SVGformer: Representation Learning for Continuous Vector Graphics using Transformers

Defu Cao[1], Zhaowen Wang[2], Jose Echevarria[2], Yan Liu[1]
[1]University of Southern California
[2]Adobe Research, Inc.
{defucao, yanliu.cs}@usc.edu, {zhawang, echevarr}@adobe.com

Abstract

Advances in representation learning have led to great success in understanding and generating data in various domains. However, in modeling vector graphics data, the pure data-driven approach often yields unsatisfactory results in downstream tasks as existing deep learning methods often require the quantization of SVG parameters and cannot exploit the geometric properties explicitly. In this paper, we propose a transformer-based representation learning model (SVGformer) that directly operates on continuous input values and manipulates the geometric information of SVG to encode outline details and long-distance dependencies. SVGfomer can be used for various downstream tasks: reconstruction, classification, interpolation, retrieval, etc. We have conducted extensive experiments on vector font and icon datasets to show that our model can capture high-quality representation information and outperform the previous state-of-the-art on downstream tasks significantly.

1. Introduction

In the last few years, there have been tremendous advances in image representation learning [22, 26, 37], and these representations lead to great success in many downstream tasks such as image reconstruction [40], image classification [14, 16], etc. However, most previous works have focused on analyzing structured bitmap format, which uses a grid at the pixel level to represent textures and colors [18]. Therefore, there is still considerable room for improving the representation of detailed attributes for vector objects [25].

In contrast to bitmap image format, scalable vector graphics (SVG) format for vector images is widely used in real-world applications due to its excellent scaling capabilities [12, 20, 33]. SVGs usually contain a mixture of smooth curves and sharp features, such as lines, circles, polygons, and splines, represented as parametric curves in digital format. This allows us to treat SVGs as sequential data and learn their compact and scale-invariant representation using neural network models. However, how to automatically learn an effective representation of vector images is still non-trivial as it requires a model to understand the high-level perception of the 2D spatial pattern as well as geometric information to support the high-quality outcome in downstream tasks.

Transformer-based models [30] have been proven to achieve start-of-the-art results when dealing with sequential data in various problems including Natural Language Processing (NLP) [34] tasks and time-series forecasting [41] problems. We argue that representation learning for SVG is different from these tasks for two reasons: Firstly, most if not all NLP tasks need a fixed token space to embed the discrete tokens, while SVGs are parameterized by continuous values which make the token space infinite in the previous setting; Secondly, the number of commands and the correlation between the commands vary greatly from one SVG to another, which is hard to handle by a pure data-driven attention mechanism. For example, the font data may vary across different families while sharing similar styles within the same font family. Such property is encoded in the sequential data explicitly and can provide geometric dependency guidance for modeling the SVG if used appropriately.

To tackle the above challenges and fully utilize the geometric information from SVG data, this work introduces a novel model architecture named SVGformer, which can take continuous sequential input into a transformer-based model to handle the complex dependencies over SVG commands and yield a robust representation for the input. Specifically, we first extract the SVG segment information via medial axis transform (MAT) [2] to convey the geometric information into the learned representation. Then we introduce a 1D convolutional embedding layer to preserve the original continuous format of SVG data, as opposed to previous vector representation learning models [8] which need a preprocessing step to discretize the input to limited discrete tokens. After that, we inject the structural relationship between commands into the proposed geometric self-attention module in the encoder layer to get the hidden representation

Method	Font-MF [3]	SVG-VAE [20]	Im2Vec [25]	DeepVecFont [33]	DeepSVG [8]	LayoutTrans [12]	SVGformer (ours)
Encoding Modality	Seq	Img	Img	Img&Seq	Seq	Seq	Seq
Decoding Modality	Seq	Img&Seq	Seq	Img&Seq	Seq	Seq	Seq
Model Architecture	GP-LVM	VAE	RNN	LSTM+CNN	Transformer	Transformer	Transformer
Sequence Format	Keypoints	Commands	Keypoints	Commands	Commands	Commands	Commands
Geometric Information	-	-	-	-	-	-	Segment

Table 1. Comparison of our SVGformer model and recent models for vector representation learning, where "Seq" donates sequence, and "Img" donates image.

of each SVG. The representation learned with the reconstruction pretext task can be used in various downstream tasks including classification, retrieval, and interpolation. To the best of our knowledge, our proposed model is the first to explicitly consider vector geometric information as well as directly deal with the raw input of SVG format in an end-to-end encoder-decoder fashion.

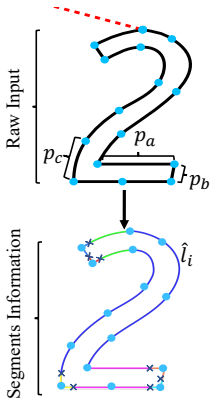
The **main contributions** of this paper includes:

- SVGformer captures both geometric information and curve semantic information with the geometric self-attention module, which synergizes the strengths of MAT and the transformer-based neural network to handle long-term sequence relationships in SVG.
- SVGformer can take original continuous format as input which can effectively reduce the token space in the embedding layer. Thereby the model is general for all continuous vector graphics without extra quantization.
- SVGformer achieves new state-of-the-art performance on four downstream tasks of both Font and Icon dataset. For example, it outperforms prior art by 51.2% on classification and 42.5% on retrieval tasks of the font dataset.

2. Related work

Transformer models in the computer version. In the realm of computer vision, many researchers have proposed using self-attention to model long-term dependencies, similar to how transformers are used in natural language processing and other real-world applications [5, 6, 34, 36]. By leveraging self-attention in the encoder-decoder architecture, these methods have shown promising results in computer vision benchmarks, overcoming the limitations imposed by inductive convolution bias [10]. The fields of those benchmarks include object detection [7], image generation [11], and classification on both image [21] and video [32]. In most related works, the self-attention layer in computer vision takes a feature map as input and computes the attention weights between each pair of features. This operation can result in an updated feature map where each position has information about any other feature in the same image which could be costly for high-resolution or long sequential inputs. Thus, there are multiple works aimed at overcoming this problem. For example, Informer [41] proposes a sparse attention mechanism by sorting the important queries; Axial-DeepLab [31] computes the attention score sequentially along the two spatial axis and [23] processes the patches of feature maps instead of the whole image. In addition, as inductive bias can always be beneficial to the learning process [35], we need to inject the position information into the transformers to help update features that are lost in the input sequence. For learning the SVG representation, we can introduce structural inductive bias into transformer-based models by obtaining the position information from the input sequence.

Representation learning for SVG. Designing and manipulating the SVG image including fonts and icons require substantial expertise and expert knowledge [3]. Machine learning-based models can automatically manipulate the outlines and skeletons of vector images by learning from specific datasets. For the previous works, Font-MF [3] utilize the Gaussian Process Latent Variable Model [17] to learn a manifold of fonts so that it can synthesize fonts in unseen fonts. Im2Vec [25] proposes a new neural network that can generate complex vector graphics with varying topologies and only requires indirect supervision from readily-available raster training images (i.e., with no vector counterparts). Instead of directly using bitmap format, DeepVecFont [33] introduces a new generative paradigm to handle unstructured data (e.g., vector glyphs) by randomly sampling plausible synthesis results and a dual-modality learning strategy that utilizes both image-aspect and sequence-aspect features of fonts to synthesize vector glyphs. SVG-VAE [20] is a generative model with an autoencoder to learn the font style feature and an SVG decoder to generate vector glyphs. In addition, LayoutTransformer [12] uses the self-attention mechanism to capture the contextual relationship between different vector elements, which provides a potential solution for SVG data. Besides, DeepSVG [8] raises a hierarchical transformer-based generative model for vector graphics and collects a large-scale dataset of SVG along with deep learning-based SVG manipulation. However, most models that process the SVG format fail to handle the continuous nature [4, 39] of the input correctly. Besides, most of them do not consider



Raw Commands	Arguments	Geometric Segments
p_0	M (13.2, 3.0)	-1
	…	
p_a	L (16.5, 18.9)	
p_b	L (16.2, 21.0)	
	…	
p_c	C (5.8, 19.1, 6.3, 17.5, 7.2, 16.2)	
	…	
p_n	EOS (∅)	-1

Figure 1. The extraction of geometric information. Starting from the original SVG commands p_i delimited by blue circles, we apply MAT [2] to obtain a new set of commands and labels $(\hat{p}_i, \hat{l}_i)$ (denoted by curve color and delimited with crosses) that may not exactly align with the original commands. A unique label l_i is finally assigned to each original command, as outlined in black color in the last column of the table.

geometric information, which can directly equip the relationship of different curves into deep learning-based methods. Refer to Table 1 for a complete comparison of related works.

3. Method

3.1. Overview

We propose a transformer-based deep neural network, SVGformer, as a general solution for multiple vector graphics downstream tasks. The overall architecture of SVGformer is illustrated in Figure 2. The raw t-th SVG's arguments contain the type and corresponding coordinates of its commands, where the coordinates with continuous values are first constructed into a tensor format, $\mathcal{X}^t \in \mathbb{R}^{n\times 8}$ and n is the number of commands. Then the input tensor is fed into the embedding layer to get the input vector for the encoder-decoder model. The learnable embedding layer contains three components to specifically abstract the information from the original input: 1) the continuous token component uses a 1D convolution to map $\mathcal{X}^t$ into an embedding vector U^t; 2) the position component captures the positional information P^t from $\mathcal{X}^t$; 3) the geometric component treats the outline labels l_i (refer to Figure 1) as a discrete token to map extra contextual geometric information S^t to the embedding vector. In addition, we build an adjacency matrix E based on the labels and the neighboring commands in order to further utilize the structural information in the training process.

Next, we combine the embedding vectors to feed the encoder module, where we introduce the geometric self-attention to replace the canonical self-attention in order to generate a compact representation with a higher receptive field efficiently. Specifically, we leverage structural relationships by encoding geometric segment information via a graph convolution layer and injecting it into spared attention score. Finally, we pass the hidden representation produced by the encoder into the SVG-decoding module consisting of the self-attention mechanism [41] and fully connected layers to fine-tune the total model. The decoder aims to reconstruct the original SVG commands from a masked input as well as predict the type of commands.

In this way, the overall network can be trained end-to-end by backpropagation of a reconstruction loss and classification loss. Precisely, the classification loss has 3 different labels as we only consider three types of commands: "Move" (M), "Line" (L) and "Curve" (C).

3.2. Geometric Information from Raw Input

As the attention mechanism generally only calculates the similarities via the commands' semantic features, we want to cooperate explicit geometric dependencies into the attention mechanism to provide our model with the ability to capture the structural relation between the commands. v We start from the raw SVG input $\{p_0, p_1, p_2, \ldots, p_n\}$, where p_i denotes the i-th command in a given SVG, and n is the number of original commands. As shown in Figure 1, we obtain semantic labeling of segments with MAT. This tool leverages the medial axis and outlines information to find new segments labeled based on different geometric relationships: $\{(\hat{p}_1, \hat{l}_1), (\hat{p}_2, \hat{l}_2), \ldots, (\hat{p}_m, \hat{l}_m)\}$, where $\hat{l}_i$ is the label of the new command $\hat{p}_i$. Commands with the same label have a strong position relationship inside the given SVG, which is usually ignored by feature engineering. Note that MAT can usually split an original command into several segments (new commands), so $m \geq n$. However, directly adding such extra commands into the neural network will complicate the task of representation learning. Instead, we propose to map the new segments and labels back to the original input. Specifically, we group different commands by start and end points so that we have $p_i = \{\hat{p}_k, \ldots, \hat{p}_j\}$. To obtain the final label l_i for p_i, we calculate the longest label by the distance between the start and end point of the label set $\{\hat{l}_k, \ldots, \hat{l}_j\}$. Then the final data we use to train the model is $\{(p_1, l_1), (p_2, l_2), \ldots, (p_n, l_n)\}$. We will introduce how to incorporate geometric information into the transformer in the next several sections.

3.3. Continuous Feature Embedding

Representation learning requires the model to learn the inductive biases from the data and training task which can facilitate the downstream task. Thus, any inductive bias from raw input is able to coach the training process. In this work, we have identified two inductive biases present in raw input: continuous values and geometric information. While previous related works have typically relied on manually discretizing continuous values, our model is able to support continuous-valued inputs. Additionally, our model

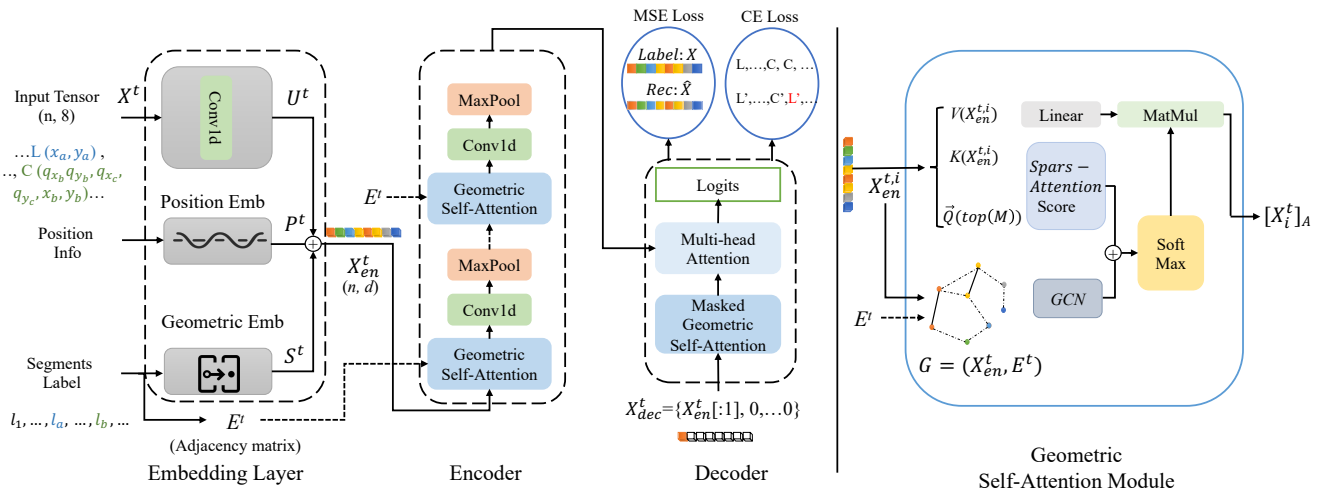


Figure 2. Overall model structure of SVGformer.

is able to automatically discover structural information from the geometric information present in the input, providing further support for the inductive biases we have identified. This section will present proposed modifications to make it compatible with continuous SVG commands input with geometric information rather than with discrete pre-processed index sequences.

Specifically, for the raw input, we first set a maximum sequence length L_x for the entire dataset, then we pad shorter samples with arbitrary values (0 in our setting) as the length of individual samples may vary considerably regarding the SVG images. Then, we generate a padding mask that adds a large negative value to the attention fraction of the padded positions before computing the self-attentive distribution with the softmax function. After that, different from previous works that can only support discrete input with fix length of the look-up table, we use a **continuous value embedding layer** with 1D convolutional operation to project the t-th SVG input $\mathcal{X}^t$ into a d-dimension vector U^t:

$$U^t = \text{CNN}(\mathcal{X}^t) \quad . \tag{1}$$

Note that this embedding layer can fully utilize the translation equivariance [1] of convolutional operation for SVG input and can have the ability of rotation invariance and size invariance with data augmentation. Corresponding to the continue value embedding layer, we preserve the input SVG commands by using the **position embedding layer**:

$$\begin{aligned} P^t(p, 2j) &= \sin(m/(2L_x)^{2j/d}), \\ P^t(p, 2j+1) &= \cos(m/(2L_x)^{2j/d}) \quad , \end{aligned} \tag{2}$$

where $j \in \{1, \ldots, \lfloor d/2 \rfloor\}$ and $m \in \{0, \ldots, L_x - 1\}$. In this way, self-attention can compute the similarity with access to global information and affordable consumption [41]. Note that this position encoding function is inductive and parameters free to handle variable input length [19]. However, the above inductive embedding functions fail to capture the geometric dependency among the different positions of raw SVG commands. Thus, we exact the segment label l_i^t for each input command p_i^t according to Section 3.2 and use a **geometric embedding layer** with a look-up table, containing all the segmented labels, to get the geometric information S. Thus, we finally maintain the feeding vector $X_{en}^t \in \mathbb{R}^{L_x \times d}$:

$$X_{en}^t = \alpha U^t + \beta P^t + \gamma S^t \quad , \tag{3}$$

where α, β, γ are the factors balancing the magnitude between the scalar projection and position/geometric embeddings.

3.4. SVGformer Encoder with Geometric Information

After the embedding layer, we feed the vector X_{en}^t to the encoder to extract latent representation. We refer the reader to the detailed description of the transformer model in the original work [30]. In this section, we will introduce the key technologies different from the vanilla transformer. Specifically, the encoder consists of several stacked geometric self-attention modules which can deal with long-term sequential dependency and geometric dependency in the attention mechanism jointly.

Sparse Query for Self-Attention. As the representation learning of SVG requires the model to have the ability to capture precise long-term dependency, we need to utilize the commands' features via an efficient attention mechanism. The canonical self-attention is defined based on the scaled dot-product of the input tuple, i.e., query $\mathbf{q} \in \mathbb{R}^d$, key $\mathbf{K} \in \mathbb{R}^{L_K \times d}$, and value $\mathbf{V} \in \mathbb{R}^{L_V \times d}$. Specifically, the i-th

query's attention is defined as a kernel smoother in a probability form [29]:

$$\mathcal{A}(\mathbf{q}_i, \mathbf{K}, \mathbf{V}) = \sum_j \frac{k(\mathbf{q}_i, \mathbf{k}_j)}{\sum_l k(\mathbf{q}_i, \mathbf{k}_l)} \mathbf{v}_j = \mathbb{E}_{p(\mathbf{k}_j|\mathbf{q}_i)}[\mathbf{v}_j] \ , \quad (4)$$

where $p(\mathbf{k}_j|\mathbf{q}_i) = k(\mathbf{q}_i, \mathbf{k}_j)/\sum_l k(\mathbf{q}_i, \mathbf{k}_l)$ and $k(\mathbf{q}_i, \mathbf{k}_j)$ selects the asymmetric exponential kernel $\exp(\mathbf{q}_i \mathbf{k}_j^\top / \sqrt{d})$. Inspired by [41], we calculate the importance of queries by Kullback-Leibler divergence:

$$M(\mathbf{q}_i, \mathbf{K}) = \max_j \{ \frac{\mathbf{q}_i \mathbf{k}_j^\top}{\sqrt{d}} \} - \frac{1}{L_K} \sum_{j=1}^{L_K} \frac{\mathbf{q}_i \mathbf{k}_j^\top}{\sqrt{d}} \quad , \quad (5)$$

where $M(\mathbf{q}_i, \mathbf{K})$ is the i-th query's sparsity score. A larger $M(\mathbf{q}_i, \mathbf{K})$ gives a higher chance to activate the dominate dot-product pairs in the self-attention. Thus, we can build a new query $\overline{Q}$ containing top u queries with query sparsity score given by M and calculate the sparsity self-attention score by:

$$A_{att} = \overline{\mathbf{Q}} \mathbf{K}^\top / \sqrt{d} \quad . \quad (6)$$

Note that we can avoid information loss by generating different sparse query-key pairs according to different attention heads and this attention score can capture the relationships between input commands from a data-driven aspect.

Geometric Dependency in Self-Attention. Segments with the same label share a strong relationship even if they are at a long distance in the input sequence, which cannot be reflected in the original self-attention module which only considers the semantic characteristics. In addition, commands can be represented as the nodes lying in a non-Euclidean space and linked by edges with the geometric information. such as the adjacency and the same segment label. In this work, we propose to apply the graph convolution network (GCN) [15] to extract the structural relationship between different commands. Specifically, we first use the geometric label from Section 3.2 to build the weight matrix E where the commands adjacent to each other or sharing the same label have an edge in the weight matrix E, otherwise there is no edge in the weight matrix E.

This has two advantages, firstly, the structure information of SVG is contained in the weight matrix, and secondly, commands sharing the same labels can distinguish local details by their own features. After getting the weight matrix, we use GCN to extract the structural information. The goal is then to learn a function of SVG commands on a graph $\mathcal{G} = (V, E)$, where V is the set of nodes' (commands in SVG) feature of each single SVG, i.e., X_{en}^t at the first layer, and E is the in the form of the adjacency matrix, which is A based on Figure 1. Our model can produce command-level output $A_{geo} \in \mathbb{R}^{N \times d}$ with L multiple layers GCN where l-th layer output H^l can be used to represent the latent vectors to get the final output :

$$H^{l+1} = f(H^l, E) = \sigma(E H^l W^l) \quad , \quad (7)$$

where H^0 is the input features of the GCN, i.e., $\mathcal{X}_{en}$ at the very first layer, f is the graph convolutional function with learnable matrix W and σ is the non-linear activation function for GCN and $H^L = A_{geo}$.

Geometric Self-Attention Module. To better encode the geometric information into attention layers, we propose a new self-attention module in SVGformer. The attention mechanism needs to estimate correlations on both semantics (features from commands) part and the geometric (dependency of each command) part. Thus, for each ordered command pair (p_i, p_j), we inject the relationship score from GCN into the sparse attention score. Concretely, the attention mechanism with the modified attention score can be represented as

$$\mathcal{A}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}(A_{att} + A_{geo})\mathbf{V} \quad . \quad (8)$$

To enhance the ability of our model on handling the permutation invariance where the order of shapes in an SVG image is arbitrary, we apply the 1D convolutional operation following an ELU activation function [9] and max pooling [38] on the output of the attention layer: $\mathbf{X}_{j+1} = \text{MaxPool}\left(\text{ELU}(\text{Conv1d}([\mathbf{X}_j]_{\mathcal{A}_j})) \right)$, where $[\cdot]_{\mathcal{A}}$ represents the j-th geometric self-attention module.

The final representation of the encoder Z is the last layer's output of the encoder module and can be used for multiple downstream tasks.

3.5. SVGformer Decoder

The decoder part is modified from the decoder structure in [41], which can reconstruct the long sequences of SVG at one forward operation rather than a step-by-step way with the input of $\mathbf{X}_{\text{de}}^t = \text{Concat}(\mathbf{X}_{\text{token}}^t, \mathbf{X}_{\mathbf{0}}^t) \in \mathbb{R}^{(L_{\text{token}}+L_y)\times d}$, where $\mathbf{X}_{\text{token}}^t \in \mathbb{R}^{L_{\text{token}} \times d}$ is the start token, $\mathbf{X}_{\mathbf{0}}^t \in \mathbb{R}^{L_y \times d}$ is a placeholder of 0 for the target sequence. Specifically, it consists of masked geometric self-attention and multi-head attention, where the masked geometric self-attention sets masked dot-products to $-\infty$ and gives up the geometric GCN component, which can prevent data leakage in the decoding stage. Besides, a fully connected layer is needed to acquire the final logits for the loss function.

3.6. Loss Function

We need to reconstruct the input commands as well as predict the type of the commands. Thus, we choose the reconstruction loss as the mean squared error (MSE) loss function between the reconstructed commands and the ground-truth command and the classification loss as the cross entropy (CE) with logits between the predicted command type and the type label.

4. Experiments

4.1. Setup

We summarize the critical settings in this section. For input SVG, we choose those font inputs with a maximum length L_x equals to 60 and icon inputs with a maximum length equals to 50. Then pad the font commands less than 60 (50 for icon commands) by 0. For the embedding layer, a 1D convolutional operation is used as the trainable module for SVG recognition with kernel width equal to 3 and stride equal to 1. Both the 1D convolutional operation and the other two components inside the embedding layer's output dimension are 512. We recommend α, β, γ to be 1 inside the embedding layer if the sequence input has been normalized. In the training stage, the sparse attention's head number is 8, and we add geometric information given by one GCN layer, which output shape fits the attention score given by the sparse self-attention. To gain the global representation for a specific SVG, we use two stacked geometric self-attention modules followed by 1D convolutional operations and maximum pooling operations to finish the fusion process of all the commands and get the hidden representation with the shape of $L_x \times 512$. For the decoder during the training stage, we mask all the commands instead of the first command (whose type is "Move") to finish the reconstruction task, which means $L_{token} = 1$ and $L_y = L_x - 1$.

In the remaining part of this section, we discuss the qualitative and quantitative results of our models on two representative datasets, Google Fonts[1] and Icons [8]. Note that the evaluation of representation learning models is difficult, and most quantitative metrics can not provide a generic measure of the novelty and veracity of the data sampled from representation learning models. Thus, we will explore 4 types of downstream tasks including reconstruction task, classification task, retrieval task, and interpolation task with specific quantitative metrics used by the various underlying methods.

4.2. Results

We evaluate the effectiveness of the proposed SVGformer on both fonts dataset and the icons dataset. We first show the reconstruction results, which is the training task. Note that simply using MSE and CE metrics cannot accurately show the coordinate values of control points on the contour line because the average of the correct target values is often not a correct value, so we use Chamfer distance (CD) metric to do the evaluation. Then, for the downstream tasks, we train a classification model to finish the classification task and use the glyph label to verify the retrieval task, which is a pure unsupervised process without any fine-tuning on the representation. After that, we finish with an interpolation task to show that SVGformer has good interpretability.

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

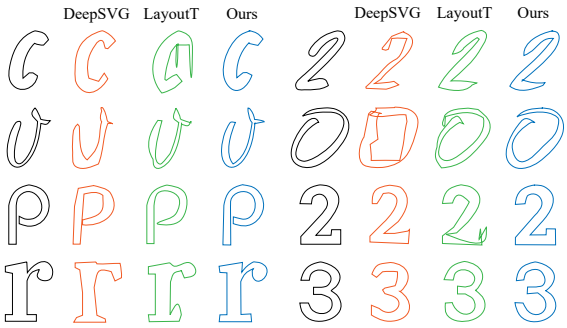


Figure 3. Reconstruction results on various glyphs. From left to right, the original SVG (black), DeepSVG (orange), LayoutTransformer (green) and our SVGformer (blue), respectively.

4.2.1 Fonts

Reconstruction task. The reconstruction task is the training task where we can get the pre-trained model and obtain the latent representation of a given SVG image. We use Chamfer distance (CD) between corresponding paths of the input SVG and reconstructed SVG to evaluate the performance of the SVGformer.

Based on Table 2, our model produces a smaller gap than the other baselines with 17.8% overperformance in a mathematical sense, proving the accuracy of SVGformer. In addition, we render the reconstructed results for quantitative comparisons on Figure 3. Benefiting from the two kinds of attention scores, SVGformer can reconstruct more details of the raw input. As confirmed by the results, we find that introducing the geometric information brings a substantial improvement in reconstruction error compared to the previous baselines.

Classification task. For the classification task, we train a linear classification model for the different hidden representations with 1000 epochs and show in Table 2 the accuracy results of glyph-based category (62 classes) and family-based category (378 font family types filtered by a minimum class size of 100 in the test set). The glyph-based classification task is used to verify that our model's latent representation can identify the outline of inputs. The family-based classification task is more challenging as it has more categories and focuses on font style which needs the latent representation to be distinctive in details. We can figure out that the geometric information can help SVGformer gain further improvement for this mature task. Specifically, we find that both the LayoutTransformer and DeepSVG fail to classify the family labels with the accuracy of 7.0% and 35.2%, respectively, which means they lose the style information during the data-driven process. Especially we find that higher dimensional data does not encompass more information because LayoutTransformer's hidden representation has a larger shape

Model	Reconstruction (CD ↓)		Classification (Acc %)			Retrieval (Acc %)		
	Fonts	Icons	Fonts Glyph	Fonts Family	Icons	Fonts@5	Fonts@10	Icons@10
LayoutTransformer	37.6	18.8	75.6	7.0	57.6	1.7	1.6	13.4
DeepSVG	32.7	15.6	85.6	35.2	45.1	51.2	45.9	22.8
SVGformer	**26.9**	**6.3**	**89.5**	**69.7**	**64.7**	**74.3**	**64.2**	**48.9**
SVGformer-no-geo	30.8	7.9	89.3	46.3	46.7	69.6	59.4	46.2
SVGformer-discrete	41.5	19.3	85.9	15.0	63.7	60.7	51.5	30.5

Table 2. Result comparison for different downstream tasks on Fonts and Icons dataset.

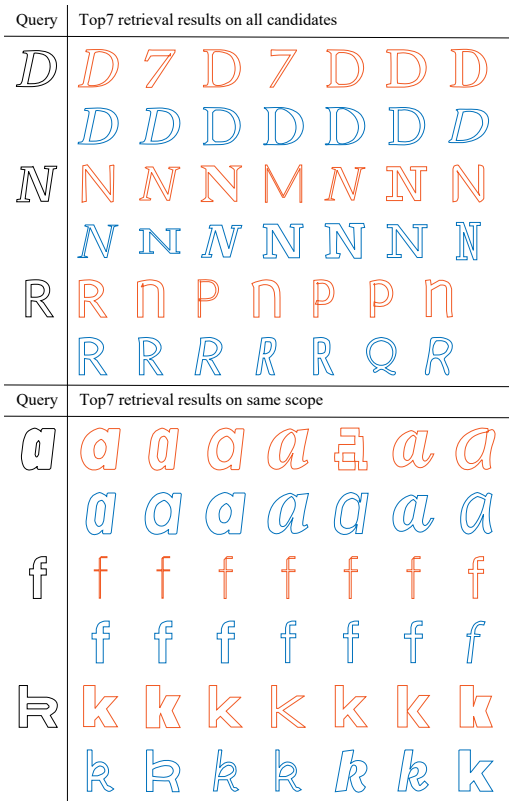


Figure 4. Retrieval results from all glyphs and same glyphs, respectively. Results for DeepSVG are shown in orange, and results for SVGformer in blue. SVGformer always retrieves the same glyph as the query when searching from all glyphs, and returns results in more similar styles when searching for a specific glyph.

than both DeepSVG and SVGformer, yet it performs almost randomly on the family classification task.

Retrieval task. From the perspective of quality analysis, we report the family accuracy of the most top 5% (@5) and 10% (@10) similar retrieval results by name on Tabel 2, where the similarity score can be calculated by the inner product of the query and all the candidates of an SVG image, which is equivalent to the cosine similarity formula after normalization. The retrieval results show that our model can offer a 42.5% performance improvement compared with DeepSVG. Besides, we use Figure 4 to show the retrieval results of some query samples which indicates SVGformer can achieve higher accuracy in general than previous baselines. Specifically, SVGformer can beat the previous state-of-the-art in two different settings: 1. retrieving in all the candidates (across families) to figure out the ability to find the same family; 2. retrieving under the target family to verify that our model's representation can align the style of the query.

Interpolation task. To verify that our model can animate SVGs by interpolating automatically from different glyphs, we use the hidden representation from DeepSVG and SVGformer to perform the comparison on the interpolation task. As shown in Figure 5, SVGformer can give more meaningful and smooth results with fewer artifacts and contortions when handling both translations and deformations in 3 different interpolation types including "Thin" → "Black", "Expanded" → "Condensed" and "Regular" → "Italic". For example, when we interpolate the fonts' size from "Thin" to "Black", we can find a specific value for "Regular" size in SVGformer's results but cannot find that one in DeepSVG's.

4.2.2 Icons

For the reconstruction task on the Icons dataset, we calculate the CD metric as mentioned before in Table 2, where SVGformer can build a new state-of-the-art by achieving 59.5% performance improvement over previous methods. In addition, we also plot the reconstruction results in Figure 6 to show that SVGformer can yield more precise reconstruction results than DeepSVG. For the classification task, we first collect icons' labels by their categories including "cinema", "city", "clothing", etc. Then we train a classification model with 59 classes and report the top 10 accuracy, where SVGformer achieves the highest accuracy. For the retrieval task, we use visual inspection to verify the retrieval results are relevant as there is no such ground truth information. As shown in Figure 7, the unsupervised representation of SVGformer has great capability to distinguish the global and local structure of SVG, achieving better retrieval results than other baselines.

4.3. Ablation Study

To better understand the effectiveness of different components in SGVformer, we evaluate two model variants, namely *SVGformer-no-geo* and *SVGformer-discrete*, by dropping the

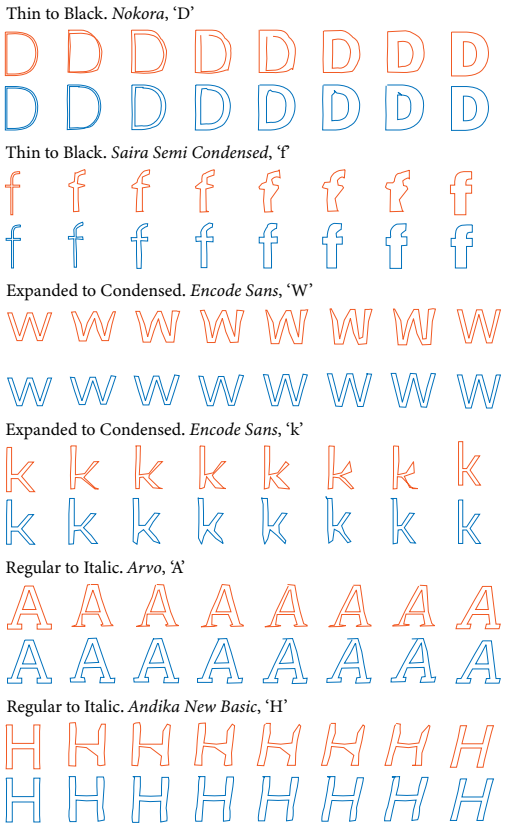


Figure 5. Examples of interpolations between glyphs. The results for DeepSVG are shown in orange, and those for SVGformer in blue. The first and last columns are the given glyphs to be interpolated, and the middle columns are the interpolations.

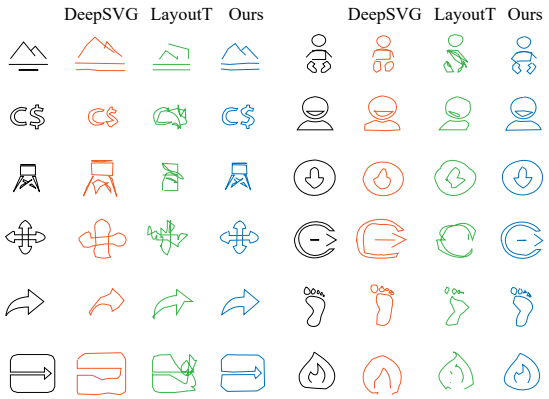


Figure 6. Example reconstruction results on Icons data. From left to right, columns show original SVG (black), DeepSVG (orange), LayoutTransformer (green), and our SVGformer (blue).

geometric information and continuous input embedding from the full model. The results on both Fonts and Icons datasets shown in the last two rows of Table 2 confirm that both components are indispensable. Specifically, *SVGformer-no-geo* aborts the geometric information inside the self-attention mechanism, which indicates the importance of structural

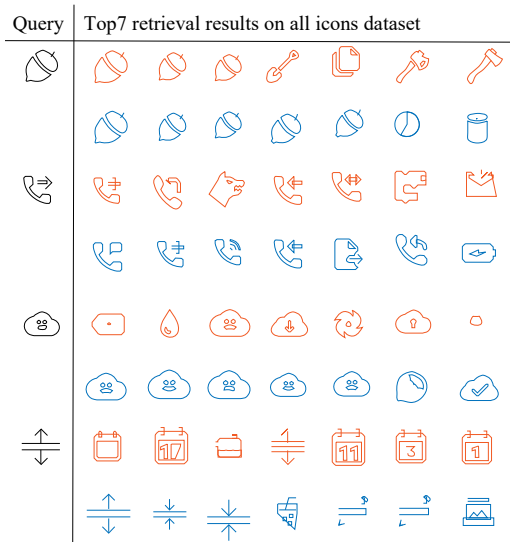


Figure 7. Retrieval results on Icons data. The results of DeepSVG are shown in orange, and the results of SVGformer in blue.

inductive bias for encoder-decoder-based transformer models. *SVGformer-discrete* provides a fair comparison of SVGformer with the previous works which only accept discrete SVG inputs. The results demonstrate that the raw input contains detailed information which is critical to high-quality hidden representation.

5. Limitation

We note some limitations of SVGformer in this section. One of the major limitations is the effectiveness of representation is based on sufficient SVGs in the training process, which is hard to guarantee in many real-world scenarios. Admittedly, although our model can yield much more precise hidden representations than previous baselines to make it identifiable for each SVG, we found that SVGformer's hidden space lacks linearity and smoothness when facing complex downstream tasks such as some interpolation samples of different style transfers. Inspired by diffusion models [13,24,27,28], one possible insight for future work can be smooth the latent space by introducing the diffusion process.

6. Conclusion

In this paper, we study representation learning for SVG and propose a transformer-based model named SVGformer. Specifically, we directly deal with the continuous input and extract MAT segment information as the inductive bias of SVG input. Besides, we design a geometric self-attention module to utilize the stylistic semantics of SVG commands and the geometric information jointly to handle long-term sequence relations. The experiments on two datasets and several downstream tasks demonstrate the effectiveness of SVGformer for robust representation learning of SVG.

References

[1] Valerio Biscione and Jeffrey S Bowers. Convolutional neural networks are not invariant to translation, but they can learn to be. *arXiv preprint arXiv:2110.05861*, 2021. 4

[2] Harry Blum. A Transformation for Extracting New Descriptors of Shape. In *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, 1967. 1, 3

[3] Neill D. F. Campbell and Jan Kautz. Learning a manifold of fonts. *ACM Trans. Graph.*, 33(4), jul 2014. 2

[4] Defu Cao, James Enouen, Yujing Wang, Xiangchen Song, Chuizheng Meng, Hao Niu, and Yan Liu. Estimating treatment effects from irregular time series observations with hidden confounders. *arXiv preprint arXiv:2303.02320*, 2023. 2

[5] Defu Cao, Jiachen Li, Hengbo Ma, and Masayoshi Tomizuka. Spectral temporal graph neural network for trajectory prediction. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1839–1845. IEEE, 2021. 2

[6] Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Congrui Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, et al. Spectral temporal graph neural network for multivariate time-series forecasting. *Advances in neural information processing systems*, 33:17766–17778, 2020. 2

[7] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020. 2

[8] Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. Deepsvg: A hierarchical generative network for vector graphics animation. *Advances in Neural Information Processing Systems*, 33:16351–16361, 2020. 1, 2, 6

[9] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *ICLR 2016*, 2016. 5

[10] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. In *International Conference on Learning Representations*, 2020. 2

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2

[12] Kamal Gupta, Justin Lazarow, Alessandro Achille, Larry S Davis, Vijay Mahadevan, and Abhinav Shrivastava. Layouttransformer: Layout generation and completion with self-attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1004–1014, 2021. 1, 2

[13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 8

[14] Ildoo Kim, Woonhyuk Baek, and Sungwoong Kim. Spatially attentive output layer for image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1

[15] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17, 2017. 5

[16] Jack Lanchantin, Tianlu Wang, Vicente Ordonez, and Yanjun Qi. General multi-label image classification with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16478–16488, June 2021. 1

[17] Neil Lawrence and Aapo Hyvärinen. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *Journal of machine learning research*, 6(11), 2005. 2

[18] Xiaolong Liu, Zhidong Deng, and Yuhan Yang. Recent progress in semantic image segmentation. *Artificial Intelligence Review*, 52(2):1089–1106, 2019. 1

[19] Xuanqing Liu, Hsiang-Fu Yu, Inderjit Dhillon, and Cho-Jui Hsieh. Learning to encode position for transformer with continuous dynamical model. In *International conference on machine learning*, pages 6327–6335. PMLR, 2020. 4

[20] Raphael Gontijo Lopes, David Ha, Douglas Eck, and Jonathon Shlens. A learned representation for scalable vector graphics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7930–7939, 2019. 1, 2

[21] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International conference on machine learning*, pages 4055–4064. PMLR, 2018. 2

[22] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 1

[23] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. *Advances in Neural Information Processing Systems*, 32, 2019. 2

[24] 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. 8

[25] Pradyumna Reddy, Michael Gharbi, Michal Lukac, and Niloy J Mitra. Im2vec: Synthesizing vector graphics without vector supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7342–7351, 2021. 1, 2

[26] Leo Sampaio Ferraz Ribeiro, Tu Bui, John Collomosse, and Moacir Ponti. Sketchformer: Transformer-based representation for sketched structure. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14153–14162, 2020. 1

[27] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep

language understanding. *arXiv preprint arXiv:2205.11487*, 2022. 8

[28] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015. 8

[29] Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. Transformer dissection: An unified understanding for transformer's attention via the lens of kernel. In *ACL 2019*, pages 4335–4344, 2019. 5

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 1, 4

[31] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In *European Conference on Computer Vision*, pages 108–126. Springer, 2020. 2

[32] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018. 2

[33] Yizhi Wang and Zhouhui Lian. Deepvecfont: synthesizing high-quality vector fonts via dual-modality learning. *ACM Transactions on Graphics (TOG)*, 40(6):1–15, 2021. 1, 2

[34] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45, 2020. 1, 2

[35] Kan Wu, Houwen Peng, Minghao Chen, Jianlong Fu, and Hongyang Chao. Rethinking and improving relative position encoding for vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10033–10041, 2021. 2

[36] Neo Wu, Bradley Green, Xue Ben, and Shawn O'Banion. Deep transformer models for time series forecasting: The influenza prevalence case. *arXiv preprint arXiv:2001.08317*, 2020. 2

[37] Shijie Yang, Liang Li, Shuhui Wang, Weigang Zhang, Qingming Huang, and Qi Tian. Skeletonnet: A hybrid network with a skeleton-embedding process for multi-view image representation learning. *IEEE Transactions on Multimedia*, 21(11):2916–2929, 2019. 1

[38] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks. In *CVPR*, pages 472–480, 2017. 5

[39] Yizhou Zhang, Defu Cao, and Yan Liu. Counterfactual neural temporal point process for estimating causal influence of misinformation on social media. In *Advances in Neural Information Processing Systems*, 2022. 2

[40] Yajing Zheng, Lingxiao Zheng, Zhaofei Yu, Boxin Shi, Yonghong Tian, and Tiejun Huang. High-speed image reconstruction through short-term plasticity for spiking cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6358–6367, June 2021. 1

[41] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11106–11115, 2021. 1, 2, 3, 4, 5