Embedding Font Impression Word Tags Based on Co-occurrence

Yugo Kubota Seiichi Uchida
Kyushu University, Japan
{yugo.kubota, uchida}@human.ait.kyushu-u.ac.jp

Abstract

Different font styles (i.e., font shapes) convey distinct impressions, indicating a close relationship between font shapes and word tags describing those impressions. This paper proposes a novel embedding method for impression tags that leverages these shape–impression relationships. For instance, our method assigns similar vectors to impression tags that frequently co-occur in order to represent impressions of fonts, whereas standard word embedding methods (e.g., BERT and CLIP) yield very different vectors. This property is particularly useful for impression-based font generation and font retrieval. Technically, we construct a graph whose nodes represent impression tags and whose edges encode co-occurrence relationships. Then, we apply spectral embedding to obtain the impression vectors for each tag. We compare our method with BERT and CLIP in qualitative and quantitative evaluations, demonstrating that our approach performs better in impression-guided font generation.

1. Introduction

Different fonts convey different impressions. Fig. 1 shows several examples of fonts and their associated impression word tags (hereafter referred to as *impression tags*) from the MyFonts dataset [2]. These impression tags are assigned to each font by users via crowdsourcing. The impressions conveyed by a font can vary significantly depending on its shape. For example, the font in the top row conveys positive and casual impressions and may be suitable for printing the word "HAPPY." In contrast, the font in the second row conveys negative impressions and may be unsuitable for the same word "HAPPY."

Understanding the relationship between font shapes and impressions is crucial from both theoretical and practical perspectives. From a theoretical standpoint, uncovering this relationship deepens our understanding of visual design principles and shape perception. From a practical perspective, understanding this relationship has wide-ranging applications in areas such as branding, design evaluation, computer-aided font generation, and other typographic tasks.

Figure 1. Examples of fonts and their impression word tags. The six characters "HERONS" are shown as they represent diverse stroke variations within each font.

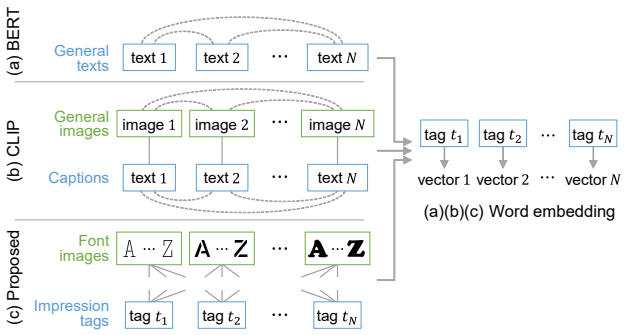


Figure 2. Conceptual comparison of word embedding methods, where (c) shows the proposed method.

To understand this relationship, a key challenge lies in the numerical representation (*i.e.*, vector representation) of font impressions. Traditional approaches typically rely on word embedding methods such as word2vec [24], with more recent work introducing methods like BERT [9] and CLIP [29]. However, as shown in Figs. 2(a) and (b), these existing methods treat impression tags as ordinary words in general texts, rather than as impressions grounded in font shapes. Indeed, as shown in Fig. 1, these impressions are inherently tied to the visual characteristics of fonts. For example, in general usage, the tags "`elegant`" and "`thin`"

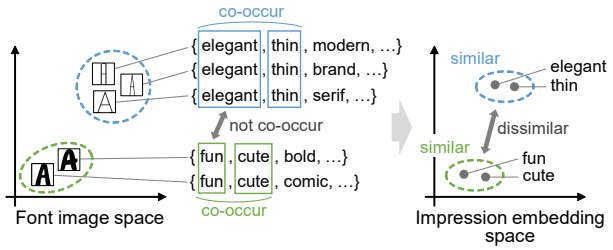


Figure 3. Impression tags "`elegant`" and "`thin`" frequently co-occur and are generally associated with fonts with similar shapes. Our co-occurrence-based method assigns similar "impression embedding vectors" to them. Consequently, their similarity reflects the visual similarity of fonts.

differ in meaning, yet both may be assigned to the same font—as seen in the third example—due to similarities from a visual perspective. In such cases, their embedding vectors should be similar and reflect their shared grounding in that particular font shape.

In this paper, we propose a co-occurrence-based embedding method for impression tags. As shown in Fig. 2(c), our method addresses the limitations of conventional word embedding methods by focusing on how impressions arise from font shapes. Since impressions are conveyed by font shapes, we assume that impression tags frequently associated with the same fonts (*i.e.*, co-occurring tags) should be represented by similar embedding vectors. For example, as illustrated in Fig. 3, two tags "`elegant`" and "`thin`" tend to co-occur in fonts with similar shapes. Consequently, by focusing on tag co-occurrence and assigning similar vectors to frequently co-occurring tags, we can obtain vector representations of impression tags that reflect the visual characteristics of fonts.

The proposed embedding method has several notable features. First, our embedding vectors are tightly linked to font shapes rather than general texts and general images, and therefore behave very differently from the vectors produced by BERT and CLIP. Second, our vectors are particularly suitable for modeling how font shapes convey specific impressions; this feature is useful when regulating neural network-based font generation processes by specifying the desired impressions.

We evaluate the proposed co-occurrence-based embedding method for impression tags through four experiments:

- First, we visualize the distribution of impression tags in the embedding space and observe their clusters.
- Second, we examine the correspondence between impression tag embeddings and actual font images, and find that tags that are similar in the proposed embedding space tend to be associated with similar fonts.
- Third, we demonstrate how the impression tag–font correspondence can help identify "missing tags" (*i.e.*, tags that should be assigned but are absent) and "noisy tags" (*i.e.*, tags that are incorrectly assigned).
- Finally, we conduct a conditional font generation experiment, in which the embedding vectors guide the generation of fonts that reflect specified impressions.

2. Related Work

2.1. Font Impression Analysis

Researchers have long studied how font shapes affect the subjective impressions people form. Early work dates back to the 1920s [7, 28], and numerous subsequent studies have demonstrated consistent associations between specific fonts and their impressions [20, 26, 32]. In practical applications, these associations serve as effective tools for conveying nonverbal cues, such as brand and product identities in marketing [8, 13, 35], or emotional tone in text-based communication [3, 4, 6, 12]. Traditionally, much of this work has been based on small-scale user studies focusing on relatively limited font sets and impression vocabularies.

Recently, studies on font–impression relationships have increasingly focused on large-scale datasets and machine-learning-based approaches [2, 27]. Several studies have leveraged these large-scale datasets alongside multimodal models to learn font-impression correspondences [5, 15, 18]. In particular, following the success of CLIP [29], contrastive learning has been used to embed fonts and impressions into a shared space that captures their correspondence [17, 33]. While previous methods vectorize both font images and impressions, our method emphasizes vectorizing impressions based on tag co-occurrence within fonts.

2.2. Generating Fonts with Impressions

From a practical perspective, the relationship between font shapes and impressions is important for impression-guided font generation. In general, machine learning-based font generation follows a few-shot paradigm that requires a couple of pre-designed characters as style references [11, 34, 38–40]. In contrast, impression-guided font generation seeks to generate fonts by specifying only the expected impressions and thus can eliminate the need for pre-designed characters. This allows non-expert users to intuitively generate new fonts by specifying impressions that describe their desired font style, thereby making the process user-friendly.

Vectorizing impressions is a key step for providing style-related information to impression-guided generative models. Matsuda *et al.* [22, 23] introduced generative adversarial approaches that utilize word2vec [24] for vectorizing impression tags. Furthermore, some approaches employ CLIP [29] to incorporate visual aspects of text into font generation [14, 33]. Currently, GRIF-DM, proposed by Kang *et al.* [16], is the state-of-the-art impression-guided font generation method, where impression vectors derived

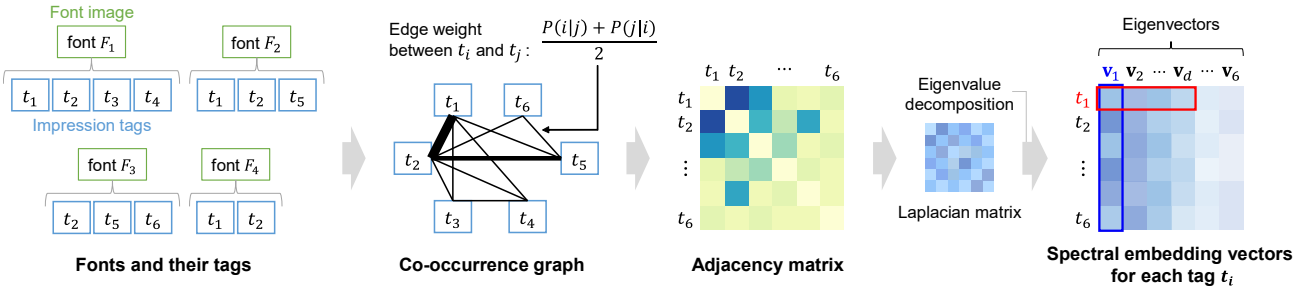


Figure 4. Overview of our impression tag embedding method. For simplicity, the tag vocabulary size is set to $N = 6$ in this illustration.

from BERT [9] are used as conditions in a diffusion model. In this paper, we develop a diffusion model based on GRIF-DM to demonstrate that the impression vectors derived from our method can be effectively used as conditions for high-quality font generation.

2.3. Word Embedding Models and LLMs

Large language models (LLMs) have become ubiquitous for vectorizing text in various tasks, owing to their training on massive corpora. Seminal examples include word2vec and BERT, which remain widely used for both standalone text processing [1, 25] and multimodal applications such as image captioning [19, 21, 37] and text-based image generation [10, 30, 31]. In the latter domain, CLIP [29] has garnered particular attention by aligning images and textual captions in a shared latent space, making the learned text features widely applicable to vision-and-language tasks.

Despite their versatility, these language models are not explicitly designed to handle the specialized scenario of font impressions. In practice, tags with distinct meanings in general usage can still apply to the same font. For instance, Fig. 1 shows that "`happy`" and "`funny`" are assigned to the same font, indicating a strong relationship in terms of font impressions despite their differing general meanings. This paper compares our proposed method with BERT and CLIP, demonstrating that it captures font-related impressions more effectively than LLM-based embeddings.

3. Font Impression Dataset

Throughout this paper, we use the MyFonts dataset [2], collected from MyFonts.com, one of the largest font marketplaces on the web. Fig. 1 shows several examples. The original dataset contains 18,815 fonts and 1,824 impression tags, assigned by both font design experts and non-expert users in an open-vocabulary manner. Consequently, the dataset inevitably contains missing and noisy tags.

Since our goal is to analyze the impressions conveyed by font shapes, we exclude dingbat fonts (*i.e.*, non-letter symbols) via manual inspection. After this exclusion, 17,380 fonts remain, each associated with a subset of the 1,818 impression tags. Each font has between 1 and 184 tags, with an average of 16.1 and a median of 11. Each tag appears in 1 to 5,991 fonts, with an average of 153 and a median of 35. In the following experiments, we split these 17,380 fonts into 15,639 for training and 1,741 for testing.

4. Embedding Impression Tags Based on Co-occurrence

4.1. Overview

Fig. 4 illustrates an overview of our proposed embedding method based on spectral embedding [36]. Specifically, we first construct a graph based on the co-occurrence of impression tags across fonts, where each node corresponds to an impression tag. An edge between two nodes is weighted according to their *conditional co-occurrence probabilities* in the dataset. We then represent the graph as a weighted adjacency matrix and perform an eigenvalue decomposition on its Laplacian matrix to embed each node into a low-dimensional embedding space. This low-dimensional representation is referred to as the spectral embedding, which we use as the vector representation for each impression tag.

4.2. Co-occurrence of Font Impression Tags

Assume we have a collection of fonts, in which each font comprises a set of character images (*e.g.*, 'A'–'Z') in a particular style. As shown in Fig. 1, each font is assigned multiple impression tags. In most cases, impression tags are single words (*e.g.*, "`cool`"), but some are hyphenated phrases (*e.g.*, "`comic-book`"). Let N denote the vocabulary size of the impression tags.

The co-occurrence of two impression tags means that both tags are assigned to the same font. For instance, in the left part of Fig. 4, the tags t_1 and t_2 co-occur in font F_1, and the same pair also co-occurs in fonts F_2 and F_4. By contrast, t_1 and t_3 co-occur only in F_1. By counting how often each tag pair co-occurs across all fonts, we construct a graph. The graph's nodes correspond to the overall vocabulary of impression tags, and each edge is weighted according to their co-occurrence in training fonts.

More specifically, each edge weight is computed as the average of the conditional co-occurrence probabilities between the two tags (nodes). Given two tags t_i and t_j, where $i, j \in [1, N]$, we calculate $P(i \mid j)$ and $P(j \mid i)$ by dividing the co-occurrence count by the individual frequencies of t_j and t_i, respectively. Finally, the edge weight between t_i and t_j is defined as $(P(i \mid j) + P(j \mid i))/2$.

A simpler approach might be to use raw co-occurrence frequencies as edge weights. However, impression tags vary widely in frequency; some are common (high-frequency), while others are rare (low-frequency). Relying solely on raw frequency would make high-frequency tags appear strongly correlated with nearly all others, obscuring the nuanced relationships involving rare tags. Instead, by averaging the conditional co-occurrence probabilities, we better capture specific associations for infrequent tags and reduce the bias toward dominant ones.

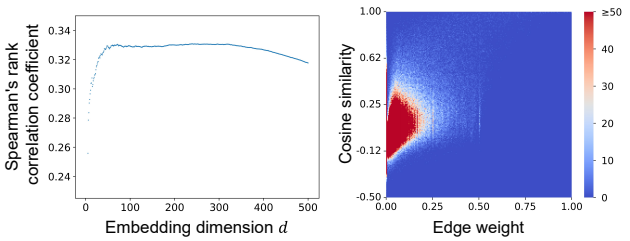
4.3. Graph Spectral Embedding of Impression Tags

As shown in Fig. 4, we use spectral embedding on the co-occurrence graph to represent each impression tag as a vector. Specifically, to compute the spectral embedding vectors, we perform eigenvalue decomposition on the Laplacian matrix of the co-occurrence graph. First, we construct an $N \times N$ adjacency matrix that defines the edge weights between nodes, and then compute the corresponding $N \times N$ symmetric normalized Laplacian matrix. Next, we perform eigenvalue decomposition on this matrix and select the first d eigenvectors corresponding to the d smallest eigenvalues.

We represent each impression tag t_i by stacking the i-th component of the first d eigenvectors and normalizing the resulting vector. More specifically, let $\mathbf{v}_1, \ldots, \mathbf{v}_d$ be the top d eigenvectors of the Laplacian matrix, each of length N (number of tags). We obtain the d-dimensional embedding for impression tag t_i by extracting the i-th component from each eigenvector and stacking them into the vector $\big(\mathbf{v}_1(i), \ldots, \mathbf{v}_d(i)\big)^\top$. Finally, we normalize this vector to have unit length.

Our method relies on a linear-algebra-based approach, in contrast to neural network-based embeddings such as BERT or CLIP. One might assume that a linear approach has limited expressive power; however, our experiments show that this spectral embedding strategy outperforms embeddings obtained from BERT or CLIP for capturing font impressions. Moreover, our method avoids dependence on complex neural architectures and iterative training procedures.

Using the training set of $15,639$ fonts introduced in Sec. 3, we embed $1,818$ impression tags into a d-dimensional space following the procedure described above. We set $d = 100$ based on the analysis provided in Sec. 4.4. In the experiments in later sections, these $1,818$ embedding vectors (each of dimension 100) serve as the numerical representations of individual impression tags.



(a) (b)

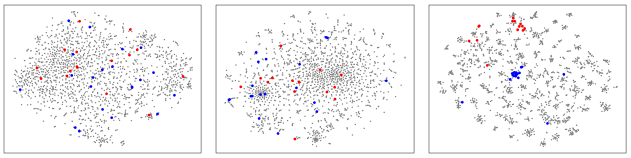
Figure 5. We determine d so that frequently co-occurring tags are placed closer together in the embedding space. (a) Spearman's rank correlation coefficient for each value of d. (b) Correlation between edge weights and the cosine similarity of embedding vectors for tag pairs, with dimensionality $d = 100$.

4.4. Determining the Embedding Dimension

Choosing the number of eigenvectors d (*i.e.*, the embedding dimension) is critical. Smaller values of d lead to insufficient representational capacity, whereas very large values result in sparse representations that obscure relationships between tags. We determine d based on the principle that tags connected by larger edge weights should be placed closer together in the embedding space. Specifically, we vary d, compute Spearman's rank correlation coefficient between the edge weight of each tag pair and the cosine similarity of their embedding vectors, and then select the dimensionality that maximizes this correlation.

Fig. 5(a) shows the correlation coefficients for each dimension d, computed between the edge weights and the cosine similarities of the embedding vectors. The figure indicates that the correlation peaks in the range of 100 to 300 dimensions and changes only slightly within this range, suggesting that 100 dimensions are almost as effective at representing the data as 300. Therefore, we set $d = 100$ in the subsequent experiments. Despite this relatively small dimension, our method outperforms BERT and CLIP, as demonstrated in the experimental results presented later.

Fig. 5(b) provides a heatmap comparing each tag pair's edge weight in the co-occurrence graph with the cosine similarity of their embedding vectors. The heatmap indicates a positive correlation: pairs of tags with larger edge weights generally exhibit higher cosine similarity, indicating that the embedding vectors effectively capture co-occurrence relationships. Note that most edge weights are small, suggesting that only a limited subset of tag pairs strongly co-occur. In contrast, some pairs with low edge weights still display relatively high similarity. This situation arises when two tags do not directly co-occur but are indirectly related. For example, tags A and B co-occur, and tags B and C also co-occur, then, even if tags A and C never co-occur (and therefore, their edge weight is zero), their embedding vectors become closer.



(a) BERT (b) CLIP (c) Proposed

Figure 6. t-SNE visualization of the embedding vector distributions. The red and blue dots represent impression tags that belong to the same cluster respectively, in the proposed embedding space, as listed in Fig. 7.

(a) (b)

Figure 7. Examples of tag clusters. The tags in (a) correspond to the red dots in Fig. 6, and those in (b) correspond to the blue dots. Shown below each cluster are three example fonts that are closely associated with the corresponding tags.

5. Distribution of Impression Vectors

5.1. Observing the Distribution by t-SNE

We compared the properties of our impression tag embeddings described in Sec. 4.3 with those derived from BERT [9] and CLIP [29]. To obtain the embedding vector for each tag using BERT, we simply input each tag as a single-word prompt and extracted its embedding vector using the publicly available checkpoint `google-bert/bert-base-uncased`. Similarly, we extracted embedding vectors from CLIP using the checkpoint `openai/clip-vit-base-patch32`.

Fig. 6 visualizes the distributions of impression tag embedding vectors obtained using BERT, CLIP, and our proposed method. In the BERT and CLIP embedding spaces, the vectors appear to be distributed uniformly, whereas in the proposed method, the distribution exhibits non-uniform density. Notably, the proposed method forms numerous small clusters, which set it apart from the other two methods. This indicates that, by explicitly accounting for each tag's relationship to font images, our approach produces embeddings that are substantially different from those of text-based methods. Moreover, the formation of numerous small clusters suggests that certain impression tags frequently co-occur, while others rarely do.

5.2. Clustering the Impression Vectors

Since the t-SNE visualization revealed the cluster structure of the vectors produced by the proposed method, we applied standard k-means clustering to all $N = 1,818$ impression tags in the 100-dimensional embedding space. The number of clusters, k, was set to 140 based on the peak of the silhouette score, a commonly used clustering quality metric.

In Fig. 6(c), the red and blue dots represent impression tags that belong to the same cluster in the proposed embedding space. Although the red dots appear slightly scattered due to the nonlinearity of t-SNE, the blue dots form a clear cluster. An important observation is that the same tags appear more widely scattered in (a) and (b), indicating that the proposed embedding method exhibits different characteristics from those of BERT and CLIP.

5.3. Cluster-Wise Observation

Figs. 7(a) and (b) show two example tag clusters along with sample fonts that have multiple tags that belong to each cluster. First, looking at the tag clusters themselves, we find that even though the proposed method is based solely on tag co-occurrence, the resulting embedding vectors of tags reflect semantic similarity in natural language and group similar tags into the same cluster (*e.g.*, "`future`" and "`futuristic`" in (a) and "`nostalgic`" and "`nostalgia`" in (b)). This observation indicates that the proposed method works effectively. Furthermore, when examining the fonts associated with these clusters, we observe that the fonts in (a) exhibit fairly consistent stroke widths, whereas in (b), they are more varied. Hence, each cluster is underpinned by a set of fonts that share certain style features.

On the other hand, there are cases where the clusters do not align with a tag's typical meaning in natural language. For instance, in tag cluster (a), the tags "`technological`", "`wide`", "`extend`", and "`square`" are grouped with "`future`" although the word "`future`" has a significantly different meaning from the others in natural language. However, when we consider font shapes, these tags are reasonably similar. Indeed, fonts labeled with "`future`" often have wide, square shapes. It is also notable that the abstract word "`future`" is associated with the geometric tags "`wide`" and "`square`". Similarly, in cluster (b), "`thick-and-thin`" is grouped with "`retro`" and "`nostalgic`", suggesting that a classic impression is often conveyed by fonts with contrasting thick and thin strokes. These results demonstrate that the proposed method can derive impression vectors that accurately reflect actual font shapes, despite relying solely on co-occurrence.

6. Relationship between Embedded Impression Tags and Font Images

6.1. Consistency between Impressions and Fonts

Fonts with similar shapes should share similar impressions; conversely, similar impression tags should be assigned to

similarly shaped fonts. We refer to this property as the *consistency* between impression tags and fonts. Our tag-embedding method needs to reflect the consistency; namely, impression tags that are close to each other in the embedded space should be assigned to the same fonts. Specifically, consider an impression tag t_i (where $i = 1, \ldots, 1,818$) and let F_{t_i} be the set of test fonts associated with this tag. We then find the tag s_i closest to t_i in the embedding space and obtain the corresponding font set F_{s_i}. If t_i is indeed similar to s_i, then the set F_{t_i} should also be similar to F_{s_i}.

Table 1 quantifies this consistency by measuring the similarity between the two font sets, F_{t_i} and F_{s_i}. We used three set similarity metrics: the Jaccard coefficient, the Dice coefficient, and the overlap coefficient. The Jaccard coefficient represents the ratio of the intersection to the union of two sets, the Dice coefficient emphasizes common elements, and the overlap coefficient measures the ratio of the intersection to the size of the smaller set. All three metrics yield values in the range $[0, 1]$, where a higher value indicates greater similarity. The values shown in the table are averaged over all t_i.

Our proposed method outperforms BERT and CLIP across all metrics, indicating that font sets associated with closely embedded impression tags tend to overlap more. In particular, this difference is especially pronounced for the overlap coefficient. Since the overlap coefficient uses the smaller of the two sets as its reference for measuring the intersection, it is more sensitive to low-frequency tags (those associated with fewer fonts). This dataset contains many low-frequency tags, mainly due to the open-vocabulary tag assignment process and missing tags. Nevertheless, when embedded using our method, even these low-frequency tags show more substantial overlap with the font sets of nearby tags. This result suggests that our co-occurrence-based approach captures shape-based relationships among impression tags more accurately than text-based BERT and the general-purpose image-text alignment of CLIP.

6.2. Robustness to Missing and Noisy Tags

An appropriately constructed embedding of impression tags can help mitigate missing and noisy tags. In the embedding space, tags positioned close to a given tag t_i can be regarded as candidate tags that could reasonably be applied to the same font as t_i. By assuming that all relevant neighbors should also be assigned to the font, we can reduce the risk of missing tags. Conversely, if a font includes a tag that is embedded far away from its other tags, this outlier tag can be considered a noisy tag that may have been inappropriately assigned.

We evaluated whether the proposed method produces an embedding space capable of addressing missing and noisy tag problems. First, we assessed the method's robustness to missing tags based on the proximity of tags assigned to each font in the embedding space. Suppose a font is associated with I distinct impression tags. For each ordered pair (t_i, t_j) of these tags (where $i, j \in [1, I]$, $i \neq j$), we determine the rank $p \in [1, N]$ of t_j among $N - 1$ impression tags based on its similarity to t_i. We then average the rank p over all $I(I - 1)$ ordered pairs.

Table 1. Quantitative evaluation of the consistency between the impression embedding space and the font image distribution, measured by the Jaccard, Dice, and Overlap coefficients. Robustness to missing and noisy tags is also evaluated.

Method	Jaccard↑	Dice↑	Overlap↑	Missing↓	Noisy↑
BERT [9]	0.023	0.037	0.091	790.2	0.122
CLIP [29]	0.044	0.068	0.175	745.3	0.163
Proposed	**0.139**	**0.198**	**0.344**	**543.5**	**0.594**

We define this average value as the font's "missing-tag robustness," where smaller values indicate greater robustness. In the ideal—but practically unachievable—scenario, every tag pair assigned to a font would be each other's nearest neighbor, resulting in an average of 1. We then compute the mean of this value across all test fonts to evaluate the embedding method's missing-tag robustness.

Next, we quantify robustness to noisy tags by assuming that a tag that is relatively dissimilar to the others can be considered a noisy tag. Given a set consisting of I tags assigned to a font and one additional tag randomly selected from the N total tags, we compute the average distance between each of the $I + 1$ tags and the remaining I tags. If the randomly selected tag yields the largest average distance, we consider it successfully identified as a noisy tag. We repeat this process for each font using ten randomly selected additional tags for each test font, and compute the proportion of trials in which the random tag is identified as a noisy tag. We define this proportion as the embedding method's noisy-tag robustness, with higher values indicating stronger robustness. In the ideal case—which, again, is unlikely in practice—this proportion would be 1, meaning that the random tag is always detected as a noisy tag in every trial.

Table 1 presents the robustness of each embedding space to missing and noisy tags. An embedding space is considered robust if it enables the detection of missing tags (overlooked) and noisy tags (erroneous). The results show that the proposed method achieves greater robustness to both missing and noisy tags. Hence, it is more effective in handling incomplete or erroneous font-impression annotations. This suggests that our co-occurrence-based approach, which grounds impression tags in font shapes, yields a clearer representation of semantic proximity, thereby facilitating more accurate tag correction and supplementation.

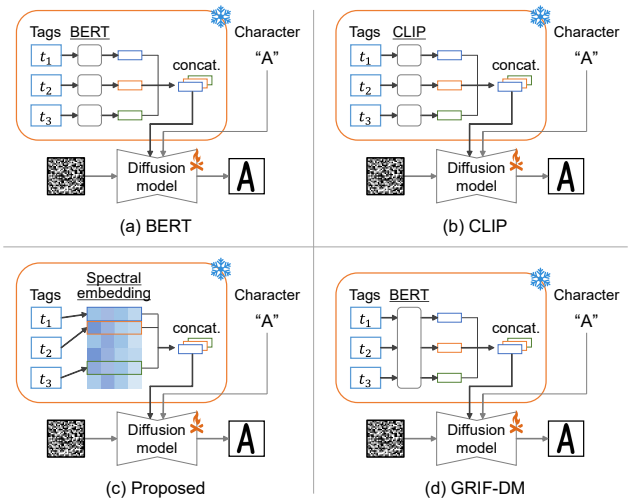


Figure 8. Impression-guided font generation using different word embedding methods.

7. Font Generation Using Impression Tags

7.1. Impression-Guided Font Generation with a Diffusion Model

Once impression tags are embedded as vectors, one natural application is to generate fonts that reflect a given impression. As described in Sec. 2.2, several methods have already pursued this goal, with GRIF-DM [16], which represents the state-of-the-art diffusion-based approach. Fig. 8(d) illustrates the architecture of GRIF-DM, in which a set of impression tags is first vectorized by BERT and then used to condition the diffusion model.

Building on the concept of GRIF-DM, we replace its impression-conditioning vectors with those derived from BERT, CLIP, or our proposed method, as illustrated in Figs. 8(a), (b), and (c), respectively. It is worth noting that in both (a) and (d), BERT serves as the underlying language model, but the impression vectors are obtained in different ways: in (a), each tag is individually embedded, whereas in (d), all tags are input collectively as a single prompt. We expect the latter approach to capture tag interrelationships more effectively through BERT's self-attention mechanism.

We also compared our impression embedding method with the state-of-the-art approach, Impression-CLIP [17]. This method associates fonts with impressions via CLIP-based contrastive learning. Impression-CLIP embeds fonts and impressions into a shared latent space such that positive font–impression pairs are closer and negative pairs are farther apart. We trained Impression-CLIP using the same training fonts and tags as our proposed method. We then used its impression encoder to vectorize impression tags and condition the diffusion model. We compared its font generation performance with that of our proposed method.

7.2. Experimental Setup

To train each diffusion model, we used the 15,639 training fonts introduced in Sec. 3. Specifically, for each training font, the model was trained to generate images of the 26 uppercase letters 'A'–'Z' by conditioning on the font's impression-tag set and a target letter. Other training details follow those of the original GRIF-DM paper [16].

We evaluated the quality of generated images using the Fréchet Inception Distance (FID) and the Structural Similarity Index (SSIM). To investigate the effect of the number of impression tags K, we evaluated $K \in \{1, 3, 5, 7, 10\}$ and the case $K = \text{all}$ (*i.e.*, all impression tags for a font). Consequently, we focus on fonts that have at least ten impression tags and randomly select 5,000 such fonts from the full set of 17,380 fonts for evaluation. [1] For each of 5,000 fonts, we randomly shuffle the complete set of its impression tags. Then, for $K = 1$, we use the first tag from the shuffled list as a condition; for $K = 3$, the first three tags; and so on. Thus, when $K = 5$, we simply use the first five tags from the shuffled list.

Regardless of K, the 5,000×26 font images in this evaluation serve as ground-truth references. We compute the FID between the distribution of the 5,000×26 ground-truth images and that of the 5,000×26 generated images conditioned on K impression tags per font. For SSIM, we compare each generated image directly with its corresponding ground-truth image and average the similarity scores across all 26 letters and all fonts.

7.3. Experimental Results

Fig. 9 shows a quantitative evaluation of the generated font images across different values of K. Overall, the results indicate that using more impression tags generally improves the quality of the generated fonts. This is because supplying additional tags provides a more fine-grained description of the font's shape, thus facilitating the generation of fonts closer to the ground truth. When comparing the proposed method with BERT and CLIP, the proposed method consistently outperforms both across all values of K. As discussed in Sec. 6.1, this superiority can be attributed to the tighter alignment of the proposed impression-tag embedding space with the font-image distributions, which facilitates impression-based font generation. In other words, the co-occurrence-based vectors appear more effective than the general-purpose embeddings from BERT and CLIP. Furthermore, our proposed method also outperforms the comparative embedding method, *i.e.*, Impression-CLIP [17].

When focusing on GRIF-DM, the proposed method performs comparably to GRIF-DM when all tags are used, showing only a slight performance difference. However,

[1] As in GRIF-DM [16], we combine training and test sets for FID evaluation to ensure a sufficient number of fonts. Note that the trends in the evaluation results remain consistent even when using only the test set.

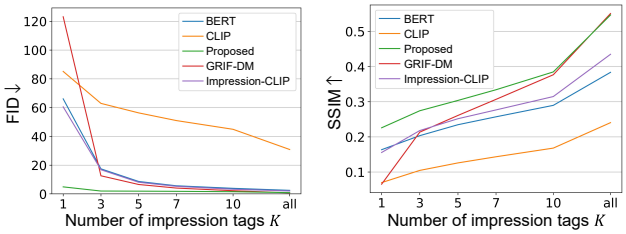


Figure 9. Quantitative evaluation of the quality of impression-guided font generation for different numbers of impression tags, K. "all" on the horizontal axis indicates that all impression tags associated with a font were used for generation.

when fewer tags are available ($K < 10$), the proposed method significantly outperforms GRIF-DM. A possible explanation is that GRIF-DM heavily relies on a self-attention mechanism to capture relationships among multiple tags, which makes it more difficult to accurately represent the font shape when only a few tags are available. Indeed, with very few tags (particularly when $K = 1$), we find that even the simpler BERT-based method—without any self-attention among tags—can outperform GRIF-DM.

Fig. 10 provides representative examples of the generated fonts. First, when all tags are used, both the proposed method and GRIF-DM generate fonts that closely resemble the ground truth across all samples. Next, when fewer tags are available, the other four methods—except for our proposed method—often fail to generate legible characters, whereas the proposed method, although its style deviates from the ground truth, still generates legible characters that reasonably match the available impression tags. In most examples, roughly $K = 5$ tags are sufficient to produce a font style that closely resembles the ground truth, and using more tags further narrows the gap. For instance, in the top example with $K = 5$, adding "`serif`" and "`heavy`" to the set {"`modern`", "`masculine`", "`elegant`"} (*i.e.*, $K = 3$) makes the generated style closer to the ground truth. In real-world applications, it is not always possible to obtain a large set of impression tags. Therefore, the ability to generate high-quality fonts from only a few tags is an important requirement. Overall, the results demonstrate that the proposed method is effective even in practical scenarios where only a limited number of tags are available.

8. Conclusion

This paper proposes a co-occurrence-based embedding method for font impression word tags, demonstrating that these embedding vectors are grounded in font shapes and more effectively capture the relationships between font shapes and impressions than general-purpose, text-centric models such as BERT and CLIP. Our experiments demonstrated that the proposed method is not only relatively robust

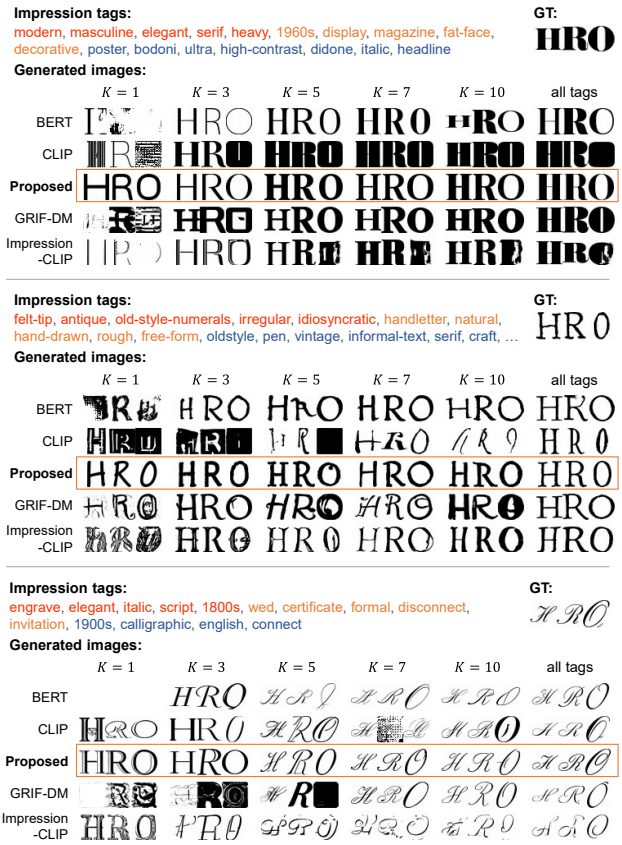


Figure 10. Generated images conditioned on different numbers of impression tags, K. In the top example, $K = 3$ indicates the use of the first three impression tags—"`modern`", "`masculine`", and "`elegant`"—taken from those associated with the ground truth (GT). Other values of K correspond to different numbers of tags. The first five and ten impression tags are highlighted in red and orange, respectively. All examples are taken from the test fonts.

to missing and noisy tags but also effective for impression-guided font generation. This practical value was confirmed in scenarios with both abundant and sparse impression tags.

Despite these advantages, our method is limited to the existing impression-tag vocabulary, unlike LLMs that can embed arbitrary words. A promising direction is to integrate LLMs with our co-occurrence-based approach to vectorize arbitrary words consistently with font shapes. Moreover, while our results suggest increased robustness to noisy or missing tags, these issues still pose challenges when co-occurrence data are severely incomplete. Incorporating dedicated mechanisms to mitigate such labeling errors—similar to those used in prior work on handling noise [23]—represents another important avenue for future research.

Acknowledgments: This work was supported by JSPS KAKENHI-JP25H01149, JST CRONOS-JPMJCS24K4, JST BOOST-JPMJBS2406, and Hoso Bunka Foundation.

References

[1] Samar Al-Saqqa and Arafat Awajan. The use of word2vec model in sentiment analysis: A survey. In *Proceedings of the International Conference on Artificial Intelligence, Robotics and Control (AIRC)*, pages 39–43, 2019. 3

[2] Tianlang Chen, Zhaowen Wang, Ning Xu, Hailin Jin, and Jiebo Luo. Large-scale tag-based font retrieval with generative feature learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9116–9125, 2019. 1, 2, 3

[3] Saemi Choi and Kiyoharu Aizawa. Emotype: Expressing emotions by changing typeface in mobile messenger texting. *Multimedia Tools and Applications*, 78(11):14155–14172, 2019. 2

[4] Saemi Choi, Toshihiko Yamasaki, and Kiyoharu Aizawa. Typeface emotion analysis for communication on mobile messengers. In *Proceedings of the 1st International Workshop on Multimedia Alternate Realities*, pages 37–40, 2016. 2

[5] Saemi Choi, Shun Matsumura, and Kiyoharu Aizawa. Assist users' interactions in font search with unexpected but useful concepts generated by multimodal learning. In *Proceedings of the International Conference on Multimedia Retrieval (ICMR)*, pages 235–243, 2019. 2

[6] Rintaro Chujo, Atsunobu Suzuki, and Ari Hautasaari. Exploring the effects of japanese font designs on impression formation and decision-making in text-based communication. *IEICE Transactions on Information and Systems*, E107-D(3):354–362, 2024. 2

[7] R. C. Davis and H. J. Smith. Determinants of feeling tone in type faces. *Journal of Applied Psychology*, 17(6):742–764, 1933. 2

[8] Maísa M. M. de Sousa, Fabiana M. Carvalho, and Rosemary G. F. A. Pereira. Do typefaces of packaging labels influence consumers' perception of specialty coffee? A preliminary study. *Journal of Sensory Studies*, 35(5):e12599, 2020. 2

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, 2019. 1, 3, 5, 6

[10] Kevin Frans, L. B. Soros, and Olaf Witkowski. CLIPDraw: Exploring text-to-drawing synthesis through language-image encoders. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5207–5218, 2022. 3

[11] Bin Fu, Fanghua Yu, Anran Liu, Zixuan Wang, Jie Wen, Junjun He, and Yu Qiao. Generate Like Experts: Multi-stage font generation by incorporating font transfer process into diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6892–6901, 2024. 2

[12] Ari Hautasaari, Minami Aramaki, Rintaro Chujo, and Takeshi Naemura. EmoScribe Camera: A virtual camera system to enliven online conferencing with automatically generated emotional text captions. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, pages 121:1–121:7, 2024. 2

[13] Pamela W. Henderson, Joan L. Giese, and Joseph A. Cote. Impression management using typeface design. *Journal of Marketing*, 68(4):60–72, 2004. 2

[14] Kota Izumi and Keiji Yanai. CLIPFontDraw: Stylizing fonts with CLIP. *IEEE Access*, 12:154811–154822, 2024. 2

[15] Jihun Kang, Daichi Haraguchi, Seiya Matsuda, Akisato Kimura, and Seiichi Uchida. Shared latent space of font shapes and their noisy impressions. In *Proceedings of the 28th International Conference on Multimedia Modeling (MMM)*, pages 146–157, 2022. 2

[16] Lei Kang, Fei Yang, Kai Wang, Mohamed Ali Souibgui, Lluis Gomez, Alicia Fornés, Ernest Valveny, and Dimosthenis Karatzas. GRIF-DM: Generation of rich impression fonts using diffusion models. In *Proceedings of the 27th European Conference on Artificial Intelligence (ECAI)*, pages 226–233, 2024. 2, 7

[17] Yugo Kubota, Daichi Haraguchi, and Seiichi Uchida. Impression-CLIP: Contrastive shape-impression embedding for fonts. In *Proceedings of the 18th International Conference on Document Analysis and Recognition (ICDAR)*, pages 70–85, 2024. 2, 7

[18] Tugba Kulahcioglu and Gerard De Melo. Fonts like this but happier: A new way to discover fonts. In *Proceedings of the 28th ACM International Conference on Multimedia (MM)*, pages 2973–2981, 2020. 2

[19] Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, Yejin Choi, and Jianfeng Gao. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *Proceedings of the 16th European Conference on Computer Vision (ECCV)*, pages 121–137, 2020. 3

[20] Ying Li and Ching Y. Suen. Typeface personality traits and their design characteristics. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, pages 231–238, 2010. 2

[21] Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. Visual relationship detection with language priors. In *Proceedings of the 14th European Conference on Computer Vision (ECCV)*, pages 852–869, 2016. 3

[22] Seiya Matsuda, Akisato Kimura, and Seiichi Uchida. Impressions2Font: Generating fonts by specifying impressions. In *Proceedings of the 16th International Conference on Document Analysis and Recognition (ICDAR)*, pages 739–754, 2021. 2

[23] Seiya Matsuda, Akisato Kimura, and Seiichi Uchida. Font generation with missing impression labels. In *Proceedings of the 26th International Conference on Pattern Recognition (ICPR)*, pages 1400–1406, 2022. 2, 8

[24] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3111–3119, 2013. 1, 2

[25] Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre,

Ilana Heintz, and Dan Roth. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, 56(2):30:1–30:40, 2023. 3

[26] Gary R. Morrison. Communicability of the emotional connotation of type. *Educational Communication and Technology*, 34(4):235–244, 1986. 2

[27] Peter O'Donovan, Jānis Lībeks, Aseem Agarwala, and Aaron Hertzmann. Exploratory font selection using crowdsourced attributes. *ACM Transactions on Graphics (TOG)*, 33(4):1–9, 2014. 2

[28] A. T. Poffenberger and R. B. Franken. A study of the appropriateness of type faces. *Journal of Applied Psychology*, 7 (4):312–329, 1923. 2

[29] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 8748–8763, 2021. 1, 2, 3, 5, 6

[30] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with CLIP latents. *arXiv preprint arXiv:2204.06125*, 2022. 3

[31] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2022. 3

[32] Audrey Dawn Shaikh, Barbara S. Chaparro, and Doug Fox. Perception of fonts: Perceived personality traits and uses. *Usability News*, 8(1):1–6, 2006. 2

[33] Yuki Tatsukawa, I-Chao Shen, Anran Qi, Yuki Koyama, Takeo Igarashi, and Ariel Shamir. FontCLIP: A semantic typography visual-language model for multilingual font applications. *Computer Graphics Forum (CGF)*, 43(2):e15043, 2024. 2

[34] Vikas Thamizharasan, Difan Liu, Shantanu Agarwal, Matthew Fisher, Michael Gharbi, Oliver Wang, Alec Jacobson, and Evangelos Kalogerakis. VecFusion: Vector font generation with diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7943–7952, 2024. 2

[35] Carlos Velasco, Alejandro Salgado-Montejo, Fernando Marmolejo-Ramos, and Charles Spence. Predictive packaging design: Tasting shapes, typefaces, names, and sounds. *Food Quality and Preference*, 34:88–95, 2014. 2

[36] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007. 3

[37] Sijin Wang, Ruiping Wang, Ziwei Yao, Shiguang Shan, and Xilin Chen. Cross-modal scene graph matching for relationship-aware image-text retrieval. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1508–1517, 2020. 3

[38] Yangchen Xie, Xinyuan Chen, Li Sun, and Yue Lu. DG-Font: Deformable generative networks for unsupervised font generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5130–5140, 2021. 2

[39] Zhenhua Yang, Dezhi Peng, Yuxin Kong, Yuyi Zhang, Cong Yao, and Lianwen Jin. FontDiffuser: One-shot font generation via denoising diffusion with multi-scale content aggregation and style contrastive learning. In *Proceedings of the AAAI conference on artificial intelligence*, pages 6603–6611, 2024.

[40] Yexun Zhang, Ya Zhang, and Wenbin Cai. Separating style and content for generalized style transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8447–8455, 2018. 2