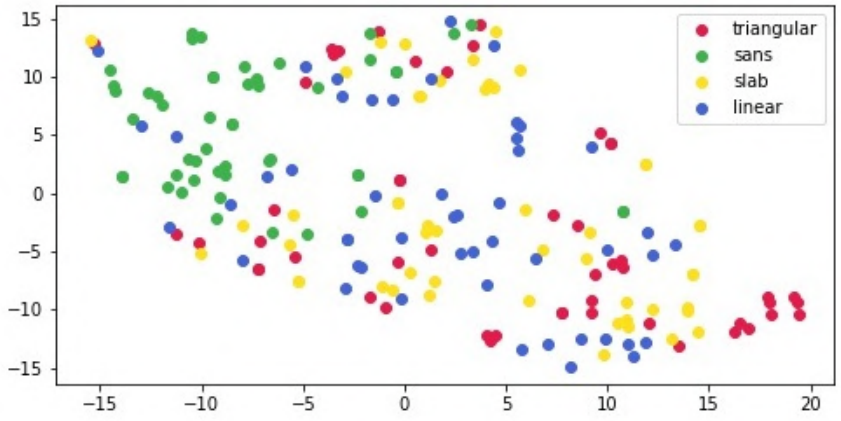
Figure 5.10: TSNE plots for 50 characters across different serif types, with unconditioned VAE trained for different latent dimension sizes.



(a) Labelled (64)

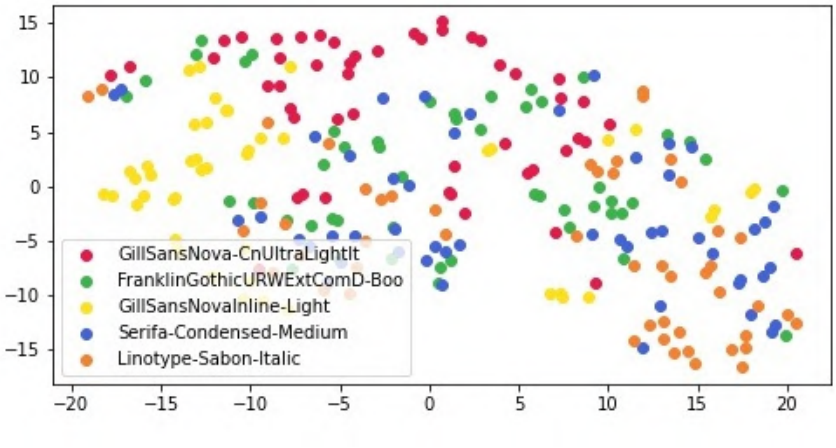


(b) Labelled (128)

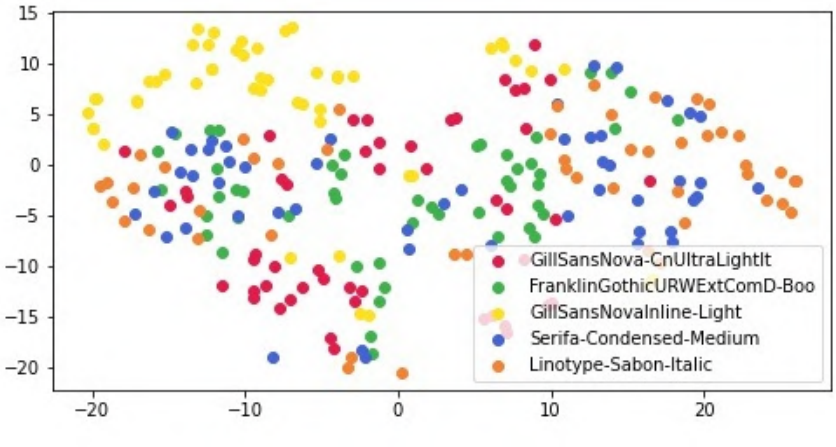


(c) Labelled (256)

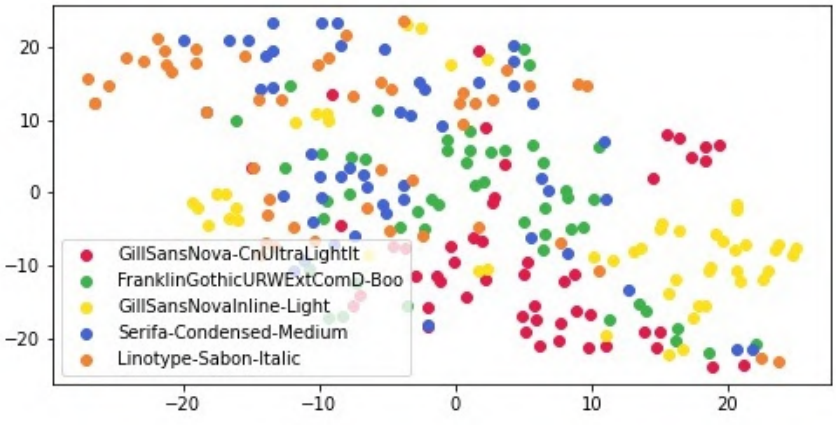
Figure 5.11: TSNE plots for 50 characters across different serif types, with conditioned (on characters) VAE trained for different latent dimension sizes.



(a) Unlabelled (64)

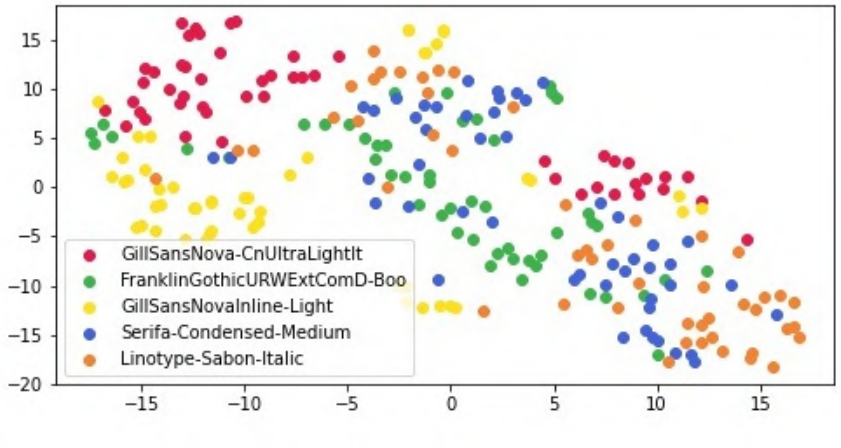


(b) Unlabelled (128)

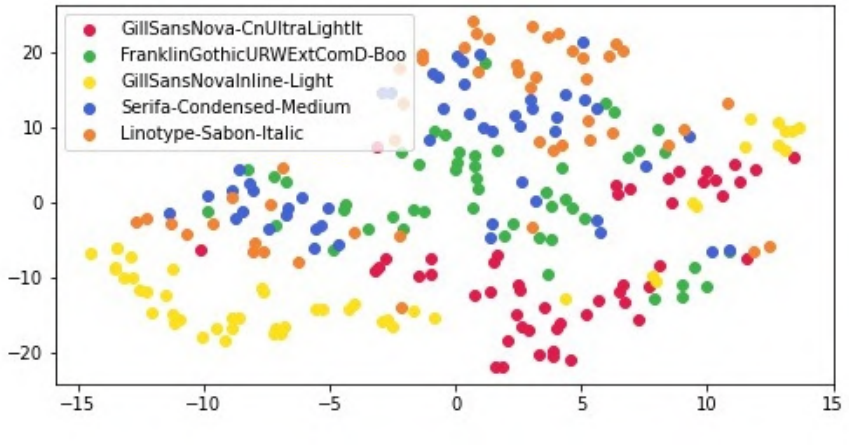


(c) Unlabelled (256)

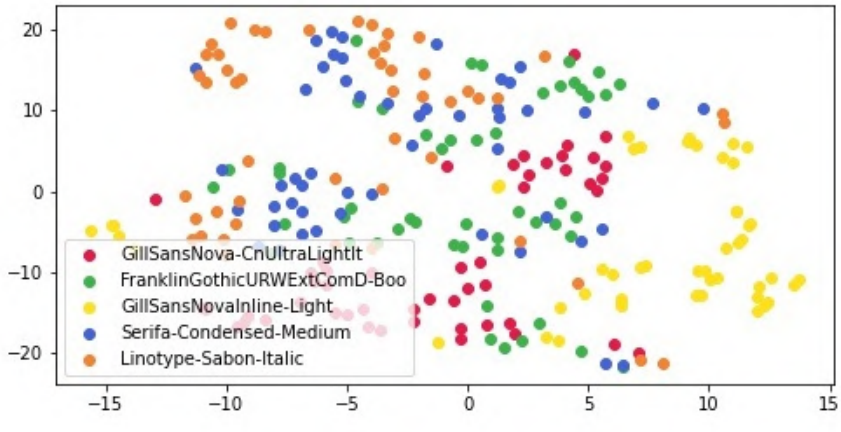
Figure 5.12: TSNE plots for 50 characters across different fonts, with unconditioned VAE trained for different latent dimension sizes.



(a) Labelled (64)

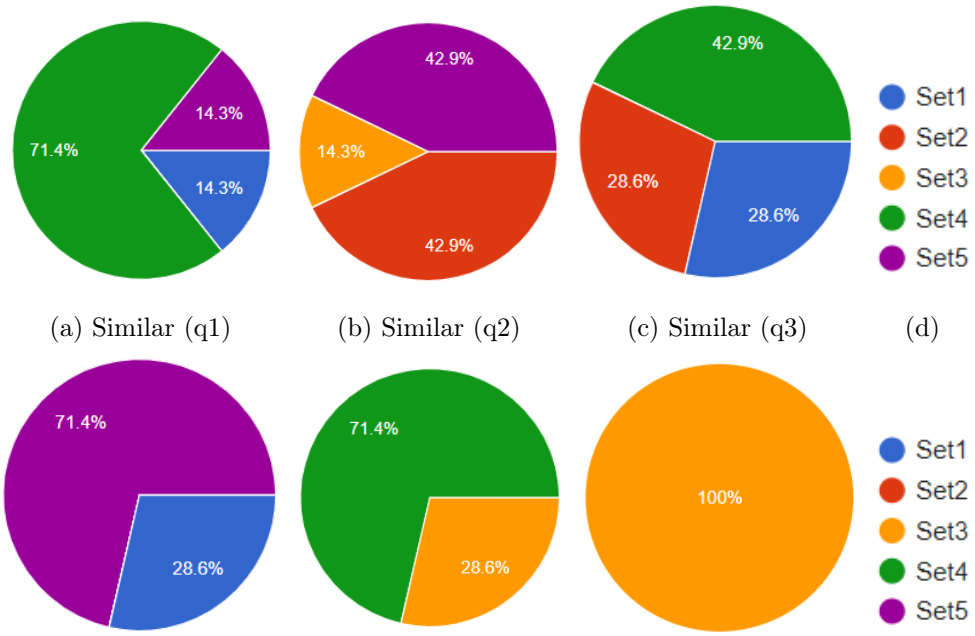


(b) Labelled (128)



(c) Labelled (256)

Figure 5.13: TSNE plots for 50 characters across different fonts, with conditioned (on characters) VAE trained for different latent dimension sizes.



(a) Similar (q1) (b) Similar (q2) (c) Similar (q3) (d)

(e) Dissimilar (q1) (f) Dissimilar (q2) (g) Dissimilar (q3) (h)

Figure 5.14: Results from the three questionnaires. Top row: most similar descriptor for each questionnaire. Bottom row: Most dissimilar descriptor for each questionnaire. The labels are: set1 (Des-AVG), set2 (Des-MAX), set3 (Des-STD), set4 (DES-ALL) and set5 (random).

Chapter 6

Summary

The focus of this thesis is on automatic typographic analysis. While typographic features are numerous (more details in chapter 2), this thesis focuses on four of them based on discussions with design experts at the EPFL+ECAL lab. The four features and associated problem statements are mentioned below:

- Contrast: Given an image of a character, can the contrast be accurately calculated?
- Recognizing Serifs: Given a dataset of images of characters of different Serifs types (San-Serif, Triangular, Linear, and Slab), can a model be built to accurately predict the Serif type?
- Typographic Similarity: 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?
- Typographic Complexity: Given a dataset of posters with typographic content, can a representation of typographic complexity be built, which could be used to associate posters of similar complexity?

6.1. Results

The results for the four focus tasks are summarized in the following sub-sections.

6.1.1 Contrast Calculation

For contrast calculation, the algorithm accurately calculated the contrast for more than 50% of the dataset with less than 20% maximum error (With the least being less than 1%). There were a few outlier cases where the algorithm failed because the character shape

was a self-intersecting polygon (because of which the employed medial axis calculation algorithm failed). Statistics across the results show that the Serif fonts are the most difficult to calculate the contrast for while the geometric and sans-serif ones are the easiest.

Even with the outliers, the mean error across the entire dataset was 39% with $T_{distance\,ratio} = 0.0001$ and arcs included as corners. This shows that the algorithm can calculate the contrast quite accurately, except for a few outlier cases where the basic assumption of the algorithm (that the SVG path is not self-intersecting) is not true.

6.1.2 Classifying Serif Type

For the serif classification, the fine-grained image classification model, TransFG, gave quite accurate results with a font-independent test accuracy of 89.4% and 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, reaching an accuracy of 80.0% and an F1-score of 0.83. However, the general image classification model, EfficientNet-B2, did not generalize well to the test set. With a font-independent accuracy of 69.2% compared to the validation accuracy of 96.7%. This shows that the model is "memorizing" the fonts rather than picking up the more subtle underlying serif features. Therefore, it can be concluded that the serif classification problem is better defined as a fine-grained image classification problem.

6.1.3 Typographic Similarity

For the typographic similarity, Variational Auto-Encoders were trained on a dataset of SVG images. A total of six different models were trained with two different label conditions (unconditioned, conditioned on character labels) and three different sizes for the latent dimension $z = (64, 128, 256)$. The results showed that the model conditioned with labels was able to learn to differentiate between serif and sans-serif fonts and group together variations of the same fonts in the latent space. However, the model was not able to differentiate between various types of serifs and ended up grouping together some very different fonts.

Changing the size of the latent dimension did not have a significant impact, which could be due to the fact that the dataset is quite limited (< 25000 images). However, visually slightly better results were obtained with a latent dimension size $z = 64$, but the difference was very subtle. In short, the model is able to learn some useful features and

representation but misses out on some of the finer details.

6.1.4 Typographic Complexity

For typographic complexity, four different feature descriptors (Des-AVG, Des-MAX, Des-STD, Des-ALL) are defined as discussed in chapter 4. Among the four descriptors, there is no clear indication for which one is the best, considering that the responses from the design experts are mixed. However, some trends can be seen. The designers seem to lean towards Des-MAX and Des-ALL, and seem to reject Des-STD. However, the current sample size is not large enough to make any solid conclusions.

6.2. Future plans

Based on the results mentioned above, some future directions of research can be seen. Firstly, with the success of the fine-grained recognition model for serif classification, it would be interesting to see if a transformer-based classifier working on SVG images can be competitive or even better than the model working on raster images. Secondly, with regard to the contrast calculation algorithm, it would be interesting to work towards dealing with the outliers and edge cases where the boundary is not a simple polygon. Additionally, for the typographic similarity, a larger dataset should be used to explore if increase the dataset can result in the model learning a better latent representation and if it is able to better group together various subtle typographic features.

Finally, in the context of the end application of the project, it would be interesting to use these new typographic features like contrast and similarity measures to come up with better descriptors to group together typographically similar posters.

Bibliography

[1] A. Heinze, M. Griffiths, A. Fenton, and G. Fletcher, "Knowledge exchange partnership leads to digital transformation at hydro-x water treatment, ltd.," *Global Business and Organizational Excellence*, vol. 37, no. 4, pp. 6–13, 2018. DOI: 10.1002/joe.21859.

[2] J. Bowen and T. Giannini, "Digitalism: The new realism?," Jul. 2014. DOI: 10.14236/ewic/eva2014.38.

[3] R. LZICAR, R. Lzicar, D. Fornari, C. Delamadeleine, and I. B. Caleffi, *MAPPING DESIGN HISTORY IN SWITZERLAND*. 2016, p. 192.

[4] R. Bringhurst, *The elements of typographic style*, 3rd ed. Hartley and Marks, 2005, p. 32.

[5] *Typography | origin and meaning of typography by online etymology dictionary*. [Online]. Available: https://www.etymonline.com/word/typography#etymonline_v_18894.

[6] B. Schwartz, "The phaistos disk," *Journal of Near Eastern Studies*, vol. 18, no. 2, pp. 105–112, 1959. DOI: 10.1086/371517.

[7] J. Needham and C. A. Ronan, *The shorter science and civilisation in China*. Cambridge University Press, 1994.

[8] J. Needham and T.-h. Tsien, *Chemistry and chemical technology*. Cambridge University Press, 1993.

[9] H.-b. Ch'on, "Typography in korea," *Koreana*, vol. 7, no. 2, pp. 10–19, 1993.

[10] *Typography - wikipedia*, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Typography.

[11] W. Tracy, *Letters of credit: A View of Type Design*, 1st ed. Gordon Fraser, 2003.

[12] T. Meyrick, "Typography and the branding of culture: A methodological investigation into the way typography is used to brand cultural festivals in australia," in *Proceedings of the Inaugural Land Dialogues Conference: Interdisciplinary Research*. 2016.

[13] M. Waldeck, "Typography and national identity," *Blucher Design Proceedings*, vol. 1, no. 5, 2014.

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

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

[16] B. Favre, "Automatic typography analysis on figurative content," Ph.D. dissertation, Ecole Polytechnique de Lausanne, 2021.

[17] N. Srivatsan, J. T. Barron, D. Klein, and T. Berg-Kirkpatrick, *A deep factorization of style and structure in fonts*, 2020. arXiv: 1910.00748 [cs.LG].

[18] D. Kingma and M. Welling, "Auto-encoding variational bayes," Dec. 2014.

[19] Z. Wang, J. Yang, H. Jin, E. Shechtman, A. Agarwala, J. Brandt, and T. S. Huang, *Deepfont: Identify your font from an image*, 2015. arXiv: 1507.03196 [cs.CV].

[20] M. Javed, P. Nagabhushan, and B. B. Chaudhuri, *Automatic detection of font size straight from run length compressed text documents*, 2014. arXiv: 1402.4388 [cs.CV].

[21] 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)*, 2015, pp. 1171–1175. DOI: 10.1109/ICDAR.2015.7333945.

[22] 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)*, vol. 01, 2017, pp. 1355–1360. DOI: 10.1109/ICDAR.2017.222.

[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. arXiv: 1906.10269 [cs.CV].

[24] S. Long, X. He, and C. Yao, "Scene text detection and recognition: The deep learning era," *International Journal of Computer Vision*, vol. 129, no. 1, pp. 161–184, 2020. DOI: 10.1007/s11263-020-01369-0.

[25] B. Shi, X. Bai, and S. Belongie, "Detecting oriented text in natural images by linking segments," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. DOI: 10.1109/cvpr.2017.371.

[26] Y. Jiang, X. Zhu, X. Wang, S. Yang, W. Li, H. Wang, P. Fu, and Z. Luo, *R2cnn: Rotational region cnn for orientation robust scene text detection*, 2017. arXiv: 1706.09579 [cs.CV].

[27] P. Lyu, C. Yao, W. Wu, S. Yan, and X. Bai, "Multi-oriented scene text detection via corner localization and region segmentation," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. DOI: 10.1109/cvpr.2018.00788.

[28] F. Wang, L. Zhao, X. Li, X. Wang, and D. Tao, "Geometry-aware scene text detection with instance transformation network," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. DOI: 10.1109/cvpr.2018.00150.

[29] M. Liao, Z. Zhu, B. Shi, G.-s. Xia, and X. Bai, "Rotation-sensitive regression for oriented scene text detection," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. DOI: 10.1109/cvpr.2018.00619.

[30] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang, "East: An efficient and accurate scene text detector," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. DOI: 10.1109/cvpr.2017.283.

[31] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee, "Character region awareness for text detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9365–9374.

[32] C. Zhang, B. Liang, Z. Huang, M. En, J. Han, E. Ding, and X. Ding, *Look more than once: An accurate detector for text of arbitrary shapes*, 2019. arXiv: 1904.06535 [cs.CV].

[33] W. Wang, E. Xie, X. Song, Y. Zang, W. Wang, T. Lu, G. Yu, and C. Shen, "Efficient and accurate arbitrary-shaped text detection with pixel aggregation network," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. DOI: 10.1109/iccv.2019.00853.

[34] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. G. i. Bigorda, S. R. Mestre, J. Mas, D. F. Mota, J. A. Almazan, and L. P. de las Heras, "Icdar 2013 robust reading competition," *2013 12th International Conference on Document Analysis and Recognition*, 2013. DOI: 10.1109/icdar.2013.221.

[35] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, and S. e. a. Lu, "Icdar 2015 competition on robust reading," *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, 2015. DOI: 10.1109/icdar.2015.7333942.

[36] N. Nayef, F. Yin, I. Bizid, H. Choi, Y. Feng, D. Karatzas, Z. Luo, U. Pal, C. Rigaud, and J. e. a. Chazalon, "Icdar2017 robust reading challenge on multi-lingual scene text detection and script identification - rrc-mlt," *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, 2017. DOI: 10.1109/icdar.2017.237.

[37] M. Al-Rawi, D. Bazazian, and E. Valveny, "Can generative adversarial networks teach themselves text segmentation?" *IEEE Proceedings of International Conference on Computer Vision Workshops*, 2019.

[38] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *ECCV*, 2018.

[39] S. Lee, M. S. Cho, K. Jung, and J. H. Kim, "Scene text extraction with edge constraint and text collinearity," in *2010 20th International Conference on Pattern Recognition*, 2010, pp. 3983–3986. DOI: 10.1109/ICPR.2010.969.

[40] X. Xu, Z. Zhang, Z. Wang, B. Price, Z. Wang, and H. Shi, "Rethinking text segmentation: A novel dataset and a text-specific refinement approach," *arXiv preprint arXiv:2011.14021*, 2020.

[41] Y. Yuan, X. Chen, and J. Wang, "Object-contextual representations for semantic segmentation," 2020.

[42] S. Bonechi, P. Andreini, M. Bianchini, and F. Scarselli, "Coco_ts dataset: Pixel-level annotations based on weak supervision for scene text segmentation," in *Artificial Neural Networks and Machine Learning - ICANN 2019: Image Processing - 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17-19, 2019, Proceedings, Part III*, I. V. Tetko, V. Kurková, P. Karpov, and F. J. Theis, Eds., ser. Lecture Notes in Computer Science, vol. 11729, Springer, 2019, pp. 238–250. DOI: 10.1007/978-3-030-30508-6_20. [Online]. Available: https://doi.org/10.1007/978-3-030-30508-6%5C_20.

[43] S. Bonechi, M. Bianchini, F. Scarselli, and P. Andreini, "Weak supervision for generating pixel–level annotations in scene text segmentation," *Pattern Recognition Letters*, vol. 138, pp. 1–7, 2020, ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2020.06.023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865520302415.

[44] 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)*, vol. 115, no. 3, pp. 211–252, 2015. DOI: 10.1007/s11263-015-0816-y.

[45] M. Tan and Q. V. Le, *Efficientnet: Rethinking model scaling for convolutional neural networks*, 2020. arXiv: 1905.11946 [cs.LG].

[46] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, *Self-training with noisy student improves imagenet classification*, 2020. arXiv: 1911.04252 [cs.LG].

[47] A. Brock, S. De, S. L. Smith, and K. Simonyan, *High-performance large-scale image recognition without normalization*, 2021. arXiv: 2102.06171 [cs.CV].

[48] 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. arXiv: 2010.11929 [cs.CV].

[49] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, *Mobilenetv2: Inverted residuals and linear bottlenecks*, 2019. arXiv: 1801.04381 [cs.CV].

[50] 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*, ser. ICML'15, Lille, France: JMLR.org, 2015, pp. 448–456.

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

[52] 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*, ser. NIPS'17, Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010, ISBN: 9781510860964.

[53] 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.

[54] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," California Institute of Technology, Tech. Rep. CNS-TR-2011-001, 2011.

[55] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei, "Novel dataset for fine-grained image categorization," in *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, 2011.

[56] G. Van Horn, S. Branson, R. Farrell, S. Haber, J. Barry, P. Ipeirotis, P. Perona, and S. Belongie, "Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 595–604. DOI: 10.1109/CVPR.2015.7298658.

[57] G. V. Horn, O. M. Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, and S. Belongie, *The inaturalist species classification and detection dataset*, 2018. arXiv: 1707.06642 [cs.CV].

[58] D. Ha and D. Eck, *A neural representation of sketch drawings*, 2017. arXiv: 1704.03477 [cs.NE].

[59] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, Dec. 1997. DOI: 10.1162/neco.1997.9.8.1735.

[60] 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)*, 2020, pp. 14 141–14 150. DOI: 10.1109/CVPR42600.2020.01416.

[61] 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)*, 2019, pp. 7929–7938. DOI: 10.1109/ICCV.2019.00802.

[62] A. Carlier, M. Danelljan, A. Alahi, and R. Timofte, *Deepsvg: A hierarchical generative network for vector graphics animation*, 2020. arXiv: 2007.11301 [cs.CV].

[63] H. Blum, "A Transformation for Extracting New Descriptors of Shape," in *Models for the Perception of Speech and Visual Form*, W. Wathen-Dunn, Ed., Cambridge: MIT Press, 1967, pp. 362–380.

[64] F. Preparata, "The medial axis of a simple polygon," in. Jan. 2006, vol. 53, pp. 443–450, ISBN: 978-3-540-08353-5. DOI: 10.1007/3-540-08353-7_166.

[65] K. P. F.R.S., "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901. DOI: 10.1080/14786440109462720.