

---

# Fixing Interpolation in Existing Latent Spaces instead of Constantly Producing New Ones

---

Moritz Dück Jonas Hübötter Nora Schneider

## Abstract

We study latent space interpolation in a latent space of SVG images based on the DeepSVG model of (Carlier et al., 2020). We survey existing approaches to augment the AE loss with discriminator-based regularization, and we introduce new approaches to interpolation within existing latent spaces by framing interpolation as a regularized optimization problem. We find that our approaches use novel interpolation paths — some with improved (subjective) visual fidelity.

## 1. Introduction

Scalable vector graphics (SVGs) are widely used to represent fonts, emojis, icons, and other digital illustrations because of their arbitrary scalability and efficient representation. As SVGs are represented by XML text files, they combine challenges present in natural language and rasterized images. A common problem is the interpolation between two SVGs, e.g., to animate the transition between two emojis or two icons. Today, this is still done “by hand”, but recent work suggests that similar results can be achieved by interpolating (linearly) between points in a latent space of SVGs. However, often the intermediate vector graphics have poor visual fidelity. In this report, we develop and evaluate new techniques for improving visual fidelity of interpolated SVGs in a pretrained latent space.

Our experiments are based on DeepSVG, a hierarchical transformer-based autoencoder model, proposed by Carlier et al. (2020). We use the same data representation and dataset, which consists of roughly 26k icons from 56 different categories (sourced from <https://icons8.com>). It is worth noting that the visual fidelity of reconstructed SVG icons after embedding them in the lower dimensional space is already noticeably worse than of the inputs, posing an upper bound for the visual fidelity of any interpolated icon. Figure 4 shows examples of such reconstructions. Nevertheless, the textual representation of SVG icons poses an interesting challenge compared to pixel-based data, for which latent spaces have been analyzed in great depth in previous works.

Our approaches can be divided into two categories: Firstly,

we are evaluating methods for improving latent space interpolation without changing the latent space. We introduce a novel framing of interpolation as regularized optimization, which is presented in Section 2.1. Secondly, we fine-tune the DeepSVG latent space using a discriminator-based regularization, which we describe in Section 2.2.

## 2. Models and Methods

### 2.1. Latent Space Interpolation

We begin by discussing approaches for interpolating between two points  $z_0$  and  $z_T$  in the latent space. We characterize this problem as finding a sequence of points  $z_1, z_2, \dots, z_{T-1}$  on the interpolation path.

#### 2.1.1. PRIOR APPROACHES

A naïve and arguably the most common approach is to *linearly* interpolate points, that is,

$$z_t = \alpha \cdot z_0 + (1 - \alpha) \cdot z_T \quad (1)$$

where  $\alpha = t/T$ . An alternative approach is to interpolate along the great circle path on a hypersphere,

$$z_t = \frac{\sin((1 - \alpha) \cdot \theta)}{\sin \theta} z_0 + \frac{\sin(\alpha \cdot \theta)}{\sin \theta} z_T \quad (2)$$

where  $\cos \theta = z_0 \cdot z_T$ . This method is known as *spherical linear interpolation* (SLERP) (Shoemake, 1985; White, 2016). SLERP reduces to linear interpolation as  $\theta \rightarrow 0$ . A more recently suggested method is to interpolate based on distribution-matching optimal transport maps (Agustsson et al., 2017), aiming to resolve the distribution mismatch between the resulting outputs of linear interpolation and the latent space prior.<sup>1</sup>

#### 2.1.2. REINTERPRETING LINEAR INTERPOLATION

In this work, we consider a generalization of linear interpolation which regularizes the interpolation path such that it is closer to “more realistic” parts of the latent space. To this

---

<sup>1</sup>Due to time constraints, and because of the generally low visual fidelity of points in the latent space (even of reconstructions), we do not evaluate this method for DeepSVG.

end, we first characterize linear interpolation as an optimization problem.

A linear interpolation path can be interpreted as the solution to minimizing the loss

$$\ell_E(\mathbf{z}_{1:T-1}) \stackrel{\text{def}}{=} \sum_{t=0}^{T-1} \|\mathbf{z}_t - \mathbf{z}_{t+1}\|_2^2. \quad (3)$$

**Theorem 2.1.** Equation (1) is the closed-form solution to minimizing  $\ell_E$ .

A proof is included in Appendix C. We also consider minimizing the loss

$$\begin{aligned} \ell_C(\mathbf{z}_{1:T-1}) &\stackrel{\text{def}}{=} \sum_{t=0}^{T-1} \angle(\mathbf{z}_t, \mathbf{z}_{t+1})^2 - \sum_{t=0}^{T-2} \angle(\mathbf{z}_t - \mathbf{z}_{t+1}, \mathbf{z}_{t+1} - \mathbf{z}_{t+2}) \end{aligned} \quad (4)$$

where  $\angle(\mathbf{x}, \mathbf{y}) \stackrel{\text{def}}{=} \mathbf{x} \cdot \mathbf{y} / \|\mathbf{x}\|_2 \|\mathbf{y}\|_2$  denotes the cosine similarity of  $\mathbf{x}$  and  $\mathbf{y}$ . Intuitively, the first term encourages points to be well-separated, and the second term encourages points to lie on a linear trajectory. More details and an illustration can be found in Appendix D.

### 2.1.3. REGULARIZATION

To find regularized interpolation paths, we consider a density over the latent space, encoding which points yield “realistic” SVGs. We discuss in Section 2.1.4, how to find such a density.

We consider densities which are a mixture of Gaussians  $p$  which is given by a distribution  $\mu$  over the finite set of Gaussians  $\{\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\}_i$ :

$$p(\mathbf{z}) \stackrel{\text{def}}{=} \mathbb{E}_{i \sim \mu} [\mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)]. \quad (5)$$

An illustration of a “latent space density” which is a Gaussian mixture can be found in Figure 5. Our objective is to minimize the surprise of samples  $\mathbf{z}_t$  with respect to  $p$ . We use the classical information-theoretic interpretation of the surprise of an event with probability  $u$  which is described by  $S[u] \stackrel{\text{def}}{=} -\log u$ . From Jensen’s inequality and the convexity of  $S$  we obtain

$$S[p(\mathbf{z})] \leq \mathbb{E}_\mu [S[\mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)]] \quad (6)$$

$$\propto \mathbb{E}_\mu [(\boldsymbol{\mu}_i - \mathbf{z})^\top \boldsymbol{\Sigma}_i^{-1} (\boldsymbol{\mu}_i - \mathbf{z})] \stackrel{\text{def}}{=} s(\mathbf{z}). \quad (7)$$

We use  $\ell_{\text{reg}}(\mathbf{z}_{1:T-1}) \stackrel{\text{def}}{=} \sum_{t=1}^{T-1} s(\mathbf{z}_t)$  for regularization.

When the set of Gaussians is large, it is useful to consider a stochastic variant of  $\ell_{\text{reg}}$ . Observe that we can obtain unbiased (gradient-)estimates of  $s(\mathbf{z})$  simply by straightforward Monte Carlo estimation.

The regularized objectives become

$$\min_{\mathbf{z}_{1:T-1}} \ell_E(\mathbf{z}_{1:T-1}) + \lambda \ell_{\text{reg}}(\mathbf{z}_{1:T-1}) \quad \text{and} \quad (8)$$

$$\min_{\mathbf{z}_{1:T-1}} \ell_C(\mathbf{z}_{1:T-1}) + \lambda \ell_{\text{reg}}(\mathbf{z}_{1:T-1}) \quad (9)$$

As  $\lambda \rightarrow 0$ , the solutions reduce to linear interpolation.

### 2.1.4. FINDING A LATENT SPACE DENSITY

**Data-based Density** A naïve approach is to consider  $n$  isotropic Gaussians with unit-variance,<sup>2</sup> each based at one training example, and to choose  $\mu$  as the uniform distribution. In this case,

$$s(\mathbf{z}) = \frac{1}{n} \sum_{i=1}^n \|\tilde{\boldsymbol{\mu}}_i - \mathbf{z}\|_2^2 \quad (10)$$

where  $\tilde{\boldsymbol{\mu}}_i$  is the latent representation of example  $i$ . To use this density for regularization, one inevitably has to resort to stochastic optimization as  $n$  is generally large.

**Cluster-based Density** To improve efficiency, we use the scikit-learn implementation of the  $k$ -means algorithm (MacQueen, 1967) to determine  $k$  clusters (where  $k \ll n$ ) with means  $\boldsymbol{\mu}_i$ . The  $k$ -means algorithm determines  $\boldsymbol{\mu}_i$  such that  $\sum_{i=1}^n \min_{\boldsymbol{\mu}_j} \|\boldsymbol{\mu}_j - \tilde{\boldsymbol{\mu}}_i\|_2^2$  is minimized. This reduces the surprise term to

$$s(\mathbf{z}) = \frac{1}{k} \sum_{i=1}^k \|\boldsymbol{\mu}_i - \mathbf{z}\|_2^2. \quad (11)$$

**General Gaussian Mixtures** While the above clustering-based approach makes the regularization more efficient, it does not take into account the relative importance and the (potentially non-spherical) shape of the clusters. To address this, we consider a general distribution  $\mu$  and arbitrary covariance matrices  $\boldsymbol{\Sigma}_i$  in addition to the means  $\boldsymbol{\mu}_i$ , such that the marginal likelihood  $\prod_{i=1}^n p(\tilde{\boldsymbol{\mu}}_i)$  is maximized approximately using the expectation-maximization (EM) algorithm (Dempster et al., 1977) as implemented by scikit-learn. In this case, the surprise term of Equation (7) is

$$s(\mathbf{z}) = \sum_{i=1}^k \mu(i) \cdot (\boldsymbol{\mu}_i - \mathbf{z})^\top \boldsymbol{\Sigma}_i^{-1} (\boldsymbol{\mu}_i - \mathbf{z}) \quad (12)$$

where  $k$  denotes the number of components. We provide additional details regarding our choice of  $k$  and the resulting model in Appendix B.

## 2.2. Regularized DeepSVG

An alternative approach to improve the fidelity of interpolated SVGs is to further adjust the latent space of

<sup>2</sup>The variance is equivalently scaled by the parameter  $\lambda$ .

DeepSVG such that linear interpolation is encouraged to appear more realistic. In Adversarial Constrained Autoencoder Interpolation (ACAI) (Berthelot et al., 2018) this is achieved with a regularized autoencoder. The model tries to fool a discriminator which predicts the mixing coefficient  $\alpha$  from a decoded (linearly) interpolated example  $z = \alpha \cdot z_1 + (1 - \alpha) \cdot z_2$ . This approach has shown promising results on image data (Berthelot et al., 2018). Inspired by this, we adapt the architecture of DeepSVG to improve interpolations.

Since the goal is to improve interpolations within a given latent space, we do not retrain DeepSVG with the discriminator-based regularization from scratch, as proposed by Berthelot et al. (2018). Instead, we fine-tune DeepSVG and a pretrained discriminator. Additionally, we fine-tune DeepSVG while freezing the discriminator.

### 2.2.1. DISCRIMINATOR ARCHITECTURE

Since SVG icons are expressed as XML text files, the input data is of sequential nature. This motivates the use of a recurrent neural network architecture. An established approach is the use of BiLSTM models for such sequential and variable length input formats. The input dimension is 14, as we are using the SVG representation introduced by DeepSVG. Each of the two LSTM cells has a hidden state of size 256, the concatenated output of size 512 is then fed into a fully connected neural network using ReLU activation functions with one hidden layer of dimension 32 and a single output neuron representing the predicted  $\alpha$  value. The  $\ell_2$  norm of the difference of the true and the predicted alpha is used as the loss function. For more details on the architecture, see Figure 9. As seen in Figure 1, we derive different datasets from the icons provided by Carlier et al. (2020). Interpolations in  $D_i$  are obtained by linear interpolation. We train a model  $F_1$  on  $D_r \cup D_i$  with the BiLSTM architecture and use it for the regularized training covered in the next section.

A natural choice would be to reuse the hierarchical transformer-based encoder from DeepSVG as the discriminator model, due to the similarity in capacity and training dynamics. However, this architecture delivered less promising results during training (see Appendix F).

### 2.2.2. REGULARIZED DEEPSVG ARCHITECTURE

In the regularized DeepSVG model, the autoencoder is encouraged to fool the discriminator into predicting the interpolation coefficient  $\hat{\alpha} = 0$ . The regularized loss between one SVG  $x$  and its reconstruction  $\hat{x}$  is defined as follows:

$$L(x, \hat{x}) \stackrel{\text{def}}{=} L^{\text{DeepSVG}}(x, \hat{x}) + \lambda \|F(\hat{x}_\alpha)\|^2. \quad (13)$$

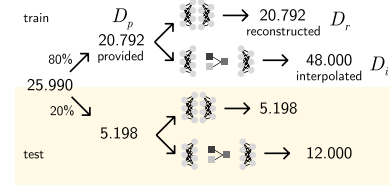


Figure 1. The three datasets  $D_p$ ,  $D_r$  and  $D_i$  created for training the discriminator for different analysis tasks. The initial 25 990 icons are the ones used by (Carlier et al., 2020).

Here,  $L^{\text{DeepSVG}}(x, \hat{x})$  is a weighted cross-entropy loss between the commands, arguments, fillings and visibility of the shapes of the input and its reconstructions (details in Carlier et al. (2020)).  $F(\hat{x}_\alpha)$  is the prediction of the discriminator, where  $\hat{x}_\alpha = \text{dec}(\alpha \cdot \text{enc}(x) + (1 - \alpha) \cdot \text{enc}(y))$  is the reconstructed interpolation between  $x$  and an arbitrary other SVG icon  $y$ . This is also summarized in Figure 2. During training,  $y$  is chosen from the same minibatch as  $x$  and  $\alpha$  is randomly sampled. Finally, enc and dec are optimized with respect to  $L$  (Equation (13)) and the discriminator  $F$  is optimized with respect to a mean squared error (see Section 2.2.1) using Adam (Kingma & Ba, 2014).

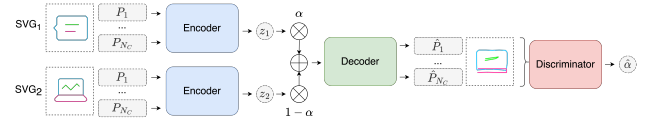


Figure 2. Regularized DeepSVG model.





























































































































## 3. Results and Discussion

Our results are illustrated in Figure 4, which shows an exemplary interpolation of the linear baseline and our proposed methods. Further examples are in Appendix H.

### 3.1. Latent Space Interpolation

We train a discriminator  $F_2$  on  $D_p \cup D_i$ , using the provided icons and interpolations thereof. We intended to use this model to evaluate the quality of different reconstructions. However, as seen in Figure 4, the model is only able to tell apart reconstructions from provided icons, but does not deliver any insights regarding the visual fidelity of the icons. A detailed discussion follows in Section 3.2. Looking at the visual fidelity of differently interpolated icons (Figure 4), we can observe that there is variation in the icons generated, however, it is difficult to judge the quality objectively.

We observe that the number of shapes on a regularized interpolation path regresses towards the mean of 4 shapes. When interpolating between icons with fewer shapes, one can ignore the additional shapes that appear during interpolation.

	Linear Interpolation	Spherical	GMM euclidean (ours)	GMM cosine (ours)	K-Means euclidean (ours)	K-Means cosine (ours)	Regularized I (ours)	Regularized II (ours)	Regularized III (ours)	
										provided
										reconstruction
										
										
										
										
										
										
										
										
										
										
										

<sup>4</sup>Regularized I: regularized DeepSVG right after drop in regularization loss. Regularized II: converged regularized DeepSVG. Regularized III: converged regularized DeepSVG with frozen discriminator.

## References

- Agustsson, E., Sage, A., Timofte, R., and Van Gool, L. Optimal transport maps for distribution preserving operations on latent spaces of generative models. *arXiv preprint arXiv:1711.01970*, 2017.
- Berthelot, D., Raffel, C., Roy, A., and Goodfellow, I. Understanding and improving interpolation in autoencoders via an adversarial regularizer. *arXiv preprint arXiv:1807.07543*, 2018.
- Carlier, A., Danelljan, M., Alahi, A., and Timofte, R. Deepsvg: A hierarchical generative network for vector graphics animation. *Advances in Neural Information Processing Systems*, 33:16351–16361, 2020.
- Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL <https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf>.
- Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- MacQueen, J. Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*, pp. 281–297, 1967.
- Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., and Chen, X. Improved techniques for training gans. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL <https://proceedings.neurips.cc/paper/2016/file/8a3363abe792db2d8761d6403605aeb7-Paper.pdf>.
- Shoemake, K. Animating rotation with quaternion curves. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pp. 245–254, 1985.
- Warde-Farley, D. and Goodfellow, I. *Adversarial Perturbations of Deep Neural Networks*, pp. 311–342. 2017.
- White, T. Sampling generative networks. *arXiv preprint arXiv:1609.04468*, 2016.



## A. Illustration of Latent Space Density

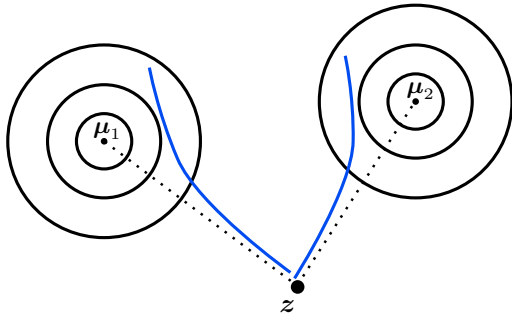


Figure 5. A figure illustrating the use of a latent space density which consists of a mixture of Gaussians centered at  $\mu_1$  and  $\mu_2$ . The level sets are shown as black circles. When considering a point  $z$  in the latent space, its surprise (inverse shown in blue) is evaluated according to each of the mixture components and then averaged. The result corresponds to a measure of how “likely”  $z$  is according to the mixture density.

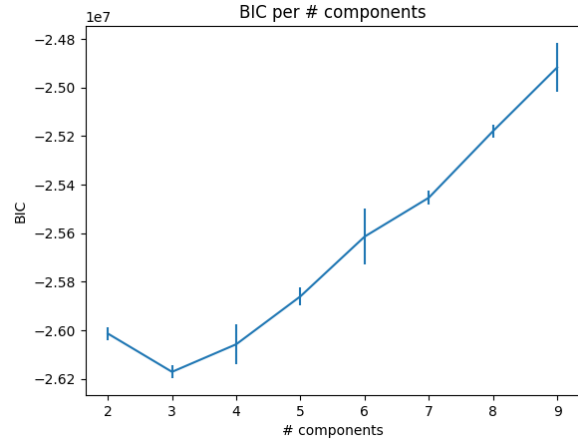


Figure 6. BIC for Gaussian mixture. Error bars correspond to three times the standard deviation of 3 independent model fits.

## B. Gaussian Mixture Model

In our experiments we choose the number of components  $k = 4$  based on the Bayesian information criterion (BIC) and the goodness of interpolations.

More concretely, as shown in Figure 6 the BIC is minimized at between 2 and 4 components. We found that of these Gaussian mixtures, the best performance was obtained with 4 components.

Figure 7 contains a visualization of the resulting model.

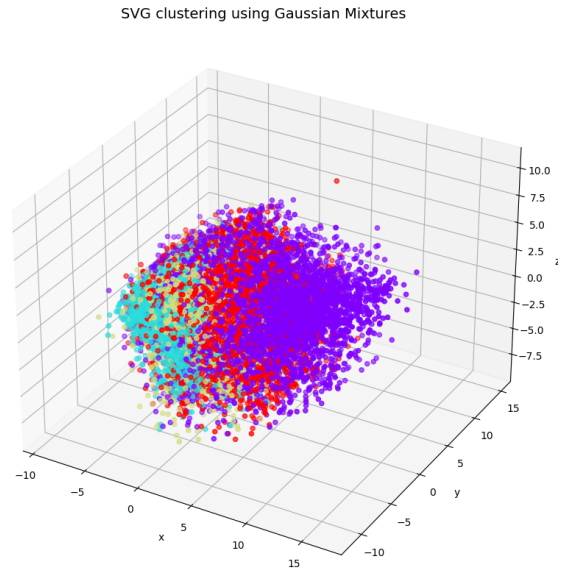


Figure 7. Clustering according to GMM with 4 components when points are projected to 3 dimensions with PCA. The explained variance is  $\approx 25\%$ .

### C. Omitted Proofs

*Proof of Theorem 2.1.* Observe that points that are a solution to the corresponding optimization problem must lie on the linear path between  $z_0$  and  $z_T$  as this path has the minimum (Euclidean) length. Moreover, observe that if points are not evenly spaced, the value of  $\ell_E$  can be reduced by evenly spacing points as larger distances contribute proportionally more to the squared term than smaller distances.  $\square$

### D. Cosine Similarity-based Loss

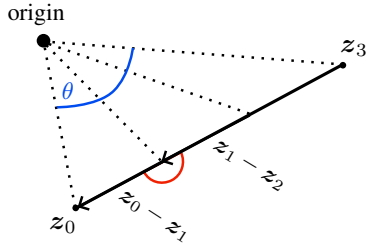


Figure 8. Illustration of linear interpolation in terms of cosine similarity. The first term of the loss  $\ell_C$  encourages that points are well-separated (shown in blue). The second term of the loss encourages that points are on the linear path between  $z_0$  and  $z_3$  (shown in red).

Recall the definition of the loss  $\ell_C$  from Equation (4).

We remark that the first term of  $\ell_C$ ,  $\angle(z_t, z_{t+1})^2$ , encourages that all consecutive points have identical cosine similarity, therefore encouraging that points are well-separated as depicted in Figure 8 (shown in blue). The degree of well-separation deteriorates as  $\angle(z_0, z_T) \rightarrow \pm 1$ .

The second term encourages that  $\angle(z_t - z_{t+1}, z_{t+1} - z_{t+2}) = 1$ , and hence, points lie on the linear interpolation path. The effect is illustrated in Figure 8 in red.

### E. BiLSTM Discriminator

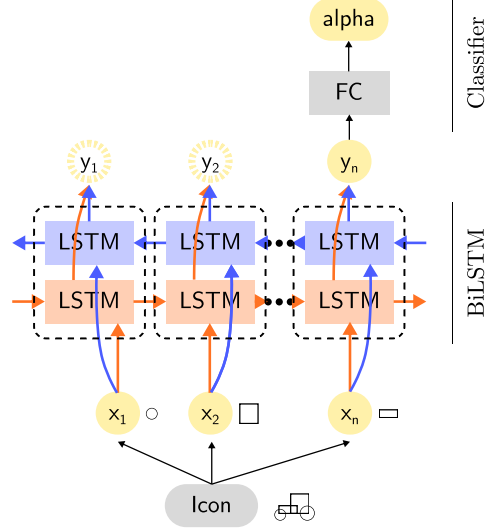


Figure 9. The architecture of our discriminator model.

Figure 10 shows that the test loss is decreasing until epoch 60. We use the models at this epoch within our paper.

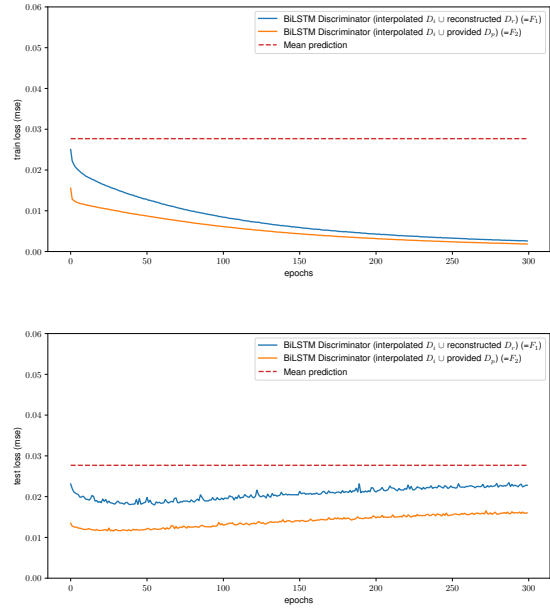


Figure 10. Training and test loss for BiLSTM discriminator.

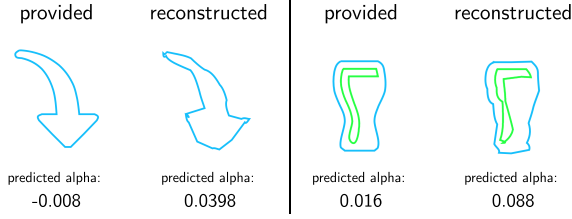


Figure 11. Sample icons and their reconstructions evaluated by  $F_2$ . One can see that human design principles like symmetry, equal spacing and smoothness are not preserved well under the model. This might be an inherent limitation of the representation of SVGs in their textual form.

## F. Transformer-based Discriminator

As the hierarchical transformer-based structure works well for the autoencoder in DeepSVG, we use a similar structure for predicting the mixing coefficient  $\alpha$  of interpolated outputs in the discriminator. We use the same datasets as described in Figure 1.

We train the resulting discriminator model from scratch. However, the discriminator seems not to learn anything as it constantly predicts the mean value. Therefore, we initialize the discriminator with the weights from the pretrained encoder in DeepSVG. While this seems more promising on the training dataset, we observe that the discriminator is overfitting to the training set after little training time and does not generalize well to the validation data. Simple regularization methods like weight decay, increase of dropout rate or reducing the complexity of the regression head do not solve the problem of overfitting. The loss functions are plotted in Figure 12. Therefore, we decide to use the BiLSTM architecture as a final discriminator model.

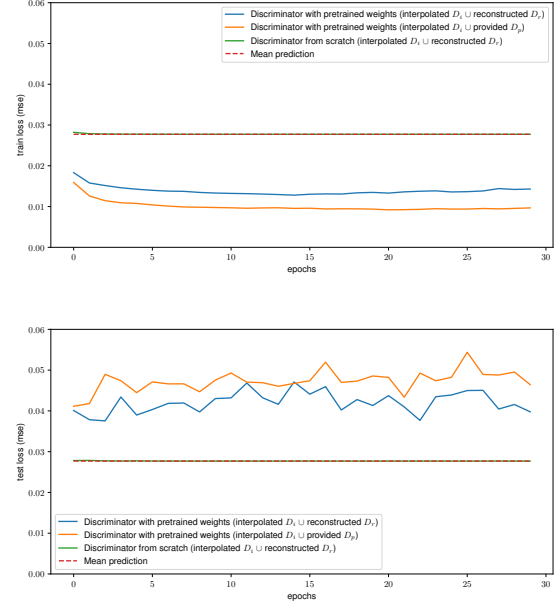


Figure 12. Training and test loss for different transformer-based discriminator configurations.

## G. Regularized DeepSVG

In Figures 13 and 17, the loss functions over time are shown for the regularized DeepSVG models. We see that DeepSVG successfully fools the discriminator after little training, as the regularization term becomes constantly close to zero. After this, DeepSVG only improves its reconstruction quality.

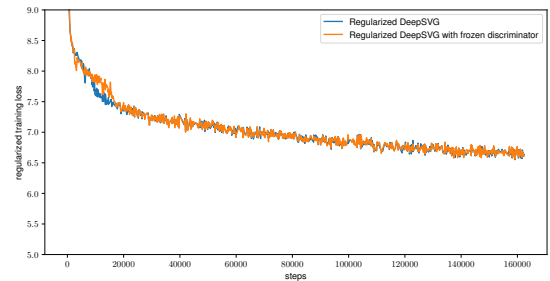


Figure 13. Training loss as described in Equation (13) during fine-tuning the regularized DeepSVG model.



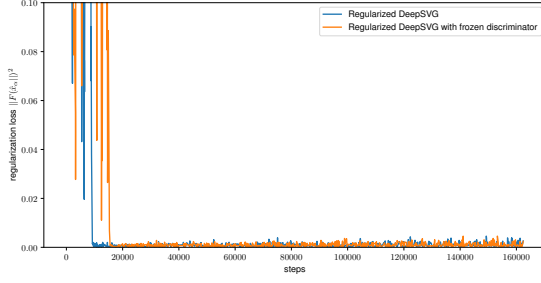


Figure 14. Regularization loss as described in Equation (13) during finetuning the regularized DeepSVG model.

## H. Exemplary Interpolations of Baselines and Developed Methods

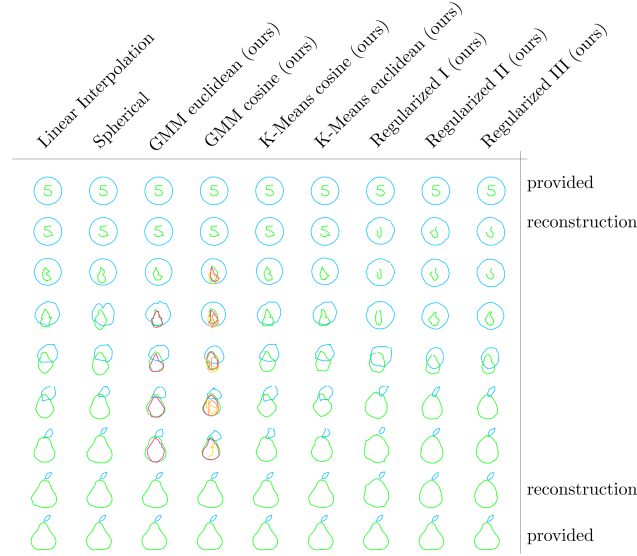


Figure 15. Example I: interpolating icons with a small number of shapes.

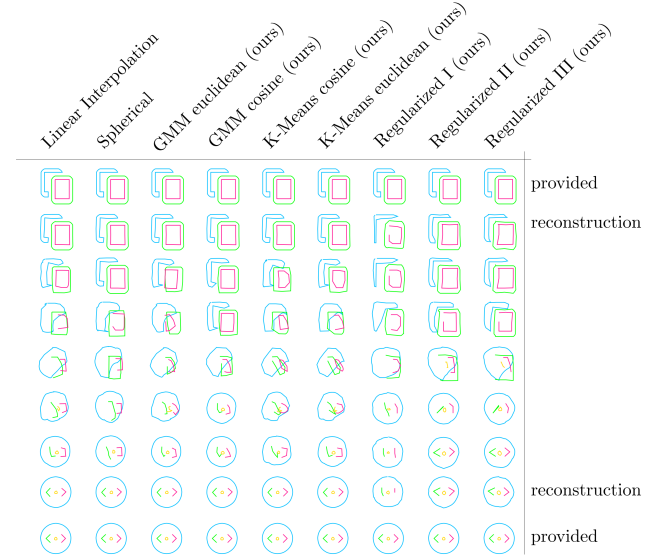


Figure 16. Example II: interpolation paths for slightly more complex icons.

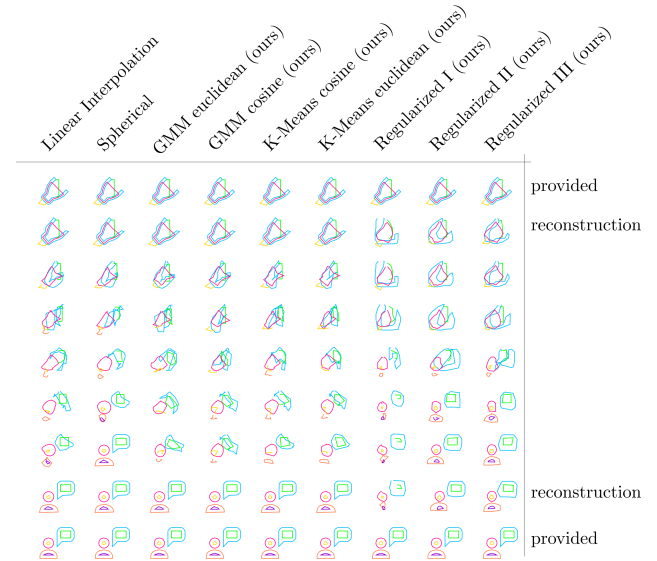


Figure 17. Example III: interpolation paths for slightly more complex icons.