You Text, We Sketch: Text-Guided Diffusion for Vectorized Sketch Generation

Yash Shah[*] **Samy Cherfaoui**[*]
{ynshah,scherfao}@stanford.edu
Department of Computer Science
Stanford University

Abstract

Vectorized sketch generation is an interesting domain of image generation dealing with vectorized sketches that not just closely resemble the way humans typically draw but also have the advantage of scaling up or down without loss in image quality, in contrast to raster (pixelated) images. With the recent popularization of text-conditioned raster image generation because of the power of human language to describe complex scenarios succinctly by Glide, DALL-E 2, Imagen, and others, we extend this idea to vectorized sketch generation in this paper. Precisely, we explore diffusion models for the problem of text-conditioned vectorized sketch generation, improving sample quality through classifier-free guidance. Across five different classes, our model achieves an average Fréchet Inception Distance (FID) score of 6.5, an average Geometry Score (GS) of 3.4, average Kynkäänniemi precision and recall of 0.57 and 0.68 respectively, and average image classifier acc@1 and acc@10 of 56.1 and 92.4 on the *Quick, Draw!* dataset, achieving competitive performance on these metrics.[1]

1 Introduction and Motivation

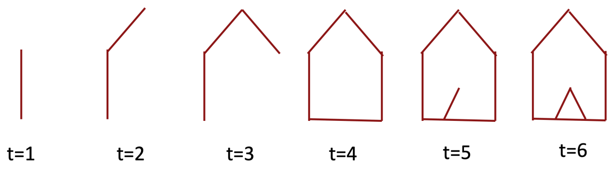
High quality image generation has become an exciting avenue for research with applications in generating human faces [1], cartoons [2, 3], medical images [4, 5], human poses [6], inpainting [7], editing [8], super-resolution [9], and much more. While these and many other works work mostly with raster (pixelated) images [10, 11], vectorized sketches that make use of strokes and point-slopes are a more natural way of thinking about how humans draw (generate) images or sketches.

They have the added advantage over raster images of being lightweight in file size, and being able to scale up or down without loss in image quality, which is possible because vector images work with strokes and point slopes rather than with individual pixels that determine image resolution.

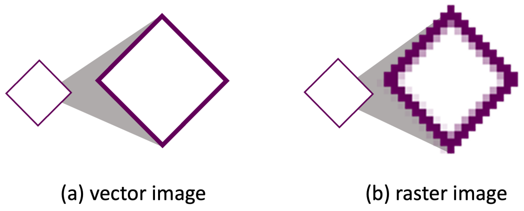
Architectures like SketchRNN [12], which uses a recurrent neural network for generating stroke-based drawings, SketchHealer [13], which uses a graph-to-sequence network for recreating partial human sketches, and SketchKnitter [14], which uses a diffusion model to generate vectorized sketches through DDIM sampling, are popular. Vectorized sketch generation is of particular interest because it enables a wide variety of use cases ranging from the ability to generate icons and logos on the fly for a UI, complex designs that fit on a business card and on a billboard, synthetic data for training, step-by-step tutorials on sketching, "healing" bad drawings or completing them, and potentially domain adapting and producing sketches for out-of-vocabulary words.

All works in the space of vectorized sketch generation have focused on either unconditional generation or conditioning on a part of the sketch for sketch completion or healing. Recent popularization of

[*]Equal contribution, order decided by coin flip.
[1]Github repository code: `https://www.github.com/ynshah3/SketchKnitter`



(a) Timelapse of how humans would typically draw a house, i.e., using strokes.



(b) Preservation of image quality with scaling up by vectorized images.

text-conditioned diffusion models like DALL-E 2 [15], Glide [16], and Imagen [17] have shown potential to generate text-conditioned images—a more natural and effective way of communicating what kind of image a model should generate through the power of human language, which is able to describe complex scenarios succinctly. In the human handwriting space, [18] and [19] are popular works. A key challenge is to come up with an idea that is able to apply text-conditioning to vector sketch generation under economic and resource constraints when text prompts are not available for training.

In this paper, we build on the SketchKnitter [14] paper and explore text-conditioned diffusion models for vectorized sketch generation. Concretely, we make the following key contributions:

1. We come up with a unique way to annotate sketches given limited money and time for crowdsourcing.
2. We condition on text during training by feeding text tokens into a Transformer [20] used as a text encoder, and introduce classifier-free guidance to improve sample quality.
3. Across five classes, our model achieves an average Fréchet Inception Distance (FID) score of 6.5, an average Geometry Score (GS) of 3.4, and average Kynkäänniemi precision and recall of 0.57 and 0.68 respectively on the *Quick, Draw!* dataset[2], achieving competitive performance on these metrics.
4. Our model achieves recognizability scores acc@1 and acc@10 of 56.1 and 92.4 on an AlexNet-based image classifier, outperforming the former state-of-the-art SketchKnitter.
5. We perform ablation studies to analyze learned text embeddings, what happens when isotropic Gaussian noise is added to them, how metric scores perform per class, and how metrics are affected by tuning sampling hyperparameters.

To the best of our knowledge, there has not been any previous paper on text-conditioned vectorized sketch generation.

2 Related Works

Our work is a follow up to and extends the following two works to combine the power of vector images and text-conditioning:

SketchKnitter [14]. This paper proposes generating sketch stroke sequences in vector format from noise by denoising diffusion implicit models (DDIMs) [21]. The generative model learns a distribution over stroke points' locations from a deformation-based denoising process which starts from noise. It also learns to predict a binary pen state for each of the stroke points representing whether the pen is "touching the paper" when "sketching" the strokes. The paper also explores conditional generation through classifier guidance for tasks such as sketch completion and healing by conditioning on certain parts of the sketch. Unconditional diffusion models rarely allow tailored and nuanced sketch generation; so, we extend SketchKnitter's idea to add text-conditioning to the model by leveraging the power of human language.

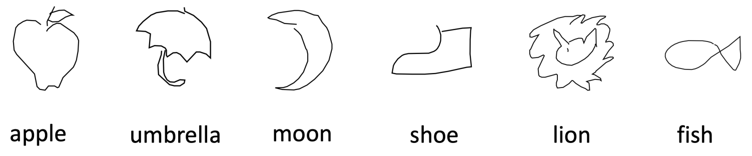
GLIDE [16]. Motivated by the ability of text-to-image models to handle free-form prompts, this paper applies guided diffusion to the problem of text-conditioned image synthesis. First, they train a 3.5 billion parameter diffusion model that uses a text encoder to condition on natural language descriptions. Then they compare two techniques for guiding diffusion models towards text prompts: CLIP guidance [22] and classifier-free guidance [23]. Because CLIP is primarily used for jointly learning an embedding for text and raster images, and given the effectiveness of classifier-free guidance, we employ Glide's strategies and apply them to SketchKnitter. Text conditioning requires lots of annotated images for training, but since we have none and are under resource and economic constraints, we come up with a unique and quick labelling strategy that is able to fulfill the requirement and can be generalized to when carefully crafted annotations are available.

3 Problem Statement

The problem that we will be investigating is how to generate vectorized sketches conditionally from a specified text prompt. More precisely, we are given a sketch $\mathbf{s}_0 = \{s^{(1)}, s^{(2)}, \ldots, s^{(N)}\}$, where $\forall i,\ s^{(i)} = (\Delta x^{(i)}, \Delta y^{(i)}, g^{(i)})$ are 3-D vectors. $(\Delta x^{(i)}, \Delta y^{(i)})$ represents the coordinate offsets during the pen's moving trajectory and $g^{(i)}$ represents the binary pen state, denoting whether the pen is touching the paper. Our goal is to learn a probability distribution over the offsets from the training data by a diffusion model conditioned on a text prompt c that identifies what the sketch $\mathbf{s}_0$ represents. Then, a sketch can be drawn given the estimated coordinate offsets for each point and the corresponding pen state inferred by a pen-state network.

3.1 Dataset

We use Google's *Quick, Draw!* dataset[2], which consists of 50 million vectorized drawings across 345 categories collected from a drawing game Google created. We select categories `apple`, `umbrella`, `moon`, `shoe`, `lion`, and `fish`, each with 70k samples. Our techniques can, however, be extended to many more categories given sufficient resources.



The *Quick, Draw!* dataset does not come with labelled annotations for every sketch, so, to condition on text prompts, we employ a unique way to get the job done and not have to manually annotate or crowdsource around 420k sketches. We randomly assign one of the following prompts to sketches from either class by constructing grammatically correct sentences:

- `This is {a,an,the} {apple,umbrella,moon,shoe,lion,fish}`
- `Sketch of {a,an,the} {apple,umbrella,moon,shoe,lion,fish}`
- `Image of {a,an,the} {apple,umbrella,moon,shoe,lion,fish}`
- `Here is {a,an,the} {apple,umbrella,moon,shoe,lion,fish}`

We construct a vocabulary for words from this mini universe after converting these prompts to lowercase, prepend and append the special tokens `<start>` and `<end>`, and convert to indices. The idea behind using a small number of prompts is that they can be quick to produce while also being representative of how working with a larger, more carefully crafted, set of text prompts given time and money could be.

For processing the sketches, we sanitize them by removing all sketch vectors with size more than 96, also removing large gaps between sketch points by clipping them to lie in the range $[-1000, 1000]$. We further normalize each sketch vector by dividing by the standard deviation of all sketch points. After these pre-processing steps, the number of samples left for training per class is visualized in figure 2.

[2]`https://github.com/googlecreativelab/quickdraw-dataset`

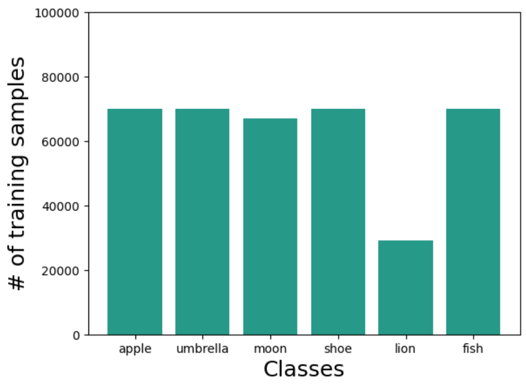


Figure 2: Number of training samples per class after pre-processing.

3.2 Evaluation Metrics

The quantitative metrics we use to evaluate our model are:

- Fréchet Inception Distance (FID) [24] which measures the similarity of generated sketches to real drawings based on the Fréchet distance between Gaussians fitted to image feature representations from the Inception network,
- Geometry Score (GS) [25] which compares the geometric properties of the generated and true image data manifolds, and
- Kynkäänniemi et al.'s image generation precision and recall metrics [26]. Kynkäänniemi precision measures the likelihood that a random image from our generated distribution P_g falls within the support of our real image distribution P_r. Recall measures the likelihood that a random image from P_r falls within the support for P_g.

We also train a multi-category classifier built on AlexNet on the training set of all 345 QuickDraw categories. We use higher recognition accuracy as a proxy to determine the recognizability of sketches. Following the practices used in SketchHealer [13] and SketchKnitter [14], we use acc@1 and acc@10 as evaluation metrics. acc@K denotes the accuracy of obtaining the true category from AlexNet within the top K predictions. We also evaluate our samples qualitatively by ensuring that they visually look like the text prompt we feed before sampling.

4 Technical Approach

We use the architecture from the SketchKnitter paper [14] as a baseline, which is as follows: During the forward process, noise is gradually added to the sample sketches by drifting each sketch's coordinate offsets along the x and y directions. Then, during the reverse process, a trainable U-Net $\epsilon_\theta^{(t)}(\mathbf{s}_t) \in \mathbb{R}^{N\times 2}$ estimates noise in the form of coordinate offsets for a sketch $\mathbf{s}_t \in \mathbb{R}^{N\times 2}$ at timestep t. A trainable embedding and a decoding layer transforms the input $\mathbf{s}_t$ into an embedding $\mathbf{e}_t \in \mathbb{R}^{N\times 128}$, and the penultimate feature embedding is converted back to coordinates. A single head attention layer is used to inject timestep t's embedding into the convolution blocks in the U-Net during downsampling following DDIM [21].

The diffusion model learns a distribution over coordinate offsets for sketch points. However, to predict a binary pen state for each of the stroke points, the feature vector from the penultimate layer of the U-Net is passed through a trainable linear layer followed by a sigmoid function to get $\hat{g}^{(i)}$. When $\hat{g}^{(i)} > 0.5$, it indicates that the pen is touching the canvas at point i. This is done for each timestep in the generative process.

We add own modifications to the above, illustrated in Figure 3. To condition on text, we follow GLIDE [16]: we feed text tokens into a Transformer [20] model. The output of this transformer is used in two ways: first, the final token embedding is used in place of a class embedding in the U-Net model; second, the last layer of token embeddings (a sequence of feature vectors) is

Algorithm 1 Training the model using classifier-free guidance

```
Require: $p_{uncond}$: probability of unconditional training
Require: $\gamma$: weight given to the pen state loss
1: repeat
2:     $(\mathbf{s}_t, c) \sim p(\mathbf{s}_t, c)$        ▷ Sample data with text prompt from the dataset
3:     $c \leftarrow \varnothing$ with probability $p_{uncond}$        ▷ Randomly discard conditioning to train unconditionally
4:     $L_d(\theta) \leftarrow \mathbb{E}\left[\|\epsilon^{(t)} - \epsilon_\theta^{(t)}(\mathbf{s}_t|c)\|_2^2\right]$        ▷ $\ell_2$ loss between estimated and generated noise
5:     $L_p(\theta) \leftarrow \frac{1}{N}\sum_{i=1}^{N}\left[-g^{(i)}\log(\hat{g}^{(i)}) - (1-g^{(i)})\log(1-\hat{g}^{(i)})\right]$        ▷ Pen state loss
6:     $L(\theta) \leftarrow L_d(\theta) + \gamma L_p(\theta)$        ▷ Total training loss
7:     Take gradient step on $\nabla_\theta L(\theta)$        ▷ Optimization of denoising model
8: until converged
```

Algorithm 2 Sampling from the model using classifier-free guidance

```
Require: $w$: guidance strength $\geq 1$
Require: $c$: conditioning information for conditional sampling
1: $\hat{\epsilon}_\theta(\mathbf{x}_t|c) \leftarrow \epsilon_\theta(\mathbf{x}_t|\varnothing) + w \cdot (\epsilon_\theta(\mathbf{x}_t|c) - \epsilon_\theta(\mathbf{x}_t|\varnothing))$
2: $\mathbf{v} \in \mathbb{R}^{N\times 128} \leftarrow$ Feature vector from penultimate layer of U-Net
3: $f_\psi \leftarrow$ Trainable linear layer
4: $\hat{g} = \mathtt{sigmoid}(f_\psi(\mathbf{v}))$        ▷ Estimated pen state
```

separately projected to the dimensionality of each attention layer throughout the U-Net model, and then concatenated to the attention context at each layer. To further improve sample quality, we employ the classifier-free guidance strategy [23] as detailed by Algorithms 1 and 2. The reason for using classifier-free guidance over classifier guidance is its simplicity, leveraging the knowledge that the diffusion model has already learned, and not having to train a separate classifier on noisy samples.

The loss function optimized during training is:

$$\mathbb{E}\left[\|\epsilon^{(t)} - \epsilon_\theta^{(t)}(\mathbf{s}_t|c)\|_2^2\right] + \gamma\frac{1}{N}\sum_{i=1}^{N}\left[-g^{(i)}\log(\hat{g}^{(i)}) - (1-g^{(i)})\log(1-\hat{g}^{(i)})\right],$$

where the first term is the ℓ_2 loss between the estimated and generated noise, and the second term is the mean binary cross entropy loss between the estimated and target pen state, weighted by γ which is a hyperparameter.

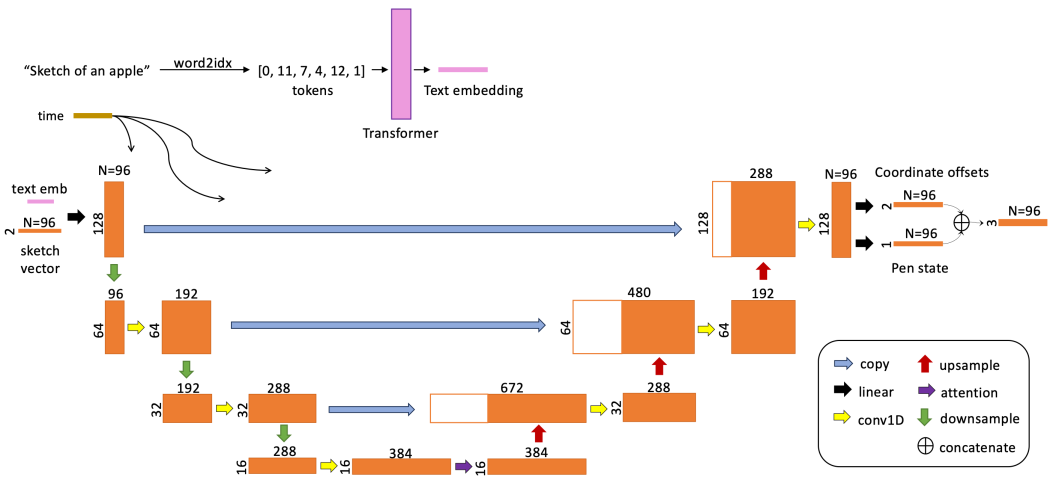


Figure 3: Our model architecture comprising of a U-Net for approximating noise and a Transformer for learning text embeddings.

Table 1: Model evaluation on FID, GS, Precision, and Recall (n=512 samples) using the best w for a given class. This is w=5 for all classes except `moon` which achieves maximum performance at w=1.

Model	FID ↓	GS ↓	Prec ↑	Rec ↑
SketchKnitter (Unconditional)	6.9	3.4	0.52	**0.88**
Ours (on text prompts for class `apple`)	6.4	3.1	**0.60**	0.75
Ours (on text prompts for class `umbrella`)	7.0	4.5	0.52	0.54
Ours (on text prompts for class `moon`)	6.6	**3.0**	0.58	0.71
Ours (on text prompts for class `shoe`)	**6.0**	3.2	0.57	0.77
Ours (on text prompts for class `lion`)	18.4	5.6	0.13	0.52
Ours (on text prompts for class `fish`)	6.3	3.1	0.59	0.74
Average across classes (w=5)	8.5	3.8	0.50	0.66
Average across classes (w/o `lion`) (w=5)	6.5	3.4	0.57	0.68

Table 2: Recognition results for SketchKnitter and our model on an AlexNet model. We use the same AlexNet parameters as those present in the SketchKnitter paper and evaluation code.

Model	acc@1	acc@10
SketchKnitter (Unconditional)	52.4	90.2
Average across classes (w=5)	55.6	92.1
Average across classes (w/o `lion`) (w=5)	56.1	92.4

5 Experiments

To run our experiments, we sample 512 images for every possible pair of sketch category ($\texttt{class} \in \{\texttt{apple}, \texttt{umbrella}, \texttt{moon}, \texttt{shoe}, \texttt{lion}, \texttt{fish}\}$) and w. Recall from Algorithm 2 that w represents the guidance strength hyperparameter and for our experiments, $w \in \{1, 2, 3, 4, 5\}$. w signifies the weight our sampling process should put on our conditioning information, where $w = 0$ is unconditional sampling (and, by extension, the output of SketchKnitter). We use this approach to evaluate the success of including conditioning information in our sampling process. The baseline we use is SketchKnitter, which is described in detail in the first half of Section 4, since it achieves state-of-the-art performance in vectorized sketch generation on all four of our evaluation metrics.

5.1 Model Hyperparameters

Our U-Net makes use of 3 residual network blocks, 4 attention heads, attention resolutions of $16, 8$, and dropout of 0.1. The transformer expects a text token length of 6, has a width of 24, 2 heads, a depth of 5, and uses Layer Norm after the output layer. We use 1000 diffusion steps for training and sampling, learning rate of 1e-4, batch size of 512, AdamW optimizer with a weight decay of 0.001, uniformly annealing LR scheduler, linear noise scheduler, and run the model for 100k iterations.

Our model is trained using a single NVIDIA T4 GPU on Google Cloud Provider.

5.2 Results

For our baseline model, which is SketchKnitter without text-conditioning, we directly take its evaluation performance from the paper. Section 3.2 provides a detailed explanation of the metrics we use for evaluation. We want to achieve low FID and GS scores (the real (human drawn) and generated image distributions should have similar features and geometric properties) and high precision and recall (the real (human drawn) and generated image distributions should overlap as much as possible).

We include the results of our experiments in Table 1, reporting them for the best w value for each class. We are able to achieve state-of-the-art performance or near state-of-the-art performance for FID, GS, and precision for 5 of our classes. Since FID and GS tend to reflect human-perceived similarity to a real image, we believe that this indicates that our samples accurately mirror their associated category. This is corroborated by our qualitative observations: Figure 4 shows 13 samples generated from each class.

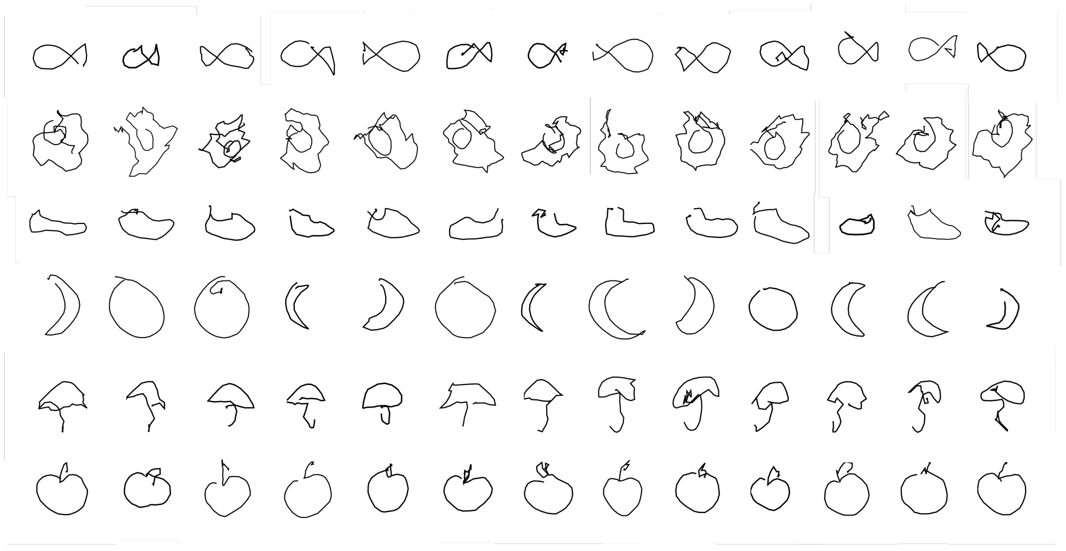


Figure 4: Samples generated by the model for $w = 5$ on different text prompts.

We do note that we were not able to achieve state-of-the-art performance on recall, even when discounting our outlier `lion` class. However, our recall is still on par with former state-of-the-art solutions such as SketchRNN [12] and SketchPix2Seq [27], which achieve recall scores of 0.72 as compared to our value of 0.68. Since our precision performs much better, this likely means that our model is mimicking the real sketch image distribution quite well but possibly only focusing on a small portion of it, indicating minor mode collapse. We also achieve state-of-the-art recognizability metrics on an AlexNet-based image classifier, surpassing the results of SketchKnitter even when factoring in our outlier class `lion`. These results are presented in Table 2.

5.3 Discussion

Though 5 of our classes achieved maximal performance when $w = 5$, it is interesting to note that `moon` achieved its best performance when $w = 1$, which is when the U-Net samples completely conditionally without any influence from the unconditionally-trained model, although there is no apparent trend that changing w has for this particular class over different metrics like other classes have. A possible reason why this might be is because `moon` is, arguably, the easiest class to draw sketches of, and increasing guidance strength, thus, does not amount to a clear improvement in sample quality. We also note that every w value for `moon` results in state-of-the-art or near state-of-the-art results across all metrics so this discrepancy may be accounted for by sampling variance.

We also note that one class `lion` performs much worse than any of the others. We have two possible hypotheses for this phenomenon. The first is that there is a training data disparity between `lion` and the other 5 classes. In fact, there are 3 times as many samples in any other class as there are in `lion`. With fewer data points, our model may have had a tougher time generalizing on this class. Our second hypothesis is that `lion` is simply harder to draw and requires more stroke points than any other class. We can observe this in Figure 4. The other classes appear to be the product of much fewer strokes and can all be drawn without lifting the "pen" once. `lion`, on the other hand, requires many more strokes and possible shifts in pen state. We note that SketchKnitter labeled `lion` as a complex category since it required >100 stroke points on average. The results of SketchKnitter also indicate that their metrics performed much worse on complex sketches. We did not separate our categories into categorical complexity due to resource constraints but this would be an interesting future analysis direction.

We note that, unlike the other metrics, our recognizability scores (for `w` = 5) perform better on all of our classes including `lion`. This likely demonstrates the importance of including any form of conditioning information during sampling since even though human perception proxy metrics perform worse, our sampling process is likely generating sketches which encode category-coded features that our AlexNet model picks up on.

6 Ablation Study

6.1 Evolution of Sketches with Time

In figure 5, we visualize the evolution of the sketch generation process for a single sketch from each of the 6 classes we train our model on. Out of the 1000 diffusion steps over which we sample, the first 600 steps make very slow progress in terms of gradually estimating the noise; the sketches are not generally recognizable before this point. In the last 400 steps, the progress toward sketching a recognizable sample is quick. While the object being sampled is "recognizable" by the 900th step, the last 100 steps work towards smoothing out the corners of the strokes, making them look more like curves and realistic.

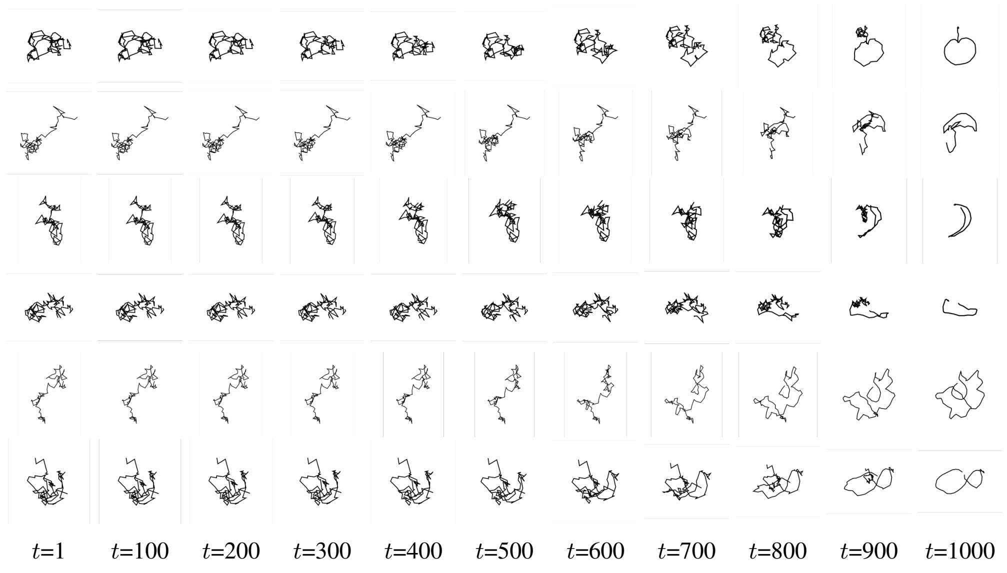


Figure 5: Evolution of sketches sampled from each class at timesteps $t \in \{1, 100, \ldots, 1000\}$.

6.2 Text Embedding

We visualize how the embeddings are learned for the text prompts we provide to the model during training in figure 6. This is done using the Uniform Manifold Approximation and Projection (UMAP) algorithm [28], which is able to perform non-linear dimensionality reduction, using the Euclidean metric.

There is an underlying structure that our model learns. All of the individual words (`<start>`, `shoe`, `an`, etc.), even though they appear in prompts for different classes, get mapped very close in the embedding space. Words that do not correspond to object names (`<start>`, `<end>`, `image`, `this`, `here`, `of`, `is`, `the`, and `sketch`) get mapped farther away from those that correspond to object names, implying that the model learns to differentiate between these subgroups. An interesting observation is that `a` and `an` also get mapped close to the object classes in the embedding space, possibly because their presence accentuates the difference between classes that are nouns (`apple` and `umbrella`) and those that are not (`fish`, `lion`, etc.).

6.3 Isotropic Gaussian Noise Perturbation of Text Embedding

The importance of text embedding over conditioning on only class labels is that, for the model, class labels act as distinct entities that need to be provided as is during the sampling stage to be able to generate a sketch for; i.e., if a model sees class labels $1, \ldots, k$ during training, it needs to see one of these k labels during sampling time to condition on. The advantage for text embedding is that similar entities are mapped closer together in the embedding space and it is possible to move through this continuous space.

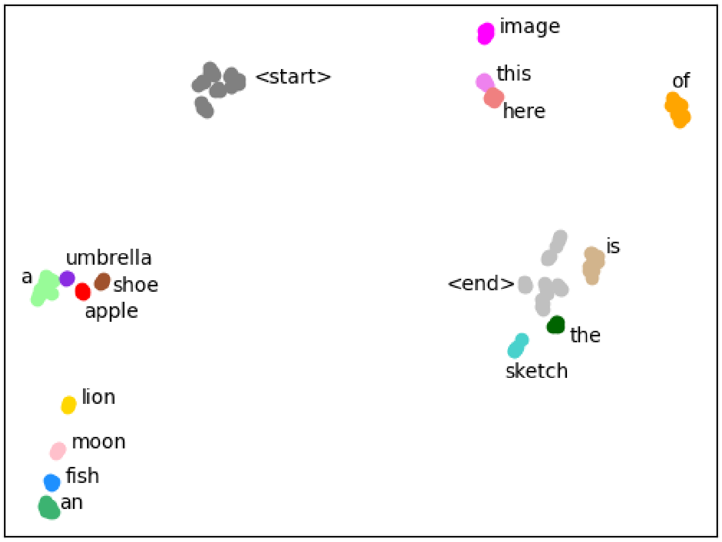


Figure 6: Visualization of text prompts used for training.

A sign of a robust model is that it is invariant to small perturbations to the text embedding. Here, we explore this question by adding isotropic Gaussian noise to a text embedding prior to feeding it to the model during the sampling stage. More precisely, if ζ is the embedding of a text prompt c, then we perturb ζ to get $\zeta + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, 1)$. Figure 7 shows that the model is still able to generate good quality samples from these perturbed embeddings, highlighting the power of using text over class labels.

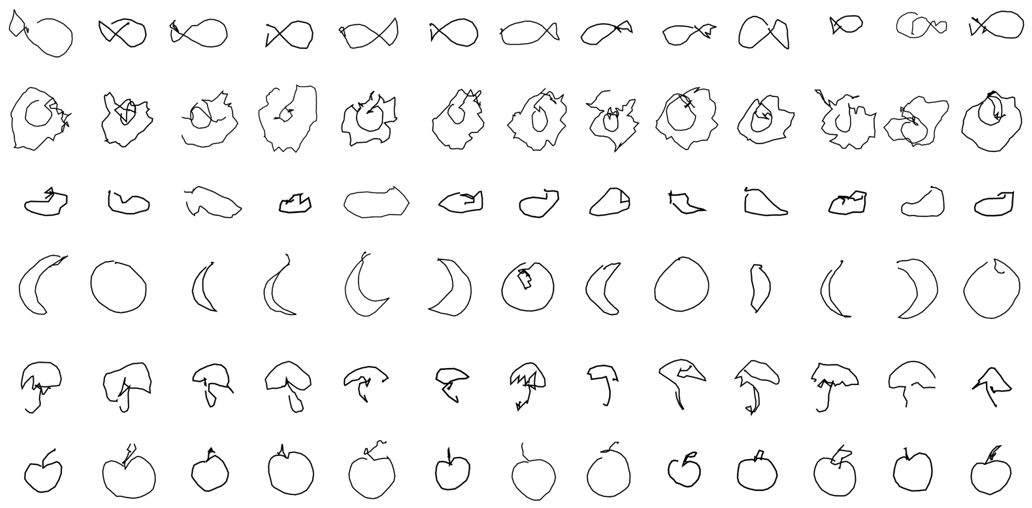


Figure 7: Samples generated after adding isotropic Gaussian noise to text embeddings.

6.4 Changing the Guidance Score

We note in Figures 8 and 9 that increasing the guidance score generally improves our metrics. The one exception, `moon`, is discussed in Section 5.3. This is in line with what is expected since higher guidance scores encode more conditional information which can be used to generate more informed samples.

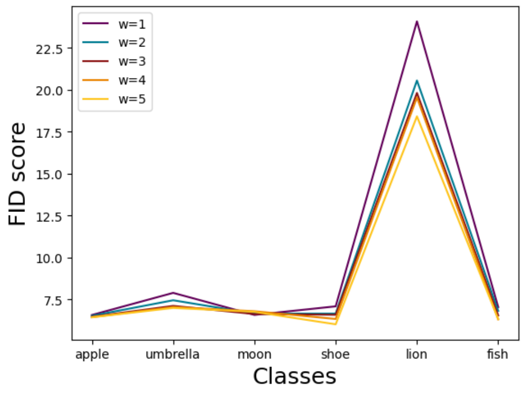


Figure 8: Plot of FID versus guidance score per class.

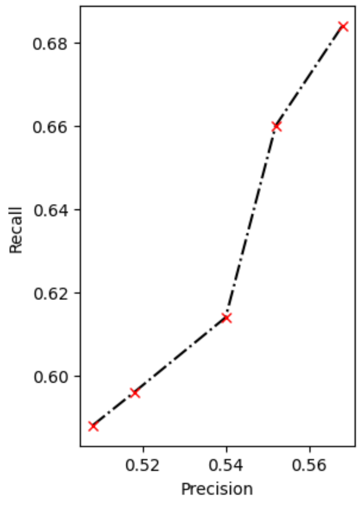


Figure 9: Precision vs Recall trade off. We sweep over guidance scores of $\{1, 2, 3, 4, 5\}$.

7 Conclusion and Future Work

Using SketchKnitter as a baseline, we were able to construct a model that uses text-conditioning to generate vectorized sketches. Using simple annotation techniques and inspired by GLIDE, we are able to condition on text during training by feeding text tokens into a Transformer serving as a text encoder. We diverge further from SketchKnitter's architecture by utilizing classifier-free guidance as opposed to classifier guidance which simplifies the sampling process while still generating impressive samples due to the knowledge learned by the diffusion model. Across five different classes, our model achieves competitive performance on various metrics. Text-conditioned vectorized sketch generation is an important yet under-explored field of generative modeling and our model hopefully represents the first step to generating accurate sketches for use in UIs, advertising, or even damaged artwork restoration.

Further research can look into how generalizable this approach is in terms of training on more carefully curated text annotations and how easily our model can extend to out-of-vocabulary words, how sampling can be sped up since 1000-step diffusion is much slower than non-diffusion approaches, and how an evaluation framework can be constructed for conditional vectorized sketch generation. Since there is no precedent for text-conditioned vectorized sketch generation, we used popular evaluation metrics from the unconditional vectorized sketch generation literature but there may be better methods more suited towards conditional approaches.

References

[1] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

[2] Yanghua Jin, Jiakai Zhang, Minjun Li, Yingtao Tian, Huachun Zhu, and Zhihao Fang. Towards the automatic anime characters creation with generative adversarial networks. *CoRR*, abs/1708.05509, 2017.

[3] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. In *International Conference on Learning Representations*, 2017.

[4] Waqar Ahmad, Hazrat Ali, Zubair Shah, and Shoaib Azmat. A new generative adversarial network for medical images super resolution. In *Scientific Reports 12, 9533*, 2022.

[5] Wei Peng, Ehsan Adeli, Qingyu Zhao, and Kilian Pohl. Generating realistic 3d brain mris using a conditional diffusion probabilistic model. 12 2022.

[6] Liqian Ma, Xu Jia, Qianru Sun, Bernt Schiele, Tinne Tuytelaars, and Luc Van Gool. Pose guided person image generation. 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.

[7] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. *arXiv preprint arXiv:1801.07892*, 2018.

[8] Peiye Zhuang, Oluwasanmi O Koyejo, and Alex Schwing. Enjoy your editing: Controllable {gan}s for image editing via latent space navigation. In *International Conference on Learning Representations*, 2021.

[9] Jingwei Guan, Cheng Pan, Songnan Li, and Dahai Yu. Srdgan: learning the noise prior for super resolution with dual generative adversarial networks. *arXiv preprint arXiv:1903.11821*, 2019.

[10] Songwei Ge, Vedanuj Goswami, Larry Zitnick, and Devi Parikh. Creative sketch generation. In *International Conference on Learning Representations*, 2021.

[11] Runtao Liu, Qian Yu, and Stella Yu. Unsupervised sketch to photo synthesis. 2020.

[12] David Ha and Douglas Eck. A neural representation of sketch drawings. In *International Conference on Learning Representations*, 2018.

[13] Guoyao Su, Yonggang Qi, Kaiyue Pang, Jie Yang, and Yi-Zhe Song. Sketchhealer: A graph-to-sequence network for recreating partial human sketches. In *BMVC*, 2020.

[14] Qiang Wang, Haoge Deng, Yonggang Qi, Da Li, and Yi-Zhe Song. Sketchknitter: Vectorized sketch generation with diffusion models. In *The Eleventh International Conference on Learning Representations*, 2023.

[15] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

[16] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

[17] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Raphael Gontijo-Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[18] Yuanzhi Zhu, Zhaohai Li, Tianwei Wang, Mengchao He, and Cong Yao. Conditional text image generation with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14235–14245, 2023.

[19] Troy Luhman and Eric Luhman. Diffusion models for handwriting generation. *arXiv preprint arXiv:2011.06704*, 2020.

[20] 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.

[21] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *CoRR*, abs/2010.02502, 2020.

[22] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[23] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[24] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[25] Valentin Khrulkov and Ivan Oseledets. Geometry score: A method for comparing generative adversarial networks. In *International conference on machine learning*, pages 2621–2629. PMLR, 2018.

[26] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *CoRR*, abs/1904.06991, 2019.

[27] Yajing Chen, Shikui Tu, Yuqi Yi, and Lei Xu. Sketch-pix2seq: a model to generate sketches of multiple categories. *arXiv preprint arXiv:1709.04121*, 2017.

[28] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.