Visual Design Informatics: Revealing Design Tendencies by Machine Learning

原口, 大地

Visual Design Informatics:
Revealing Design Tendencies by Machine Learning

Daichi Haraguchi

Department of Information Science and Technology
Doctor of Philosophy

KYUSHU UNIVERSITY

February 2024

Visual Design Informatics:
Revealing Design Tendencies by Machine Learning

Daichi Haraguchi

Department of Information Science and Technology
February 2024
Doctor of Philosophy

Abstract

This thesis aims to tackle *visual design informatics*, which is a research field analyzing visual designs and revealing unseen design tendencies using informatics approaches. When designers create visual designs, they express additional messages in their designs. For example, designers use "red" color to express a hot impression and "blue" color to express a cool impression. Visual design informatics will discover such tendencies and lead to understanding implicit designers' knowledge.

In this thesis, I analyze two types of visual designs: font images and lyric videos. In font image analysis, I propose and tackle a novel font identification task, which identifies whether a pair of font images come from the same font or not. I first use a convolutional neural network to confirm that the font identification task is solvable by machine learning. Then, I use another neural network model, called vision transformer, for the identification task. A merit of using the vision transformer is that it can reveal the image regions that characterize the font style. I call this characterization a local style awareness and use it to another task, font generation. In the generation task, local style awareness plays a role in emphasizing the detailed font style.

I also tackle a novel task of lyric video analysis. A lyric video is a music video where lyrics are displayed and animated synchronously with the music. Creating lyric videos requires considering the relationship between the graphical and musical expressions. To understand the relationship, I analyze the correlation between font style, word motion, and music style in lyric videos. To analyze these three modalities, extraction of each style feature is necessary. Therefore, I develop a font-style extractor and a word motion tracker. Additionally, I utilize a music-style estimator to extract music-style features. My experimental result of the lyric video analysis showed slight yet interesting correlations between each modality. For example, I could see a correlation between "Fancy" and "Sans-Serif" fonts and active word motions.

Acknowledgments

I would like to thank my supervisor, Professor Seiichi Uchida, for his guidance and encouragement. Additionally, I appreciate my committee members, Professor Shimada, and Professor Bise, for their help with the supervision of this thesis. Besides my supervisor and committee members, I also sincerely appreciate another faculty member, Professor Brian Kenji Iwana, for collaborating with my research and giving me a lot of advice. Thanks to them, I could write this thesis. Finally, I am grateful for all lab members in the Human Interface laboratory.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Our daily life is surrounded by various visual designs or graphic designs. An example is font; we can find various 'A's printed in different styles. Every day, new font designs are created by type-designers in the world. Another example is a logo. Companies and commercial products often have their own logo as their representative symbol. Logos are also designed by special designers and sometimes renewed according to the change in company policies and/or display devices.

We can also find compounded visual designs. For example, posters and web advertisements, i.e., typographic designs, comprise text lines in different fonts, photographs, and graphic elements (such as boxes and frames). Those components are carefully laid out spatially with different colors and sizes. Moreover, the designs of individual components are also carefully correlated to each other. For example, as the color of a text line on a black background box, dark colors will not be chosen — the designers will be careful of color contrast between the text and its background.

Videos can be a rather new modality of visual designs. They are often "moving typography." Lyric videos are good examples of lyric words being shown (while often moving) on graphical artwork. In lyric videos, the music itself is also a design

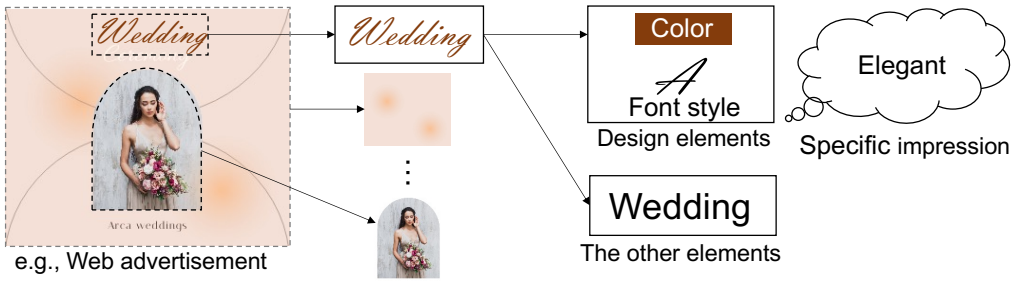


Figure 1-1: Visual designs often consist of a combination of several visual designs. In particular, design elements express the impression that the designer wants to convey. The other elements express the main messages.

modality. Therefore, lyric video creators must consider the relationship between the graphical and musical expressions while creating the videos.

One important observation is that visual designs give specific impressions. For example, in a web advertisement shown in Figure 1-1, the design elements in the text design, such as color and font style, express an elegant impression that the designers want to convey. Therefore, focusing on the analysis of design elements is essential to understanding the tendencies of visual designs.

Analyzing visual designs will help in understanding how designers create their designs. As I mentioned above, designers create visual designs expressing specific impressions in the designs. Analyzing design tendencies leads to understanding designers' knowledge objectively and quantitatively. Therefore, visual design analysis helps non-expert designers understand designers' knowledge and incorporate design tendencies into their products or works.

Visual design analyses have been conducted in various research fields. Previous visual design analyses have generally been conducted in affect engineering and psychology [3, 4, 5]. In these analyses, the number of subjects is often limited; therefore, the experimental results might not be objective enough. In recent years, visual design analyses have been conducted with informatics approaches [6, 7, 8]. This is due to the availability of large-scale datasets and the development of machine-learning techniques. These analyses can find more general and objective tendencies than previous

approaches. In this thesis, I define the latter visual design analysis using informatics approaches as visual design informatics. Note that there is still room for further analysis because the types of visual designs are too varied, and there is no established analysis method.

1.2 Challenges in Visual Design Informatics

In this thesis, I tackle visual design informatics, which is a research field analyzing visual designs using informatics approaches. The purpose of visual design informatics is to quantitatively and objectively reveal designers' knowledge and design tendencies that have not been discovered yet.

To this end, I tackle three tasks in visual design informatics, as shown in Figure 1-2.

(a) *Feature extraction of design elements.* As I mentioned above, design elements convey specific impressions that designers want to convey. Therefore, analyzing design elements is important. To this end, we first need to extract design elements independent of the other elements by using machine learning or image processing approaches.

(b) *Analysis of design tendencies.* Design tendencies can be revealed by analyzing design elements. In this thesis, I reveal design tendencies from only the design elements themselves. In this approach, I consider analyzing the design elements using unsupervised (or self-supervised) approaches (e.g., clustering) or conducting correlation analysis (e.g., bi-clustering) on several design elements. Note that I consider other approaches using metadata (e.g., genre, impression, etc.). This approach is out of scope in this thesis. Visual design informatics in this approach can be seen in [9, 10].

(c) *Visual design generation by using design tendencies.* As an application task, I

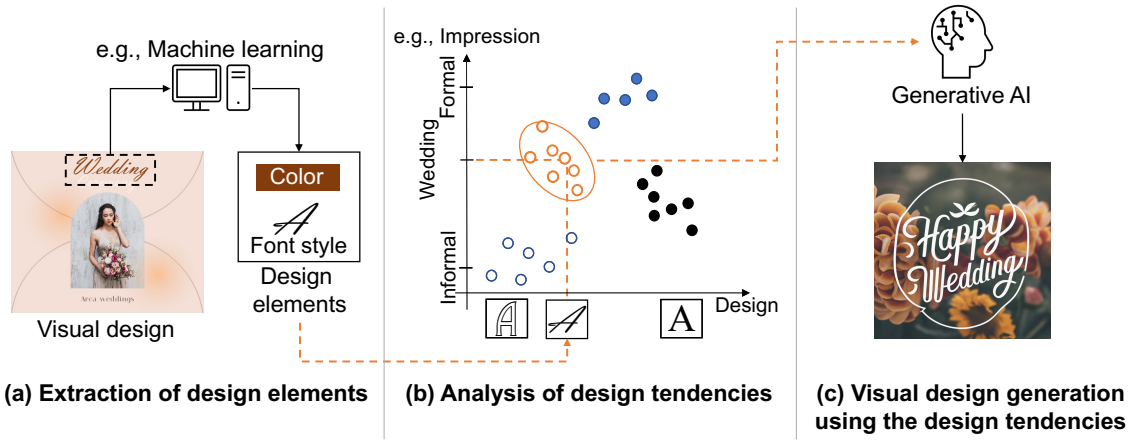


Figure 1-2: Three tasks in visual design informatics, including an application task.

conduct visual design generation utilizing the result of visual design analysis. By quantitatively analyzing the design tendencies in (b), I can easily integrate the analysis results into generation models of visual designs. By integrating the results, the generation model can focus on detailed designs or design tendencies. It contributes to creating a generative model specialized for specific visual design generation and thus improving the performance of generating visual design.

In this thesis, I tackle visual design informatics, focusing on two types of visual designs: font images and lyric videos. Font images are the simplest visual design because font images consist of a single design element of font style and another design element (i.e., character shape). However, font image analysis is challenging because the character shape is more dominant than the font style in its looks. For this analysis, I conduct three tasks of visual design informatics: (a) extraction of font style, (b) analysis of the parts that represent font style, and (c) application task of font generation related to Figure 1-2.

A lyric video is a music video where lyrics are displayed and animated synchronously with the music. It is difficult to address all modalities because lyric videos contain various visual designs. Therefore, I focus on three modalities: font style, word motion, and music style, which are the most important design elements of lyric

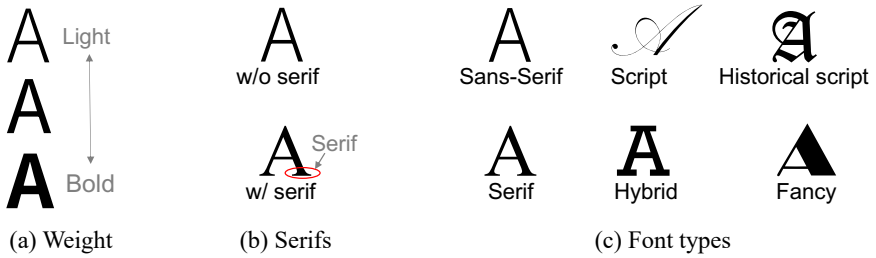


Figure 1-3: Several examples of font styles. Font types are defined in [1].

videos. For this analysis, I conduct (a) extraction of style features from each modality and (b) correlation analysis between each design modality related to Figure 1-2.

1.3 Motivation and Purpose

1.3.1 Font Style Analysis through Font Identification

I analyze one of the simplest visual designs, font images, to understand subtle yet important parts representing font style. A font is a graphic representation defined by its outline shape without intricate coloring or texture, as shown in Figure 1-3. Despite this simplicity, minute differences in shape, such as the presence or absence of serifs (Figure 1-3(b)) and controlled curvature, contribute to the diversity of fonts. Interestingly, these subtle variations substantially influence readability and overall impression.

In this thesis, I propose and tackle a novel font identification task that identifies whether two font images come from the same font or not. I first use a convolutional neural network (CNN) to confirm that the font identification task is solvable by machine learning. Then, I use another neural network model, called vision transformer (ViT), for the identification task. A merit of using ViT is that it can reveal the image regions that characterize the font style. Font identification is relevant to Figure 1-2 (a).

As a font style analysis, I visualize the important parts representing a font style. By utilizing the merit of ViT, I can easily visualize such parts. I call this visualization of font style as local style awareness. This is relevant to Figure 1-2 (b).

As an application task, I conduct font generation using local style awareness. I utilize local style awareness to the weight of reconstruction loss of several font generation models. Thanks to the weighting, I can improve the performance of the font generation models. This is relevant to Figure 1-2 (c).

1.3.2 Multi-Modal Design Analysis on Lyric Videos

I analyze lyric videos consisting of a combination of multiple visual designs to address more complicated visual designs than font images. In this analysis, I focus on the three most important design modalities in lyric videos: font style, word motion style, and music style. A lyric video is a music video where lyrics are displayed and animated synchronously with the music. Creating lyric videos requires considering the relationship between the graphical and musical expressions. To understand the relationship, I conduct a correlation analysis between each design modality.

There are mainly two purposes for conducting the correlation analysis. First, it is expected that the analysis will lead to a deeper understanding of the typographic designs of lyric videos. This analysis will provide hints as to how experts can use their knowledge of typography in music videos. Second, the relationships revealed by the analysis will help non-experts create lyric videos or help in the development of lyric video creation tools such as TextAlive [11]. The relationships can also be helpful to suggest suitable font styles for specific music styles.

For the analysis, first, I conduct feature extraction from each modality. I develop a font style extractor and a lyric word tracker to extract font style and word motion style. To extract music style, I utilize a music style estimator [2, 12]. This is relevant to Figure 1-2 (a).

In the analysis, I conduct co-clustering between each modality. As a result, I

reveal weak but interesting tendencies. For example, the result shows a correlation between "Fancy" and "Sans-Serif" fonts and active word motions. This is relevant to Figure 1-2 (b).

1.4 Contributions

Contributions of this thesis are as follows.

- I propose a novel font identification task, which identifies whether two font images come from the same font. Font identification is challenging due to the differences between characters generally being greater than the differences between font styles. However, the proposed neural network models can identify the unseen font with more than 90 % accuracy. This is despite using different characters as representatives of their font. This indicates that the proposed models can capture font styles independent of their character shapes.

- I unveil the key parts representing a font style and visualize such parts as local style awareness of font images. Local style awareness is acquired by solving a font identification task. The proposed model is based on ViT instead of CNN. The attention mechanism of ViT helps us to have finer local style awareness that can catch a small style structure such as serifs.

- As an application task, I utilize local style awareness to generate font images whose local structures are realized more accurately. In this application, I simply use the local style awareness as the weight for the reconstruction loss function. This simplicity allows me to use the local style awareness in various state-of-the-art font generation models. In my experiments, I prove that the proposed loss can improve the performance of three baseline models quantitatively and qualitatively.

- I tackle the novel task of lyric video analysis to understand the relationships

between three style modalities: word motion, font style, and music style. To conduct this analysis, I develop an original font style estimator and a lyric word tracker. Additionally, I extract music style features utilizing music style estimator [2, 12].

- I conduct a clustering-based co-occurrence analysis of the style modalities from 100 lyric videos. The results indicated several tendencies in the style combinations. For example, I could see a correlation between "Fancy" and "Sans-Serif" fonts and active word motions. I can catch such tendencies in the videos in an objective and reproducible manner without manual annotations.

1.4.1 Organization of Thesis

Chapter 2 explains a font identification task, which identifies whether the input image comes from the same font or not. Section 2.1 shows the background and purpose of font identification and related work of identification tasks. Section 2.2 shows related work of font identification and other identification tasks. Section 2.3 shows the proposed method to solve the font identification task. Section 2.4 shows the experimental result of font identification quantitatively and qualitatively. This part corresponds to the contents of [13].

Chapter 3 explains font identification by ViT and visualization of important parts that represent font style, called local style awareness. Section 3.1 shows the background and purpose. Section 3.2 shows related work. Section 3.3 shows the method and the experiments of extracting local style awareness. Section 3.4 shows font generation by utilizing local style awareness. This part corresponds to the contents of [14].

Chapter 4 explains lyric video analysis focusing on three modalities: font style, word motion, and music style. Section 4.1 shows the background and purpose of lyric video analysis. Section 4.2 shows related work. Section 4.3 shows the lyric video

dataset. Section 4.4 shows the feature extraction of each modality. Section 4.5 shows the correlation analysis between each modality in lyric videos. This part corresponds to the contents of [15].

Lastly, Chapter 5 describes the conclusion and future works of this thesis.

Chapter 2

Character-Independent Font Identification

2.1 Background and Purpose

In this chapter, I tackle a novel font identification across different character classes. The font identification is a task that discriminates whether the two given font images come from the same font or not. Figure 2-1 (a) shows examples of input pairs for the font identification. It is easy to identify that Figure 2-1 (b) is the same font pair. It is also easy to identify that Figure 2-1 (c) is a different font pair. In contrast, the examples in Figure 2-1 (d) and (e) are more difficult. Figure 2-1 (d) shows the same font pairs, whereas (e) shows different font pairs.

This task is very different from the traditional font identification task, such as [16, 17, 18]. In the traditional task, given a character image (a single character image, a single word image, or a sentence image), we need to answer the font name (e.g., Helvetica) and the name of the font type (e.g., Blackletter [1]). In a sense, it is rather a classification task than an identification task. This is because, in the traditional task, we can only identify the fonts that are registered in a system in advance. In

[1] Blackletter is a finer font type that is classified as Historical script shown in Figure 1-3 (c).

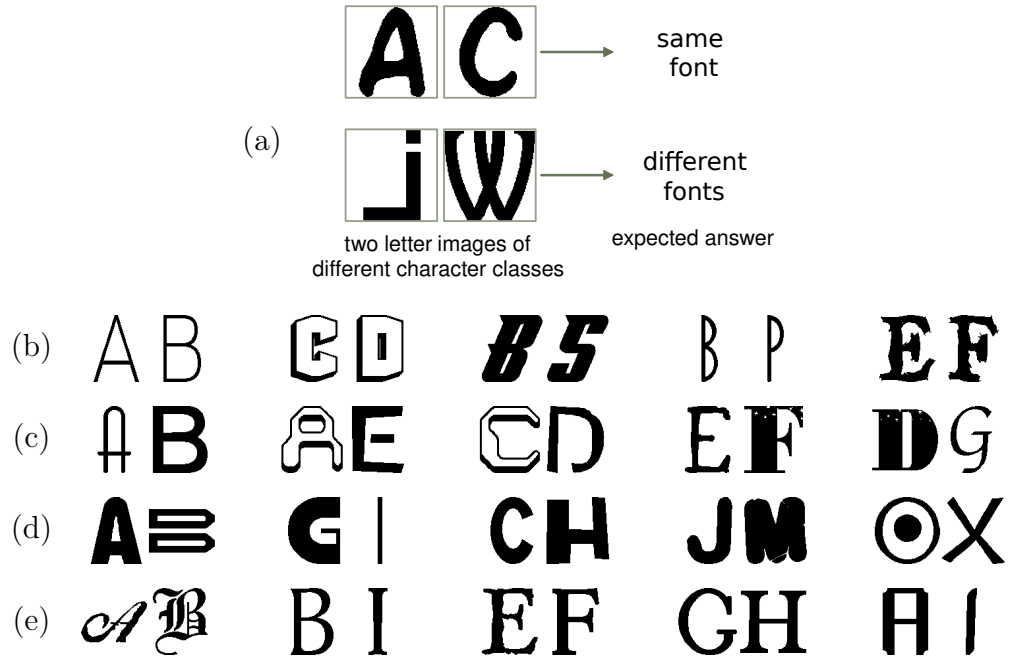


Figure 2-1: Explanation of my task and examples of character image pairs from different classes. (a) is an explanation of my task. The pairs in (b) show the same font, whereas (c) shows different font pairs. In contrast, (d) and (e) are difficult cases; (d) shows the same pairs, whereas (e) shows different pairs.

other words, it is a multi-class font recognition (i.e., classification) task, and each class corresponds to a known font name. In contrast, my font identification task is a two-class task to decide whether a pair of character images come from the same font or different fonts without knowing those font names beforehand.

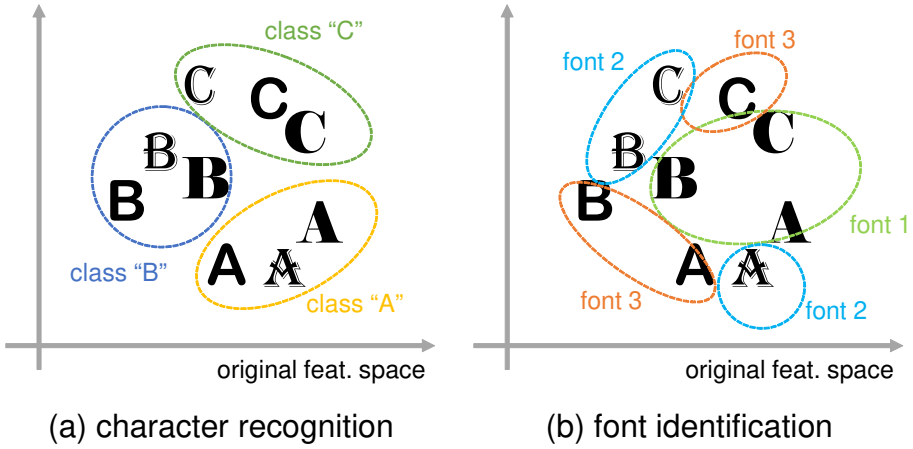
To address this issue, I propose a method to solve the proposed font identification task, including a pair of different character classes. The proposed method is practically useful because the method will have more flexibility than the method for the traditional font identification task. As noted above, the traditional methods only "recognize" the input image as one of the fonts that are known to the methods. However, it is impossible to register all fonts to the method because new fonts are generated every day in the world. (In the future, the variations of fonts will become almost infinite since many automatic font generation systems have been proposed,

such as [19, 20, 21, 22, 23, 24]. See in Section 3.2.1 for more details about font generation systems.) Accordingly, traditional systems will have a limitation in dealing with those fonts that are "unknown" to them. Since the proposed method does not assume specific font classes, it can deal with arbitrary font images. This property is significant for analyzing unknown fonts.

Moreover, since the proposed method assumes a pair of single-character images as its input, I can perform font identification even if a document contains a small number of characters. For example, analysis of incunabula or other printed historical documents often needs to identify whether two pieces of documents are printed in the same font or not. A similar font identification task from a limited number of characters can be found in forensic research. For example, forensic experts need to determine whether two pieces of documents are printed by the same printer or not.

In addition to the above practical merits, the proposed font identification is a challenging scientific task. Even though the proposed font identification is formulated as just a binary classification problem, the task remains difficult. Figure 2-2 illustrates the distribution of image samples in a feature space. As the success of multi-font optical character recognition (OCR) [25] proves, the samples from the same character class form a cluster, and the clusters of different character classes are distant in the feature space. This is because inter-class variance is much larger than intra-class variance; that is, the discrepancy among the character classes is larger than the difference by the fonts. This fact can be confirmed by imagining the template matching-based identification. Although I can judge the class identity of two images (in different fonts) even by template matching, I can not totally judge the font identity of two images (in different character classes). Consequently, the proposed method needs to disregard large differences between character classes and emphasize tiny differences (such as the presence or absence of serif) in fonts. I find a similar requirement in the text-independent writer identification task, such as [26].

In Chapter 2, I experimentally show that even a simple two-stream convolutional



(a) character recognition (b) font identification

Figure 2-2: Comparing multi-font character recognition (a), my font identification (b) is a more difficult classification task.

neural network (CNN) can achieve high accuracy for my font identification task in spite of the above-anticipated difficulty. The proposed CNN is not very modern (like a CNN with a feature disentanglement function [27, 28, 29]) but simply accepts two-character image inputs and makes a decision for the binary classification (i.e., the same font or not). In addition, I show a detailed analysis of the identification results. For example, I will observe which alphabet pairs (e.g., 'A'-'K') are easier or more difficult for identification. Though there is a difference in the font identification performance among alphabet pairs, the proposed method has enough potential to identify unknown fonts. This indicates that the proposed method can capture font style independent of characters.

The main contributions of this chapter are summarized as follows:

- To the best of my knowledge, this is the first attempt at font identification for different character classes.

- Through a large-scale experiment with more than 6,000 different fonts, I prove that even a simple two-stream CNN can judge whether two-character images come from the same font or not with high accuracy ($> 90\%$), in spite of the

essential difficulty of the task. It is also experimentally shown that trained CNN has a generalization ability. This means that the representation learning by the simple CNN is enough to extract font style features while disregarding the shape of the character class.

- By analyzing the experimental results, I prove the identification accuracy is dependent on the character class pairs. For example, 'R' and 'U' are a pair with high accuracy, whereas 'I' and 'Z' have a lower accuracy.

2.2 Related Work

2.2.1 Font Identification and Recognition

To the best of my knowledge, this is the first trial of font identification in my difficult task setting. Most past research on font identification is font recognition (i.e., classification), where a set of fonts are registered with their names, and an input character image is classified into one of those font classes. These systems traditionally use visual features extracted from characters. For example, Ma and Doermann [17] use a grating cell operator for feature extraction, and Chen et al. [30] use a local feature embedding. In addition, visual font recognition has been used for text across different mediums, such as historical documents [18] and natural scene text [30]. Font recognition has also been used for non-Latin characters, such as Hindi [31], Farsi [32], Arabic [33, 34], Korean [35], Chinese [36], etc. Recently, neural networks have been used for font identification. DeepFont [16] uses a CNN-based architecture for font classification.

However, these font identification methods classify fonts based on a set number of known fonts. In contrast, the proposed method detects whether the fonts come from the same class or not, independent of known or unknown fonts. This means that the proposed method can be used for fonts that are not in the dataset, which can be an

important task, given the growing popularity of font generation [37, 19, 22, 23, 24] (See in Section 3.2.1 for more details).

In order to address unknown fonts, an alternative approach would be only to detect particular typographical features or groups of fonts. Many classical font recognition models use this approach and detect typographical features such as typeface, weight, slope, and size [38, 39, 40]. In addition, clustering has been used to recognize groups of fonts [41, 42].

2.2.2 Other Identification Tasks in Document Analysis and Recognition Field

The task of font identification can be considered as a subset of script identification. Script identification is a well-established field that aims to recognize the script of text, namely, the set of characters used. In general, these methods are designed to recognize the language for individual writing-system OCR modules [43]. Similar to font identification, traditional script identification uses visual features such as Gabor filters [44, 45] and text features [46, 47].

Furthermore, font identification is related to the field of signature verification and writer identification. In particular, the task of the proposed method is similar to writer-independent signature verification in that both determine if the text is of the same source or different sources. Notably, there are methods in recent times that use CNNs [48, 49] and Siamese networks [50, 51] that resemble the proposed method.

2.3 Font Identification by Convolutional Neural Networks

Given a pair of the character images $\mathbf{x}_c$ and $\mathbf{x}_d$ of font class c and d, respectively, my task is to determine whether the pair of characters come from the same font ($c = d$)

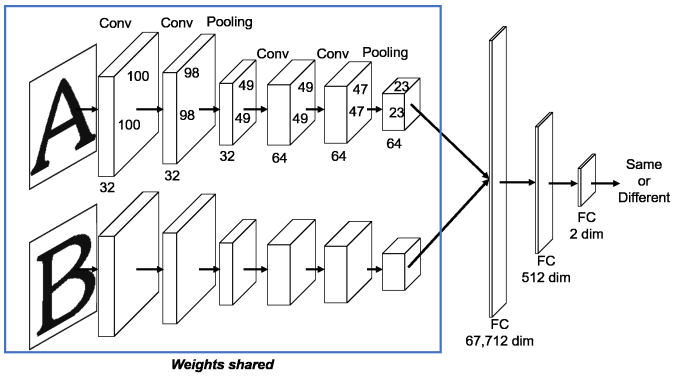


Figure 2-3: Structure of the neural networks for font identification.

or different fonts ($c \neq d$). In this way, the classifier assigns a binary label indicating a positive match and a negative match. The binary label is irrespective of the character or actual font name of the character used as an input pair.

In order to perform font identification, I proposed a two-stream CNN-based model. As shown in Figure 2-3, a pair of input characters are fed to separate streams of convolution layers, which are followed by fully connected layers and then the binary classifier. In addition, the two streams of convolution layers have the same structure and shared weights. This is similar to a Siamese network [52], typically used for metric learning due to the shared weights. However, it differs in that I combine the streams before the fully connected layers and have a binary classifier with cross-entropy loss.

Each stream consists of four convolution layers and two max-pooling layers. The kernel size of the convolutions is 3×3 with stride 1, and the kernel size of the pooling layers is 2×2 with stride 2. The features from the convolutional layers are concatenated and fed into three fully-connected layers. Rectified Linear Unit (ReLU) activations are used for the hidden layers, and softmax is used for the output layer. During training, dropout with a keep probability of 0.5 is used after the first pooling layer and between the fully-connected layers.

2.4 Quantitative and Qualitative Evaluation of Font Identification

2.4.1 Font Dataset

The dataset used for the experiment was 6,628 fonts from the Ultimate Font Download[2]. Although the total font package is originally comprised of about 11,000, I removed "dingbat" fonts (i.e., icon-like illustrations and illegible fonts) for the experiments, and the 6,628 fonts remain. This font dataset still contains mainly fancy fonts; I discuss another dataset with more formal fonts in Section 2.4.4. To construct the dataset, I rasterized the 26 uppercase alphabet characters into 100×100 binary images. I only used uppercase characters in this section for experimental simplicity. Although, it should be noted that several fonts contain lowercase character shapes as uppercase characters.

The 6,628 fonts were divided into three font-independent sets: 5,000 for training, 1,000 for validation, and 628 for testing. Within each set, I generated uppercase alphabet pairs from the same font (positive pairs) and different fonts (negative pairs). Each of the pairs contains different alphabetical characters. Furthermore, each combination of characters is only used one time, i.e., either A'-'B' or 'B'-'A' is used, not both. Therefore, I made ${}_{26}C_2 = 325$ total pairs of each font. Consequently, the training set has $5,000 \times 325 \approx 1.60 \times 10^6$ positive pairs. An equal number of negative pairs were generated by randomly selecting fonts within the training set. Using this scheme, I also generated $2 \times 3.25 \times 10^5$ for validation and approximately $2 \times 2.04 \times 10^5$ for testing. In addition, as outlined in Section 2.4.4, a second experiment was performed on an external dataset to show the generalization ability of the trained model on other fonts.

[2] `http://www.ultimatefontdownload.com/`

Table 2.1: Confusion matrix of the test set.

GT\predicted	same	different
same	$196,868 \pm 1,758$	$7,232 \pm 1,758$
different	$24,331 \pm 2,014$	$179,769 \pm 2,014$

2.4.2 Quantitative Evaluation

I conducted a 6-fold cross-validation to evaluate the accuracy of the proposed CNN. The identification accuracy for the test set was $92.27 \pm 0.20\%$. The high accuracy demonstrates that it is possible for the proposed method to determine if the characters come from the same font or not, even when they come from different characters. Table 2.1 shows a confusion matrix of the test results. From this table, it can be seen that different font pairs have more errors than the same font pairs. This means that similar but different font pairs are often misidentified as the same font.

I found that the difficulty of font identification depends on the character pairs. As shown in Figure 2-4, the pairs including 'I' or 'J' are more difficult. This is because 'I' and 'J' do not have distinctive style features due to their simplicity.

Additionally, I found that character pairs with similar features are predictably easier to differentiate, and character pairs with different features are difficult. In other words, the amount of information that characters have, such as angles or curves, is important for separating matching fonts and different fonts. For example, in Figure 2-4, the number of misidentifications of the 'I'-'T' pair is the lowest of any pair combination which includes an 'I' because 'T' has a similar shape to 'I.' I also find that the number of misidentifications for 'D,' 'K,' 'R,' and 'U' are the least because they have the most representative features of straight lines, curves, or angles.

This is consistent with other characters with similar features. The character pair with the worst classification rate is 'I'-'Z,' and the character pair with the highest accuracy is 'R'-'U,' as outlined in Figure 2-5. From this figure, I can see that many characters with similar features have high accuracies. For example, 'B'-'P,' 'B'-'D,'

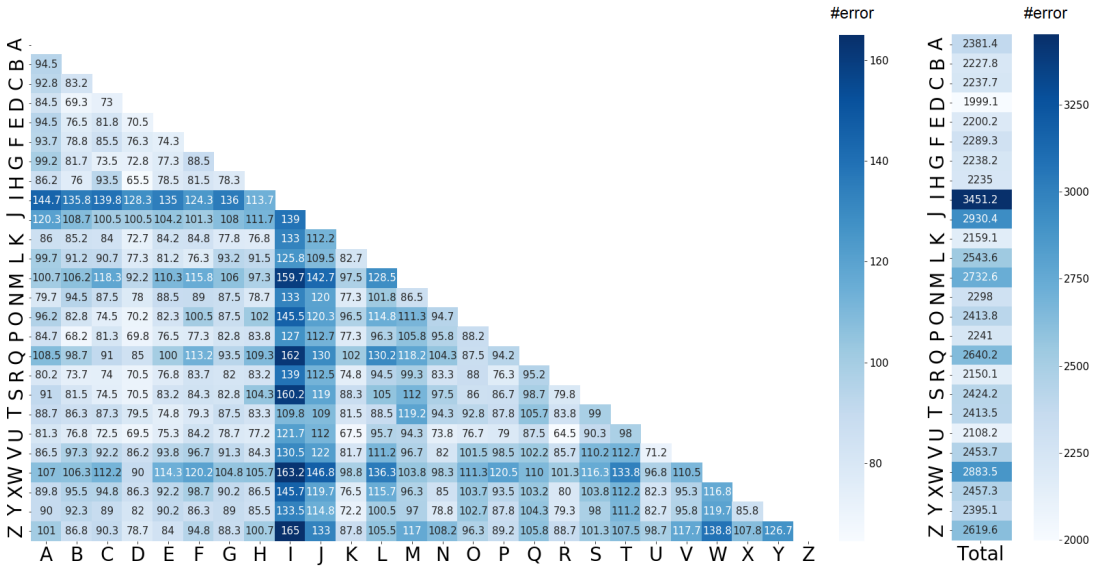


Figure 2-4: Misidentification by class. The left figure shows the number of misidentifications by each pair. The right figure shows the number of misidentifications by each character.

The 20 worst
‘I’ - ‘Z’, ‘I’ - ‘W’, ‘I’ - ‘Q’, ‘I’ - ‘S’, ‘I’ - ‘M’, ‘J’ - ‘W’, ‘I’ - ‘X’, ‘I’ - ‘O’, ‘A’ - ‘I’, ‘J’ - ‘M’, ‘C’ - ‘I’, ‘I’ - ‘J’, ‘I’ - ‘R’, ‘W’ - ‘Z’, ‘L’ - ‘W’, ‘G’ - ‘I’, ‘B’ - ‘I’, ‘E’ - ‘I’, ‘T’ - ‘W’, ‘I’ - ‘Y’

The 20 best
‘R’ - ‘U’, ‘D’ - ‘H’, ‘K’ - ‘U’, ‘B’ - ‘P’, ‘B’ - ‘D’, ‘D’ - ‘U’, ‘D’ - ‘P’, ‘D’ - ‘O’, ‘D’ - ‘S’, ‘D’ - ‘R’, ‘D’ - ‘E’, ‘U’ - ‘V’, ‘K’ - ‘Y’, ‘C’ - ‘U’, ‘D’ - ‘K’, ‘D’ - ‘G’, ‘C’ - ‘D’, ‘C’ - ‘G’, ‘B’ - ‘R’, ‘N’ - ‘U’

Figure 2-5: The character pairs with the 20 worst and 20 best accuracies.

and ‘O’-‘D.’ As a whole, ‘C’-‘G’ and ‘U’-‘V’ pairs have fonts that are easy to identify. These pairs are not likely to be affected by the shape of the characters.

Interestingly, the top 5 easiest characters paired with ‘B’ for font identification are ‘P,’ ‘D,’ ‘R,’ ‘H,’ and ‘E’ and the top 5 for ‘P’ are ‘B,’ ‘D,’ ‘R,’ ‘E,’ and ‘F.’ In contrast, the top 5 easiest font identifications with ‘R’ are ‘U,’ ‘D,’ ‘B,’ ‘C,’ and ‘K.’ ‘B’ and ‘P’ have the same tendency when identifying fonts. However, font identification with ‘R’ seems to use different characteristics despite ‘B,’ ‘P,’ and ‘R’ having similar shapes. This is because ‘B’ and ‘P’ are composed of the same elements, curves, and a vertical line, whereas ‘R’ has an additional component.

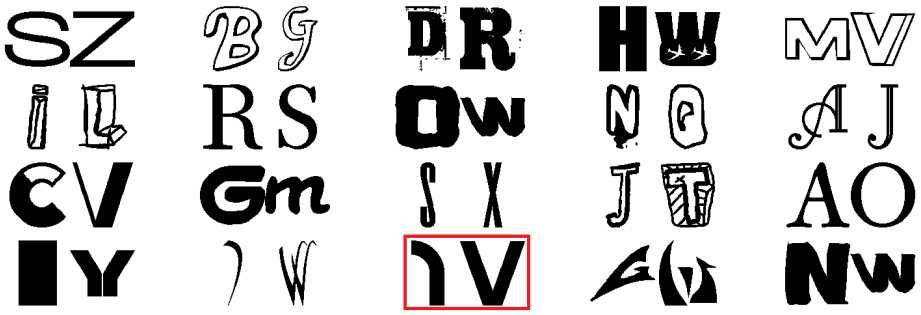


Figure 2-6: Examples of correctly identified pairs (GT: same → prediction: same). The font pair marked by the red box has a nonstandard character.

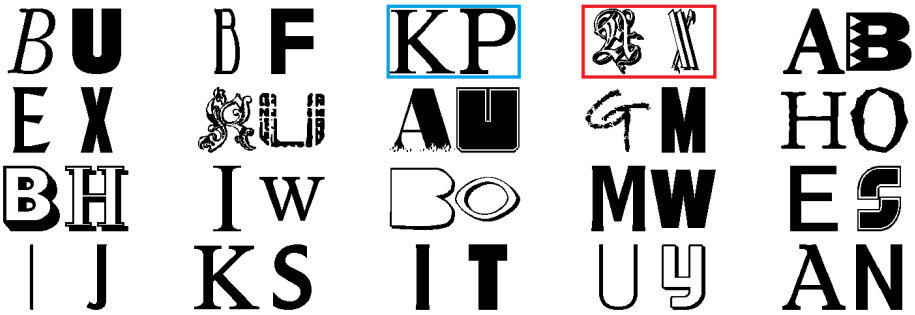


Figure 2-7: Examples of correctly identified pairs (GT: different → prediction: different). The font pairs marked by the blue boxes have similar but different fonts. The red box indicates fonts that are almost illegible.

2.4.3 Qualitative Evaluation

I show some examples of correctly identified pairs in Figure 2-6. In the figure, the proposed method is able to identify fonts despite having dramatically different features, such as different character sizes. However, the weight of the correctly identified fonts tends to be similar (refer to Figure 1-3(a) for about the weight of fonts). Also notably, in Figure 2-6, in the 'A'-'O' pair, although the 'O' does not have a serif, the proposed method is able to identify them as the same font. Furthermore, the character pair highlighted by a red box in Figure 2-6 is identified correctly. This is surprising due to the first character being unidentifiable and not typical of any character. This reinforces that the matching fonts are determined heavily by font-weight.

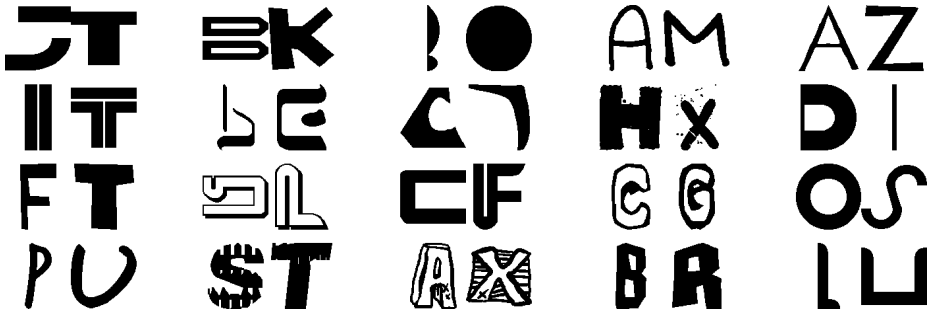


Figure 2-8: Examples of misidentified pairs (GT: same → prediction: different).

Figure 2-9: The font with the most identification errors (GT: same → prediction: different).

It is also easy for the proposed method to correctly identify different font pairs that have obviously different features from each other. Examples of different font pairs that are correctly identified are shown in Figure 2-7. Almost all of the pairs have different features like different line weights or the presence of serif. On the other hand, the proposed method was also able to distinguish fonts that are similar, such as 'K'-'P' highlighted by a blue box.

There are also many examples of fonts that are difficult with drastic intra-font differences. For example, Figure 2-8 shows examples of fonts that had the same class but were predicted to be from different classes. Some of these pairs are obviously the same fonts, but most of the pairs have major differences between each other, including different line weights and different shapes. In particular, the font in Figure 2-9 is difficult as there is seemingly no relation between the characters. This font had the lowest accuracy for the proposed method.

There are several fonts that look similar, although they are different. Therefore, it is difficult to identify such fonts with the proposed method. Figure 2-10 shows

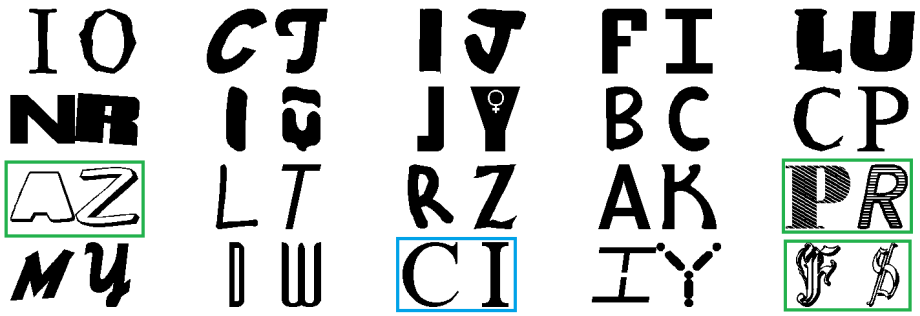


Figure 2-10: Examples of misidentified pairs (GT: different→ prediction: same). The green boxes indicate font pairs, which are outline fonts, and the font pair with the blue box is the font is difficult even for humans.

examples of font pairs that are misidentified as the same font, although they are actually different fonts. These fonts are very similar to each other. It is also difficult even for us to identify as different. Their font pairs have similar features, including line weights, slant lines, and white areas.

2.4.4 Font Identification Using a Dataset with Less Fancy Fonts

The dataset used in the above experiment contains many fancy fonts, and thus, there was a possibility that my evaluation might overestimate the font identification performance; this is because fancy fonts are sometimes easy to identify by their particular appearance. I, therefore, use another font dataset, called the Adobe Font Folio 11.1[3]. From this font set, I selected 1,132 fonts, which are comprised of 511 serif fonts, 314 sans-serif fonts, 151 serif-sans hybrid fonts, 74 script fonts, 61 historical script fonts, and (only) 21 fancy fonts. Note that this font type classification for the 1,132 fonts is given by [1]. I used the same neural network trained by the dataset of Section 2.4.1, i.e., trained with the fancy font dataset and tested on the Adobe dataset. Note that for the evaluation, 367,900 positive pairs and 367,900 negative pairs are prepared using the 1,132 fonts. Using the Adobe fonts as the test, the identification accuracy was 88.33±0.89%. This was lower than 92.27% of the original dataset. However,

[3]https://www.adobe.com/jp/products/fontfolio.html

considering the fact that formal fonts are often very similar to each other, I can still say that character-independent font identification is possible even for formal fonts.

2.5 Summary

Character-independent font identification is a challenging task due to the differences between characters generally being greater than the differences between fonts. Therefore, I proposed the use of a two-stream CNN-based method, which determines whether two characters are from the same font or different fonts. As a result, I was able to demonstrate that the proposed method could identify fonts with an accuracy of $92.27 \pm 0.20\%$ using 6-fold cross-validation. This is despite using different characters as representatives of their font.

Furthermore, I performed qualitative and quantitative analyses of the results of the proposed method. Through the analysis, I found that there are differences in identification accuracy between character pairs. This is due to certain characters containing information about the font within their native features. Additionally, I found that the proposed method could not identify the same fonts without common features.

In the next chapter, I utilize the proposed font identification to extract font styles and analyze the important parts that represent font style.

Chapter 3

Local Style Awareness of Font Images

3.1 Background and Purpose

To understand font styles, a reasonable choice is to observe local shapes. Each character has a shape representing font style; however, the whole character shape is unnecessary to understand its style. Assume that a character 'A' is printed with `Helvetica` (a famous sans-serif font), and we want to understand the style of `Helvetica` from it. In this case, we must ignore the global shape that makes 'A' as 'A.' In other words, we need to focus on local shapes, such as serifs, corners, stroke width, and local curvatures, which are rather independent of character class 'A.'

Through a *contrastive learning* scheme, this chapter tries to determine *local style awareness* representing important local shapes for particular font styles. Imagine a person who has only seen `Helvetica` in his/her lifetime — then, the person cannot determine the style of `Helvetica`. In other words, we can understand the particular style of `Helvetica` by contrasting (i.e., comparing) it with other fonts, such as `Times New Roman` and `Optima`. Moreover, as noted above, we need to ignore the whole letter shape and focus on local shapes during the comparison in some automatic way.

Figure 3-1 shows the overall structure of the proposed model to determine local style awareness. This figure also shows a heatmap representing local style awareness

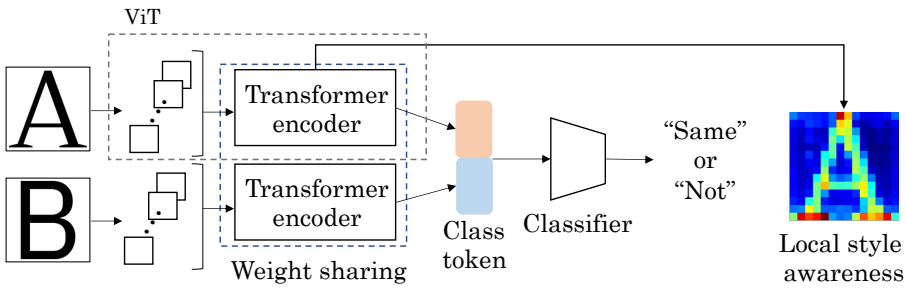


Figure 3-1: Overview of the proposed technique to determine *local style awareness*, which indicates important local shapes to describe font styles. The local style awareness is obtained as a fine attention map through a contrastive learning scheme that identifies whether two given character images belong to the same font. Note that local style awareness can be extracted from one input image, not a paired image.

for the input 'A.' This model is trained via a *font identification* task, as mentioned in Chapter 2. This task aims to determine whether two given character images come from the same font. In the case of this figure, a serif-style 'A' and a sans-serif 'B' are given, and therefore, the model must answer "Not." To answer this task correctly, the model needs to ignore the whole shapes of 'A' and 'B' and enhance their local style differences; therefore, the model needs to determine the local style awareness internally. By visualizing this internal representation as a spatial map, I will have local style awareness.

The following two points must be considered for determining local style awareness via the font identification task. First, the task is implicitly formulated as a contrastive learning task. As noted above, font style is determined by comparing the target font with other fonts. Therefore, the model of Figure 3-1 is trained to enhance local style differences. Second, I need to compare two different alphabets, such as 'A' and 'B,' instead of the same alphabet, such as 'A' and 'A.' If I only compare the same alphabet in the font identification task, it reduces to a trivial task. The model can give perfect identification results by checking whether two inputs are entirely the same or not. In other words, the model cannot learn the local style awareness. By training the model with font pairs including different alphabets, the model can learn local style differences while ignoring the global letter shapes. Additionally, font style should be

consistent between the same fonts, and therefore, the comparison between different characters is also important.

In this chapter, the proposed technique uses Vision Transformer (ViT) [53] by expecting the merits from its attention mechanism instead of CNN described in Chapter 2. In ViT, a character image is decomposed into small patches, and these patches are fed into a transformer encoder. In the encoder, the attention of each patch is calculated by using the mutual relationship between patches. By comparing font images in the font identification task, ViT will give larger attention to the local patches that are more important for representing the style. The heatmap of the local style awareness in Figure 3-1 is an attention map given by the proposed technique, and each element of the heatmap corresponds to a patch.

It should be emphasized that the ViT-based model of Figure 3-1 is trainable very efficiently in *a quasi-self-supervised manner* for the font identification task. The ground truth for my task is whether two character images are from the same font or not. Accordingly, if I prepare the character images from specific font sets, I know the font name of each image and give the ground truth for each image pair without any manual annotation cost. For example, if I prepare font sets of `Helvetica` and `Optima`, the pair 'A' and 'B' from `Helvetica` should have the ground truth of "Same," and the pair 'C' from `Helvetica` and 'D' from `Optima` have "Not." My experimental results show that the attention learned in this efficient manner becomes larger around important local shapes for individual styles, as expected.

In this chapter, I further utilize this attention mechanism to realize local style-aware font generation models. Specifically, as shown in Figure 3-2, I utilize the local style awareness representing the importance of individual patches for weighting the reconstruction loss function in various font generation models. This weighting scheme contributes to a more accurate reproduction of the important local parts in the generated images, as proved in my experiments of few-shot font generation in three different font generation models.

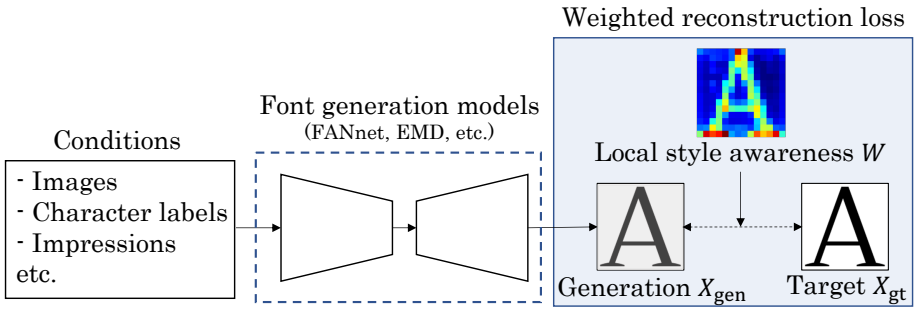


Figure 3-2: Training the font generation model using the reconstruction loss weighted by local style awareness.

My contributions are as follows.

- To determine local style awareness, I propose an efficient contrastive learning framework to solve the font identification task. Through the solution of the task, my network model can determine the local parts important to describe the font style. Note that the model can be trained without any manual annotation.
- I experimentally prove that the above framework can determine the important local parts called local style awareness.
- I apply the local style awareness to the weight of reconstruction loss in the font generation model. This weighting scheme can be easily introduced in any model trained with a reconstruction loss.
- My experimental results show that the weighting scheme improves the quality of few-shot font generation.

3.2 Related Work

3.2.1 Font Generation

In recent years, many researchers tackled font generation, especially few-shot font generation [54, 23, 55, 56, 57, 58]. Few-shot font generation is a task that accepts

content images or labels (i.e., a character label) and a few source images for extracting the style and then generating font images with the content and the style. This generation task has practical benefits for creating novel fonts. For example, a font designer can create fonts of a few characters and then automatically create the rest. Most of the few-shot font generation approaches inadequately handle aesthetic details in fonts. Fonts have their impression in their details (local structures) [59]; therefore, the aesthetic details in fonts are essential.

Some font generation for Chinese characters studies seeks the local-aware font generation to utilize radical information or structure of characters [54, 23, 55]. However, the more detailed font styles in the radicals are not addressed. Additionally, there is no study of local-aware font generation for alphabets.

Some studies address the imbalance between character regions (foreground) and the background or sharpness of characters [58, 57]. In my experiment, I use these approaches for comparative methods; therefore, I describe the details of these approaches in Section 3.4.2.

3.2.2 Fine-Grained Tasks

Fine-grained image recognition and classification focus on learning subtle yet discriminative features. Some studies utilize attention maps to extract such features [60, 61, 62, 63]. These methods estimate attention maps that localize the discriminative regions through end-to-end training for fine-grained image recognition or classification. Then, they utilized the attention map to emphasize the discriminative features. They do not need extra annotations for the regions; however, they need additional branches to estimate the attention map in each model. Therefore, they need to propose a model for each task to obtain and utilize an attention map.

I obtain the attention map of local style awareness through font identification. In contrast to the above fine-grained recognition and classification, I utilize the attention map for font generation tasks independent of font identification. I do not need to

prepare the different models to obtain the attention map for each font generation method. If I train the ViT by font identification once and obtain the local style awareness, I can utilize the local style awareness to arbitrary font generation model with reconstruction loss.

3.2.3 Explainable AI

Various approaches have been proposed to visualize the local parts of an image that are relevant to the decisions of neural networks [64, 65, 66, 67, 68, 69]. These parts help humans to interpret the decisions. These visualization approaches are one of the explainable AI (XAI) techniques.

Layer-wise Relevance Propagation (LRP) [64], guided-back propagation [65], Class Activation Map (CAM) [66], and Grad-CAM [67] are well-known XAI techniques used in CNN. LRP [64] and guided-back propagation [65] visualize the relevance by analyzing backpropagation from the output of neural networks. CAM visualizes a heatmap by weighting the final convolution layer outputs by global average pooling (GAP) layer information and output information. Grad-CAM is the generalized approach of CAM, where GAP layer information is replaced with gradient information.

For the Transformer, several XAI techniques have also been proposed [68, 69]. Attention rollout [68] can visualize the importance of individual patches by using the result of self-attention. Thanks to self-attention, the visualization considers the relationship between patches. Therefore, I can obtain the patch-wise attention map, realizing a higher spatial resolution.

For local style awareness, the attention map with a higher spatial relationship is very important to capture the local style (structure) of font images. Grad-CAM is a popular approach; however, visualizing the spatial relationship between the distant pixels is difficult. On the other hand, attention rollout can visualize the patch-wise attention map, considering a higher spatial relationship than Grad-CAM. Therefore, I used the attention rollout in my experiments.

3.3 Extraction of Local Style Awareness in Font Images

3.3.1 Methodology

Font Identification Using Vision Transformer

To determine important local parts for font styles, I propose a font identification model by contrastive learning. As noted in Section 3.1, font styles are defined by comparing various fonts and enhancing their differences. Font identification is the task of determining such differences between two input images by comparing them in a contrastive manner. Therefore, solving the font identification task fits my aim to determine local style awareness.

Figure 3-1 shows the ViT-based model for the font identification task. A pair of character images are prepared, and each is fed into a ViT, i.e., a transformer encoder, after decomposing into small patches. Each ViT outputs a feature vector of a class token. A pair of class tokens are concatenated and fed to a classifier consisting of fully connected layers to make the binary decision, "Same" or "Not."

Determining Local Style Awareness by Attention

ViT, or transformer encoder, has a patch-wise self-attention mechanism, which evaluates the mutual relationship not only between neighboring patches but also between *distant* patches. This mechanism is useful for acquiring local style features because the style-aware local parts, such as serifs, often exist at distant locations. For example, serifs of 'I' exist at the top and bottom of the vertical stroke. By training the model of Figure 3-1 for style identification, this self-attention mechanism is expected to be more sensitive to the local style difference and less sensitive to the global shapes that make, for example, 'A' as 'A.'

Accordingly, if I measure the value of patch-wise self-attention, I can get local style awareness as an attention map. (If an image is decomposed into $M \times N$ patches, the map has $M \times N$ resolution.) Roughly speaking, in the task of font identification, the attention value will become higher (or lower) at patches that are important (or unimportant) for the identification. In my model of Figure 3-1, I have two $M \times N$ self-attention maps corresponding to two image inputs. The attention map for each image will show the local style awareness of the image.

To measure the attention values, I use *attention rollout* [68]. Attention rollout is an explainable AI (XAI) technique and can visualize the importance of individual patches by using the result of self-attention. For my task of font identification, attention rollout will give higher (or lower) attention to the important (or unimportant) patches for the identification.

For local style awareness, it is very important that the patch-wise attention map with attention rollout realizes a higher spatial resolution than other XAI techniques, such as Grad-CAM [67], which is a popular XAI to visualize the regions that contribute to the decision in Convolutional Neural Networks (CNN). It is well-known that the spatial resolution by Grad-CAM is very low because it depends on the size of the deepest convolution layer. In contrast, mine has $M \times N$ resolutions, and theoretically, using smaller patches makes M and N larger. In practice, however, using too small patches is not good to describe the local shape. The current resolution of the local style awareness in Figure 3-1 is a good compromise between resolution and descriptive power and is still finer than Grad-CAM.

Quasi-Self-Supervised Learning

To train the model of Figure 3-1, I need to give ground truth ("Same" or "Not") for each character image pair. This ground truth information, fortunately, can be given without any manual annotation effort. As noted in Section 3.1, if I can prepare a set of fonts (say, `Helvetica` and `Optima`), they automatically indicate which character

images come from `Helvetica` or `Optima`. Such indications are enough to give the ground truth. Since this framework still needs external information (on preparing font sets), it is not fully self-supervised, which does not require any external information. Therefore, I call it *quasi*-self-supervised learning. From a practical viewpoint, however, it is equivalent to self-supervised learning because its annotation cost is zero after font set preparation.

Implementation Details

The transformer encoder in ViT follows the implementation of ViT [53] pretrained by ImageNet-21K. The classifier in Figure 3-1 consists of two fully connected layers. The number of layers and heads in the transformer is 12 respectively. The class token is a 768-dimensional vector. The size of an input image and the patch size are 224×224 and 16×16, respectively. (Therefore, $M = N = 14$.) Batch size and learning rate are set at 64 and 10^{-5}, respectively. I use Adam for the optimizer and cross-entropy loss for the loss function.

3.3.2 Qualitative Evaluations of Local Style Awareness

Dataset

I used the font dataset from Google Fonts [1] as follows. First, using metadata, I obtained font family name, category name[2], and a character subset, which shows languages included in each font. I chose the fonts with a character subset of "Latin" and discarded the others. I also discarded incomplete fonts. Then, I divided the font into a training, validation, and testing set to 8:1:1. During division, I did my best to avoid very similar fonts in different sets by checking font family names. As a result, I prepared 2,094 fonts, 230 fonts, and 249 fonts for the training, validation, and testing

[1] `https://github.com/google/fonts`

[2] I use four categories of fonts included in Google Fonts. In more detail, there are 1,283 Sans-Serif fonts, 630 Serif fonts, 457 Display (i.e., decorative) fonts, and 203 Handwriting fonts.

sets, respectively. For simplicity (by avoiding the disturbances of small caps.), I only used 26 capital letters in the following experiments.

The original font data is the vector format (TTF); therefore, I render it to bitmap images of 224×224 pixels with a margin of 5 pixels to use the experiment of font identification. In font generation, I resize these images to 64×64 or 80×80 to adapt to the experimental setting for each baseline of the font generation models.

Comparative Models

Although there is neither similar work nor a baseline, I designed two comparative models for evaluating local style awareness in font images.

- One is a ViT trained for the font category classification task (instead of font identification). Then, I obtained its attention map by attention rollout. Font category classification is a task that classifies the input font image into one of the four font categories, "Serif," "Sans Serif," "Handwriting" and "Display." These categories are given in Google Fonts. ViT pretrained by ImageNet-21K was fine-tuned for font category classification. The hyper-parameters in the model are the same as the ones in the font identification in Section 3.3.1.

- The other is CNN (instead of ViT) trained for the font identification task. Then, I obtained a heatmap using Grad-CAM. I employed ResNet-18 [70] as the CNN. The way of making pairs in the training phase is the same as the identification by ViT.

The test accuracy of font identification by ViT, font category classification by ViT, and font identification by CNN were 94.69%, 86.38%, and 94.59%, respectively. Note that font category classification is difficult because of fuzzy class boundaries between four categories. (Especially the boundary between Sans-Serif and Display is often confusing.)

(a) ViT + Identification (b) ViT + Classificaiton (c) CNN + Identification

Figure 3-3: Visualization of local style awareness. Red regions show strong attention, and blue regions show weak attention. Note that (a) and (b) can be obtained by only one input image. In contrast, (c) requires paired images; therefore, I randomly made pairs.

Visualization of Local Style Awareness

Figure 3-3 (a) visualizes local style awareness obtained by the proposed model. In the first row, strong attention is found in a part of shadows. (Note that the bottom part of these characters are shadows.) I can also see the consistency of attention to serif parts in the second row. In the third row, strong attention is found not only in the serif parts but also curves in 'U' and 'C.' For the sans-serif fonts in the fourth row, attention is found at the bottom, where the stroke thickness and straightness are clearly represented. The fonts in handwriting style in the fifth row show attention around their curvy stroke ends and intersection parts. From observing those maps, my attention maps, showing local stroke awareness, can find local parts that represent font-specific local structures.

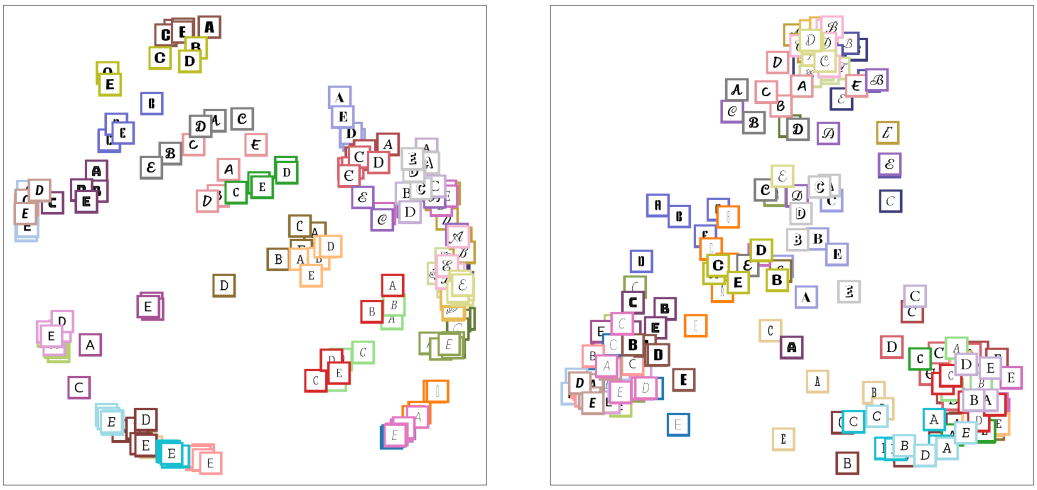
The comparison between (a) and (b) suggests how the font identification task is more suitable for local style awareness than the font category classification task. In the second row of (b), the comparative model of category classification could capture serifs

because the model is trained to discriminate serif fonts from others. However, except for the serif parts, the comparable model of (b) often fails to catch representative local parts. For example, for the samples in the third row, this model totally ignores the curves because the curves are not important for the current category classification task. Similarly, in the fourth row, the model also seems to ignore the thickness and straightness — it focuses on the corners to check the existence of serifs. To summarize these observations, this model mainly focuses on the corners to discriminate between serifs and sans-serifs and thus is rather insensitive to other local parts representing the unique structure specific to the font. In the next section, I will see how this comparative model captures different style features from the proposed model.

The differences between ViT (a)(b) and Grad-CAM (c) in their resolution and accuracy are obvious. As expected, the map by Grad-CAM is very coarse and difficult to understand the important local parts for representing font styles. Moreover, the map by CNN shows strong attention not only to the character region but also to the background region — although the background regions are often important for specifying font styles, the Grad-CAM highlight on 'F' and 'B' seems irrelevant to describe the font style.

Distributions of Local Style Features

For a further comparison between the proposed model and the comparative model trained for the category classification, I visualized the distributions of their class tokens, that is, style features, by ViT. Figure 3-4 shows their distributions for the samples of five alphabets from 'A' to 'E' by two-dimensional PCA. The comparative observation of (a) and (b) shows that the characters from the same font are more clustered in (a) than (b). Consequently, the proposed model based on font identification was more sensitive to the style and can ignore the whole letter shapes.



(a) ViT + Identification (b) ViT + Classificaiton

Figure 3-4: Distributions of class tokens (i.e., style features) by two-dimensional PCA. The same box color indicates the same font.

3.4 Boosting Font Generation Quality by Local Style Awareness

3.4.1 Methodology

In this section, I utilized local style awareness in font generation tasks to realize local style-aware font generation. As shown in Figure 3-2, local style awareness represents the importance of individual patches. Thus, I can use it for weighting the reconstruction loss function in various font generation models. If local style awareness really represents the font style, the weight will contribute to a more accurate reproduction of the important local parts in the generated images. Note that this experiment aims not only to evaluate the usefulness of local style awareness but also to evaluate the fact that local style awareness represents font style.

L1 loss weighted by local style awareness W is as follows:

$$L = ||(W + \alpha) \odot (X_{\text{gt}} - X_{\text{gen}})||_1, \tag{3.1}$$

where $\odot$ is an element-wise product, and α is a constant value for computing normal L1 loss. In the following experiment, I set $\alpha = 0.1$ to fully emphasize the weight of local style awareness (maximum value is 1). Additionally, X_{gt} and X_{gen} indicate a ground truth image and a generated image, respectively. In this experiment, I used local style awareness extracted by the font identification model trained in Section 3.3.2.

3.4.2 Quantitative and Qualitative Evaluation Experiments

Few-Shot Font Generation

I evaluated the usefulness of the proposed loss through few-shot font generation. Few-shot font generation is the task that accepts a few source images and content images (or a content label), and then the style of source images is transferred into the content.

In the training phase of few-shot font generation, almost all of the models optimize a reconstruction loss between generated image X_{gen} and the target image (*e.g.,* ground truth) X_{gt}. Therefore, introducing the proposed reconstruction loss into the font generation model is very straightforward.

In this experiment, I used the same dataset used in Section 3.3.2. I set the image size to 64×64 or 80×80 according to the experimental setting for each baseline of the font generation models. To this end, I resized the attention maps to the same size as each input image. I randomly selected a few source images and generated font images of 'A' to 'Z' for each font.

Three Baseline Models of Few-shot Font Generation

I picked up three baseline models[3] for few-shot font generation and observed the usefulness of the proposed loss for them.

[3]There are indeed newer font generation methods, and the proposed model can be introduced even to them. Since they have rather complex structures, which might obfuscate the effects of different loss functions (original, L1, and proposed), I did not use them.

- **FANnet** [71] is the font generation model that can edit the characters while retaining the font style of a source image. In more detail, FANnet accepts an image of the source font and a one-hot vector corresponding to the character label and then generates the characters with the same style as the source font. In the training phase, it employs L1 loss for reconstruction loss. When I conduct the few-shot font generation, I use average features extracted from source images.

- **EMD** [58, 72] is the style transfer model for font style and has often been used as a baseline of the font generation task. EMD accepts content images; therefore, I fixed the content images to a simple sans-serif font in the evaluation. EMD employs L1 loss weighted by the character regions to consider the imbalance between background and character regions. I compare the loss with the proposed in my experiments.

- **Srivatsan *et al.*** [57] proposed a font generation model that disentangles content from style in font images and combines them. They optimized the model by projecting an image onto the frequency using the Discrete Cosine Transform (DCT-II) instead of directly reconstructing an image. Specifically, they impose a Cauchy distribution, which is a heavy-tailed distribution in the projected space to generate sharper images. I also compare the loss with the proposed one. Note that this model includes a loss function for disentangling the style. Therefore, reconstruction loss is not dominant compared with the other methods.

Evaluation Metrics

I evaluated the quality of font generation with various evaluation metrics[4]. L1, LPIPS (Learned Perceptual Image Patch Similarity), and SSIM (Structural Similarity) are evaluation metrics commonly used for font generation. Hausdorff distance and IoU

[4]Some metrics might take infinite value when the generated image becomes empty. Therefore, I excluded such images.

Table 3.1: Quantitative evaluation of few-shot font generation. In this experiment, I used five style images. The metrics in magenta are expected to be more sensitive to local differences. The loss of "original" is the loss function proposed in each baseline model.

Baseline	loss	L1 ↓	weighted L1↓	Hausdorff↓	PHD↓	LPIPS↓	IoU ↑	SSIM↑
FANnet [71]	L1	0.0855	0.0347	7.172	1083	0.1542	0.6503	0.6810
	proposed	**0.0843**	**0.0319**	**5.602**	**803**	**0.1294**	**0.6764**	**0.6896**
EMD [58, 72]	original	**0.0916**	0.0364	8.997	2049	**0.1523**	0.6404	**0.6829**
	L1	0.0938	0.0376	9.267	2139	0.1564	0.6306	0.6794
	proposed	0.0988	**0.0358**	**8.600**	**2013**	0.1543	**0.6434**	0.6666
Srivatsan *et al.* [57]	original	0.1219	0.0429	6.0439	1026	0.2182	**0.6486**	0.6182
	L1	**0.0901**	**0.0362**	6.866	990	0.1339	0.6335	**0.6735**
	proposed	0.1007	0.0378	**5.699**	**888**	**0.1130**	0.6307	0.6417

are used for the quantitative evaluation of several font generations [73, 9, 74]. When I calculated the Hausdorff distance and IoU, I binarized the image using Otsu's method. In the Hausdorff distance, I conducted canny edge detection as preprocessing. Additionally, I used Pseudo Hamming Distance (PHD) [75], an evaluation metric, to calculate the similarity between fonts. PHD might be the most appropriate way to evaluate font styles among the above metrics because PHD can directly evaluate the difference between two shapes. Hausdorff distance also directly evaluates the difference. Roughly speaking, PHD evaluates an average difference over all shape contours, whereas Hausdorff distance evaluates the maximum difference. Consequently, it is sensitive to slight font shape differences and has been used for evaluating the similarity between font images. I also evaluated the quality of font generation using the proposed loss function Eq. 3.1 to set $\alpha = 0$ and call it weighted L1.

Quantitative Evaluation

As shown in Table 3.1, the proposed loss could improve all evaluation metrics for FANnet. FANnet is a simple model; therefore, the proposed loss could directly improve the font generation quality. In EMD, the proposed loss was better than the others in more than half of the metrics. In particular, the proposed loss was best in Hausdorff distance and PHD. These two metrics are more sensitive to the little

difference between the images than L1 loss. This indicates that the proposed loss contributes to generating fonts, keeping detailed styles more than the others. Original loss takes into the imbalance between character regions and background regions. However, to sustain the font style, using local style awareness for font generation was more effective than the original one.

In Srivatsan *et al.*, the proposed loss was better than the other model in several metrics. In particular, the proposed loss was much better than the original loss function in almost all metrics. The proposed loss was worse than the L1 loss in several evaluation metrics (such as weighted L1 and SSIM). This model includes not only reconstruction loss but also a loss function for disentangling the style. The balance between the loss function and reconstruction loss is crucial. Therefore, the order difference between the proposed loss and the L1 loss might be one of the reasons for the lower results than the simple L1 loss. It is a limitation of the proposed loss to tune the hyper-parameter (e.g., weight between loss).

Through the experiment of all three baseline models, the results by the proposed loss tended to be better in Hausdorff distance and PHD, as shown in Table 3.1. This indicates that the proposed loss contributes to improving font generation quality in detailed font styles because these metrics are sensitive to differences between images. In particular, PHD is a metric to evaluate the similarity between fonts. Note that the proposed loss aims to sustain the detailed local styles in font generation; therefore, seeing a clear improvement in font generation by using the proposed loss might be difficult in the other evaluation metrics.

However, in some metrics, degradation was caused by two reasons. The first reason is the characteristics of evaluation metrics. For example, I sometimes had better (low) L1 scores when the font generation model generated empty images than generating deformed font images. The second reason is the limitation of the proposed loss function. The proposed loss function focused on local shapes representing the style; this implies that some parts unimportant for the style sometimes become noisy.

A typical case is 'J' in the first example of Figure 3-5 (c), whose stroke width is not constant.

Qualitative Evaluation

Figure 3-5 shows the font generation examples by each baseline model and loss. To generate fonts, I used five source images marked by orange boxes. The source images were chosen randomly.

In the first example in FANnet (a), the L1 loss tends to defect to thin strokes. Especially, 'A,' 'J,' and 'M' are likely to defect their strokes. However, the proposed loss is effective in fonts with thin strokes. This is because the proposed loss is correctly weighting to the local style awareness of the font with thin strokes. The second example in (a) shows that the proposed has serif parts more clearly than L1, especially 'E,' 'F,' and 'T.' From this example, the proposed loss effectively generates font while keeping the local style.

In the first example of EMD (b), the original can not generate serif parts precisely, and some images defect the strokes. L1 can not capture the stress of stroke width (e.g., 'C' and 'G'). In contrast, the proposed can clearly generate fonts while keeping its serif style. In the second example, there is not much difference between the generated fonts. However, I emphasize that only the proposed can generate 'J' correctly. This result comes from the proposed loss functions advantage that can pay attention to local style awareness.

In Srivatsan *et al.* (c), the first examples show that the proposed can generate fonts sustaining the detailed serif parts, especially the top serif of 'A.' This trend can be seen in the proposed method. Local style awareness contributes to generating serif parts like the above 'J.' The second example shows that the proposed can generate thin fonts compared with L1. This trend is the same as in (a). The original method also generates the images; however, several images have a blurry noise.

(a) FANnet

(b) EMD

(c) Srivatsan *et al.*

Figure 3-5: Results of few-shot font generation by each baseline model and loss. Orange boxes show the source, which is used for extracting font styles. "Target" shows the ground truth.

3.5 Summary

In this chapter, I analyzed and visualized important parts that represent particular font styles. I called them local style awareness. Local style awareness was acquired by solving a font identification task in a contrastive learning scheme. This task was solved very efficiently in a quasi-self-supervised learning manner where no manual annotation was necessary. In other words, I could obtain local style awareness without human effort. The proposed model was based on ViT instead of CNN because ViT and its attention mechanism helped us to have finer local style awareness that can catch a small style structure such as serifs.

As an application task, I utilized local style awareness in a few-shot font generation to generate font images whose local structures are realized more accurately. In this application, I simply used the local style awareness as the weight for the reconstruction loss function; this simplicity allowed us to use the local style awareness in various state-of-the-art font generation models. In my experiments, I proved quantitatively and qualitatively that the proposed loss could improve the performance of three baseline models. Note that this result emphasizes the fact that local style awareness represents the font style.

Chapter 4

Multi-Modal Design Trend Analysis of Lyric Videos

4.1 Background and Purpose

Lyric videos (a.k.a., kinetic typography videos) have become a popular approach for promoting songs on video-sharing services, such as YouTube and social network services. In lyric videos, the lyric words are displayed and animated synchronously with the music. The display style of the lyric words is very different from that of still video captions. Figure 4-1 shows a series of video frames taken from a lyric video. In this video, the lyric words are shown in a decorative font style and move dynamically along with the video frames.

Similar to conventional typographic designs, such as book covers, posters, and web advertisements, creating lyric videos requires that the video creator have expertise in graphic design and that the relationship between the graphical and musical expressions be considered. The creators need to carefully choose the font style for the lyric words while considering the style (mood) of the music. Moreover, creators need to design the word motions. For example, lyric words might be shown with fewer motions for quieter music and with more flashy movements for energetic music.

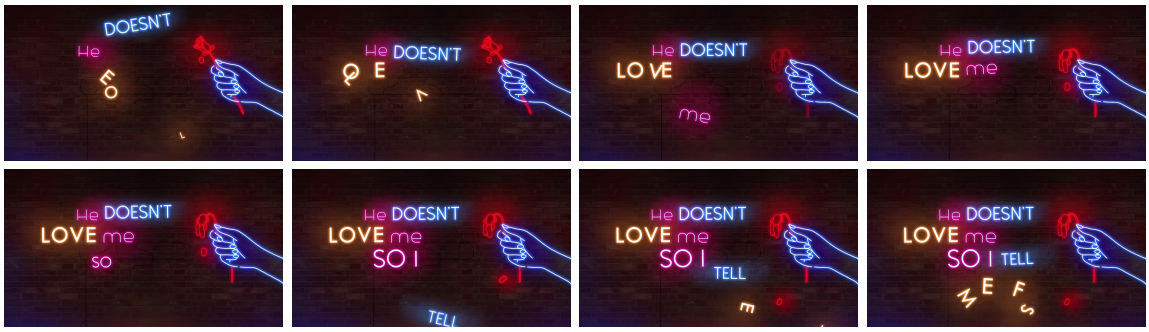


Figure 4-1: Example of video frames captured from an existing lyric video (from upper-left to lower-right).

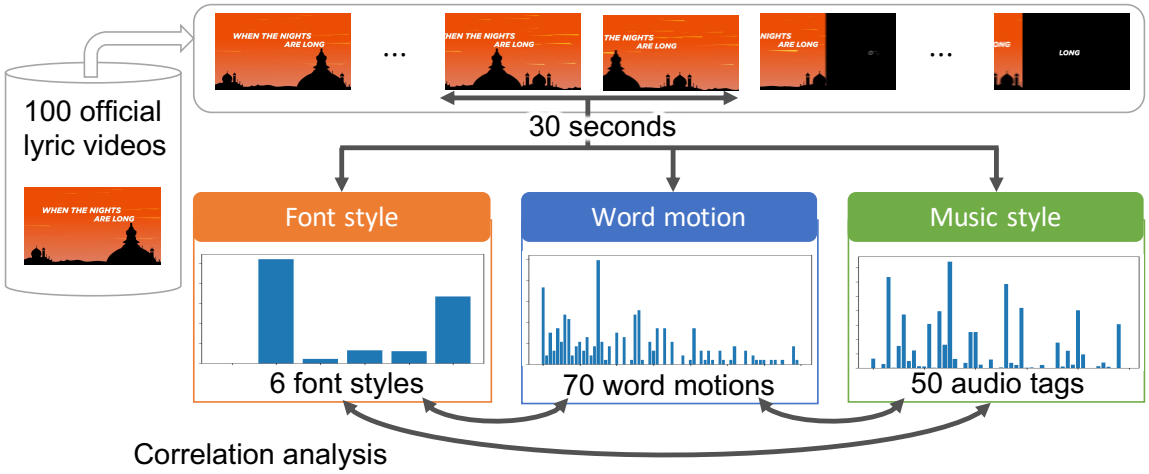


Figure 4-2: Overview of the proposed lyric video analysis.

Moreover, the motions are often designed to be synchronized with the rhythm (i.e., the beat) and the vocal timing.

This characteristic means that lyric words are often displayed in various decorated fonts. Therefore, elaborate visual designs are sometimes hard to read, even for humans. In addition, the background images of the video frames can be photographic images, illustrations, or mixtures of the two, often making it difficult to read the lyrics. As noted later, this means that lyric word detection and recognition for lyric videos is a difficult task, even for state-of-the-art scene text detectors and recognizers.

The purpose of this study is to explore the relationships between three style modalities of lyric videos: font style, word motion, and music style, as shown in Figure 4-2. For this study, I need to develop or employ appropriate techniques to quantify these

three style modalities. For example, to quantify word motions, I first need to detect and recognize lyric words in individual frames and then track them over multiple frames.

After quantifying the three style modalities, statistical analyses are conducted to reveal the correlations between the modalities. This correlation analysis is meaningful in two ways. First, it will lead to a deeper understanding of the typographic designs of lyric videos. This analysis provides hints as to how experts can use their knowledge of typography in music videos. Second, the relationships discovered by the analysis will help non-experts create lyric videos or help in the development of lyric video creation tools such as TextAlive [11]. The relationships could also be used to suggest suitable font styles for specific music styles.

Despite its meaningfulness, correlation analyses between these three style modalities for lyric videos are underexplored and remain challenging because of the following difficulties.

1. Word motion quantification is not a simple task. Lyric word detection and recognition for lyric videos are difficult tasks, even for state-of-the-art scene text detectors and recognizers. Various decorated fonts and background images prevent the accurate detection and recognition of lyric words.

2. Even though quantification of the music style is possible using a standard style estimator, such as musicnn [2, 12], there is no standard tool for quantifying the font style. The font style in lyric videos has wide varieties, and therefore the employed font style estimator needs to be capable of dealing with them.

3. The correlation between the style modalities will likely be very subtle and weak. Styles largely depend on the designer's subjective choices and may undergo multiple artistic and artificial variations. For example, the same font style may be used for music with completely different styles. This indicates that style correlations will not have simple or clear (such as linear) trends or distinctive peaks.

In fact, my preliminary regression analysis experiment using XGBoost [76] was unable to capture a clear correlation between the style modalities.

4. Because lyric videos are a relatively new multi-media video resource with typographic artwork, there is not yet a standard video dataset available for analyses. This situation is very different from other well-studied video analysis tasks, such as the Text REtrieval Conference Video Retrieval Evaluation (TRECVID).

To address the first of the above difficulties, I propose a lyric word detection and tracking method, called *lyric-frame matching*. Its key idea is to utilize the lyric word sequence, which is given as metadata, to improve the tracking performance. More specifically, state-of-the-art scene text detectors and recognizers are first applied to each video frame to obtain candidates for the lyric word locations. Then, dynamic programming (DP)-based optimization is applied to determine the optimal matching between the candidates and the lyric word sequences over the frames. The matching result gives a reliable spatio-temporal trajectory for each lyric word in a given sequence.

For the second difficulty, I propose a font style estimator based on a convolutional neural network (CNN). Basically, the estimator is simply realized by training the CNN with a font image dataset where the font style (e.g., "Sans-Serif") is annotated to each font. Because there is no standard font style class definition, I roughly define six font styles and represent the style of a given word image using a six-dimensional class probability vector. In addition, because the word images extracted from the lyric video frames have various backgrounds and distortions, I need to train the CNN not with a clean font image but with synthetic font images that mimic actual lyric word images.

For the third difficulty, I make full use of cluster analysis. Even though clustering is a classic and simple method, it is useful for my correlation analysis task. Clustering involves vector quantization and, therefore, gives a rough view of the variations in the styles. Moreover, clustering can deal with highly nonlinear style trends because

of its non-parametric nature. In this chapter, I first apply k-means clustering to each modality independently and then apply a biclustering technique to understand the correlation between two modalities via the co-occurrence of their (quantized) styles.

For the fourth difficulty, I prepare a new lyric video dataset containing 100 lyric videos created by design experts. I manually attached the lyric word bounding boxes to 1,000 video frames to evaluate the accuracy of the lyric word tracking result. A list of the videos and the bounding box data are publicly available at `https://github.com/uchidalab/Lyric-Video`.

The main contributions of this chapter can be summarized as follows:

- To the best of the authors' knowledge, this is the first study to analyze the design of lyric videos in a quantitative manner. Because of the design factors specific to lyric videos, I focus on three style modalities: font style, word motion, and music style. A correlation analysis between these style modalities will provide basic knowledge concerning kinetic typography designs in music videos. In fact, the analysis results reveal interesting trends between the three style modalities; for example, "Fancy" fonts tend to be used for "pop" and "guitar" music, and active motions are often printed in "Fancy" and "Sans-Serif" fonts.

- This is also the first attempt to detect and then track lyric words in lyric videos. I propose a novel word tracking technique using an optimal lyric-frame matching algorithm based on DP.

4.2 Related Work

Since this chapter is the first attempt at a design analysis of lyric videos, there are presently no similar studies. In this section, instead, I review previous attempts to extract or analyze word motion, font style, and music style for more general subjects.

4.2.1 Word Motion Analysis

There are several tasks involved in detecting and tracking words in video frames. The most typical task is caption detection [77, 78, 79, 80, 81, 82, 83, 84, 85]. Captions are defined as text superimposed on video frames. Captions, therefore, have characteristics that differ from scene text. Even though most studies have dealt with static captions (i.e., captions without motions), Zedan [82] addressed not only static captions but also moving captions. They referred to the vertical or horizontal scrolling of caption text as moving captions.

Recently, video text tracking [86, 87, 88, 89, 90, 91, 92, 93, 94] has also been attempted, as reviewed in [95]. Because such methods try to track words in a scene captured by a moving camera, they introduce a common assumption that the words are static in the scene and are captured by the moving camera. Therefore, they assume, for example, that neighboring words will move in similar directions. The paper [96] introduces "moving MNIST" for video prediction tasks. The paper focuses on synthetic videos capturing two digits moving with respect to a uniform background.

My study is very different from these previous attempts with respect to the following three points at least. First, my target words in lyric videos move far more dynamically and freely, invalidating the assumption used in previous studies. Second, I can utilize lyric information during tracking, whereas previous attempts did not include such guiding information.

4.2.2 Font Style Estimation

Most previous font image analysis studies have focused on the so-called font identification (or font recognition). This involves identifying the font name (such as "Helvetica") of a given text image. Zramdini and Ingold [97] presented a pioneering trial recognizing 10 different fonts. Recently, deep neural networks have also been used for font identification [16, 98, 99].

In this study, I use font style estimation, which is different from font identification. Font styles are defined as Serif, Sans-Serif, Script, and so on. If a method can estimate the style of an arbitrary font, it can be applied to lyric words printed with rare or even brand-new fonts. However, font style estimation is less common than font identification because font style classes are not well defined[1]. Shinahara et al. [40] developed a font style estimation method based on six font classes (Serif, Sans-Serif, Hybrid, Script, Historical Script, and Fancy) defined in a font guidebook [1]. Examples of these font classes are shown in Figure 1-3(c). They used simple pattern matching for the classification. In this chapter, as described in Section 4.4.2, I develop my own neural network-based font style estimator following the same six classes. Note that there have been several classical attempts (see Table 1 of [101]) to classify font images into Roman, bold, and italic classes. I do not use these three classes because they are appropriate for font images from ordinary text documents but not for various font images of lyric words.

4.2.3 Music Style Estimation

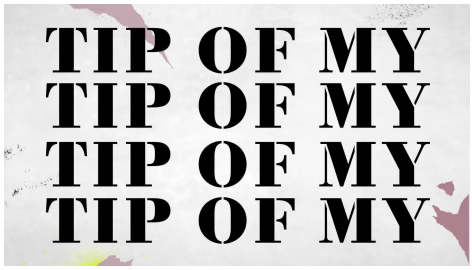
Music audio tagging, including mood/emotion estimation, is a popular research topic in the music information retrieval community. Various approaches have already been proposed for music audio tagging [102, 103, 104, 105, 106, 107, 108, 109, 12, 110, 111]. Recently, Pons and Serra released *musicnn* [2, 12], which can provide a "taggram" for each music segment using CNNs. Each taggram is a 50-dimensional vector, and each element corresponds to 1 of 50 tags (defined in the MagnaTagATune (MTT) dataset [112]). This is not a one-hot vector but rather a non-negative real-valued vector. Each value represents the property of the music segment or the corresponding tag.

In this chapter, I use the 50-dimensional taggram given by musicnn as the music

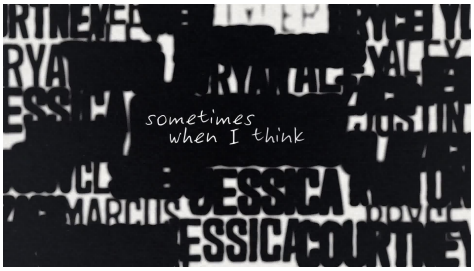
[1]The PANOSE System [100] was expected to be a good standard for font styles; however, most fonts currently do not follow it.

(a) Showing lyric words.

(b) No lyric word (interlude).



(c) Duplicated lyric words.



(d) Words including those unrelated to the lyrics.

Figure 4-3: Variations of lyric video frames.

style. As in the case of the font styles, the music styles do not have any standard definition; this is because music styles are defined by multiple factors, such as instrument types and genres. Fortunately, taggram by musicnn covers these factors. Of the 50 tags, some tags indicate musical instruments (such as "drums" and "guitar"), some indicate vocal types (such as "male vocal" and "choral"), some indicate music genres (such as "rock" and "techno"), and some indicate moods (such as "loud" and "slow").

4.3 Lyric Video Dataset

As the lyric video dataset to be analyzed, 100 videos were collected via the following steps. First, a list of lyric videos was generated by searching YouTube with the keywords "official lyric video" (on July 18, 2019). The keyword "official" was added to find videos with not only long-time availability but also professional quality. The latter is very important because I want to exclude incomplete or thoughtless video designs from my analysis. Then, the videos in the list were manually checked to

exclude videos with only static motion words (i.e., videos whose lyric words did not move). Finally, the top-100 videos on the list were selected as my experimental target[2]. The frame image size is 1,920 × 1,080 pixels. The average, maximum, and minimum lengths of the videos in the dataset are 5,471 frames (3 min 38 s), 8,629 frames, and 2,280 frames, respectively. The average, maximum, and minimum numbers of lyric words are 338, 690, and 113, respectively.

Figure 4-3 shows four examples of lyric video frame variations. Figure 4-3 (a) depicts a frame showing lyric words. Typically, several words (i.e., a phrase in the song) are shown in a single frame. In the introduction, interlude, and ending parts, frames with no lyrics are often found, as shown in Figure 4-3 (b). In Figure 4-3 (c), the same word is duplicated, as in the refrain of a song. Sometimes, as shown in Figure 4-3 (d), the background image contains words unrelated to the lyrics.

To perform a quantitative evaluation of the word tracking method in Appendix A, bounding boxes were manually attached to the lyric words for 10 frames in each video. These frames were selected automatically. Specifically, for each video, the top 10 frames with the most words were selected from the frames sampled at three-second intervals. The lyric words were detected using the method described in Appendix A.1, and a bounding box was attached to each word in the lyrics. I attached non-horizontal bounding boxes[3] to the rotated lyric words. Consequently, I obtained $10 \times 100 = 1,000$ ground-truth frames with 7,770-word bounding boxes for the dataset.

[2]A list of all 100 videos and their annotations is published at `https://github.com/uchidalab/Lyric-Video`.

[3]To attach non-horizontal bounding boxes, I used the labeling tool `roLabelImg` available at `https://github.com/cgvict/roLabelImg`.

4.4 Feature Extraction of Each Style

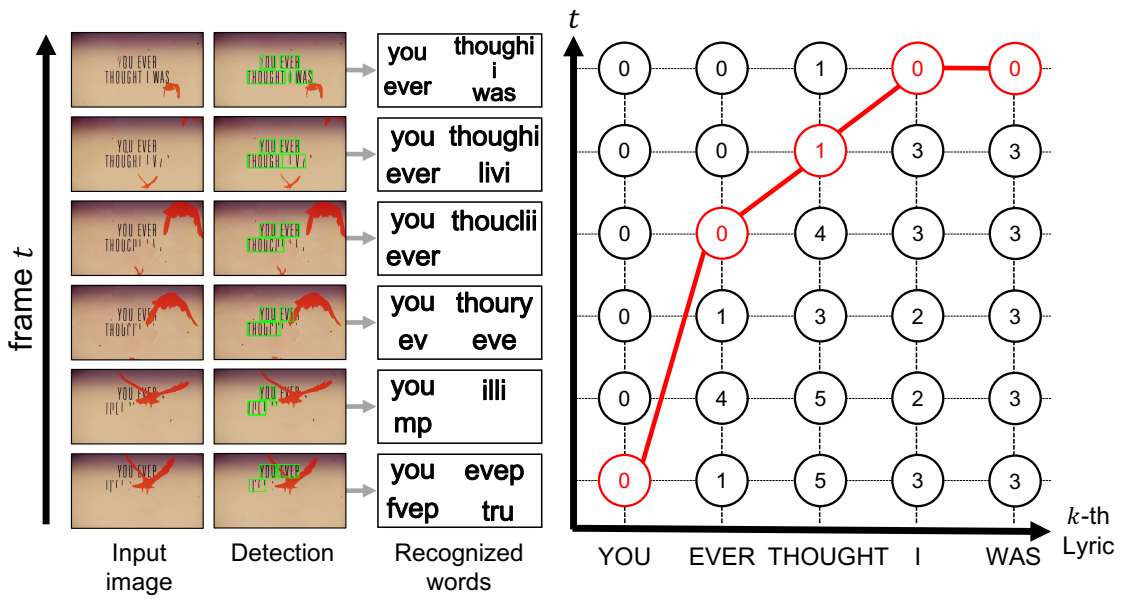
4.4.1 Word Motion Style

Lyric Word Tracking

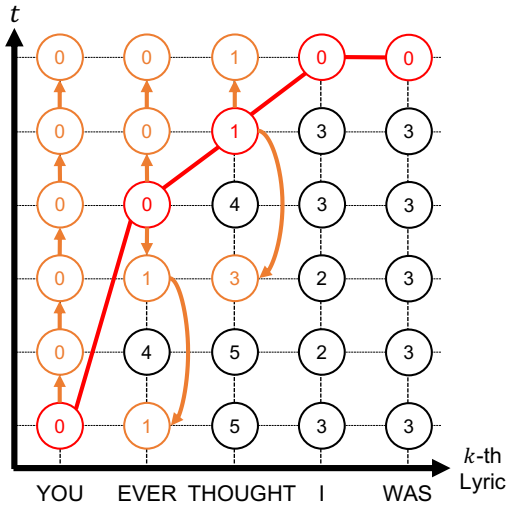
I propose a word tracking method for extracting individual word motions and then quantifying their style. The proposed method is specialized to accurately track lyric words while utilizing lyric word information (which is text data available in the meta-data of the lyric video). The tracking method has three steps: word detection, word recognition, and lyric-frame matching. In the first step, lyric word candidates are detected and recognized by the method presented in Appendix A.1, as shown on the left-hand side of Figure 4-4 (a).

After detection and recognition, lyric-frame matching is conducted to establish the correspondence between the words on frames and the lyric word information (i.e., text data of lyrics). The matching algorithm is detailed in Appendix A.2. The red path on the right-hand side of Figure 4-4 (a) represents the optimal correspondence of the frames and lyric words. If the path passes through the grid (k, t), it means that the tth frame is determined to be the most confident frame for the kth lyric word. I then search the frames around the tth frame to find the same kth lyric word. The vertical orange paths in Figure 4-4 (b) depict the search results for individual lyric words. This search was done not only using simple spatio-temporal closeness but also by evaluating the word similarity of the kth word. As shown in Figure 4-4 (b), there are many misrecognized words; therefore, I cannot use the exact match with the lyric word in this search. Details are given in Appendix A.3.

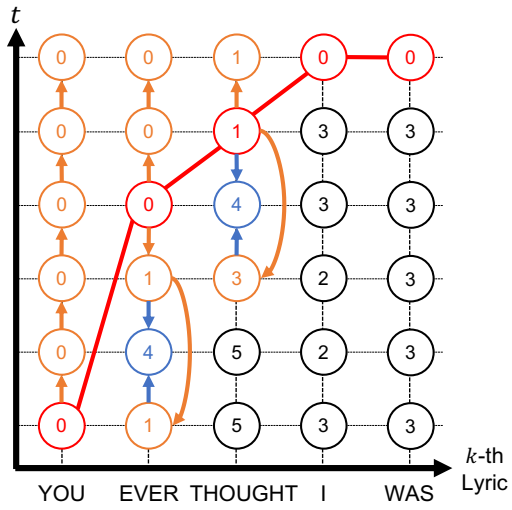
The vertical orange paths for "EVER" and "THOUGHT" in Figure 4-4 (b) include skipped frames. For example, "EVER" was not detected in the second frame. Such missed detections occur because of occlusion and severe misrecognition. Therefore, I need to perform the interpolation process shown in Figure 4-4 (c) to complete the



(a) Detection, recognition, and lyric-frame matching.



(b) Tracking via neighbor search.



(c) Interpolation.

Figure 4-4: Lyric word detection and tracking. The circled number shows the distance $D(k,t)$ between the kth word and the frame t.

spatio-temporal tracking process of each lyric word. Roughly speaking, if a missed frame is found for a lyric word, the polynomial interpolation process determines the location of the lyric word in that frame. Details are given in Appendix A.3. Figure 4-5 shows the final result of the tracking process for the two lyric words "YOU" and

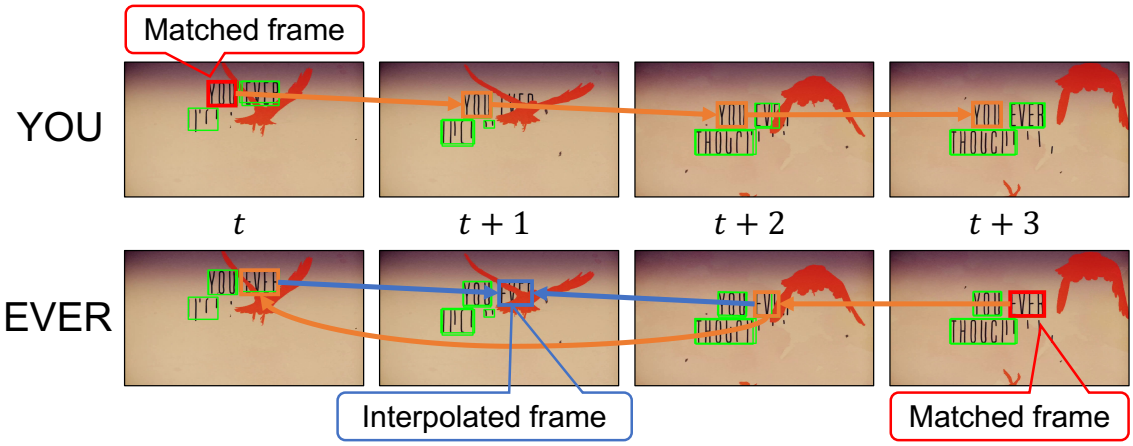


Figure 4-5: Tracking result of "YOU" and "EVER." Especially, interpolation is successfully performed for "ever."

"EVER."

Even though the above tracking method is not perfect because of the various difficulties, the quantitative evaluation uses the ground-truth bounding boxes attached to the video frames. Specifically, as detailed in Appendix A.5, the tracked trajectories according to the above method show high precision. Therefore, I believe that the following word motion style analysis based on the tracking result is sufficiently reliable.

Representative Word Motions

Later, in the correlation analysis between word motion style, font style, and music style, I represent the word motion style of each 30-second time window in a so-called "bag-of-words" manner. The word motions are very varied, and it is difficult to analyze all of them. Therefore, I extract representative word motion styles in a bag-of-words manner. Specifically, the motion trajectories of all the lyric words in the video are quantized into B representative word motions, and a histogram with B bins is created. Each bin corresponds to one representative motion and shows how many word trajectories are quantized to that motion. I, therefore, need to select the representative word motions in advance of the word style representation.

The steps to select the $B(= 70)$ representative motion trajectories of all the lyric videos in the dataset are as follows. First, each motion trajectory is represented as a sequence of four-dimensional vectors (x_1, y_1, x_2, y_2), as shown in Figure 4-6, where (x_1, y_1) represents the location of the center of the word bounding box, and (x_2, y_2) is defined as the upper-right corner of a square whose center is (x_1, y_1) and whose edge length is the bounding-box height. The coordinates (x_2, y_2) indirectly represent the size (word height) and rotation of the bounding box in a manner consistent with (x_1, y_1). Second, each motion trajectory is translated such that its first location (x_1, y_1) becomes $(0, 0)$. Third, the trajectories are grouped by their duration: $0.5 \sim$ 1.0s (5,107), $1.0 \sim$ 1.5s (4,423), $1.5 \sim$ 2.0s (3,581), $2.0 \sim$ 2.5s (2,744), $2.5 \sim$ 3.0s (1,658), $3.0 \sim$ 4.0s (1,742), and $4.0 \sim$ 5.0s (973). The numbers in parenthesizes count the trajectories in the individual groups. Extremely short (< 0.5s) and long (> 5.0s) trajectories were rare and were excluded. Finally, k-medoid clustering ($k = 10$) was performed to a dynamic time warping distance metric at each group, and 70 representative word motions were obtained.

Figure 4-7 shows the 10 word motions (i.e., 10 medoids) in each of the seven duration groups, where (x_2, y_2) is omitted. The center of each plot is the origin $(0, 0)$ (i.e., the starting point of the trajectory), and the change in the color saturation (white to vivid) indicates the transition of time. The majority consists of rather simple motions: vertical, horizontal, or no-motion (i.e., staying at the origin). In addition, I show a histogram of the number of trajectories in each of the 70 clusters for all 100 lyric videos. In each of the seven duration groups, the orange bin indicates the cluster having a no-motion trajectory in which the lyric words do not move; lyric words often appear and disappear without movement.

4.4.2 Font Style

To facilitate the correlation analysis performed later, I represent the font style of each video as a likelihood vector of six typical font styles: Serif, Sans-Serif, Hybrid (of Serif

Figure 4-6: Four-dimensional representation of the word location and rotation.

and Sans-Serif), Script, Historical Script, and Fancy (i.e., Display). Accordingly, the font style is given as a six-dimensional real-valued vector. For this purpose, I developed a six-class font style classifier that gives the likelihood vector of each word image. The font style estimates were derived using ResNet18, a CNN trained using a large number of word images synthesized by SynthText [113]. More specifically, I first collected 510, 314, 151, 74, 58, and 704 different fonts for the Serif, Sans-Serif, Hybrid, Script, Historical Script, and Fancy classes, respectively. The class of each font was specified in a font guidebook [1]. I then generated 19,000 synthetic word images for each of the six fonts using SynthText. The images were separated into training (80%), validation (10%), and test (10%) sets, and these sets were font-disjoint. Finally, ResNet was trained as a six-class classifier using the training and validation sets. I used the six-dimensional likelihood vector given before the softmax layer of the trained ResNet as the font style vector. Note that the performance of the brute-force classification (into one of six font classes) by the trained ResNet was 81.10% for the test dataset.

Figure 4-8 shows the font style vectors for word images extracted from the lyric videos. The horizontal axis corresponds to the six font classes, and the vertical axis indicates their likelihood. The top row shows four cases with a high likelihood only at the correct single class. The middle row shows cases estimated as being a mixture of several font styles. The bottom row shows style vectors for word images taken from the same lyric video with consistent font styles. Note that, in the experiment

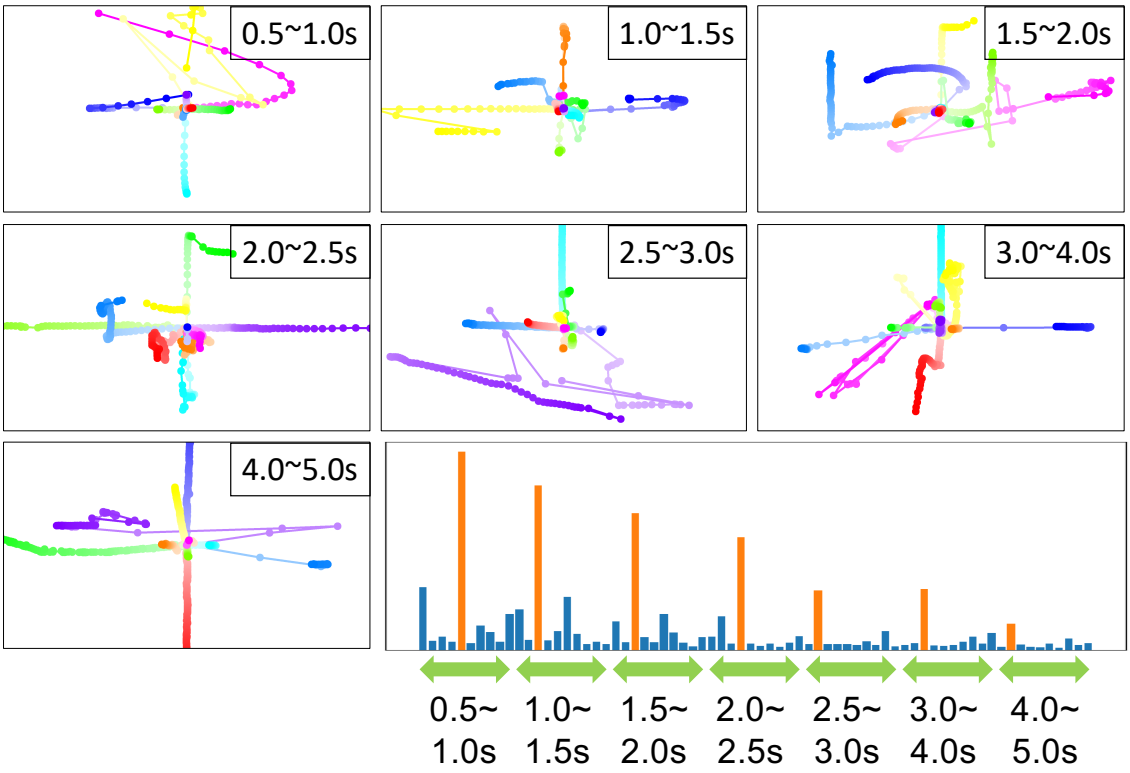


Figure 4-7: The 70 word motions (7 duration groups × 10 k-medoid clusters) and a histogram showing each cluster size. Note that this k-medoid clustering for word motions is different from the cluster analysis used to understand the style modality correlation, as described in Section 4.5.

in Section 4.5, the font style vectors of all lyric words detected within each 30-second time window were averaged and then used as the font style vector for the time period.

4.4.3 Music Style

The music style was obtained as a 50-dimensional vector using musicnn[4][2, 12]. I used the "MTT_musicnn" model pre-trained on the MTT dataset [112]. For the audio in a 30-second time window, I estimated a 50-dimensional tag likelihood vector corresponding to the 50 MTT tags, including instrument-related tags such as guitar

[4] https://github.com/jordipons/musicnn

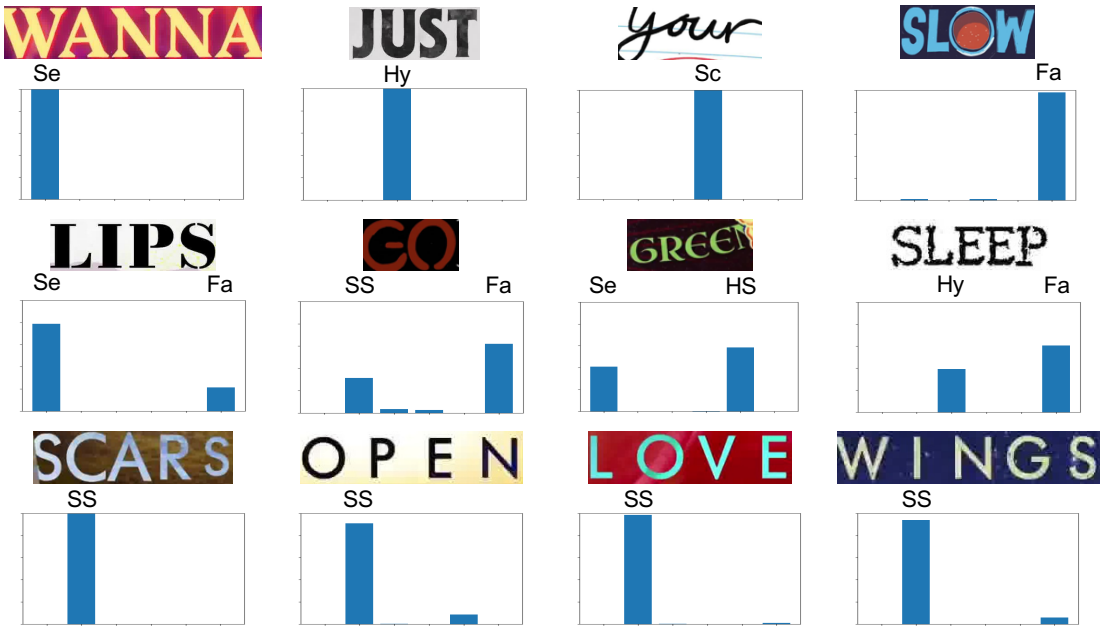
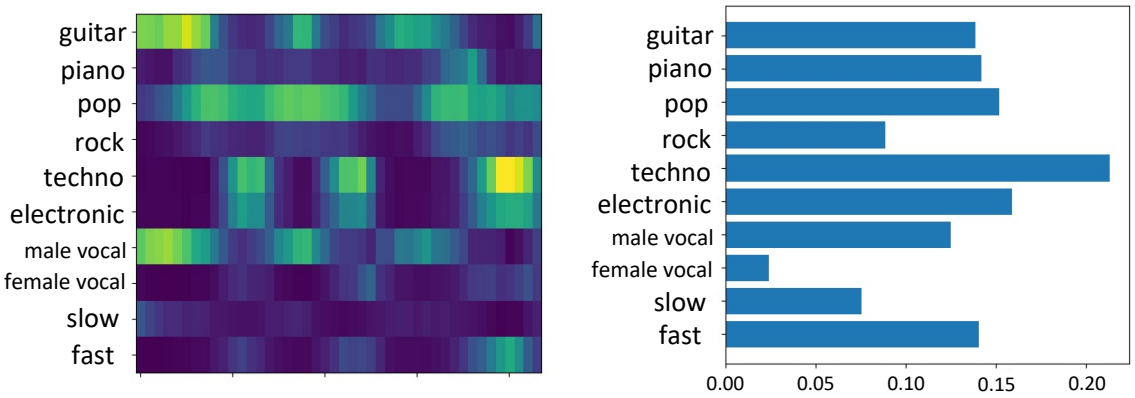


Figure 4-8: Font style estimation results for the lyric words in the lyric videos. The bar chart visualizes the likelihood of the six styles; its horizontal axis corresponds to Serif (Se), Sans-Serif (SS), Hybrid (Hy), Script (Sc), Historical Script (HS), and Fancy (Fa) from left to right.

and drums, tempo-related tags such as slow and fast, and vocal-related tags such as male and female. I simply refer to this as the music style vector, even though the estimated tags are not always style-related tags and represent a variety of musical attributes of a song, which is desirable for my research purposes. Each vector was estimated from a 30-second time window, and the estimation was performed every 5 s (at five-second intervals). For example, a sequence of 19 music style vectors can be extracted from a 120-second song ($(120 - 30)/5 + 1 = 19$).

Figure 4-9 (a) visualizes an actual music style vector sequence as a heatmap, where the horizontal axis indicates time and the vertical axis shows 10 tags of the 50 tags. Yellow indicates the highest value (i.e., 1). In this example, the music style changes along the time axis because of various interludes. Figure 4-9 (b) shows the averaged style vector over the same song and indicates that this music is sung by a male and has a techno mood with a fast tempo. Note that I do not use the averaged vector in



(a) Transition of music style. (b) Averaged style over the entire song.

Figure 4-9: Music style estimation by musicnn [2] of the lyric video of "Something Just Like This" by The Chainsmokers & Coldplay. (a) Heatmap of music style vector sequence, where the horizontal axis indicates time and the vertical axis shows 10 tags of the 50 tags. (b) The averaged style vector over the same song. Note that I extracted the averaged style vectors in 30-second time windows in the following experiments.

Figure 4-9 (b) but rather the vector sequence in Figure 4-9 (a) in the later analysis.

4.5 Correlation Analysis Between the Three Style Modalities

4.5.1 Ten Representative Types of Each Style Modality

As shown in Figure 4-2, I conducted a correlation analysis between the three style modalities of word motion, font style, and music style. Each feature vector of the three style modalities was extracted from a 30-second time window with a 5-second interval. Consequently, every 5-second, I obtained 70-, 6-, and 50-dimensional vectors for the word motion, font style, and music style, respectively. Note that time windows with no lyric words or only a few words were discarded from the analysis. This resulted in 3,494 feature vectors for each style modality from the 100 lyric videos. I used all of these vectors in the following clustering-based correlation analysis.

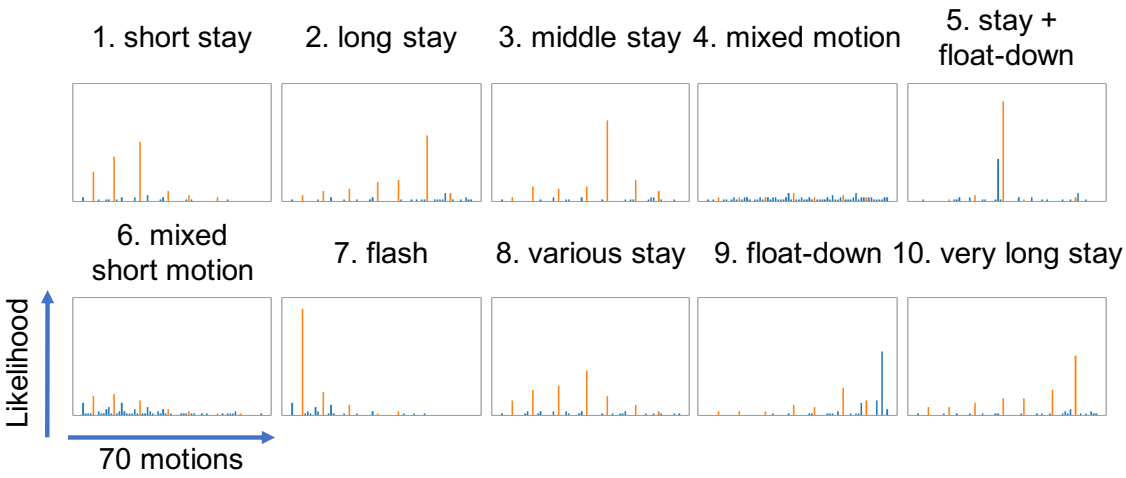


Figure 4-10: Ten word motion style types by k-means clustering. The horizontal axis corresponds to the 70 word motions presented in Figure 4-7. The orange bars correspond to no motion (i.e., stay) with seven different durations. A brief description, such as "flash" (very short presence), is attached to each type.

In advance of the correlation analysis, standard k-means clustering[5] was performed to quantize the style vectors of each modality. As noted in Section 4.1, cluster analysis is more promising for my task than orthodox multivariate analysis techniques, such as deep regression. Determination of the number of clusters (hyper-parameter k) relies on several criteria, such as the silhouette coefficient [115] that evaluates the average distances between the samples, the Calinski Harabasz score [116] that evaluates the inter-/intra-cluster variance, and the Davies-Bouldin index [117] that evaluates similarity between clusters. I examined these criteria but found no unanimous suggestion for the value of k. I, therefore, took the intermediate value of $k = 10$ for all of the style modalities. Consequently, I had $k = 10$ representative types (representative centroid vectors), as shown in Figures 4-10, 4-11, and 4-12 for the word motion, font

[5]I compared the results of k-means clustering and the results of agglomerative clustering (so-called hierarchical clustering) by using the adjusted rand index [114], and the results showed that the index score of word motion style types, font style types, and music style types is 0.28, 0.60, and 0.49, respectively. This index becomes 1 when two clustering results are completely the same and zero when there is no correlation. Although the word motion style has a weak correlation, the font style and music style have strong correlations. Therefore, the choice of clustering algorithms is not very sensitive in my task.

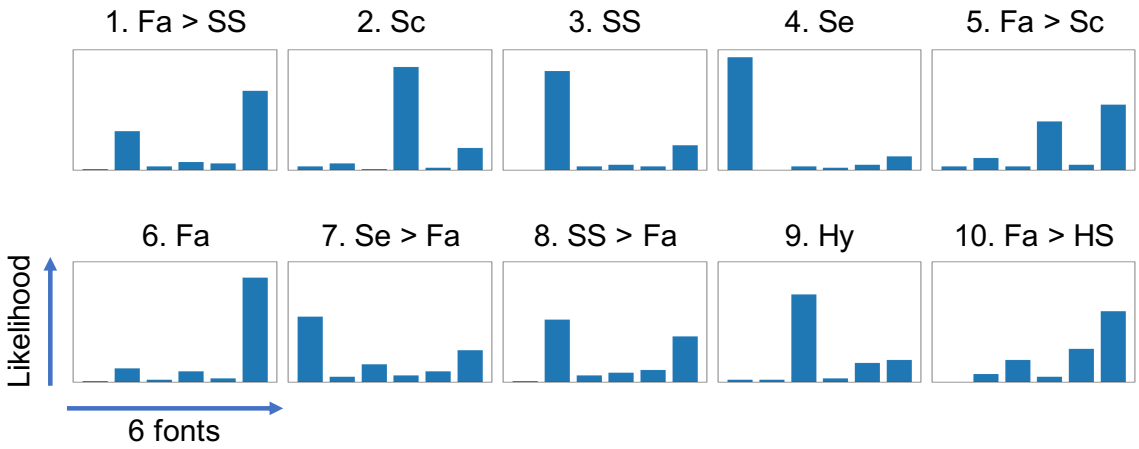


Figure 4-11: Ten font style types by k-means clustering. The horizontal axis corresponds to Serif (Se), Sans-Serif (SS), Hybrid (Hy), Script (Sc), Historical Script (HS), and Fancy (Fa), from left to right. A brief description, e.g., "Se > Fa" (Serif is presented more than Fancy), is attached to each type.

style, and music style modalities, respectively.

4.5.2 Co-occurrence Analysis Between the Style Modalities

As noted above, for every 5 s, word motion, font style, and music style feature vectors were obtained via an analysis of the 30-second time window. Let those feature vectors be denoted by $\mathbf{w}_{t,s} \in \mathbb{R}_+^{70}$, $\mathbf{f}_{t,s} \in \mathbb{R}_+^{6}$, and $\mathbf{m}_{t,s} \in \mathbb{R}_+^{50}$, respectively, where t is the frame index and $s \in [1, 100]$ is the lyric video ID. I quantized these vectors into the nearest vector of the 10 representative vectors (types) in each modality. Consequently, I obtained $W_{t,s}$, $F_{t,s}$, and $M_{t,s}$, each of which represents the nearest vector index $\in [1, 10]$. Then, I obtained a 10×10 co-occurrence matrix for each pair of two modalities. For example, the co-occurrence matrix $\mathbf{C}_{f,m}$ between the font style and the music style was created by adding 1 to the $(F_{t,s}, M_{t,s})$th element of the matrix for all t and s.

Figure 4-13 shows the co-occurrence matrices for all three pairs of style modalities. The matrices were pre-processed with biclustering (row-wise and column-wise reordering) such that blocks (sub-matrices) became more visible. Via careful observations of

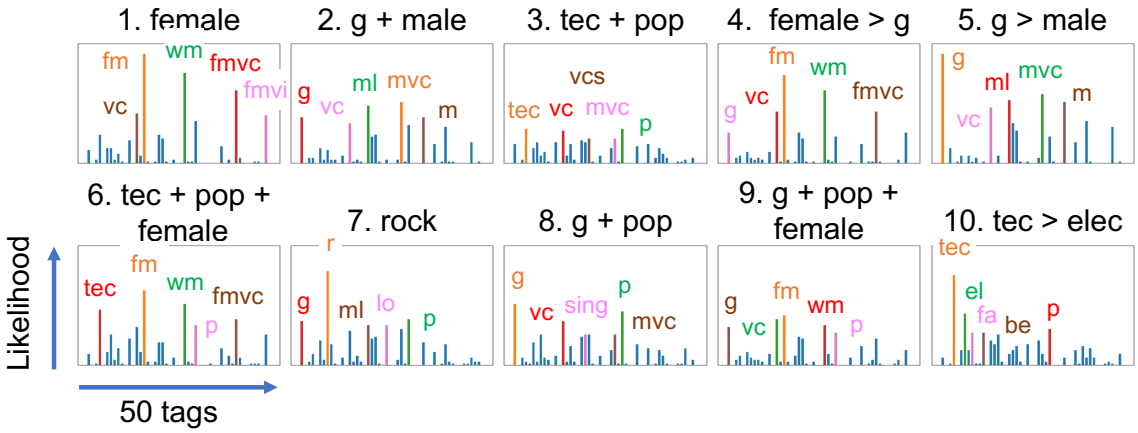


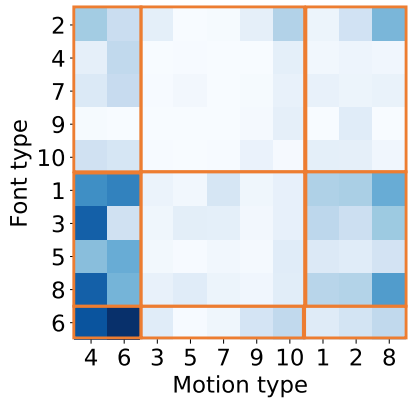
Figure 4-12: Ten music style types by k-means clustering. The five tags with the highest likelihood are printed in orange, green, red, brown, and pink. The abbreviations are as follows: vc: vocal, fm: female, wm: woman, fmvc: female vocal, fmvi: female voice, g: guitar, ml: male, mvc: male vocal, m: man, tec: techno, vcs: vocals, p: pop, r: rock, lo: loud, sing: singing, el: electronic, fa: fast, and be: beat.

the matrices, the following trends in the lyric video designs were indicated.

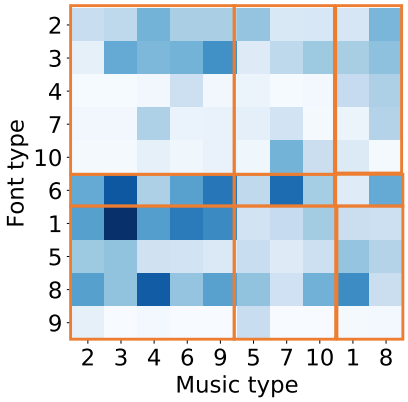
- *Word motion and font style* (Figure 4-13 (a)): There is a block with high co-occurrence in the bottom-left area of the matrix. This block is the intersection of two motion types $W_{t,s} = 4$ (mixed motion), 6 (mixed short motion) and five font types $F_{t,s} = 1$ (Fa>SS), 3 (SS), 5 (Fa>Sc), 8 (SS>Fa), 6 (Fa). This indicates that lyric words with active motions (i.e., not "no-motion") are often printed in Fancy and Sans-Serif. See examples in Figure 4-14.

- *Font style and music style* (Figure 4-13 (b)): There is another block with high co-occurrence near the bottom-left area of the matrix. This block is the intersection of five music types $M_{t,s} = 2$ (g+male), 3 (tec+pop), 4 (female>g), 6 (tec+pop+female), 9 (g+pop+female) and four (or five) font types $F_{t,s} = 6$ (Fa), 1 (Fa>SS), 5 (Fa>Sc), 8 (SS>Fa) ($F_{t,s} = 9$ (Hy), which is weaker). This block suggests that "Fancy" fonts tend to be used for "guitar" and "pop" music.

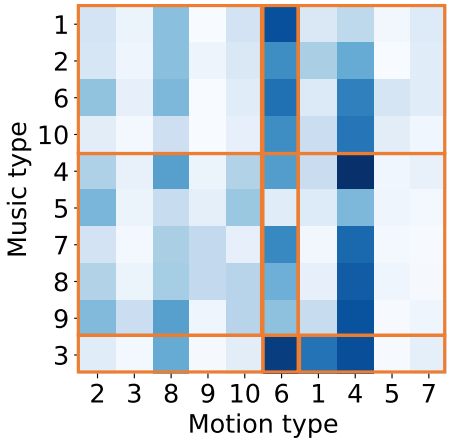
- *Word motion and music style* (Figure 4-13 (c)): Motion types $W_{t,s} = 4$ (mixed motion), 6 (mixed short motion) are scattered across all music types except for $M_{t,s} = 5$ (g>male). However, observing local correlations, there are strong peaks



(a) $\mathbf{C}_{w,f}$: Word motions and font styles.



(b) $\mathbf{C}_{f,m}$: Font styles and music styles.



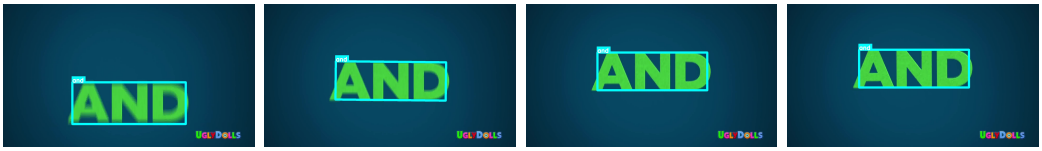
(c) $\mathbf{C}_{w,m}$: Word motions and music styles.

Figure 4-13: Co-occurrence matrices for each modality pair. Each type index corresponds to the types described in Section 4.5.1. Biclustering was applied to the matrix for better visibility. Each orange box indicates a bicluster in the matrix.

at $W_{t,s} = 6$ (mixed short motion) - $M_{t,s} = 3$ (tec+pop), $W_{t,s} = 6$ (mixed short motion) - $M_{t,s} = 1$ (female), and $W_{t,s} = 4$ (mixed motion) - $M_{t,s} = 4$ (female>g). These co-occurrences suggest that mixed motions (i.e., active motions) tend to be used for music with female vocals. Moreover, for music type $M_{t,s} = 3$ (tech+pop), various short and active motions are used for the lyric words.

There are also other interesting strong co-occurrences in Figure 4-13; for example, "Fancy" and "Historical Script" fonts ($F_{t,s} = 6, 10$) are usually used for "rock" music ($M_{t,s} = 7$) as shown in Figure 4-13 (b).

(a) Scaling motion with the "Fancy" font style.



(b) Sliding motion with the "Sans-Serif" font style.

Figure 4-14: Scenes with active motions and the font styles of "Fancy" and "Sans-Serif." These videos have remarkable trends in their relationships. You can see these videos on YouTube. Note that the four still-frame images are arranged in order from left to right.

These trends found in my analysis could be useful for assisting in the design of lyric videos. Even though I highlighted strong co-occurrences in the above analysis, no or low co-occurrences might also provide useful information concerning the trends in lyric videos. However, I do not emphasize those low co-occurrences in this chapter because they may be caused by insufficient lyric video data, and a much larger dataset might prove the importance of such low co-occurrences in future research.

4.6 Summary

In this chapter, I tackled the novel task of analyzing lyric videos to understand the relationships between three style modalities: word motion, font style, and music style. To conduct this analysis, I developed an original lyric word tracking method, which is detailed in Appendix A, and an original font style estimator. Moreover, the clustering-based co-occurrence analysis of the style modalities from 100 lyric videos indicated several trends in the style combinations. That is, I was able to catch such trends in the videos in an objective and reproducible manner without manual annotations.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, I tackled visual design informatics, which is a research field analyzing visual designs using informatics approaches. In particular, I analyzed two types of visual designs: font images and lyric videos from the viewpoint of visual design informatics.

In Part 2, I proposed and tackled a novel font identification task, which identifies whether the pair of input images come from the same font or not. This font identification is totally different from previous font identification approaches similar to font recognition. As a result, I was able to demonstrate that the simple CNN could identify fonts with an accuracy of 92.27±0.20% using 6-fold cross-validation. This is despite using different characters as representatives of their font. This indicates that CNN could capture the font style to identify fonts.

In Part 3, I solved the font identification using Vision Transformer (ViT) and visualized the attention of the ViT. In the experiments, the visualized attention showed the important parts that represent font style. I named this visualized attention to local style awareness. As an application task, I utilized local style awareness for font generation as the weight of loss function. I showed that the proposed loss contributes

to improving the font generation performance.

In Part 4, I conducted a lyric video analysis focusing on three modalities: font style, word motion, and music style. To this end, I developed an original style feature extractor and lyric word tracker, and I utilized a music style estimator. Moreover, the clustering-based co-occurrence analysis of the style modalities from 100 lyric videos indicated several tendencies in the style combinations. For example, the result shows a correlation between "Fancy" and "Sans-Serif" fonts and active word motions.

5.2 Future Work

The future works of this thesis are three-fold.

- **Analysis of contrarian tendencies of font images:** A lot of font image analyses aside from this thesis have been conducted, such as correlation analysis of font images and impression words attached to the fonts [9] and font style usage analysis in real images [10]. These analyses revealed the general tendencies of font style usage. However, design elements, including fonts, are sometimes used for a contrarian situation. For example, distinguishable fonts, such as fancy or calligraphy fonts, are generally used for food package designs, though simple fonts might be used to stand out from other packages. Tendencies that deviate from general tendencies are unique to the design domain. First of all, I will start the analysis of contrarian design tendencies using font images, which is one of the simplest visual designs.

- **Expansion of lyric video analysis including generation task:** In this thesis, I tackled lyric video analysis focusing on three design modalities: font style, word motion, and music style. In the lyric videos, there are a lot of modalities aside from them, such as background, text color, etc. Therefore, to conduct a more comprehensive analysis, I will take into account more diverse design modalities. Additionally, I will apply the results of the lyric video analysis to generation tasks. I expect that the analysis results are applicable to generate font images that match

a target music. If a more comprehensive analysis, including background, color, and so on, is conducted, the results can contribute to the automatic generation of lyric videos.

- **Analysis of visual designs aside from text designs:** I will analyze visual designs aside from text designs, such as logos and pictograms. There are infinite variations of visual designs in the world. I will pursue finding more general tendencies of visual designs through the analysis.

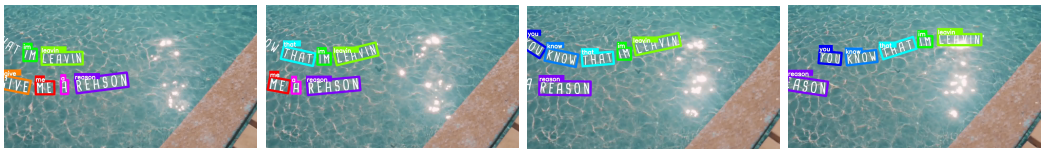
Appendix A

Lyric Word Detection and Tracking by Using Lyric Information

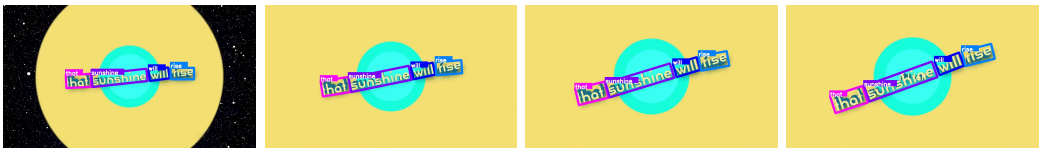
I introduce the methodology [118] used to detect and track lyric words in a lyric video. The technical highlight of the methodology is the use of the lyric word information (i.e., the text data of the lyric word sequence of the song) to obtain accurate tracking results.

A.1 Lyric Word Candidate Detection

First, lyric word candidates are detected as bounding boxes using two pre-trained state-of-the-art scene text detectors, PSENet [119] and CRAFT [120]. The detected bounding boxes are then fed into a state-of-the-art scene text recognizer `TPS-Resnet-BiLSTM-Attn`, which was proposed in [121]. If bounding boxes detected by the above detectors overlap by more than 50%, and the recognition results are the same, these bounding boxes are regarded as duplicates. Accordingly, I remove either box in the later process.



(a) Wavy motion.



(b) Scaling and rotating motion.

Figure A-1: Successful results of lyric word detection and tracking under various motion types. The series of video frames is arranged in order from left to right. The bounding boxes of the detected words are shown.

A.2 Lyric-Frame Matching

As I described in 4.4.1, the lyric-frame matching was conducted by associating the word sequence and the frame sequence of the given lyrics after detection and recognition. The red matching path shown in Figure 4-4 (a) was determined by evaluating the distance $D(k,t)$ between the kth word and the tth frame. A smaller value of $D(k,t)$ means that the probability of the kth lyric word existing in the tth frame is high. More precisely, the distance $D(k,t)$ is defined as $D(k,t) = \min_{b \in B_t} d(k,b)$, where B_t is the set of bounding boxes detected in the tth frame and $d(k,b)$ is the edit function between the kth lyric word detected in the tth frame and the bth word in the same frame. If the kth lyric word is perfectly detected in the tth frame, the distance is $D(k,t) = 0$.

Using the distance $\{D(k,t)|\forall k, \forall t\}$ for dynamic programming (DP), I can efficiently obtain the globally optimal lyric frame matching as shown in the red path in Figure 4-4 (a). In the dynamic time warping (DTW) algorithm, the DP recursion is calculated for each (k,t) from $(k,t) = (1,1)$ to (K,T) as follows:

$$g(k,t) = D(k,t) + \min_{t-\Delta \leq t' < t} g(k-1,t'),$$

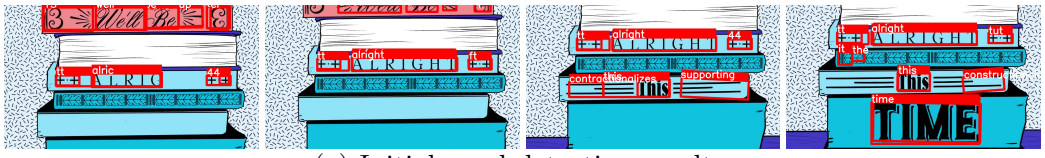
where $g(k,t)$ shows the minimum accumulated distance from $(1,1)$ to (k,t). The parameter Δ indicates the maximum frame skipped on the path. In the experiment, I set $\Delta = 1,000$. This means that a video with 24 fps is allowed to skip approximately 40 s. The calculation complexity of the algorithm is $O(\Delta TK)$.

Note that this lyric-frame matching process using lyric information is essential for lyric videos. For example, the word "the" appears many times in the lyric text; this means that the spatio-temporal location of a certain "the" is ambiguous. Therefore, the lyric-frame matching process needs to fully utilize the continuity of the lyric words, as well as the video frames, to determine the most reliable frame for each lyric word.

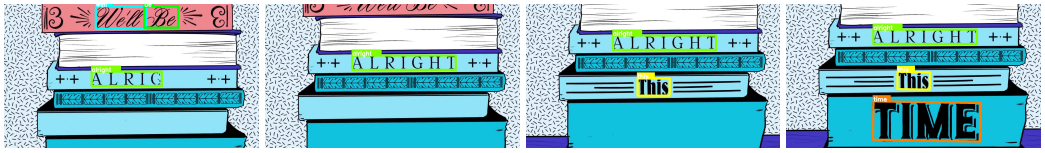
A.3 Tracking of Individual Lyric Words

In the above lyrics-frame matching step, the kth lyric word is only matched to the tth frame; however, this word may also appear around the tth frame. Therefore, I search for such frames around the tth frame, as shown in Figure 4-4 (b). This search is done not only via simple spatio-temporal similarity but also by evaluating the word similarity with the kth word in the neighboring tth frames. If both similarities are larger than a threshold in the t'th frame, we conclude that the same kth word is also found in the t'th frame.

Finally, as shown in Figure 4-4 (c), we conduct an interpolation process as post-processing. If a lyric word is seriously misrecognized and/or occluded in a certain frame, I cannot track the word around the frame using the above simple searching process. If such a missed frame is found, polynomial interpolation is performed between the neighboring frames. The average running time of lyric word tracking per frame is approximately 440 ms.



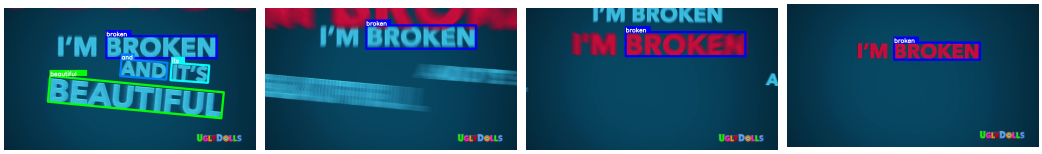
(a) Initial word detection results.



(b) Final tracking results.

Figure A-2: Effect of lyric information. The lyric words in these frames show "we'll be alright this time".

(a) Heavy distortion by partial occlusion ("PREY").



(b) Tracking error caused by multiple appearances of the same word within a short period ("BROKEN").

Figure A-3: Failure results of lyric word detection and tracking.

A.4 Qualitative Evaluation of Lyric Word Detection and Tracking

I applied the above method to all of the frames of the 100 collected lyric videos (approximately 547,100 frames in total) and obtained tracking results for all of the lyric words (approximately 33,800 words in total). Figure A-1 shows several successful results of lyric word detection and tracking. In Figure A-1 (a), I can see that a word with wavy motion can be correctly tracked. As shown in Figure A-1 (b), a word under scaling or rotation for each frame can also be correctly tracked.

Table A.1: Quantitative evaluation of the lyric word detection and tracking. MA: Lyric-frame matching. TR: Tracking. IN: Interpolation. TP: #True-positive. FP: #False-positive. FN: #False-negative. P: Precision (%). R: Recall (%). F: F-measure.

MA	TR	IN	TP	FP	FN	P = $\frac{TP}{TP+FP}$	R= $\frac{TP}{TP+FN}$	F= $\frac{2PR}{P+R}$
✓			72	12	7,698	85.71	0.93	0.0183
✓	✓		5,513	462	2,257	92.27	70.95	0.8022
✓	✓	✓	5,547	550	2,223	90.98	71.39	0.8000

Figure A-2 shows the effect of using lyric information in the lyric-frame matching process and subsequent tracking process to improve accuracy. Because these frames have a complex background (character-like patterns), unnecessary bounding boxes are found in the first word detection step; however, only the correct lyric words remain after matching and tracking the lyric frames.

Figure A-3 shows typical failure cases. The failure in Figure A-3 (a) is caused by severe distortion resulting from the complicated visual design of the video. The word "PREY" is always partially occluded and, therefore, never detected, even by the state-of-the-art word detector. The failure in Figure A-3 (b) is caused by a refrain of the same phrase "I'M BROKEN" in the lyrics. In lyric videos, an important lyric word or phrase sometimes appears repeatedly (i.e., excessively) while changing its appearance, even though the lyric text contains it only one time.

A.5 Quantitative Evaluation of Lyric Word Detection and Tracking

Table A.1 shows the result of a quantitative evaluation of the lyric word detection and tracking using 1,000 frames described in Section 4.3 as ground-truth data. If the bounding boxes of a lyric word according to the proposed method and the corresponding ground-truth data have IoU > 0.5, the detected box is considered to be a successful result. The evaluation result of the lyric-frame matching step and the later tracking step indicates that the precision is 90.98%. From this, I can see that the

false positives are more successfully suppressed than in the case of only lyric-frame matching. The introduction of the interpolation step increased the true positives as expected, even though false positives were also unfortunately increased, and the precision value was slightly decreased. The recall is approximately 71%. The main reasons for false positives are too many decorations and distortions in the word appearance, lyric-frame matching errors resulting from ambiguities in matching, and inconsistency between official lyric texts and actual sung lyrics.

Appendix B

Videos Shown in the Figures

The figures in this paper can be seen in the frame of the following videos. For URLs, the common prefix "`https://www.youtube.com/watch?v=`" is omitted in the list. Note that the URL list of all 100 videos and their annotation data can be found at `https://github.com/uchidalab/Lyric-Video`.

- Figure 4-1: Dua Lipa, New Rules, `AyWsHs5QdiY`
- Figure 4-2: Major Lazer & DJ Snake, Lean On, `rn9AQoI7mYU`
- Figure 4-3: (a) Kelly Clarkson, Broken & Beautiful, `6l8gyacUq4w`; (b) Green Day, Too Dumb to Die, `qh7QJ_jLam0`; (c) Rita Ora, Your Song, `i95Nlb7kiPo`; (d) Selena Gomez, Only You, `T2urfFpDX1c`
- Figures 4-4 and 4-5: Freya Ridings, Castles, `pL32uHAiHgU`
- Figure 4-6: (left) Anne-Marie, 2002, `1tvLIhEaEKo`; (right) Loud Luxury feat. brando, Body, `IetIg7y5k3A`
- Figure 4-14: (a) Ed Sheeran, Shape Of You, `_dK2tDK9grQ`; (b) Kelly Clarkson, Broken & Beautiful, `6l8gyacUq4w`
- Figure A-1: (a) blackbear, wanderlust, `YCRnw3WELY4`; (b) 311, What The?!, `gUGxyD-NOGo`

- Figure A-2: Ed Sheeran, Perfect, `iKzRIweSBLA`
- Figure A-3: (a) Imagine Dragons, Natural, `V5M2WZiAy6k`; (b) Kelly Clarkson, Broken & Beautiful, `6l8gyacUq4w`

Bibliography

[1] *Type Identifier for Beginners*, ser. 978-4-416-11346-2. Seibundo Shinkosha Publishing, 2013.

[2] J. Pons and X. Serra, "musicnn: Pre-trained convolutional neural networks for music audio tagging," in *Late-breaking/demo session in the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

[3] S. Mukai, H. Hibino, and S. Koyama, "Differences in ratings of impressions between japanese calligraphic styles and a japanese font," *International Journal of Affective Engineering*, vol. 16, no. 2, pp. 53–56, 2017.

[4] P. W. Henderson, J. L. Giese, and J. A. Cote, "Impression management using typeface design," *Journal of Marketing*, vol. 68, no. 4, pp. 60–72, 2004.

[5] T. L. Childers and J. Jass, "All dressed up with something to say: Effects of typeface semantic associations on brand perceptions and consumer memory," *Journal of consumer psychology*, vol. 12, no. 2, pp. 93–106, 2002.

[6] M. Ikoma, B. K. Iwana, and S. Uchida, "Effect of text color on word embeddings," in *Proceedings of International Workshop on Document Analysis Systems (DAS)*, 2020, pp. 341–355.

[7] N. Zhao, Y. Cao, and R. W. Lau, "What characterizes personalities of graphic designs?" *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–15, 2018.

[8] T. Kulahcioglu and G. de Melo, "Paralinguistic recommendations for affective word clouds," in *Proceedings of the International Conference on Intelligent User Interfaces (IUI)*, 2019, pp. 132–143.

[9] J. Kang, D. Haraguchi, S. Matsuda, A. Kimura, and S. Uchida, "Shared latent space of font shapes and their noisy impressions," in *Proceedings of the International Conference on Multimedia Modeling (MMM)*, 2022, pp. 146–157.

[10] N. Yasukochi, H. Hayashi, D. Haraguchi, and S. Uchida, "Analyzing font style usage and contextual factors in real images," in *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, 2023.

[11] J. Kato, T. Nakano, and M. Goto, "TextAlive: Integrated design environment for kinetic typography," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI)*, 2015, pp. 3403–3412.

[12] J. Pons, O. Nieto, M. Prockup, E. M. Schmidt, A. F. Ehmann, and X. Serra, "End-to-end learning for music audio tagging at scale," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018, pp. 637–644.

[13] D. Haraguchi, S. Harada, B. K. Iwana, Y. Shinahara, and S. Uchida, "Character-independent font identification," in *Proceedings of the International Workshop on Document Analysis Systems (DAS)*, 2020, pp. 497–511.

[14] D. Haraguchi and S. Uchida, "Local style awareness of font images," in *Proceedings of International Conference on Document Analysis and Recognition (ICDAR) Workshop*, 2023, pp. 242–256.

[15] D. Haraguchi, S. Sakaguchi, J. Kato, M. Goto, and S. Uchida, "Fonts that fit the music: A multimodal design trend analysis of lyric videos," *IEEE Access*, vol. 10, pp. 65 414–65 425, 2022.

[16] Z. Wang, J. Yang, H. Jin, E. Shechtman, A. Agarwala, J. Brandt, and T. S. Huang, "Deepfont: Identify your font from an image," in *Proceedings of the ACM International Conference on Multimedia (ACM Multimedia)*, 2015, pp. 451–459.

[17] H. Ma and D. Doermann, "Font identification using the grating cell texture operator," in *Proceedings of the Document Recognition and Retrieval XII*, 2005, pp. 148–156.

[18] A. Gupta, R. Gutierrez-Osuna, M. Christy, R. Furuta, and L. Mandell, "Font identification in historical documents using active learning," *arXiv preprint arXiv:1601.07252*, 2016.

[19] H. Hayashi, K. Abe, and S. Uchida, "GlyphGAN: Style-consistent font generation based on generative adversarial networks," *Knowledge-Based Systems*, vol. 186, p. 104927, 2019.

[20] R. Suveeranont and T. Igarashi, "Example-based automatic font generation," in *Proceedings of the International Symposium on Smart Graphics (SG)*, 2010, pp. 127–138.

[21] Q. Li, J.-P. Li, and L. Chen, "A bezier curve-based font generation algorithm for character fonts," in *Proceedings of the International Conference on High Performance Computing and Communications (HPCC)*, 2018, pp. 1156–1159.

[22] T. Miyazaki, T. Tsuchiya, Y. Sugaya, S. Omachi, M. Iwamura, S. Uchida, and K. Kise, "Automatic generation of typographic font from small font subset," *IEEE Computer Graphics and Applications*, vol. 40, no. 1, pp. 99–111, 2019.

[23] L. Tang, Y. Cai, J. Liu, Z. Hong, M. Gong, M. Fan, J. Han, J. Liu, E. Ding, and J. Wang, "Few-shot font generation by learning fine-grained local styles," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 7895–7904.

[24] C. Wang, M. Zhou, T. Ge, Y. Jiang, H. Bao, and W. Xu, "Cf-font: Content fusion for few-shot font generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 1858–1867.

[25] S. Uchida, S. Ide, B. K. Iwana, and A. Zhu, "A further step to perfect accuracy by training cnn with larger data," in *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016, pp. 405–410.

[26] H. T. Nguyen, C. T. Nguyen, T. Ino, B. Indurkhya, and M. Nakagawa, "Text-independent writer identification using convolutional neural network," *Pattern Recognition Letters*, vol. 121, pp. 104–112, 2019.

[27] Y. Liu, F. Wei, J. Shao, L. Sheng, J. Yan, and X. Wang, "Exploring disentangled feature representation beyond face identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2080–2089.

[28] A. H. Liu, Y.-C. Liu, Y.-Y. Yeh, and Y.-C. F. Wang, "A unified feature disentangler for multi-domain image translation and manipulation," in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2018, pp. 2590–2599.

[29] O. Press, T. Galanti, S. Benaim, and L. Wolf, "Emerging disentanglement in auto-encoder based unsupervised image content transfer," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[30] G. Chen, J. Yang, H. Jin, J. Brandt, E. Shechtman, A. Agarwala, and T. X. Han, "Large-scale visual font recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 3598–3605.

[31] Y. Bagoriya and N. Sharma, "Font type identification of hindi printed document," *International Journal of Research in Engineering and Technology*, vol. 03, no. 03, pp. 513–516, 2014.

[32] H. Khosravi and E. Kabir, "Farsi font recognition based on sobel–roberts features," *Pattern Recognition Letters*, vol. 31, no. 1, pp. 75–82, 2010.

[33] S. B. Moussa, A. Zahour, A. Benabdelhafid, and A. M. Alimi, "New features using fractal multi-dimensions for generalized arabic font recognition," *Pattern Recognition Letters*, vol. 31, no. 5, pp. 361–371, 2010.

[34] I. M., S. Hamdy, and M. G., "Deep arabic font family and font size recognition," *International Journal of Computer Applications*, vol. 176, no. 4, pp. 1–6, 2017.

[35] C.-B. Jeong, H. K. Kwag, S. Kim, J. S. Kim, and S. C. Park, "Identification of font styles and typefaces in printed korean documents," in *Proceedings of the International Conference on Asian Digital Libraries (ICADL)*, 2003, pp. 666–669.

[36] Z. Yang, L. Yang, D. Qi, and C. Y. Suen, "An EMD-based recognition method for chinese fonts and styles," *Pattern Recognition Letters*, vol. 27, no. 14, pp. 1692–1701, 2006.

[37] K. Abe, B. K. Iwana, V. G. Holmér, and S. Uchida, "Font creation using class discriminative deep convolutional generative adversarial networks," in *Proceedings of the Asian Conference on Pattern Recognition (ACPR)*, 2017, pp. 232–237.

[38] M.-C. Jung, Y.-C. Shin, and S. N. Srihari, "Multifont classification using typographical attributes," in *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, 1999, pp. 353–356.

[39] B. Chaudhuri and U. Garain, "Automatic detection of italic, bold and all-capital words in document images," in *Proceedings of the International Conference on Pattern Recognition (ICPR)*, 1998, pp. 610–612.

[40] Y. Shinahara, T. Karamatsu, D. Harada, K. Yamaguchi, and S. Uchida, "Serif or sans: Visual font analytics on book covers and online advertisements," in *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, 2019, pp. 1041–1046.

[41] S. Oöztuörk, "Font clustering and cluster identification in document images," *Journal of Electronic Imaging*, vol. 10, no. 2, p. 418, 2001.

[42] C. Avilés-Cruz, J. Villegas, R. Arechiga-Martínez, and R. Escarela-Perez, "Unsupervised font clustering using stochastic versio of the em algorithm and global texture analysis," in *Proceedings of the Iberoamerican Congress on Pattern Recognition (CIARP)*, 2004, pp. 275–286.

[43] D. Ghosh, T. Dube, and A. Shivaprasad, "Script recognition—a review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 12, pp. 2142–2161, 2010.

[44] T. Tan, "Rotation invariant texture features and their use in automatic script identification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 7, pp. 751–756, 1998.

[45] W. Pan, C. Y. Suen, and T. D. Bui, "Script identification using steerable gabor filters," in *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, 2005, pp. 883–887.

[46] A. M. Elgammal and M. A. Ismail, "Techniques for language identification for hybrid arabic-english document images," in *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, 2001, pp. 1100–1104.

[47] U. Pal and B. B. Chaudhuri, "Identification of different script lines from multi-script documents," *Image and Vision Computing*, vol. 20, no. 13-14, pp. 945–954, 2002.

[48] L. G. Hafemann, R. Sabourin, and L. S. Oliveira, "Learning features for offline handwritten signature verification using deep convolutional neural networks," *Pattern Recognition*, vol. 70, pp. 163–176, 2017.

[49] Y. Zheng, W. Ohyama, B. K. Iwana, and S. Uchida, "Capturing micro deformations from pooling layers for offline signature verification," in *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, 2019, pp. 1111–1116.

[50] Z.-J. Xing, F. Yin, Y.-C. Wu, and C.-L. Liu, "Offline signature verification using convolution siamese network," in *Proceedings of the International Conference on Graphic and Image Processing (ICGIP)*, vol. 10615, 2018, pp. 415–423.

[51] V. Ruiz, I. Linares, A. Sanchez, and J. F. Velez, "Off-line handwritten signature verification using compositional synthetic generation of signatures and siamese neural networks," *Neurocomputing*, vol. 374, pp. 30–41, 2020.

[52] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *Proceedings of the International Conference on Machine Learning (ICML) Workshop*, 2015.

[53] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[54] Y. Kong, C. Luo, W. Ma, Q. Zhu, S. Zhu, N. Yuan, and L. Jin, "Look closer to supervise better: One-shot font generation via component-based discriminator,"

in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 13 482–13 491.

[55] W. Liu, F. Liu, F. Ding, Q. He, and Z. Yi, "Xmp-font: self-supervised cross-modality pre-training for few-shot font generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 7905–7914.

[56] Y. Gao, Y. Guo, Z. Lian, Y. Tang, and J. Xiao, "Artistic glyph image synthesis via one-stage few-shot learning," *TOG*, vol. 38, no. 6, pp. 1–12, 2019.

[57] N. Srivatsan, J. Barron, D. Klein, and T. Berg-Kirkpatrick, "A deep factorization of style and structure in fonts," in *Proceedings of the The Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 2195–2205.

[58] Y. Zhang, Y. Zhang, and W. Cai, "Separating style and content for generalized style transfer," in *Proceedings of the The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 8447–8455.

[59] M. Ueda, A. Kimura, and S. Uchida, "Which parts determine the impression of the font?" in *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, 2021, pp. 723–738.

[60] H. Zheng, J. Fu, T. Mei, and J. Luo, "Learning multi-attention convolutional neural network for fine-grained image recognition," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 5209–5217.

[61] J. Fu, H. Zheng, and T. Mei, "Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4438–4446.

[62] H. Zheng, J. Fu, Z.-J. Zha, and J. Luo, "Looking for the devil in the details: Learning trilinear attention sampling network for fine-grained image recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 5012–5021.

[63] P. Zhuang, Y. Wang, and Y. Qiao, "Learning attentive pairwise interaction for fine-grained classification," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2020, pp. 13 130–13 137.

[64] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PloS one*, vol. 10, no. 7, p. e0130140, 2015.

[65] J. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," in *Proceedings The International Conference on Learning Representations (ICLR) workshop*, 2015.

[66] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[67] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 618–626.

[68] S. Abnar and W. Zuidema, "Quantifying attention flow in transformers," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020, pp. 4190–4197.

[69] H. Chefer, S. Gur, and L. Wolf, "Transformer interpretability beyond attention visualization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[70] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[71] P. Roy, S. Bhattacharya, S. Ghosh, and U. Pal, "Stefann: Scene text editor using font adaptive neural network," in *Proceedings of the The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 13 228–13 237.

[72] Y. Zhang, Y. Zhang, and W. Cai, "A unified framework for generalizable style transfer: Style and content separation," *TIP*, vol. 29, pp. 4085–4098, 2020.

[73] Y. Wang, Y. Gao, and Z. Lian, "Attribute2font: Creating fonts you want from attributes," *TOG*, vol. 39, no. 4, pp. 69–1, 2020.

[74] Q. Wen, S. Li, B. Han, and Y. Yuan, "Zigan: Fine-grained Chinese calligraphy font generation via a few-shot style transfer approach," in *Proceedings of the ACM International Conference on Multimedia (ACM Multimedia)*, 2021, pp. 621–629.

[75] S. Uchida, Y. Egashira, and K. Sato, "Exploring the world of fonts for discovering the most standard fonts and the missing fonts," in *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, 2015, pp. 441–445.

[76] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016, pp. 785–794.

[77] X. Zhao, K. H. Lin, Y. Fu *et al.*, "Text from corners: A novel approach to detect text and caption in videos," *IEEE Transactions on Image Processing*, vol. 20, no. 3, pp. 790–799, 2010.

[78] Y. Zhong, H. Zhang, and A. K. Jain, "Automatic caption localization in compressed video," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 4, pp. 385–392, 2000.

[79] Z. Yang and P. Shi, "Caption detection and text recognition in news video," in *Proceeding of 5th International Congress on Image and Signal Processing (CISP)*, 2012, pp. 188–191.

[80] H. Yang, B. Quehl, and H. Sack, "A framework for improved video text detection and recognition," *Multimedia Tools and Applications*, vol. 69, no. 1, pp. 217–245, 2014.

[81] D. Zhong, P. Shi, D. Pan, and Y. Sha, "The recognition of Chinese caption text in news video using convolutional neural network," in *Proceeding of IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, 2016, pp. 658–662.

[82] I. A. Zedan, K. M. Elsayed, and E. Emary, "Caption detection, localization and type recognition in Arabic news video," in *Proceedings of the 10th International Conference on Informatics and Systems (INFOS)*, 2016, pp. 114–120.

[83] L. H. Chen and C. W. Su, "Video caption extraction using spatio-temporal slices," *International Journal of Image and Graphics*, vol. 18, no. 2, p. 1850009, 2018.

[84] Y. Xu, S. Shan, Z. Qiu, Z. Jia, Z. Shen, Y. Wang, M. Shi, I. Eric, and C. Chang, "End-to-end subtitle detection and recognition for videos in East Asian languages via CNN ensemble," *Signal Processing: Image Communication*, vol. 60, pp. 131–143, 2018.

[85] W. Lu, H. Sun, J. Chu, X. Huang, and J. Yu, "A novel approach for video text detection and recognition based on a corner response feature map and transferred deep convolutional neural network," *IEEE Access*, vol. 6, pp. 40 198–40 211, 2018.

[86] X. Qian, G. Liu, H. Wang, and R. Su, "Text detection, localization, and tracking in compressed video," *Signal Processing: Image Communication*, vol. 22, no. 9, pp. 752–768, 2007.

[87] P. X. Nguyen, K. Wang, and S. Belongie, "Video text detection and recognition: Dataset and benchmark," in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2014, pp. 776–783.

[88] L. Gomez and D. Karatzas, "MSER-based real-time text detection and tracking," in *Proceedings of 22nd International Conference on Pattern Recognition (ICPR)*, 2014, pp. 3110–3115.

[89] L. Wu, P. Shivakumara, T. Lu, and C. L. Tan, "A new technique for multi-oriented scene text line detection and tracking in video," *IEEE Transactions on Multimedia*, vol. 17, no. 8, pp. 1137–1152, 2015.

[90] S. Tian, X. Yin, Y. Su, and H. Hao, "A unified framework for tracking based text detection and recognition from web videos," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 542–554, 2017.

[91] C. Yang, X. C. Yin, W. Y. Pei *et al.*, "Tracking based multi-orientation scene text detection: A unified framework with dynamic programming," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3235–3248, 2017.

[92] W. Y. Pei, C. Yang, L. Y. Meng, J. B. Hou, S. Tian, and X. C. Yin, "Scene video text tracking with graph matching," *IEEE Access*, vol. 6, pp. 19 419–19 426, 2018.

[93] Y. Wang, L. Wang, and F. Su, "A robust approach for scene text detection and tracking in video," in *Proceedings of the 19th Pacific Rim Conference on Multimedia (PCM)*, 2018, pp. 303–314.

[94] Y. Wang, L. Wang, F. Su, and J. Shi, "Video text detection with fully convolutional network and tracking," in *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, 2019, pp. 1738–1743.

[95] X. C. Yin, Z. Y. Zuo, S. Tian, and C. L. Liu, "Text detection, tracking and recognition in video: A comprehensive survey," *IEEE Transactions on Image Processing*, vol. 25, no. 6, pp. 2752–2773, 2016.

[96] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised learning of video representations using LSTMs," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015, pp. 843–852.

[97] A. Zramdini and R. Ingold, "Optical font recognition using typographical features," *TPAMI*, vol. 20, no. 8, pp. 877–882, 1998.

[98] N. Goel, M. Sharma, and L. Vig, "Font-ProtoNet: Prototypical network-based font identification of document images in low data regime," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshop*, 2020, pp. 556–557.

[99] O. Bychkov, K. Merkulova, G. Dimitrov, Y. Zhabska, I. Kostadinova, P. Petrova, P. Petrov, I. Getova, and G. Panayotova, "Using neural networks application for the font recognition task solution," in *Proceedings of 55th International Scientific Conference on Information, Communication and Energy Systems and Technologies (ICEST)*, 2020, pp. 167–170.

[100] B. Bauermeister, *A manual of comparative typography: The PANOSE system.* Van Nostrand Reinhold Company, 1988.

[101] F. Slimane, S. Kanoun, J. Hennebert, A. M. Alimi, and R. Ingold, "A study on font-family and font-size recognition applied to arabic word images at ultra-low resolution," *Pattern Recognition Letters*, vol. 34, no. 2, pp. 209–218, 2013.

[102] D. Eck, T. Bertin-Mahieux, and P. Lamere, "Autotagging music using supervised machine learning," in *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR)*, 2007, pp. 367–368.

[103] M. D. Hoffman, D. M. Blei, and P. R. Cook, "Easy as cba: A simple probabilistic model for tagging music," in *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR)*, 2009, pp. 369–374.

[104] E. Coviello, L. Barrington, A. B. Chan, and G. R. G. Lanckriet, "Automatic music tagging with time series models," in *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, 2010, pp. 81–86.

[105] M. I. Mandel, D. Eck, and Y. Bengio, "Learning tags that vary within a song," in *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, 2010, pp. 399–404.

[106] G. Marques, M. A. Domingues, T. Langlois, and F. Gouyon, "Three current issues in music autotagging," in *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, 2011, pp. 795–800.

[107] E. Coviello, Y. Vaizman, A. B. Chan, and G. R. G. Lanckriet, "Multivariate autoregressive mixture models for music auto-tagging," in *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, 2012, pp. 547–552.

[108] D. Liang, J. W. Paisley, and D. Ellis, "Codebook-based scalable music tagging with poisson matrix factorization," in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014, pp. 167–172.

[109] K. Choi, G. Fazekas, and M. Sandler, "Automatic tagging using deep convolutional neural networks," in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 805–811.

[110] J. Choi, J. Lee, J. Park, and J. Nam, "Zero-shot learning for audio-based music classification and tagging," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 67–74.

[111] K. M. Ibrahim, E. V. Epure, G. Peeters, and G. Richard, "Should we consider the users in contextual music auto-tagging models?" in *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020, pp. 295–301.

[112] E. Law, K. West, M. I. Mandel, M. Bay, and J. S. Downie, "Evaluation of algorithms using games: The case of music tagging," *Proceedings of 10th International Society for Music Information Retrieval Conference (ISMIR)*, pp. 387–392, 2009.

[113] A. Gupta, A. Vedaldi, and A. Zisserman, "Synthetic data for text localisation in natural images," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2016, pp. 2315–2324.

[114] L. Hubert and P. Arabie, "Comparing partitions," *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985.

[115] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[116] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," *Communications in statistics-theory and methods*, vol. 3, no. 1, pp. 1–27, 1974.

[117] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE transactions on pattern analysis and machine intelligence*, vol. PAMI-1, no. 2, pp. 224–227, 1979.

[118] S. Sakaguchi, J. Kato, M. Goto, and S. Uchida, "Lyric video analysis using text detection and tracking," in *Proceedings of the 14th IAPR International Workshop on Document Analysis Systems (DAS)*, 2020, pp. 426–440.

[119] W. Wang, E. Xie, X. Li, W. Hou, T. Lu, G. Yu, and S. Shao, "Shape robust text detection with progressive scale expansion network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9336–9345.

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

[121] J. Baek, G. Kim, J. Lee, S. Park, D. Han, S. Yun, S. J. Oh, and H. Lee, "What is wrong with scene text recognition model comparisons? dataset and model analysis," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 4715–4723.